N° d'ordre: 3243

THÈSE

présentée

devant l'Université de Rennes 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention INFORMATIQUE

par

Martín VARELA RICO

Équipe d'accueil : ARMOR - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :

Évaluation Pseudo–Subjective de la Qualité d'un Flux Multimédia et ses Applications au Contrôle

soutenue le 10 Novembre 2005 devant la commission d'examen

M. Erol	GELENBE	Professeur	Imperial College, Londres	Rapporteur
M. Peter	REICHL	Directeur de Recherche	FTW, Vienne	Rapporteur
M. Alain	JEAN–MARIE	Directeur de Recherche	INRIA Sophia-Antipolis	Rapporteur
M. Michel	BANÂTRE	Directeur de Recherche	INRIA Rennes	Président du Jury
M. Héctor	CANCELA	Professeur	U. de la República, Montevideo	Examinateur
M. Gerardo	RUBINO	Directeur de Recherche	INRIA Rennes	Directeur de Thèse

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein.

Remerciements

Je tiens à remercier les membres du jury d'avoir accepté d'évaluer mes travaux de thèse. Je remercie en particulier M. Gelenbe, M. Reichl et M. Jean–Marie d'avoir eu la gentillesse d'être rapporteurs de mon travail.

Je remercie à tous les membres du projet ARMOR, qui m'ont très bien accueilli, despuis mon début en tant que stagiaire, et en particulier à Yezekael et Francine, qui ont été des superbes voisins pendant ces trois dernières années.

Á Karina et Helena, qui à leur tour ont su me soutenir tout au long de la thèse, et à ma famille, qui a été un appui indispensable au fil de mes années d'études.

Finalement, je remercie spécialement M. Gerardo Rubino, qui a su me guider pour mener ce travail à bon terme, pour tout son support pendant ces dernières quatre années.

Contents

Chapter 1

Évaluation Pseudo–subjective de la Qualité d'un Flux Multimédia et ses Applications au Contrôle

1.1 Introduction

Au cours de ces dernières années, l'explosion d'Internet a donnée lieu à une nouvelle génération d'applications multimédia qui l'utilisent comme moyen de diffusion. De telles applications, par exemple la transmission de la vidéo en temps-réel et à la demande, la téléphonie, ou la téléconférence, sont devenues courantes et elles constituent aujourd'hui un domaine de recherche très actif ainsi qu'un nouveau secteur du marché des télécommunications.

En fait, à l'heure actuelle, plusieurs offres commerciales existent pour la distribution de services multimédia sur des liens haut–débit (par exemple, des chaînes numériques de TV sur des liens xDSL, ou encore les services de Voix sur IP, à des prix très compétitifs qui sont inclus dans certains forfaits de haut–débit), et même sur les réseaux mobiles (par exemple, des services de vidéoconference sur des téléphones mobiles de troisième génération, ou 3G). Afin d'être viables, ces services doivent fournir des niveaux de qualité comparables à ceux du téléphone et des services de TV traditionnels. Bien que quelques concessions peuvent être faites, il est peu probable que les utilisateurs payent pour des services dont la qualité est inférieure à celle dont ils ont l'habitude. Par conséquent, il devient nécessaire de développer des mécanismes qui permettent de surmonter les problèmes techniques présentés par les réseaux IP actuels aux applications du type temps réel.

Ces applications ont certains besoins vis-à-vis des services du réseau, dues à des contraintes temporelles qui leur sont propres. Or, l'architecture d'Internet n'a pas été conçue pour supporter le temps-réel, ce qui entraîne plusieurs problèmes à résoudre lorsque l'on veut implémenter des applications ayant ces types de contraintes. Ainsi, la qualité de ce type d'applications est fortement dépendante de la capacité, la charge et la topologie du réseau. Plusieurs résultats [14, 25, 27, 58, 117] montrent que certains paramètres du réseau tels que le taux de pertes, la distribution de pertes, le délai et la gigue influencent la qualité perçue par les utilisateurs. Cependant, la façon dont ces paramètres affectent ladite qualité reste inconnue.

Afin de développer les mécanismes mentionnés ci-dessus pour contrôler la qualité perçue des applications multimédia qui tournent sur un réseau sanas garanties de qualité de service, il est important de pouvoir évaluer la qualité perçue des dites applications, pour savoir quels ajustements doivent être faits, et quand ils doivent être faits. Pour cela, il ne suffit pas d'avoir une évaluation qualitative simple ; les applications de contrôle doivent avoir une vision précise et quantitative de la qualité perçue. D'ailleurs, il est clairement utile de pouvoir réaliser cette évaluation en temps réel, afin de pouvoir réagir aux changements rapides des conditions du réseau. Si l'influence des différents facteurs sur la qualité perçue est connue, on peut penser à optimiser ladite qualité. Une fois que la qualité est déterminée, une certaine forme de contrôle peut être mise en œuvre.

L'ampleur de ce contrôle dépendra de l'application, et du réseau sur lequel elle tourne. Dans le cas le plus général (i.e. le service *best–effort* que l'on trouve sur l'Internet), les améliorations de la qualité doivent être faites au niveau applicatif. Si le réseau met en œuvre des mécanismes de QoS, ceux-ci peuvent également être employés pour améliorer la qualité perçue.

Mis à part les applications de contrôle, la viabilité commerciale des applications multimédia en temps-réel se fonde sur l'existence des accords de niveau de service (SLA) et des schémas de tarification qui permettent à l'utilisateur d'avoir certaines garanties en ce qui concerne le service qu'il reçoit, et l'opérateur de réseau (et fournisseur de contenu, le cas échéant) d'optimiser leurs bénéfices. Tant pour les SLAs comme pour la tarification, l'évaluation de la qualité est très importante.

1.2 Contributions

Les contributions présentées dans ce document ont pour but d'aider à fournir une meilleure compréhension de la qualité des flux multimédia transmis sur un réseau de paquets, et de permettre le développement de mécanismes de contrôle en temps réel de cette qualité. Pour cela, nous nous basons sur une technique d'évaluation de la qualité originalement proposée par Samir Mohamed [98], que nous appelons *évaluation pseudo–subjective de la qualité* (PSQA par son nom en anglais). Cette méthode permet d'avoir une très bonne estimation de la qualité d'un flux multimédia telle qu'elle est perçue par un utilisateur moyen. Ces estimations peuvent être employées pour mieux comprendre comment les différents facteurs affectent la qualité perçue, et donc pour mettre en œuvre des mécanismes de contrôle permettant de l'améliorer.

Nos contributions peuvent être classées dans les points suivants :

- Étude approfondie de PSQA et évaluation de ses performances.
- Analyse de la qualité des flux VoIP (*one–way* et interactifs), et les facteurs que la définissent sous plusieurs contextes de réseau (notamment des réseaux filaires et sans fil).

- Intégration de PSQA avec des méthodes traditionnelles d'évaluation de performance.
- Développement des mécanismes de contrôle dynamique de la qualité, basés sur la qualité perçue.

1.2.1 Étude approfondie de PSQA et Évaluation de sa Performance

Notre première contribution est une étude complète de PSQA, en particulier dans le contexte de l'évaluation de qualité de voix sur IP. Les travaux précédents faits par Mohamed dans ce domaine avaient laissé certains aspects de côté. Nous fournissons donc une analyse détaillée de la technique et de ses performances.

PSQA permet, par l'intermédiaire d'un estimateur statistique (tel qu'un réseau de neurones), d'imiter la manière dont un sujet humain moyen évalue la qualité d'un flux multimédia. Contrairement à d'autres méthodes d'évaluation qui se fondent parfois sur des modèles de perception humaine pour réaliser l'évaluation, PSQA établit des liens entre certains facteurs qui affectent la qualité, et la qualité elle même. De cette façon, PSQA peut *apprendre* la relation entre ces facteurs et la qualité telle que perçue par les utilisateurs.

Nous étudions les performances de la technique mis en œuvre avec des Réseaux de Neurones Aléatoires (RNN) [42], qui nous permettent d'obtenir des évaluations très précises de la qualité perçue pour un flux multimédia. Notre étude couvre plusieurs architectures de RNN, le rapport entre le nombre d'échantillons employés pour entraîner l'outil et sa performance en termes de corrélation avec les évaluations fournis par les utilisateurs, et sa généralité et robustesse.

Nous comparons également la performance de PSQA à celle d'autres techniques d'évaluation de qualité de la littérature. Nous montrons que PSQA est comparable avec la meilleur parmi elles, et présentons des avantages additionnels, tels que la capacité de fournir les évaluations en temps réel, et illustrons la relation entre chaque paramètre et la qualité elle-même.

Ces résultats peuvent être trouvés dans le Chapitre 5 de ce document.

1.2.2 Analyse de la Qualité des Flux VoIP

Un résultat direct de la manière dont PSQA fonctionne est qu'elle permet d'observer et analyser la qualité d'un flux multimédia sous un éventail de conditions différentes. Comme mentionné avant, nous concentrons nos études principalement sur la qualité de la VoIP, mais le même genre d'analyse peut être réalisée sur la vidéo ou des flux mixtes audio / vidéo.

La deuxième contribution de cette dissertation est l'étude de la qualité perçue des flux VoIP, et comment elle est affectée par les facteurs applicatifs et de réseau qui la définissent. Nous analysons la manière dont chacun de ces facteurs affecte la qualité perçue, sous différent contextes de réseau.

Nous étudions des flux VoIP *one–way* sur des réseaux IP filaires et sans fil, sous un éventail de conditions différentes. Nous présentons également des résultats préliminaires pour les flux interactifs de voix, pour lesquels il n'existe, à notre connaissance, aucune autre étude d'une portée semblable. L'information obtenue par ce type d'évaluation est utile par exemple pour le calcul des dimensions et la conception des réseaux, et cruciale pour développer les mécanismes dynamiques de contrôle de qualité efficaces pour les applications multimédia.

Nos résultats principaux dans ce domaine sont présentés dans les Chapitres 6 et 8.

1.2.3 Intégration de PSQA avec des Méthodes Traditionnelles d'Évaluation de Performance

La troisième contribution principale de cette dissertation est une méthode pour intégrer PSQA avec d'autres techniques d'évaluation de performance, telles que l'analyse des files d'attente ou la simulation. L'idée est qu'il pourrait être utile, pour concevoir ou dimensionner un réseau, d'être capable de prévoir ce que sera la qualité pour certaines applications (en particulier, celles multimédia).

Nous montrons comment, en dérivant des métriques de performance d'un modèle du réseau, on peut obtenir des évaluations de qualité pour des flux multimédias. Nous illustrons ceci avec un cas d'étude simple, en utilisant un modèle simple de files d'attente pour représenter le réseau, ou son goulot d'étranglement, et nous fournissons une expression analytique simple pour la qualité en termes de la charge et la capacité du réseau. Nous étendons aussi cette étude avec un modèle multi–classe, afin d'estimer et comprendre les effets de l'addition de la redondance à un flux de voix sur l'état de réseau et la qualité perçue.

Ce mécanisme d'intégration est décrit dans le Chapitre 7, et employé dans le Chapitre 8, où nous proposons un modèle stochastique pour un réseau sans fil, et étudions la qualité des flux VoIP dans ce contexte.

1.2.4 Contrôle Dynamique de la Qualité des Applications Multimédias

Notre dernière contribution concerne le contrôle dynamique d'une application multimédia. Nous visons à fournir des mécanismes pour contrôler la qualité perçue d'un flux multimédia en temps réel, basés sur la qualité courante. Il y a plusieurs méthodes de contrôle proposées dans la littérature pour différents types d'applications multimédia. Elles sont généralement basées sur la mesure d'un ou deux paramètres de réseau, typiquement le taux de perte de paquets et le délai, et la connaissance de qu'une augmentation quelconque d'un d'eux est nuisible à la qualité. Cependant, aucune de ces méthodes considère la qualité perçue réelle.

Nous proposons est l'utilisation de PSQA pour surveiller la qualité perçue, afin de pouvoir réagir quand elle atteint certains seuils. Ceci nous permet de maintenir des niveaux de qualité acceptable autant que possible, mais également de réduire la consommation de ressources quand les conditions de réseau le permettent, pour mieux partager ces ressources avec les autres applications utilisant le réseau.

Nous proposons deux algorithmes *proof–of-concept* pour illustrer ce que peut être fait avec PSQA pour le contrôle des applications de voix. Nous comparons la qualité obtenue quand ils sont employés à celle obtenue avec des configurations statiques, sous des conditions de réseaux variées. Les algorithmes proposés sont conçus pour des flux *one–way*, mais nous prouvons qu'ils peuvent être facilement adaptés pour des flux interactifs.

Finalement, nous étudions également des mécanismes pour améliorer la qualité de la VoIP sur des réseaux IP sans fil, et nous discutons comment ils pourraient être employés pour faire du contrôle dynamique de ladite qualité. Nous proposons un schéma de priorités pour le traitement des paquets sur le lien sans fil, et nous montrons comment il affecterait la qualité perçue pour des flux de voix et de vidéo, et son impact sur le trafic des autres applications qui tournent sur le réseau.

Ces résultats se trouvent dans le Chapitre 9.

1.3 Sur les Méthodes d'Évaluation de la Qualité Perçue

Dans la littérature, on trouve deux types de métriques de qualité : les *évaluations subjectives* (e.g. celle définie dans [80, 85]), qui sont réalisées par un groupe de personnes dans un environnement contrôlé, et les *évaluations objectives*, basées sur des formules ou des algorithmes et qui permettent d'obtenir une mesure de la qualité d'un échantillon de voix (ou bien d'audio au sens large).

Étant donnée que la qualité d'un flux multimédia est un concept qui reste très subjectif, il n'est pas étonnant que les évaluations subjectives soient celles qui donnent les “vrais valeurs” de la qualité perçue. Le problème de ce type d'approche est que ces évaluations sont très chères à réaliser et aussi très encombrantes, car pour être réalisées, elles ont besoin d'une logistique assez importante. Il est donc intéressant de développer des méthodes d'évaluation objective de la qualité permettant d'éviter les difficultés associées aux méthodes subjectives. Ces méthodes présentent d'autres problèmes, surtout concernant leurs performances.

1.3.1 Évaluation Subjective

Les mécanismes d'évaluation subjective de la qualité d'un flux multimédia consistent principalement en des expériences de laboratoire, dans lesquelles un certain nombre de sujets sont exposés à des échantillons d'audio ou de vidéo qui ont été dégradés et doivent donner des points à chacun. Après un filtrage statistique, on prend la moyenne des opinions (*mean opinion score - MOS*) comme mesure de la qualité de l'échantillon. Ces mécanismes ont été standardisés [85] et sont très répandus [58].

Bien que l'évaluation subjective est le mécanisme le plus précis pour déterminer la qualité des flux audio transmis sur un réseau de paquets, elle présente plusieurs inconvénients, à savoir :

- très coûteuse, tant en temps qu'en ressources,
- encombrante à réaliser, et
- inadaptée aux applications temps-réel (et donc à la tarification ou au contrôle des flux).

1.3.2 Évaluation objective

Les difficultés associées à l'évaluation subjective de la qualité ont donnée lieu au développement d'autres métriques de qualité qui, bien que moins "performantes", sont beaucoup plus pratiques et moins coûteuses. Parmi les méthodes objectives les plus connues on trouve le Signal-to-Noise Ratio (SNR), le Segmental SNR (SNRseg), le Perceptual Speech Quality Measure (PSQM) [12], les Measuring Normalizing Blocks (MNB) [128], l'ITU E–model [81], l'Enhanced Modified Bark Spectral Distortion (EMBSD) [136], le Perceptual Analysis Measurement System (PAMS) [111] et PSQM+ [10]. Ces méthodes ne donnent pas toujours des bonnes corrélations avec la perception humaine et donc leur utilité comme remplacements des évaluations subjectives est limitée. De plus, à l'exception de l'E-model, toutes ces métriques proposent une façon de comparer l'échantillon reçu avec l'original et ne sont donc pas adaptées aux applications en temps réel.

Étant donné que la qualité d'un flux VoIP est influencée par beaucoup de paramètres ([27, 58], il n'est pas facile de concevoir une formule synthétique qui puisse les relier et qui donne une "bonne" évaluation quantitative. Cette complexité nous a mené à développer une méthode alternative pour l'évaluation de la qualité de ces flux.

1.4 PSQA – Une Approche Pseudo–subjective : l'Évaluation avec des Réseaux de Neurones Aléatoires

La méthode que nous étudions [100, 117] est un hybride entre l'évaluation subjective et l'évaluation objective, qui peut être appliqué aussi bien à la VoIP, qu'à l'audio *hi-fi* et à la vidéo. L'idée est de faire évaluer subjectivement plusieurs échantillons dégradés et utiliser ensuite ces résultats pour entraîner un RNN, en lui apprenant la relation entre les paramètres à l'origine de la dégradation des échantillons et la qualité perçue.

Pour que cela fonctionne, on doit considérer un ensemble de P paramètres qui peuvent avoir un effet sur la qualité. Pour la voix, par exemple, on peut choisir le codec, le taux de pertes dans le réseau, le délai moyen de bout en bout, etc. On appellera cet ensemble $\mathcal{P} = \{\pi_/, \ldots, \pi_\mathrm{P}\}$. Une fois $\mathcal{P}$ défini, on doit choisir des valeurs représentatifs pour chaque π_i, avec une valeur minimal $\pi_{\min}$ et une valeur maximal $\pi_{\max}$, selon les conditions sous lesquelles on pense que le système fonctionnera. Soit $\{p_{i1}, \cdots, p_{iH_i}\}$ l'ensemble de ces valeurs, où $\pi_{\min} = p_{i1}$ and $\pi_{\max} = p_{iH_i}$. Le nombre de valeurs à choisir dépend de la taille de l'intervalle choisi et de la précision désirée. Dans ce contexte, on appellera *configuration* à un ensemble $\gamma = \{v_1, \ldots, v_P\}$ où v_i est l'une des valeurs choisies pour p_i.

Comme le nombre total de configurations peut être très important, il faut choisir un sous-ensemble de configurations à utiliser pour l'évaluation subjective. Ce choix peut être aléatoire, mais il est important d'avoir des valeurs aux extrémités des intervalles considérés pour chaque paramètre. Il est aussi important d'avoir plus de points dans la région de l'espace où se trouvent les configurations qui seront les plus courantes dans l'usage quotidien de l'application considérée. Une fois que les configurations on été choisies, il faut générer des échantillons dégradés par la transmission sur le réseau avec chaque configuration considérée. Pour cela, on peut utiliser un simulateur, ou une maquette de réseau.

Plus formellement, on doit choisir un ensemble de M échantillons (σ_m), $m = 1, \cdots, M$ d'un type et d'une durée en accord à ce qui est spécifié par exemple dans [85]. On a aussi besoin d'un ensemble de S configurations $\{\gamma_1, \cdots, \gamma_S\}$, où $\gamma_s = (v_{s1}, \cdots, v_{sP})$ et v_{sp} est la valeur du paramètre π_p dans la configuration γ_s. À partir de chaque échantillon σ_i on construit un ensemble $\{\sigma_{i1}, \cdots, \sigma_{iS}\}$ d'échantillons qui ont été transmis sur le réseau sous des conditions variées. C'est à dire, l'échantillon σ_{is} est la séquence qui a été reçue lorsque l'émetteur à envoyé σ_i à travers le système source–réseau où les P paramètres considérés ont les valeurs correspondantes à la configuration γ_s.

Une fois que les échantillons ont été générés, il faut réaliser une évaluation subjective, pour associer une valeur de qualité à chaque échantillon dégradé (et donc à chaque configuration). Après un filtrage statistique des résultats, la séquence σ_{is} reçoit une note μ_{is} (souvent cette note est un MOS). Une fois que l'on a une valeur de qualité associée à chaque configuration, on peut entraîner le RNN qui pourra par la suite donner des estimations de qualité pour des flux semblables (dans le sens où les paramètres considérés aient des valeurs dans les intervalles considérés, ou pas trop loin) à ceux que l'on a fait évaluer subjectivement. Pour entraîner le réseau, on utilise une partie des résultats obtenus avec les tests subjectifs et on garde une autre partie pour vérifier que le RNN est capable de faire des bonnes évaluations pour des configurations qu'il ne connaît pas (si les résultats sont acceptables, on dit que le réseau est validé).

Cette méthode permet d'avoir des bonnes estimations de la qualité pour des variations importantes de tous les paramètres considérés, au coût de l'évaluation subjective de quelques échantillons.

1.4.1 RNN : Réseaux de Files d'Attente comme Outils d'Apprentissage

Dans cette Section nous présentons brièvement les principes mathématiques sous-jacents aux RNN. Un RNN est en fait un type de réseau de files d'attente qui a été développé assez récemment [42, 45, 51].

Il s'agit d'un réseau Markovien ouvert, avec des clients positifs et négatifs, aussi appelé un G-Network. On a N nœuds (ou neurones) qui sont des files $./M/1$ (on notera le taux de service du nœud i par ν_i), inter-connectés, qui reçoivent et renvoient des clients depuis et vers l'environnement. Le processus d'arrivée des clients positifs (négatifs) est Poisson avec un taux λ_i^+ (λ_i^-). À la sortie de la file i, un client abandonne le réseau avec une probabilité d_i, va vers la file j en tant que client positif avec une probabilité r_{ij}^+, ou en tant que client négatif avec une probabilité r_{ij}^-. Lorsqu'un client négatif arrive dans une file, il tue le dernier client (s'il y en avait) et il disparaît. Les transferts entre files sont instantanés et donc on ne voit jamais un client négatif dans le réseau. Á n'importe quel instant, on observe seulement des clients positifs dans le réseau. Les clients négatifs jouent un rôle de *signaux*, modifiant l'état du système.

Notons X_t^i le nombre de clients dans la file i à l'instant t. Il a été prouvé dans [43, 45] que lorsque le processus Markovien $\vec{X}_t = (X_t^1, \cdots, X_t^N)$ est stable, sa distribution stationnaire est du type forme–produit. Donc, si l'on impose que $(\vec{X}_t)$ est stationnaire, on a :

$$\Pr(\vec{X}_t = (k_1, \cdots, k_N)) = \prod_{i=1}^{N} (1 - \varrho_i)\varrho_i^{k_i}.$$

Les facteurs $\varrho_1, \cdots, \varrho_N$ sont les charges des nœuds dans le réseau. Ces charges ne se calculent pas avec un système linéaire comme pour les réseaux de Jackson, mais avec le système non-linéaire suivant :

$$\varrho_i = \frac{\lambda_i^+ + \sum_{j=1}^{N} \varrho_j \nu_j r_{ji}^+}{\nu_i + \lambda_i^- + \sum_{k=1}^{N} \varrho_k \nu_k r_{ki}^-}.$$

On peut prouver que lorsque ce système a une solution $\varrho_1, \cdots, \varrho_N$ telle que pour chaque nœud i on a $\varrho_i < 1$, alors le processus est stable et le résultat forme–produit est valable (cf. [43]).

Pour utiliser un tel réseau de files d'attente comme un outil d'apprentissage, on fait comme suit : les variables d'entrée du système (dans notre cas, les paramètres de réseau et codage considérés) correspondant à une configuration γ_t sont normalisées en [0,1] et on les associe avec les taux d'arrivée des clients positifs à P nœuds spécifiques, $\lambda_1^+, \cdots, \lambda_P^+$. Tous les autres taux d'arrivée de clients positifs depuis l'environnement sont mis à 0, de même que tous les taux d'arrivée de clients négatifs depuis l'environnement. La qualité associée à γ_t lors de l'évaluation subjective est associée (après normalisation dans [0,1]) à la charge d'un nœud spécifique o. Le problème est alors réduit à trouver un réseau tel que lorsque $\lambda_1^+ = v_{1t}, \cdots, \lambda_P^+ = v_{Pt}$, la charge du nœud o est proche de μ_t et ceci pour toutes les configurations que l'on souhaite utiliser pour l'apprentissage. Ceci est un problème d'optimisation où les variables de contrôle sont les paramètres restants du système, c'est à dire, les taux de service ν_i et les probabilités de routage r_{ij}^+ and r_{ij}^-.

Pour tous les nœuds i tels que $d_i < 1$ (c'est à dire, ceux qui n'envoient pas tous leurs clients vers l'environnement), on note $w_{ij}^+ = \nu_i r_{ij}^+$ et $w_{ij}^- = \nu_i r_{ij}^-$. Ces facteurs sont appelés *poids*, comme pour les réseaux de neurones classiques et ils jouent un rôle similaire dans ce modèle. Normalement, lors de l'apprentissage, au lieu d'optimiser par rapport aux taux de service et aux probabilités de routage, on le fait par rapport aux poids et on fixe les taux de service des neurones "de sortie" (i.e., qui ont $d_i = 1$) à une valeur arbitraire.

1.5 Comparaison de la Performance de PSQA et d'Autres Méthodes Objectives

1.5.1 Performances des Méthodes Objectives

Pour déterminer la valeur d'une méthode d'évaluation de qualité dans un contexte donné, il faut savoir comment elle se comporte par rapport aux méthodes subjectives, qui sont la référence dans ce domaine.

Dans cette section nous présentons des données de performance de quelques unes des méthodes d'évaluation objective les plus répandues disponibles dans la littérature (e.g. [55, 136]). Nous présentons aussi des mesures de performance de PSQA, tirées de quelques expériences que nous avons menées. Il faut remarquer que nous ne pouvons pas comparer les résultats directement, étant donnée que les données proviennent d'expériences indépendantes. Néanmoins, la comparaison permet d'avoir une idée des performances relatives des différentes approches.

La plupart des méthodes objectives que l'on trouve actuellement dans la littérature ont été conçues pour déterminer les pertes de qualité introduits par exemple par les techniques de compression, etc. Généralement ces métriques ne sont donc pas prévues pour prendre en compte les dégradations que peut subir un flux en traversant le réseau. De plus, la plupart de ces métriques proposent une façon d'estimer la qualité essentiellement en comparant le signal émis et le signal reçu et elles ne sont donc pas adaptées à une utilisation en temps réel (au moins comme métriques passives).

Dans la suite, nous présentons quelques mesures de performance pour les métriques suivantes : SNR, SNRseg, BSD, MBSD, EMBSD, PSQM, PSQM+, MNB(1,2), E-model, et PAMS. Ces mesures prennent en compte l'utilisation avec des dégradations dues au codage et aussi à la combinaison de codage et de la transmission à travers un réseau IP.

Pour déterminer la performance d'une métrique, on mesure le coefficient de corrélation des valeurs qu'elle produit avec les valeurs d'une évaluation subjective (MOS dans les cas que nous présentons). La Table 1.1 (sources : [136], [128], [113] et [111]) montre les coefficients de corrélation pour plusieurs métriques quand on ne considère que les dégradations dues au codage. On peut observer que les méthodes les plus simples (SNR et SNRseg) ont une performance assez pauvre mais aussi il y a des méthodes, telles que PSQM et PSQM+ qui donnent de très bons résultats. Lorsqu'il y a plusieurs valeurs pour le coefficient de corrélation, cela veut dire que la métrique à été évaluée avec des échantillons qui ont subi différents niveaux de dégradation.

Métrique	Corrélation avec MOS
SNR	0.22-0.52
SNRseg	0.22-0.52
BSD	0.36-0.91
PSQM	0.83-0.98
PSQM+	0.87-0.98
MNB2	0.74-0.98
PAMS	0.64-0.89
MBSD	0.76
EMBSD	0.87

Table 1.1: Coefficients de corrélation de quelques méthodes objectives d'évaluation de qualité avec le MOS. Ces résultats ont été pris dans littérature et seules les dégradations dues au codage sont considérées.

Lorsque nous considérons aussi quelques paramètres de réseau (cf. Table 1.2), la performance de ces métriques exhibe une baisse importante. L'étude comparative la plus complète de ces métriques objectives dont nous avons connaissance est celle de [136]. Malheureusement, à cause de contraintes de confidentialité imposées à l'auteur lors de la publication de son travail, tous les résultats ne sont pas donnés explicitement (il y a plusieurs méthodes qui ne sont pas nommées). De toute façon ceci ne gêne pas notre étude, car nous voulons avoir une idée de la performance de notre méthode par rapport aux autres méthodes objectives considérées. Les conditions de réseau considérées correspondent plutôt à un réseau de type GSM ou CDMA (erreurs au niveau des bits reçus, décalage temporel, *front clipping*, perte de paquets et variations de niveau), mais cela sert à montrer que lorsque l'on introduit d'autres facteurs de dégradation, les métriques ont du mal à fonctionner correctement. Il faut aussi noter que les coefficients de cette table ont été calculés à partir des courbes de régression et non pas des vraies estimations, ce qui peut les faire monter légèrement.

La Table 1.3 présente des résultats publiés dans [55, 136], qui montrent les coefficients de corrélation obtenus pour les méthodes énumérées lors de leur utilisation avec des vrais flux VoIP. Les valeurs les plus élevées correspondent à celles présentées dans [136], exceptées celles de l'E-model, qui n'y est pas considéré. L'auteur n'a pas explicité les conditions de réseau; il

Métrique	Corrélation avec MOS
A	0.87
B	0.85
C	0.56
D	0.86
E	0.90
F	0.86
MBSD	0.24
EMBSD	0.54

Table 1.2: Coefficients de corrélation pour quelques méthodes d'évaluation objective avec le MOS, pour des dégradations dues au codage et à quelques paramètres de réseau que l'on peut trouver sur un réseau de type GSM ou CDMA. Noter que toutes les métriques n'ont pas été nommées explicitement.

Métrique	Corrélation avec MOS
EMBSD	0.39 – 0.87
MNB1	0.61 – 0.83
MNB2	0.63 – 0.74
E-model	0.62 – 0.86

Table 1.3: Coefficients de corrélation pour EMBSD, MNB(1 & 2) et l'E-model avec MOS pour VoIP, pris de [136] et [55].

s'est limité à dire qu'il s'agissait des flux VoIP. Dans [55] les paramètres de réseau considérés sont le taux de pertes (0%, 1% et 5%, la distribution des pertes n'étant pas spécifiée) et la gigue (0, 50 et 100ms de variation pour un délai de bout en bout non spécifié). Nous croyons que les différences de performance entre les deux études sont données par des conditions de réseau un peu plus dures dans [55], ce qui peut avoir dégradé davantage les échantillons et donc affecté aussi davantage la performance des métriques.

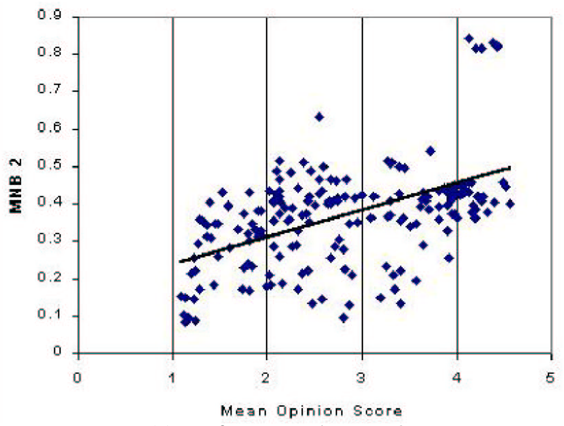
Parmi toutes les méthodes d'évaluation objective de la qualité des flux de voix que l'on a considérés, seul l'E-model est conçu pour travailler sans accéder au signal original, ce qui, avec ses faibles besoins de puissance de calcul fait de cette approche une bonne option pour des applications en temps-réel. Cette métrique a été conçue pour aider à la conception de nouveaux réseaux, et non pas comme une métrique de qualité en soi [81]. Ceci explique son manque de précision dans certaines conditions. D'ailleurs, il est encore en développement, et par exemple, c'est juste depuis Mars 2003 qu'il y a une provision explicite pour considérer les pertes de paquets comme un paramètre en entrée, et là encore, il n'y a pas de provision pour les rafales de pertes.

Comme un exemple des problèmes que l'on peut trouver aujourd'hui avec les méthodes objectives disponibles dans la littérature, nous présentons dans la Figure 1.1 deux *scatter plots*, l'un pour MNB2 et l'autre pour l'E-model. Il est facile de voir qu'il y a beaucoup de valeurs inconsistantes entre ces métriques et les évaluations subjectives (points avec le même valeur pour l'évaluation objective et des écarts très importants pour le MOS, et vice-versa).

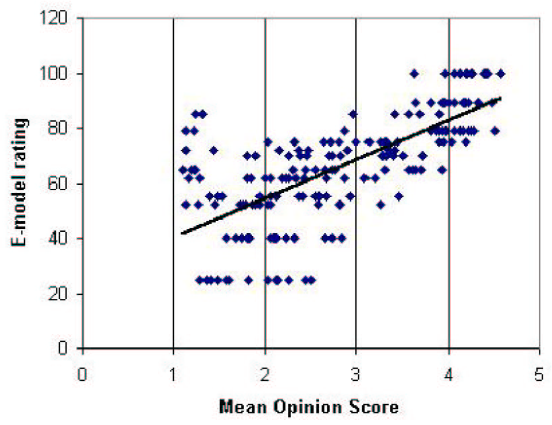
1.5.2 Performance de l'Approche Pseudo–subjective

Nous présentons ici des mesure de performance de PSQA sur des flux VoIP. Nous avons utilisé le Robust Audio Tool (RAT) [96] sur une maquette de réseau où l'on a généré des pertes selon un modèle de Gilbert [54] simplifié, qui permet d'obtenir des processus de pertes très semblables à ceux observables sur Internet [17, 135]. On a aussi utilisé un mécanisme de correction d'erreurs à l'avance (*Forward Error Correction* – FEC) et on considère ses paramètres dans notre évaluation. La Table 1.4 montre les paramètres que nous avons considéré dans notre expérience (il faut noter que PSQA permet de considérer n'importe quel ensemble de paramètres qui soient d'intérêt).

Nous avons gardé environ 115 configurations et à partir de 12 échantillons originaux on a généré 115 groupes de 4 séquences et nous avons procédé à réaliser une évaluation subjective avec un groupe de 17 sujets. Après le filtrage statistique des résultats, nous avons éliminé les résultats d'un des sujets. Avec ces résultats, nous avons entraîné plusieurs RNN, avec des différentes architectures, et avec plusieurs variations dans les tailles des ensembles d'entraînement et de validation. Les résultats obtenus ne présentent pas beaucoup de variations et on a trouvé que même avec des architectures très simples pour le RNN, on peut avoir de très bons résultats. Les coefficients de corrélation (entre la sortie du RNN et les MOS) obtenus vont de 0.73 à 0.93 (les résultats plus faibles correspondent aux tests avec une faible taille de l'ensemble d'entraînement). La Figure 1.2 montre un *scatter plot* pour un ensemble de validation et un RNN *feed forward* de 3 couches. Dans la Figure 1.3 nous montrons les MOS réels et estimés



(a) Performance de MNB2



(b) Performance de l'E-model

Figure 1.1: *Scatter plot* des résultats de MNB2 et l'E-model contre des MOS pour un ensemble de échantillons dégradés par le codage et la transmission sur le réseau (pris de [55]).

par la RNN pour le même ensemble de valeurs de validation. Nous avons constaté que l'erreur absolue la plus importante est de 0.58 points dans l'échelle de cinq points utilisée. Une différence de cette magnitude est à peine perceptible pour un utilisateur moyen.

Paramètre	**Valeurs**
Taux de perte	0%...15%
Taille moyenne des rafales de perte	1...2.5
Codec	PCM Linear 16 bits, GSM
FEC (redondance)	ON(GSM)/OFF
Décalage de la FEC	1...3
Taille des paquets	20, 40, et 80ms

Table 1.4: Paramètres de codage et de réseau utilisés pour notre expérience.

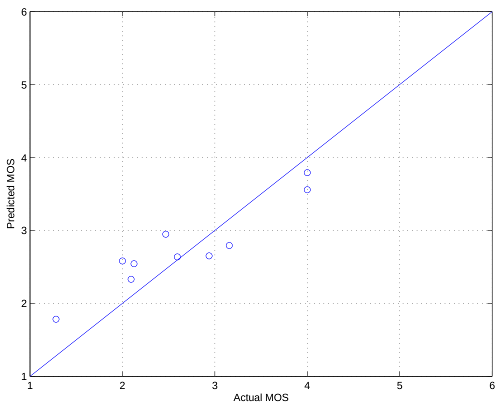


Figure 1.2: Scatter plot pour un ensemble de validation (pas connu du RNN). Coefficient de corrélation = 0.93

1.6 Qualité de la Voix sur IP

Nous avons analysé la qualité des flux VoIP sous plusieurs conditions de réseau variées. Nous nous sommes particulièrement intéressés aux flux *one–way* sur les réseaux filaires, mais nous avons également étudié ces flux sur des réseaux sans fil du type IEEE 802.11 (*wi–fi*), et nous avons aussi commencé à étudier des flux interactifs transmis sur Internet.

En utilisant PSQA, nous pouvons, à partir de simples métriques de performance du réseau, et en connaissant certains paramètres applicatifs, avoir une bonne estimation de la qualité telle que perçue par un utilisateur lambda. Nous avons donc étudié cette qualité pour de nombreuses combinaisons des paramètres qui la définissent.

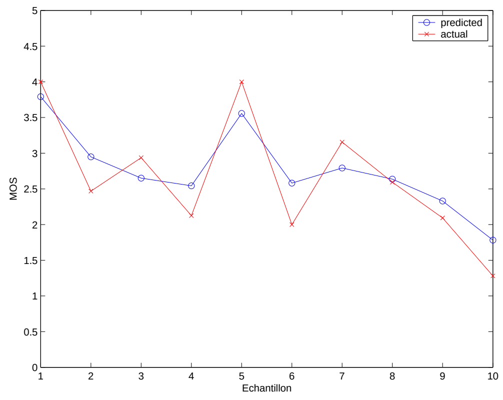


Figure 1.3: MOS réel et estimé par le RNN pour l'ensemble de validation. Erreur absolue maximale : 0.58

Les Figures 1.4 et 1.5 montrent, par exemple, l'évolution de la qualité d'un flux VoIP quand les pertes de paquets dans le réseau varient. Nous pouvons observer que l'ajout de la redondance à ce type de flux permet d'obtenir des meilleurs résultats, même sous des conditions de réseau assez sévères.

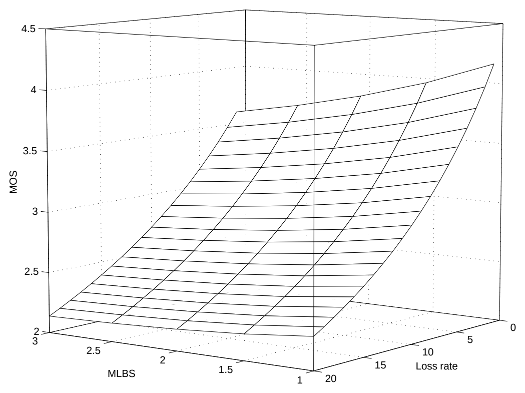


Figure 1.4: La qualité perçue en fonction du taux de perte (Loss rate) et de la taille moyenne des rafales de perte (MLBS), en utilisant de la redondance (GSM, avec décalage de 3 paquets).

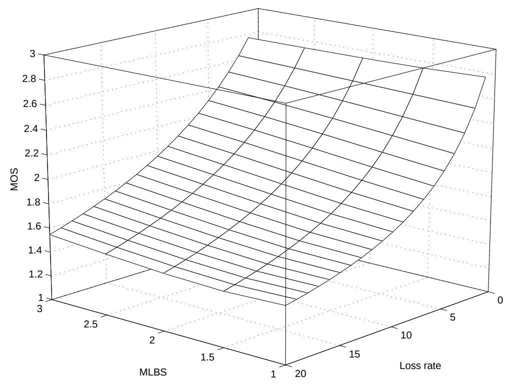


Figure 1.5: La qualité perçue en fonction du taux de perte (Loss rate) et de la taille moyenne des rafales de perte (MLBS), sans redondance.

L'étude de la qualité des flux interactifs est une affaire beaucoup plus compliquée que celle de l'étude des flux *one–way*. Ceci est dû à plusieurs raisons, la principale parmi elles étant que la qualité conversationnelle n'est pas seulement la qualité des flux audio qui circulent dans chaque sens. Il y a aussi d'autres facteurs, beaucoup plus difficiles à mesurer correctement, telle l'interactivité de la conversation ou la superposition des voix des interlocuteurs, par exemple. Ainsi, il n'y a pas, dans la littérature d'autres études aussi larges que celle que nous conduisons actuellement. Étant donné qu'au moment de la rédaction cette étude est toujours en cours, nous avons seulement des résultats préliminaires pour l'instant. Nous pouvons voir qu'il y a des comportements intéressants pour ces flux. Le plus intéressant parmi eux, s'il s'avère être correct, est que le délai joue un rôle beaucoup moins important que les pertes dans la définition de la qualité conversationnelle. Il est connu, dans la littérature qu'il existent des seuils au–delà desquels le délai devient très gênant, mais il semblerait que quand nous considérons les pertes au même temps, l'influence du délai sur la qualité est beaucoup moindre. Dans la Figure 1.6 nous pouvons apprécier que la variation parmi les courbes est vraiment très faible par rapport à la variation de la qualité par rapport aux pertes dans le réseau.

1.7 Intégration de PSQA avec des Méthodes Traditionnelles d'Évaluation de Performance

Traditionnellement, lorsque nous voulions avoir une estimation de la performance d'un réseau pendant sa phase de conception, il fallait se contenter avec des métriques relativement simples telles que le taux de pertes ou le délai que l'on pouvait estimer à l'aide, par exemple, des modèles de files d'attente. Nous proposons, dans cette thèse, d'aller un peu plus loin dans la prédiction de la performance du réseau, en permettant d'en avoir une vue de plus haut niveau, en permettant au concepteur de prédire la qualité de certaines applications (en particulier celles multimédias) sur le futur réseau.

Pour cela, nous proposons l'intégration de PSQA avec des modèles de files d'attente, ou bien des simulations. L'idée principale est que si nous pouvons dériver certaines métriques de performance du réseau, nous pouvons par la suite les utiliser pour estimer la qualité d'un flux multimédia à l'aide de PSQA. De cette façon, on peut avoir une vision plus claire de la performance du réseau du point de vue de l'utilisateur et ceci en ayant seulement le modèle, et en connaissant la capacité W et la charge ρ prévues pour le réseau, par exemple,.

Nous illustrons cette approche à l'aide d'un exemple simple dans lequel nous modélisons le réseau à l'aide d'une file $M/M/1/W$. Dans ce modèle, le taux de pertes est donné par

$$p_L = \frac{\rho^W(1-\rho)}{1-\rho^{W+1}} \tag{1.1}$$

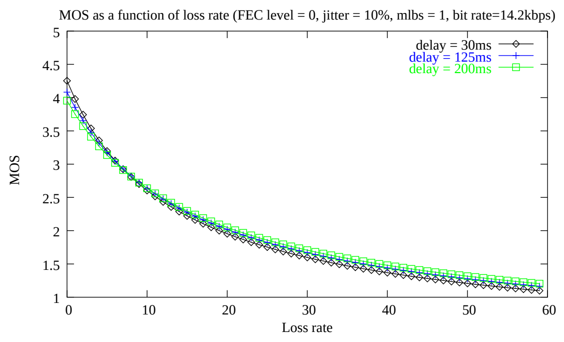
où

$$\rho = \frac{\lambda}{\mu} \neq 1.$$

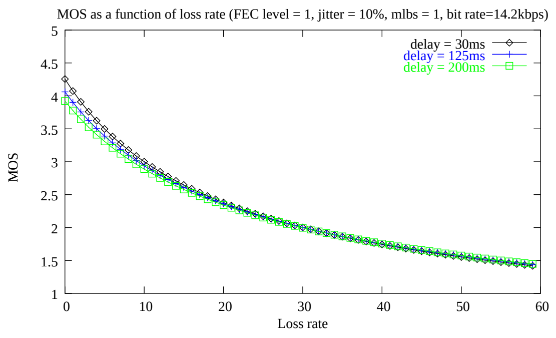
Si $\rho = 1$, alors

$$p_L = \frac{1}{W+1}.$$

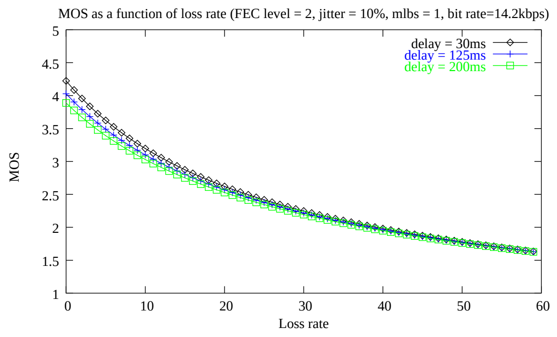
Dans ce modèle, la probabilité qu'une rafale ait une taille j, $j \geq 1$ est égal a $p^{j-1}q$, avec $p = \rho/(1+\rho)$ et $q = 1 - p = 1/(1+\rho)$. Nous pouvons par la suite déduire aussi la taille



(a)



(b)



(c)

Figure 1.6: Qualité perçue en fonction du taux de perte, pour plusieurs valeurs du délai. (a) Sans redondance. (b) Redondance de niveau 1. (c) Redondance de niveau 2.

moyenne des rafales de perte (MLBS), qui est donnée par:

$$\mathrm{MLBS} = \sum_{j \geq 1} j p^{j-1} q = \frac{1}{q} = 1 + \rho.$$

Avec ces métriques, il est possible d'estimer la qualité d'un flux *one–way*, et donc savoir, par exemple, si le réseau a la capacité nécessaire pour assurer ce type de service, ou bien s'il faut prévoir plus de capacité. La Figure 1.7 montre l'évolution de la qualité perçue en fonction de la charge du réseau, pour deux valeurs différentes de capacité.

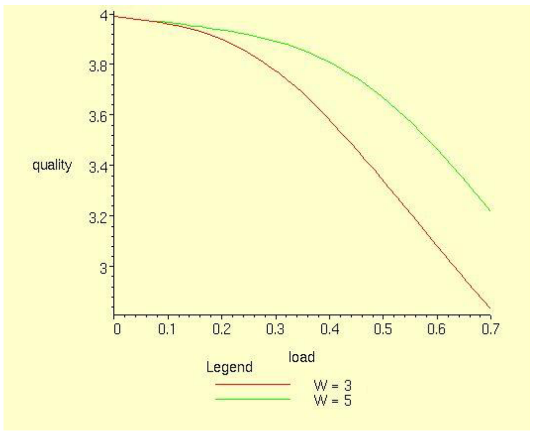


Figure 1.7: Qualité en fonction de la charge du réseau, pour deux valeurs différentes de la capacité.

Bien évidemment, cette méthode est également applicable à d'autres modèles de réseau, tant qu'ils permettent de dériver les métriques nécessaires à l'application de PSQA (taux de pertes et taille des rafales dans cet exemple, mais il pourrait être nécessaire d'avoir d'autres paramètres, telles le délai et la gigue, par exemple, pour travailler dans un contexte interactif).

1.7.1 Qualité des Flux VoIP sur un Réseau du Type IEEE 802.11

Nous avons appliqué l'intégration de PSQA avec des simulations pour étudier la performance des réseaux *wi–fi* pour le support d'applications multimédias. Ce type de réseau a été très rapidement adopté pour les réseaux à la maison, du fait de sa simplicité de mise en œuvre et d'opération. Malheureusement, les performances des réseaux sans fil restent très modestes par rapport aux réseaux filaires, ce que fait d'eux un mauvais choix pour certains types d'applications.

Nous avons développé un modèle stochastique qui reproduit le comportement d'un réseau du type 802.11, faisant spécialement attention aux variations de performance qui sont propres à ces réseaux. Nous avons par la suite utilisé ces simulations pour avoir des estimations de la qualité pour des flux VoIP et vidéo transmis sur un tel réseau. Les résultats (cf Figure 1.8 pour un exemple) montrent que ce type de réseau n'est pas encore au point pour les applications en temps-réel.

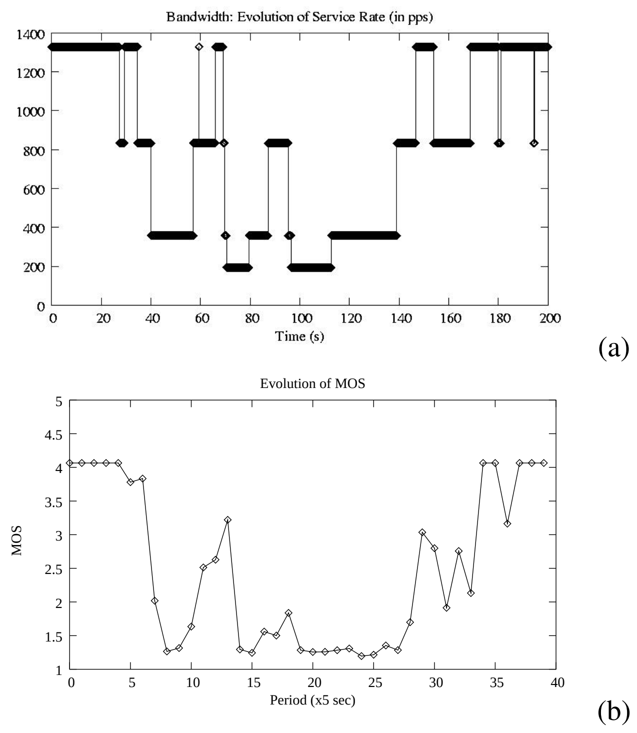


Figure 1.8: Évolution de la bande passante disponible dans le réseau sans fil (a) et de la qualité d’un flux VoIP (b) pendant une période de 200s.

1.8 Contrôle Dynamique de la Qualité des Flux Multimédias avec PSQA

Dans cette thèse, nous nous intéressons aussi aux applications de PSQA pour le contrôle de la qualité des flux multimédias. Comme mentionné dans l’introduction, le développement de mécanismes de contrôle dynamiques de la qualité a besoin de bonnes estimations de cette qualité. PSQA est capable de les fournir, et qui plus est, nous pouvons avoir cette estimation sans avoir accès au signal original, ce qui permet de travailler en temps-réel.

Nous proposons donc, d’utiliser les mesures fournis par PSQA pour contrôler, dans la mesure du possible, la qualité perçue. Pour cela, nous considérons d’abord un contexte de réseau filaire, où il n’y a aucun mécanisme de contrôle de la qualité de service, et un réseaux sans fil, sur lequel on peut contrôler les politiques de gestion de la file d’attente du point d’accès *wi–fi*.

Pour le cas filaire, nous proposons deux algorithmes qui permettent d’améliorer la qualité perçue, tout en économisant l’utilisation de ressources. Pour cela, nous définissons des seuils de qualité acceptable, et nous modifions des paramètres applicatifs tels que la redondance, la taille des paquets et le codec utilisé. Les résultats obtenus montrent que ces algorithmes permettent d'obtenir une qualité égale ou supérieure à des configurations statiques qui utilisent de la redondance, tout en faisant une meilleure utilisation de la bande passante disponible. La Figure 1.9 montre un exemple de la performance de l'un de ces algorithmes.

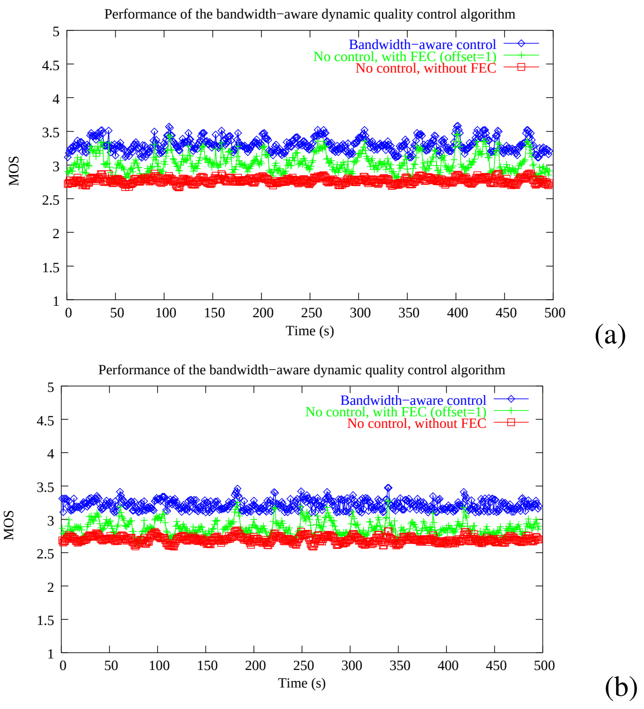
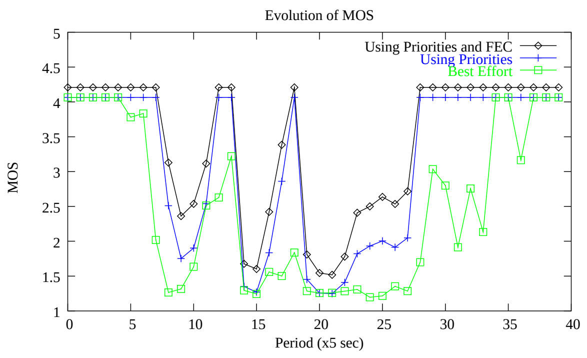
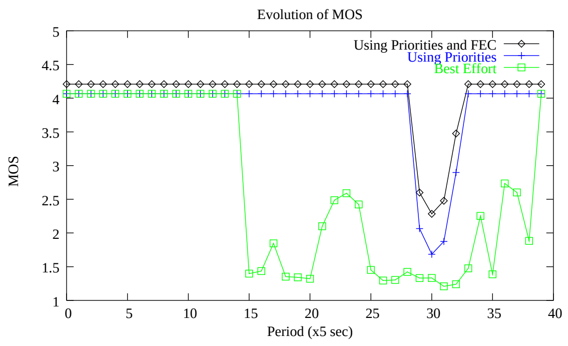


Figure 1.9: Performance d'un des algorithmes de contrôle de la qualité. Il minimise la consommation de bande passante dès que cela devient possible, tout en permettant d'avoir une qualité meilleure que celle offerte par une configuration statique avec de la redondance.

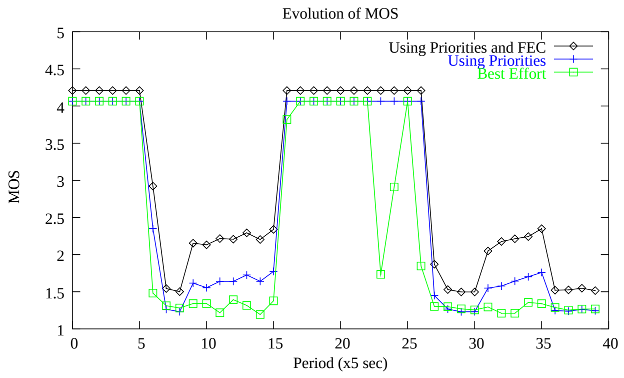
Finalement, pour le cas sans fil, nous avons proposé un mécanisme simple de gestion des files d'attente dans le point d'accès *wi–fi*, que permet de favoriser les flux d'applications temps-réel, pour minimiser leurs pertes et leurs délais. Ceci implique, en revanche, une dégradation de la performance des autres applications, et l'utilisation de ce mécanisme doit donc être réservée aux cas où elle est vraiment nécessaire. Ceci peut être déterminée en temps-réel à l'aide de PSQA. Même avec cette amélioration du traitement des flux qui nous voulons protéger, il s'avère que les réseaux *wi–fi* ne sont pas en mesure de fournir une qualité de service acceptable dans toutes les situations. La Figure 1.10 montre que l'utilisation combinée du mécanisme proposé et de la redondance permet un incrément notable de la qualité perçue pour la VoIP, même si des bons niveaux de qualité ne sont pas toujours atteints.



(a)



(b)



(c)

Figure 1.10: Évolution de la qualité de trois flux VoIP sur un réseau *wi–fi*.

1.9 Conclusions et Perspectives

Dans cette thèse, nous avons proposé plusieurs contributions dans le domaine de l'évaluation de qualité et le contrôle des flux multimédias. Nous avons construit sur la l'approche automatique d'évaluation de qualité en temps réel (que nous appelons évaluation pseudo–subjective de qualité, ou PSQA) originalement proposée par Mohamed dans [98], et nous l'avons utilisée pour mieux comprendre la qualité des flux multimédias transmis sur un réseau tel que l'Internet. Nous nous sommes particulièrement intéressés aux applications de Voix sur IP, dont la popularité a considérablement augmenté pendant les dernières années. Plusieurs opérateurs traditionnels de téléphonie commencent à fournir des services de VoIP, et un certain nombre de logiciels de téléphonie IP ont été publiés dans le monde entier et largement adoptés par des millions d'utilisateurs. Par exemple, un outil récemment lancé de téléphonie IP, Skype$^{\text{TM}}$, a été téléchargé plus de 140 millions de fois, et a une base installée de plusieurs millions d'utilisateurs.

Notre première contribution concerne la performance de PSQA, en particulier pour les flux VoIP. Les résultats obtenus montrent que, même si la technique n'est pas universelle (dans le sens où un RNN entraîné dans un contexte donné n'est pas capable d'estimer la qualité dans un autre contexte complètement différent), elle reste capable de fournir des estimations correctes (ou facilement adaptable) même quand le contexte d'utilisation est très différent à celui d'entraînement. Nos résultats montrent que la performance de PSQA, en termes de corrélation avec des évaluation subjectives, est meilleure que celle de la plupart des méthodes objectives, et comparable à celle de la meilleure méthode objective que l'on trouve aujourd'hui dans la littérature. De plus, PSQA peut fournir ces estimations en temps réel, ce qui n'est pas possible avec les autres techniques.

La deuxième contribution de cette dissertation concerne l'analyse de la qualité des flux VoIP en fonction de plusieurs paramètres que la définissent. Nous avons utilisé PSQA pour décrire l'effet de 6 paramètres différents et leurs interactions sur la VoIP *one–way*. Nous présentons aussi des résultats préliminaires pour la qualité des flux VoIP interactifs, qui présentent une complexité beaucoup plus importante.

Notre troisième contribution consiste en une méthode pour l'intégration de PSQA avec des techniques classiques d'évaluation de performance. Nous illustrons cette méthode avec un exemple basé sur une file $M/M/1/W$, et nous utilisons les résultats obtenus pour étudier l'effet d'un des mécanismes de redondance les plus répandus pour la VoIP sur la qualité perçue. Les résultats obtenus montrent que, contrairement à certaines prédictions dans la littérature, l'utilisation de ce type de redondance est toujours bénéfique pour la qualité. Nous avons aussi utilisé cette technique d'intégration de PSQA et autres méthodes d'évaluation de performance pour étudier la qualité de la voix sur des réseaux sans fil du type *wi–fi*, et nous avons montré les limitations de cette technologie pour les applications en temps réel.

Finalement, notre quatrième contribution se situe dans le domaine du contrôle dynamique de la qualité des flux multimédias. Pour cela, nous nous servons des estimations de qualité fournies par PSQA en temps-réel. Nous proposons deux algorithmes simples pour améliorer la qualité de la voix sur IP sur des réseaux filaires. Le premier algorithme est naïf, et tente d'améliorer la qualité par tous les moyens dont il dispose. Le deuxième vise à obtenir des bons niveaux de qualité, tout en minimisant la consommation des ressources du réseau. Nous montrons que ces algorithmes permettent d'obtenir une qualité meilleure que celle obtenue avec des configurations statiques, avec une meilleure utilisation du réseau. Dans le domaine du sans fil, nous avons proposé un mécanisme de marquage des paquets pour donner la priorité aux flux multimédias. Nous montrons que, combiné avec la redondance, ce modèle de marquage permet d'obtenir des meilleurs niveaux de qualité même dans quand les conditions de réseaux son mauvaises. Ici, PSQA permet de déterminer le moment auquel l'utilisation du marquage devient nécessaire, de façon à ne pas gêner le bon fonctionnement d'autres applications qui tournent sur le réseau si ce n'est pas strictement nécessaire.

Les perspectives de ce travail sont nombreuses. Tout d'abord, nous sommes très intéressés par une étude plus approfondie de la qualité des flux interactifs, qui est un domaine relativement peu exploré, et fort intéressant. Un autre point qui mérite d'être étudié avec plus de détail est la performance des réseaux sans fil comme support pour applications multimédias. Nous sommes, a l'heure actuelle en train de travailler sur une simulation plus détaillée (sur NS–2) des réseaux IEEE 802.11, avec les extensions pour la qualité de service proposées dans le standard 802.11e. Il nous intéresse de pouvoir combiner ces mécanismes à d'autres (par exemple celui proposé dans cette thèse), pour essayer d'obtenir des niveaux de qualité acceptables tout le temps. Finalement, il serait très utile de continuer les recherches entamés par Mohamed dans le domaine de la qualité de vidéo, en étudiant la qualité des flux de haute définition. De plus, il reste du travail à faire dans les domaines de l'audio de haute qualité, et des flux mixtes audio / vidéo.

Chapter 2

Introduction

2.1 Motivation

In recent years, the growth of the Internet has spawned a whole new generation of networked applications, such as voice over IP (VoIP) [3, 32, 59], videoconferencing, video on demand [130], music streaming, etc. which have very specific, and stringent, requirements in terms of network Quality of Service (QoS). The current Internet infrastructure was not designed with these kinds of applications in mind, so multimedia applications' quality is very dependent on the capacity, load and topology of the networks involved, as QoS provisioning mechanisms are not widely deployed. Normally, all Internet traffic is considered equal, and the network provides a *Best Effort* service, with no guarantees of any kind.

As of this writing, several commercial offerings exist for the distribution of multimedia services over broadband links (for instance, digital TV over DSL lines, or VoIP services provided very cheaply with broadband subscriptions), and even on mobile networks (e.g. video services on 3G mobile phones). In order to be viable, these services must provide quality levels comparable to those of traditional phone and TV services. Although some concessions can be made, it is unlikely that users will use, let alone pay, for sub–standard quality services. Therefore, it becomes necessary to develop mechanisms which allow to overcome the technical deficiencies presented by current networks when dealing with real–time applications.

Real–time applications are very sensitive to packet losses, end–to–end delay and jitter. Multimedia applications, in particular, are also sensitive to the encoding used, which generally degrades their quality. Audio and video streams tend to be naturally redundant, and in order to decrease bandwidth consumption, the encoding process takes away some of this redundancy. This in turn, makes the streams more vulnerable to packet loss or transmission errors. While it is easy to see that these (and other) factors affect the perceived quality of a multimedia stream, it is not so easy to know just *how* they affect it, i.e. what is the impact of each factor on the perceived quality.

In order to develop the mechanisms mentioned above for controlling the perceived quality of multimedia networked applications, it is important to be able to assess this perceived quality, so as to know which adjustments need to be done, and when to do them. It is not enough to have a simple qualitative assessment; control applications need to have a fine–grained, quantitative vision of the perceived quality. Moreover, it is clearly very useful to be able to perform this assessment in real–time, so as to have the capacity to react to rapidly–changing network conditions. If the influence of the different quality–affecting factors on the perceived quality can be known, it can help optimize said quality. Once the quality is determined, some form of control may be implemented. The extent of this control will depend on the application itself, and on the network it runs on. For the most general case (i.e. best–effort service on the Internet), there's not much to be done at the network level, so application–level quality improvements must be sought. If the network does implement some form of QoS provisioning, it may also be used to improve the perceived quality.

Besides control applications, the commercial viability of real–time multimedia applications relies on the existence of Service Level Agreements (SLA) and pricing schemes which allow the user to have certain guarantees with respect to the service he receives, and the network operator (and content provider, if applicable) to optimize their profits. For both SLAs and pricing, quality assessment is very important. From a user's perspective, it allows him to verify that he is getting the service quality for which he is paying. This can be done by implementing a quality assessment module at the user's site. From the service (or content) provider's point of view, being able to assess the perceived quality at several points in the network allows him to ensure he is complying with his SLAs, and to detect and troubleshoot technical problems that may arise. Furthermore, it is useful for optimizing their pricing schemes, since a user's valuation of a service will be closely related to the service's quality. Currently, this valuation is usually estimated by means of somewhat arbitrary *utility functions* in most pricing schemes. A good quality estimation could make those functions more accurate, hopefully resulting in higher profits for the provider, and better service for the user.

2.2 Contributions

The contributions presented in this dissertation aim to provide a better understanding of networked multimedia applications' quality (with a special focus in voice over IP) and to enable the development of real–time control of the quality as perceived by the end–user. We develop on a quality assessment method proposed by Samir Mohamed [98], which is able to provide accurate assessments in real–time, and is thus a good option for control applications. This technique, which we call *Pseudo-Subjective Quality Assessment* (PSQA), allows to obtain accurate estimations of the perceived quality of a multimedia stream in real–time. These estimations can be used to better understand how different factors affect the perceived quality, and therefore to implement control mechanisms which allow to improve it.

Our contributions can be summarized into the following points:

- In–depth study and performance analysis of PSQA.

- Analysis of VoIP quality (both one–way and interactive), and its defining factors under various networking contexts (e.g. wired and wireless networks).
- Integration of PSQA with traditional performance evaluation techniques.
- Quality–driven, dynamic control of multimedia applications.

Let us briefly describe here the main points of each of these contributions.

2.2.1 In–depth Study and Performance Analysis of PSQA

Our first contribution is a complete study of PSQA, particularly in the context of voice quality assessment. Previous work by Mohamed in this area had left certain aspects aside, so we provide a detailed analysis of the technique and its performance.

PSQA allows, via a statistical estimator (such as a neural network), to mimic the way an average human subject assesses the quality of a media stream. Unlike other assessment techniques, which sometimes rely on models of human perception to perform the assessment, PSQA builds a mapping between certain factors which affect quality, and the quality as perceived by human subjects. In this way, it is able to *learn* the relation between those factors and the perceived quality.

We study the performance of the technique when implemented with Random Neural Networks (RNN) [42], which allow us to obtain very accurate estimations of the perceived quality for a media stream. Our study covers several RNN architectures, the relationship between the number of samples used to train the tool, and its performance in terms of correlation with human–produced scores, and its generality and robustness.

We also compare the performance of PSQA to that of other quality assessment techniques found in the literature. We show that PSQA is on par with the best among them, and presents additional advantages, such as being able to provide real–time assessments, and to give insight into the relation between each parameter and the quality itself.

These results can be found in Chapter 5 of this document.

2.2.2 Analysis of VoIP Quality

A direct result of the way in which PSQA works is that it makes it possible to observe and analyze the quality of a multimedia stream under a wide range of conditions. As mentioned before, we focus our studies mainly on VoIP quality, but the same kind of analysis can be performed on video or mixed streams.

The second contribution of this dissertation is the study of the perceived quality of VoIP flows, and how it is affected by the applicative and network factors that define it. We analyze the way in which each of these factors impacts the perceived quality, under different networking contexts.

We consider one–way VoIP flows over wired and wireless IP networks, in a wide range of conditions. We also present preliminary results for interactive voice streams, for which no other studies of a similar scope exist, to the best of our knowledge. The insight gained by this kind of assessment is useful for example for network dimensioning and design, and crucial for developing effective dynamic quality control mechanisms for multimedia applications.

The main results in this area are presented in Chapters 6 and 8.

2.2.3 Integration of PSQA with Traditional Performance Evaluation Techniques

The third main contribution of this dissertation is a method for integrating PSQA with other performance evaluation techniques, such as queuing analysis or simulation. The rationale for this is that it would be useful, when designing or dimensioning the network, to be able to predict what the quality will be for certain applications (in particular, multimedia ones).

We show how one can, by deriving performance metrics from a network model, obtain quality estimations for media streams. We illustrate this with a simple case study, using a basic queuing model to represent the network, or its bottleneck, and provide a closed–form expression for the quality in terms of network load and capacity. We then extend this case study to a multi–class model, in order to estimate and understand the effects of the addition of redundancy to a voice stream on both the network state and the perceived quality.

The integration or *coupling* mechanism is described in Chapter 7, and used in Chapter 8, where we propose a stochastic model for a wireless network, and study the quality of voice streams on it.

2.2.4 Quality–driven Dynamic Control of Multimedia Applications

Our final contribution is on the subject of dynamic control of a multimedia application. We aim to provide mechanisms for controlling the perceived quality of a stream in real–time, based on the current quality. There are several control mechanisms proposed in the literature for different kinds of multimedia applications. They are generally based on the measurement of one or two network parameters, typically packet loss rate and delay, and the knowledge that an increase of any of those two is detrimental to quality. However, none of them take the actual perceived quality into account.

What we propose is the use of PSQA to monitor the perceived quality, so as to be able to react when it reaches certain thresholds. This allows us to maintain acceptable quality levels whenever possible, but also to try and reduce resource consumption when the network conditions allow it, so as to share those resources more fairly among competing applications.

We propose two proof–of–concept quality–driven control algorithms for voice applications, and compare the quality obtained when they are used to that obtained with static configurations, under different network conditions. The proposed algorithms are designed for one–way streams, but we show that they can be easily adapted for interactive streams.

Finally, we also study quality–enhancing mechanisms for VoIP over wireless IP networks, and discuss how they could be used for dynamic control purposes. We propose a priority scheme for packet treatment on the wireless link, and show how it would affect the perceived quality for both voice and video streams, and its impact on non–real–time traffic.

2.3 Document Outline

The rest of this document is laid out as follows. Chapter 3 introduces basic concepts concerning quality of service in IP networks, and in particular the factors which affect multimedia applications. We also discuss application–level quality factors, for both audio and video applications. Quality–enhancing mechanisms, both application and network–level are also covered in this chapter.

Chapter 4 summarizes the state of the art in quality assessment for multimedia applications. We discuss both subjective and objective assessment techniques, their characteristics, and their shortcomings. PSQA is introduced in Chapter 5, along with the analysis of is performance, and the comparison with other techniques. The RNN model which we use to implement PSQA is also described in this chapter.

Chapter 6 provides an analysis of the perceived quality of voice applications done with PSQA. It covers both one–way and interactive streams (although results for the latter are still preliminary). We provide a detailed study of how different factors, such as network packet loss, coding mechanisms, other parameters affect the quality as perceived by the end–user.

In Chapter 7 we discuss how to integrate PSQA with other performance evaluation techniques, and we provide two case studies for VoIP. The technique proposed in Chapter 7 is applied in Chapter 8 to study the quality of VoIP in a wireless network context. A stochastic model for a IEEE 802.11b network is developed, and based on the performance metrics derived from it, we perform our study of voice quality.

Chapter 9 deals with control mechanisms for multimedia applications. We present and analyze two quality–driven control algorithms for voice applications. In this chapter, we also discuss what would be needed to control the quality of multimedia applications in the networking context described in Chapter 8, and propose and analyze a priority scheme for treating packets in the wireless link.

Finally, we present the conclusions of our work in Chapter 10.

Chapter 3

Quality of Service and Perceived Quality: Basic Concepts

In this chapter we discuss different factors related to the quality of a multimedia stream transported over a packet network. We start by presenting the network protocols used for multimedia transmission and control. We then describe different network quality of service aspects that have an impact on the perceived quality and some of the mechanisms available to improve it at the network level. Finally, we proceed in an analogous fashion for application–level factors and quality improvement mechanisms.

There are several factors that will affect the perceived quality of a media stream running over a packet network. The way in which each of these factors actually modifies the quality varies according to the type of application (for instance, is it an interactive application, or is it a one–way stream?), the target audience (e.g. is it high quality streaming audio for audiophiles, or is it a low–cost VoIP application for long distance calls?), etc.

We can easily identify three different kinds of quality–affecting factors, namely:

- environmental factors,
- network related factors,
- application related factors.

Environmental factors range from the ambient noise or lighting of the place where the user is, to the quality of the hardware used, etc. We will not concern ourselves with these parameters, since it is generally difficult to measure them (that is, without the proper equipment, which is usually not found everywhere), and there's little we can do to counter any impairment caused by them in the normal usage scenarios (i.e. outside a lab). Indeed, from the point of view of a quality control module (be it network or application–level), it is normally impossible to modify the ambient noise level, or the lighting of the room.

That leaves us with the network and application–level factors, which we discuss in the following sections.

3.1 Protocols for Real–Time Networked Multimedia

In this section we provide short overviews for some of the most widely used protocols for multimedia applications. While these protocols are not directly related to the quality of an stream, they do often play an important role in the structure of a media application. Moreover, they sometimes provide mechanisms for feedback and control channels, which can be used for control. Of the three protocols we discuss in this section, the first one (the Real–time Transport Protocol) is a transport–layer protocol, while the other two (H.323 and SIP) provide basically application–layer signaling capabilities.

3.1.1 Real–time Transport Protocol

The Real–time Transport Protocol (RTP) [67] provides support for real–time (especially multimedia) networked applications at the transport–layer level. Among the services provided by RTP, there are sequence numbering, payload identification, and time-stamping of packets.

RTP works in a different fashion from other protocols, in that it is based on application–level framing. This means that each application needs to “complete” RTP to fit its own needs, and this for each media type it handles. Normally, applications using RTP implement libraries which take care of creating the RTP packets. These are generally different for each application / media type. Thus, each application needs to “extend” RTP for its own use, by defining:

a profile, which defines codes and map them to payload formats, and may also define extensions or modifications to the RTP headers, and

a payload format specification, which defines how the actual data is transported over RTP.

Although RTP flows can be run on top of several protocols, the most widely used stack for multimedia applications is RTP/UDP/IP, with RTP and UDP providing the transport layer. The use of UDP allows RTP flows to be multiplexed, and applications can use the UDP checksum to verify payload integrity. RTP/TCP is not as widely used, since TCP tends to provide low performance for multimedia applications, as mentioned in Section 3.2. However, this combination may be useful in the presence of firewalls or Network Address Translation (NAT) devices that may hinder UDP flows, since UDP is a stateless protocol.

An important part of RTP is its associated control protocol, RTCP, which allows to provide feedback on the transmission quality to the participating nodes, and to a lesser extent, to allow the transmission of session control information. However, RTCP might be insufficient to perform dynamic control of the perceived quality, due to two limitations in its design:

- The rate at which RTCP reports can be sent is limited to one every five seconds, since RTP is designed to be very scalable, and usable in multicast environments. Thus, if every participating node sent feedback to all the others at a high rate, congestion would ensue. However, sudden changes in network conditions may require a higher rate of reports in order to improve the quality than the one that RTCP supports.
- The control features in RTCP are limited (they aim mainly at session control issues, such as managing the active members in a conference, for example), and therefore might not be sufficient to perform the operations needed to dynamically control the quality.

3.1.2 The H.323 Stack

H.323 [84] is an ITU standard which provides a framework for multi–point multimedia transmission (voice, video, digital white-boards, etc.) over an unreliable packet network. The H.323 recommendation is accompanied by several others which describe the different components of the proposed framework. The main components of an H.323 system are the following:

Terminals, such as telephones, video–phones, soft–phones, etc.

Multi–point Controller Units, which manage multi–point conferences. They are composed of a Multi–point Controller (MC) which handles signaling, and optionally one or more Multi–point Processors (MP), which handle media processing.

Gateways, which provide interfacing with other networks (be it H323 or not). They are composed of a Media Gateway Controller (MGC) and / or a Media Gateway (MG), which handle signaling and media respectively.

Gatekeeper, an optional component which handles admission control, address resolution, and other services such as forward–on–busy, etc.

Border Elements, which generally work with a Gatekeeper in order to provide address resolution and call authorization between different networks or domains.

The transport layer used for H.323 channels is chosen according to the channel's function. Media is generally streamed over RTP / UDP on unreliable networks such as the Internet. Signaling and control data channels normally run on top of TCP, in order to maintain integrity.

H.323 defines standard codecs for audio and video, and ensures compatibility between endpoints. Audio terminals must implement at least the ITU G.711 [83] standard for voice compression, and the G.723 codec is widely used due to its lower bandwidth consumption. Video terminals must provide H.261 support, and they may also support H.263. Signaling support for the new H.264 video codec has recently been added to the stack. Furthermore, the terminals can communicate their capabilities during the connection, so that the best possible configuration can be used.

The H.323 stack is independent of the network architecture used, and of the platforms on which the terminals run. Multi–party (many–to–many) communications are possible thanks to the MCUs, and multicasting is also possible.

Several implementations of H.323 (be it hardware, in the form of video-phones for instance, or in software, such as Mircosoft NetMeeting, or Open-H.323) exist, and are widely deployed.

3.1.3 Session Initiation Protocol

The Session Initiation Protocol (SIP) [77] provides signaling and control for multimedia sessions. It works at the application level, and it is not integrated with media–transport protocols, but it was designed to be used as a component in multimedia communication systems. It is often said that SIP is the IETF's response to ITU's H.323. It is a simpler protocol than those provided by the H.323 stack, and it is text–based instead of binary, as H.323.

Unlike H.323, SIP moves control over the sessions to the end–points. The protocol itself its simple and it is intended to ease the development and deployment of multimedia communication systems. SIP messages closely resemble HTTP messages. Many request and error codes are taken from the HTTP 1.1 specification.

SIP provides primitives to initiate, modify and terminate a multimedia session. Other services such as inviting someone to a session, and joining multicast sessions are also available. According to the RFC which defines SIP [77], it provides means for:

User location, which consists of determining the terminals to be used for communication.

User availability, which consists of determining if the called party is willing to establish communication.

User capabilities, which is the determination of the media parameters to be used.

Session startup, which is the establishment of the session parameters at the end–systems (this is called *ringing*).

Session management, which manages call termination, call transfers, and modification of session parameters.

3.2 Network QoS

The current Internet was not designed with multimedia, or more broadly, real–time applications in mind. The services it offers are unreliable, and provide absolutely no guarantees in terms of quality of service. This means that packets do not necessarily arrive at their destination, and if they do, they do not necessarily arrive in an orderly and timely fashion. Packets can be lost in the network due to congestion, or to transmission errors. They get delayed in routers' queues, and each packet in a stream may experience a different delay than the previous ones. They may even take different routes to their destination, which in turn may lead to them arriving in a different order that the one they were sent in.

For data transfer applications such as FTP, these problems are usually solved by using a reliable transport layer protocol such as TCP, which ensures that all packets are delivered to the application, and that they are delivered in the right order. Real–time applications, on the other hand, cannot generally work well with the side effects of TCP. When using TCP, the loss of a packet will be detected at the transport layer, and the packet will be sent again. For general data transfer applications, this is not a problem, but for real–time ones it is, since the second copy of the packet will probably be too old to use when it arrives at its destination. Moreover, the built–in rate control mechanisms in TCP make for variations in bandwidth, which multimedia streams do not accommodate well to, either.

This is why multimedia applications need to deal with an unreliable network by themselves. Of course, the better the service provided by the networks, be it in terms of packet loss, delay, or raw bandwidth, the better the application quality will be. This is a very intuitive proposition. However, it is very difficult to understand just *how* and *how much* the network QoS affects the quality of an application as perceived by the end–user.

In the following subsections we explore the different impairments induced by best–effort networks.

3.2.1 Packet Loss

Perhaps the most noticeable problem of not having a reliable networking infrastructure is that some packets get lost. The losses may be due to several reasons, such as routing problems, transmission errors and network congestion. Of these, losses due to congestion are the most commonly found. In a best–effort network, routers normally have *drop–tail* queues, which means that when their buffers are full, new packets are discarded as they arrive until the departure of previous packets free up some space in the buffers. Other factors, such as bit–level errors or expired TTLs due to routing problems may also cause packets to be dropped.

It is important to analyze how this end–to–end loss process takes place, in order to better understand how it will affect the applications' quality, and what can be done about it. The characterization of the loss processes in the Internet has been the subject of a large body of work (see for instance [5, 13, 118, 120, 135]). Several models, ranging from the very simple (such as independent losses [13] or fixed–size loss bursts [58, 99]) to complex ones (k^{th} order Markov chains [135]) have been proposed. One of the most widely used is a simplified version of the Gilbert model [54], which we describe next.

In the Gilbert model, the channel has two states, one in which the transmission is perfect, and another one in which error occurs with a probability d. Figure 3.1 shows the Markov chain that drives this model. The simplified model (which is widely used in the literature, cf [3, 14, 118, 135]), does away with the loss probability d, losing instead every packet when the chain is in state 1 (as seen in Figure 3.2). This model still allows to capture the burstiness of the loss process quite accurately. We have used this model against the traces provided by the authors of [135], and it was able to accurately mimic the loss behavior of the network. We compared the loss rate, mean size of the loss bursts, and the standard deviation of the mean loss burst size of the original traces to those of the traces generated by the model. The values measured for the real traces and for the generated ones were remarkably close. The traces used are for audio traffic, on both national and international links, and in unicast and multicast scenarios.

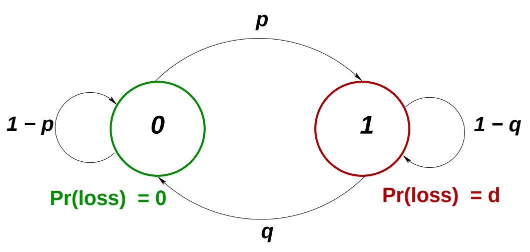


Figure 3.1: The Gilbert model. When in state 0, the transmission is error–free. In state 1, errors occur with probability d.

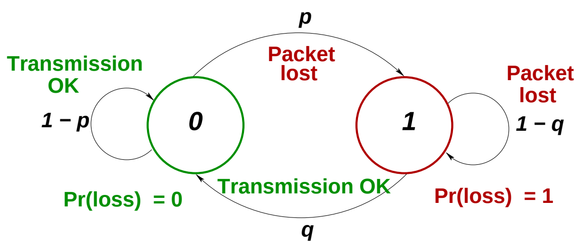


Figure 3.2: The simplified Gilbert model. When in state 0, the transmission is error–free. In state 1, all packets are lost. Note that the transition from state 0 to state 1 implies a loss, and that in the opposite direction, the packet is not lost.

Each transmitted packet either arrives successfully at its destination or it doesn't (either because some of its bits are corrupted or, more likely, because it is dropped by a router, etc.) Let the successful arrival be coded by '0' and the loss event by '1'. Let us denote by X_n the result of the transmission of packet n, for $n \geq 1$. The sequence $X = (X_n)_{n \geq 1}$ is then considered to be a Markovian homogeneous stochastic process, with values in $\{0, 1\}$, and we denote

$$\Pr(X_{n+1} = 1 \,|\, X_n = 0) = p,$$
$$\Pr(X_{n+1} = 0 \,|\, X_n = 1) = q.$$

Here, p is the probability of a packet being lost when the previous transmission was correct, and $1 - q$ is the probability of losing a packet when the previous transmission was not correct.

Let LBS be the size of a *burst* of losses, that is, a (maximal) sequence of consecutive losses not strictly included in another such sequence. Since the holding times of X are geometrically distributed, we have

$$\Pr(\text{LBS} = k) = (1 - q)^{k-1} q, \qquad k \geq 1.$$

If MLBS is the mean size of a burst of losses, we then have $\text{MLBS} = q^{-1}$.

Now, if PER is the Packet Error Rate, that is, if PER is the (unconditional) probability of losing a packet (in equilibrium),

$$PER = \pi_0 p + \pi_1 (1 - q),$$

where (π_0, π_1) is the stationary distribution of X. We have $\pi_0 = q(p + q)^{-1}$ and $\pi_1 = p(p + q)^{-1}$. We then have

$$\begin{aligned} PER &= \frac{q}{p+q} p + \frac{p}{p+q}(1 - q) \\ &= \frac{p}{p+q} \\ &= \pi_1 \quad \text{as expected.} \end{aligned}$$

Since we must set $q = \text{MLBS}^{-1}$, we deduce

$$p = q \frac{PER}{1 - PER} = \frac{1}{\text{MLBS}} \frac{PER}{1 - PER}$$

We can now calibrate the model, provided that we have a long enough trace. The PER can then be measured together with the average of the bursts of losses, and we denote their respective values PER_m and MLBS_m. We have then

$$p = \frac{1}{\text{MLBS}_m} \frac{PER_m}{1 - PER_m}, \qquad q = \frac{1}{\text{MLBS}_m}.$$

See that if there are losses (at least one) and if not every transmission is a loss, then necessarily $\text{MLBS}_m > 1$ and $0 < PER_m < 1$, leading to $0 < p, q < 1$.

3.2.2 Delay

End–to–end delay plays an important role on the perceived quality of multimedia applications, especially for those that are interactive, since large delay values induce a loss of interactivity. In the case of real–time communications, from the application point of view, the end–to–end

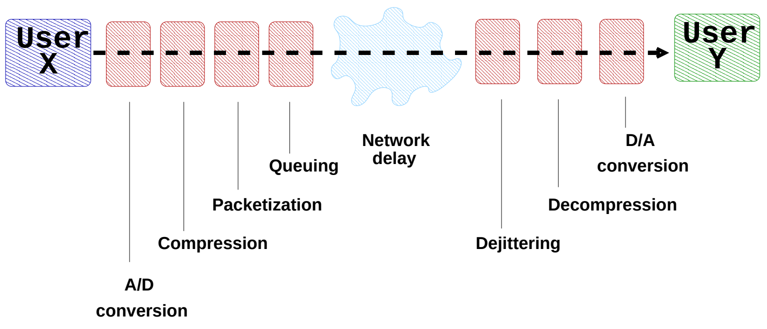


Figure 3.3: The end–to–end (*mouth–to–ear*) delay has several components. The communication is delayed due to Analog/Digital conversion, compression and decompression of the signal, packetization, queuing at the network interface, propagation and queuing times in the network, and the time needed for the dejittering of the stream at the receiver.

delay has several components as shown in Figure 3.3.

Several studies [82, 91] have been carried out to determine the impact of delay on two–way communications, in particular for telephony applications. It has been stated that there are some delay thresholds that define the quality of the conversation in terms of interactivity. Table 3.1 shows the influence of one–way delay on conversational quality, according to the ITU [82]. Of course, the effects of delay on quality largely depend on the type of application and the expectations of the user.

Delay Values (ms)	Conversation Quality
< 150	Acceptable
150–400	Noticeable degradation of interactivity
> 400	Unacceptable for general use

Table 3.1: Influence of one–way delay on interactive communications according to ITU G.114.

3.2.3 Jitter

The time a packet takes to go from one host to another one over the Internet is generally not constant. This variation of the delay is called jitter. Unlike delay, jitter affects both interactive and one–way streams in a noticeable way. Media streams are usually consumed at a steady rate by the receiver. If packets don't arrive at a constant rate, they may arrive too late to be played, and from the application point of view, they are considered as lost.

This problem is usually solved by using a *dejittering buffer*, which allows the client to absorb delay variations without losing packets. However, packets sitting in the dejittering buffer waiting to be played imply an increase of the end–to–end delay, which in some cases (especially in interactive applications) can lead to further degradation of the perceived quality. Therefore, in the presence of jitter, there is usually a trade–off to be made between "lost" packets (those that arrive too late to be played), and delay. Much research (i.e. [20, 41, 115, 122]) has been conducted in this area in order to provide an optimal buffer size that maximizes the stream's quality, or to assess the impact of these parameters (and their interaction with other parameters, such as FEC) on the perceived quality.

3.2.4 Packet De-sequencing

The "fire and forget" nature of the Internet makes each IP packet in a stream independent from the others. As such, it could be routed through a different path. While this kind of situation does not happen very often (consecutive packets going from one host to another normally take the same route), applications that do not use a reliable transport layer should know how to deal with it.

In the case of multimedia streams, the norm is to have some kind of sequencing mechanism defined for the stream (for instance, RTP's sequence number). In this way, if a packet arrives before its play-out time, it can be played whether it arrived in the right position in the stream or not. Some application–level protection schemes such as interleaving or media–dependent forward error correction (c.f. Section 3.3.2) may be affected by out–of–sequence packets, since they usually have tighter time constraints at the play-out buffer.

3.2.5 Improving Quality at the Network Level

3.2.5.1 Integrated Services (IntServ)

The IETF's Integrated Services architecture [68, 69] was designed to provide support for real–time traffic in addition to the best–effort service currently found in the Internet. It is a fine–grained system, and it was conceived with multicast applications in mind.

The main idea behind IntServ is that network resources (such as bandwidth) must be explicitly managed. In order to manage these resources, two basic principles were prescribed:

Resource reservation, which creates and maintains a per–flow state in each of the routers along each flow's path. The application defines the desired QoS parameters for its flow via a *flowspec*, which is propagated by the reservation protocol, and used for admission control and parametrization of the packet scheduler.

Admission control, which helps decide if a new flow can make a reservation without compromising the quality of previously reserved flows. Admission control is performed at each node along a path for which QoS guarantees are needed.

Although more than one resource reservation protocols may be used with IntServ, the RFC suggests the use of the Resource Reservation Protocol (RSVP) [70]. RSVP is a signaling protocol that deals with one–way flows, so in the case of an interactive application, at least two reservations need to be made. The receiving hosts are responsible for defining the QoS parameters they need, and RSVP then carries these parameters, in the form of a flowspec, along the path that the flow will take (this is not necessarily the end–to–end path if the application is multicast, in which case only the path to the node which joins the distribution tree is considered).

Given that all the resource reservation information is stored for each flow in each router along its path, it is easy to see that this approach does not scale well. This is the main reason why it is not usually deployed.

3.2.5.2 Differentiated Services (DiffServ)

The Differentiated Services architecture [71] aims to improve the quality of streaming media over the Internet. The DiffServ architecture provides a scalable way to improve the QoS of aggregated flows at the IP level. Packets are forwarded following different policies according to the value of their DiffServ Code Point (DSCP) field. The differentiated treatment is done at a per–hop level, unlike the IntServ approach mentioned above, which requires complex signaling protocols and reservation of resources.

Packets which go through a *DiffServ cloud* are marked when they enter it (at the *ingress router*). Once inside the cloud, they are treated at each router according to their class.

Several modes of operation (called Per–Hop–Behavior, or PHB) have been defined, among which the most common are:

Default Forwarding (DE) [72] simply provides a best–effort service.

Assured Forwarding (AF) [74] provides four different classes of flows. Packets in each class can be assigned three different drop probabilities, which allows users to protect certain packets or flows more than others against packet losses. For example, an intra–flow priority scheme, such as the one described for video in [105], or for VoIP in [119]. These schemes protect the most important packets in a flow against losses, providing a marked quality improvement without needing to mark the entire flow as being of the highest priority. The AF class should be used when an application requires a high probability of packets being forwarded. This is indeed the case as long as the aggregate traffic from each site does not exceed the subscribed information rate (profile). This class also caters to a situation where a site may exceed the subscribed profile with the understanding that the excess traffic is not delivered with as high a probability as the traffic that is within the profile.

Expedited Forwarding (EF) [76] was originally designed in a way such that high–priority packets would preempt low priority ones, and dropping packets of aggregates that exceeded a certain maximum rate. In its current incarnation, it is intended to provide low delay, low jitter and low loss service. It does this by ensuring that EF packets are served at a given rate, according to a formal (mathematical) definition of the desired behavior. In practice, EF flows will encounter short or empty queues, since one of the principles of this PHB is to minimize queuing time. The EF class is very well adapted to interactive conferencing applications, since they need small end–to–end delays to function properly.

EF is conceived to behave in a similar way as a leased line, which is a more strict approach than the one taken by AF. The AF service provides a more relaxed approach based on a set of basic QoS mechanisms. Routers implementing AF have several queues (one for each class), and bandwidth is distributed among them by a scheduler. Furthermore, each queue provides Active Queue Management (AQM), which enforces the priorities within each class. The most widely known AQM scheme is called Random Early Detection [38], which detects incipient congestion by analyzing the average queue size, and allows to control congestion by randomly dropping packets as they arrive at the router, or using Explicit Congestion Notification (ECN) [73]. ECN works by marking the packets to notify the receiver that congestion is imminent. The reasoning behind this is that the rate adaptation algorithm at the sender / receiver pairs will be triggered by the drop (or more explicitly by the mark in the packet, if ECN is used).

3.2.5.3 Alternative Best Effort Service (ABE)

The Alternative Best Effort Service [63–65] (formerly known as Asymmetric Best Effort Service) is a new way of providing low, bounded queuing delays for applications that have timing constraints. It considers two classes of packets, namely blue and green packets, which have different requirements from the network. The idea is to give applications the possibility of a trade–off between throughput and delay. Green packets arriving at a router are treated within a bounded time; however, in order not to penalize blue traffic, green packets receive less throughput when congestion arises.

Being a best–effort service, ABE will give little throughput to all flows when the network is very loaded. When the load is moderate, on the other hand, it allows some flows to improve the end–to–end delay at the expense of bandwidth. This is useful, for example, for VoIP applications, which do not require much bandwidth, but have severe timing constraints.

An interesting aspect of ABE is that, while it was designed with TCP–friendly multimedia applications in mind, it copes effectively with non–responsive applications as well. What this means is that if a streaming application using green packets tries to increase its rate as a result of a lower end–to–end delay[1], the blue flows' throughput will not be degraded more than they are in a plain best–effort scenario. By design, ABE strives to ensure that:

1. The delay for blue packets in an ABE scenario is not longer than in a plain best–effort one.
2. A blue packet that would not be dropped in a plain best–effort scenario is not dropped when ABE is used.
3. Assuming that all sources use a rate adaptation algorithm which responds to a loss–throughput formula, a stream composed entirely of green packets should receive less or at most equal throughput than if it were blue.

The first two requirements define what is called *local transparency to blue*, which means that blue packets do not notice green ones. The third one provides what is called *throughput transparency for blue*, which means that throughput for blue streams is at least the same in ABE than in plain best–effort. There's an extra requirement for ABE routers: packets of the same color should be served in a FIFO fashion.

ABE uses a virtual queue for scheduling (where "duplicates" of the packets are enqueued), and two real queues, one for each color. Blue packets arriving at the router, if they can be enqueued in the virtual queue, are then enqueued in the blue queue with an associated deadline corresponding to its service time in the virtual queue, and otherwise dropped. Blue packets should be served as close to their deadline as possible. Green packets are enqueued in the green queue as they arrive, with an associated deadline equal to the arrival time plus the delay bound. A duplicate of the green packet is enqueued in the virtual queue if there is enough space. The scheduling policy is as follows: if the next blue packet to be served has enough time left before its deadline to let the first green packet in the queue pass, the green packet is sent, if its deadline has not expired. Otherwise, it is dropped. If the next blue packet does not have enough time left (to serve another green packet) before its deadline, it is served.

3.3 Application–level Quality

The network aspects discussed above are not the only ones affecting the quality of a media stream. Several application–level factors exist which have a more or less important impact on the perceived quality. For instance, the codec used will determine the maximum quality that the application can attain. Sometimes trade–offs need to be done, for example, using a higher compression ratio in order to be able to work with lower bandwidth requirements will likely decrease the quality of the stream. Similarly, one can choose to add some form of redundancy to the stream, in order to protect it from packet losses, at the expense of bandwidth and delay.

[1]Note that a decrease in delay for a given route may be seen by the sender as a decrease in network load.

Several techniques to improve the perceived quality may be implemented at the application level, and many of them may be adjusted in real–time, which is useful when we are trying to control quality dynamically. Most of these techniques concern themselves with correcting errors due to packet loss in the network [15, 16, 106], since this is one of the most important factors on the perceived quality. Furthermore, sometimes the effect of other network impairments, such as a very long delay, for instance, can be seen as losses from the application's point of view. For example, if a packet arrives after its play-out time, it is a lost packet for the decoder. Some of the mechanisms described below allow for at least partial recovery in this kind of situation.

3.3.1 Quality–affecting Parameters at the Application Level

3.3.1.1 Coding

When transmitting multimedia streams, it is often desirable to eliminate some of the natural redundancy found in the media in order to fit the bit rate of the stream to the available bandwidth. More frequently, due to bandwidth constraints, we need to find "smart" ways of sending just the data that is *needed* to reproduce the stream at the receiver's end. Of course, this may influence the perceived quality, since

- removing redundancy makes the stream more vulnerable to network losses, and
- most of the high compression ratio codecs used are *lossy*.

The use of lossy codecs allows to achieve much higher compression ratios by sacrificing some "details" of the stream. Some of the most famous lossy compression schemes are the MPEG Layer–3 (MP3) codec for audio, and the JPEG codec for still pictures. Lossless codecs, on the other hand, yield the same quality as the original media, but their compression ratios are generally too low to be used in a networked environment.

For audio, we can find three main classes of coding techniques, namely:

Waveform coding, which keeps the encoded signal very close to the original one. A very simple technique that uses waveform coding is Pulse Code Modulation (PCM), which simply samples and quantizes the signal. PCM is a very simple technique, which can yield very good quality, at the cost of a somewhat high bit rate. Two well known examples of PCM are the A–law and μ–law codecs (also known as ITU–T G711). More advanced forms of waveform coding can be found in Differential Pulse Code Modulation (DPCM) and Adaptive Differential Pulse Code Modulation (ADPCM). These techniques take advantage of the correlation between speech samples in order to predict future waveforms from past ones, and transmit the error between the predicted sample and the real one, which can be encoded with a lower bit rate. The adaptive version of this codec lets the predictor adapt to match the characteristics of the speech being currently coded.

Source coding, which allows to achieve very high compression ratios by using speech models in order to reduce the data needed to reconstruct the signal. The main idea is to determine the model parameters that give the best fit to the original signal, and send them over the network instead of the signal itself. On the receiving side, the parameters are fed to the model, and the speech is recreated and played. This kind of codecs are also known as *vocoders*, and they can produce intelligible output at bit rates as low as 2.4kbps. The resulting speech does not generally have a high quality, since it is synthesized, and thus loses important characteristics. A well known example of this technique is the U.S. Department of Defense's Linear Predictive Coding (LPC) algorithm.

Hybrid coding, which as it name suggests, stands in between waveform coding and source coding. Hybrid techniques are similar to source coding in that they use a model, but they try to better reconstruct the original samples by using an excitation signal that allows for a better match between the reconstructed sample and the original one. This excitation signal is then sent along with the model parameters. Known examples of this technique are the Code–Excited Linear Predictive (CELP) family of codecs (such as the new open–source Speex [124] codec, which provides good quality speech compression in bit rates varying from 2kbps to 44kbps).

Given the differences between codecs, the perceived quality of a speech / audio stream can vary significantly from one codec to the next. Table 3.2 [93] presents quality scores for several widely used codecs, along with rate and algorithmic complexity information.

Codec	Bit rate(kbps)	Complexity (ips x 10^4)	Quality (out of 5 points)
μ–law	64	4	4.2
ADPCM	32	5	4.1
GSM	13.2	7	3.6
G.723.1	5.3/6.3	7.3	3.5/4
LPC	2.4/5.6	6	2.4/3

Table 3.2: Quality scores, bit rate and algorithmic complexity for several widely used codecs (taken from [93]). Note that the quality scores are just given as a reference, and may slightly vary between tests.

Video compression is more complex than audio compression, and therefore, for real–time systems, there are compromises to be made between low bit rates and fast coding, unless specialized hardware is used. Of course, the same trade–off between bit rate and quality found in audio coding is present in this case too. For video coding, the main characteristics of video streams that can be exploited in order to reduce the bit rate needed for transport are:

Spatial redundancy due to neighboring pixels in an image generally being very similar.

Temporal redundancy due to the similarity of a video frame and the ones surrounding it (except of course in the case of sudden changes in the scene).

Spectral redundancy due to the correlation of color planes in an image.

As for speech / audio, there exist lossless and lossy coding schemes for video. There also exist hybrid techniques, which use lossless compression for some frames, and lossy for others. Some well known codecs such as H.263 and MPEG 1,2, and 4 use hybrid coding techniques, with excellent quality and compression ratios, though some of these codecs are not well adapted for streaming.

In *spatial* (or *block*) coding, the algorithm exploits the redundancy of pixels within the same frame. The picture is decomposed into blocks (generally 8x8 pixels). The values representing each pixel are centered around 0 (say, if a color–depth of 8 bits per pixel is being used, values will range from -127 to 128). A discrete cosine transform is then applied, and its coefficients are quantized (this quantization implies a certain loss of quality, and allows a trade–off between quality and compression rate). Later, each block of coefficients is mapped into a vector and are further compressed by, for instance, a run–length encoder.

Temporal coding is used to remove the temporal redundancies found in the encoded stream. This is done by dividing the frame into *macro blocks* (usually 16x16 pixels), which is then compared to neighboring macro blocks in the following frames. If the same macro block (or a good enough approximation) is found in one or more of the following frames, it is then only encoded in the first frame, and replaced with a motion vector which indicates its current position in the others.

Among the most widely used video codecs, we find:

MPEG 1 and 2, developed by the ISO/IEC Moving Picture Experts Group (MPEG). These codecs use several layers which are useful for searching within the stream, correcting errors, or synchronizing with an associated audio stream. Three types of frames are used in MPEG coding: Intra frames (I) which are encoded without any temporal compression, and Inter frames (P and / or B), which are encoded using motion prediction. Each P frame is predicted by the I or P frame which precedes it. B frames, on the other hand, are predicted from the frames surrounding them, which makes for higher compression ratios, but may propagate errors across frames. Normally, frames are sequenced into a group of pictures (GOP), such as IBBPBBP...

H.261, which is an ITU standard conceived for use over ISDN lines. It uses a similar, but simpler algorithm to that of MPEG. It provides data rates ranging from 40kbps to 2Mbps. H.261 only supports two resolutions (CIF, 352x288 pixels and QCIF, 176x144 pixels). It does not use B frames, and its GOP is IPPPPP...

H.263, which is also an ITU standard originally designed for low bit rates of up to 64kbps.

The bit rate limitation has since been removed, and H.263 then offers some advantages over H.261 in terms of robustness.

H.264 / MPEG–4 part 10 AVC, is the current state of the art in video coding for networked applications, jointly developed by the ITU and the ISO/IEC MPEG. It was designed as a solution for, among other things: broadcast over cable, satellite, cable modem, xDSL; interactive and serial storage on optical and magnetic media; conversational services over ISDN, LANs, xDSL; video on demand (VOD), streaming services; Multimedia Messaging Services (MMS). The standard provides a Video Coding Layer (VCL), designed to represent video content efficiently, and a Network Abstraction Layer (NAL) which handles the VCL representation of video and provides header information for a number of transport layers and storage media. The NAL provides support for streaming video over RTP, for storage in several container formats such as the ISO MP4, for integration with ITU H.32X. A comprehensive overview of H.264 can be found in [133].

3.3.1.2 Echo Cancellation and Side-tone

Echo is a well–known artifact of voice communications, which consists of the reception by the person currently talking, of a delayed and attenuated copy of his speech. In older analog Public Switched Telephone Networks (PSTN), it usually happened due to leakage in the four–wire to two–wire hybrid conversion [92] and / or handset. In modern digital telephone networks and VoIP, it does not normally occur, except when using a hands–free environment (which is a common situation for many workstation–based multimedia applications, or in audio–conference situations when more than several parties share a room), in which case the sound coming from the speakers is fed back to the microphone. Another scenario in which echo might occur is in an hybrid call where one of the end–points is a regular telephone, and the other one is a VoIP device. In this case, there might be an echo created.

If the delay between speech and its echo is sufficiently small, it may go unnoticed, as it would be considered as *side-tone*, which is the speaker hearing his own voice through the earpiece as he speaks[2], and is in fact a desirable [34] effect. However, if the delay exceeds a few milliseconds, as it is often the case, an *echo canceler* needs to be implemented lest the quality be severely degraded. Currently, most applications and devices implement some form of echo cancellation in order to avoid this problem.

3.3.1.3 Silence Detection and Suppression

Speech normally consists of *talk–spurts* and *silence gaps*. As no useful information is contained in the silence gaps, they can be suppressed from the stream [89], thereby reducing the bandwidth consumption. Besides, detection and suppression of silence gaps provides other advantages such as

[2]For a hands–free system, this would be the speaker hearing himself talk, so the echo should be almost synchronous with his own speech.

- allowing echo suppression based on the silence detector output,
- allowing per–spurt play out delay adjustment, which is useful for countering the effects of jitter by adjusting the dejittering buffer size [95].

A silence detector compares the current frame's energy to a threshold value in order to decide if there is a change of state from spurt to gap or vice–versa. The threshold is dependent on the current state, and is normally lower for the gaps, since if speech fails to be detected, the talk–spurt is clipped. In order to avoid clipping at the end of a talk spurt (due to premature detection of silence), a *hangover* period is defined, which allows to prevent clipping at the end of the talk-spurt. Both the thresholds and the hangover time can be dynamically adapted (as for instance in the G.729 Voice Activity Detector [8], which does the adaptation automatically) if need be.

3.3.2 Improving Quality at the Source

The quality–improving techniques described in this section require the participation of the sender, in that some modifications need to be performed on the original stream in order for them to work. Of course, the receiver needs to be able to make proper use of the added or modified data.

3.3.2.1 Forward Error Correction

Forward Error Correction (FEC) is based on the addition of redundancy to the stream, so as to be able to reconstruct at least some of the information missing due to packet losses in the network. There are two different kinds of FEC for multimedia streams, namely:

Media–independent FEC schemes, that use algebraic techniques to produce extra packets which contain enough information to reconstruct the missing ones (examples of this type of FEC are parity coding, Reed–Solomon codes, or complex XOR schemes like the one described in [37]).

Media–dependent FEC schemes, which benefit from knowledge about the stream's structure and coding properties in order to provide efficient protection. Media–dependent FEC (in particular the scheme described in [16, 17]; see below for details on how it works) is the kind most commonly found in tools and in the literature.

The performance of all FEC schemes is dependent on the distribution of the packet losses, so some adjustments may need to be done in order to optimize it. Also, the extra bandwidth needed by FEC depends on the correction technique used. Media–independent FEC imposes a constant overhead, since it works at the bit–level. With media–dependent schemes, the amount of redundancy may be modified following the need for it at any given moment, and the extra bandwidth consumed can be reduced by up to 30% [106] with respect to that of media–independent FEC. In any case, the overhead introduced by FEC is generally not very significant, and its advantages are very noticeable. FEC plays an essential role in the perceived quality of a stream, and examples of its impact can be found in Chapters 6, 7, and 8.

In [16, 17], Bolot proposes a simple redundancy scheme for VoIP traffic that consists of piggybacking a compressed copy[3] of the contents of packet N in packet $N + i$. In the event of losing packet N, the receiver waits for the arrival of packet $N+i$, and then uses the compressed version of packet N to fill the gap in the stream. The FEC offset i, which in the original version of this scheme is set to 1, may be adjusted in order to compensate for longer loss burst sizes if need be. It should be noted that while increasing i may increase the level of protection against losses, it also increases the mouth–to–ear delay, since a larger play-out buffer is needed. Figure 3.4 gives a graphical description of this FEC scheme, and Figure 3.5 shows how it works.

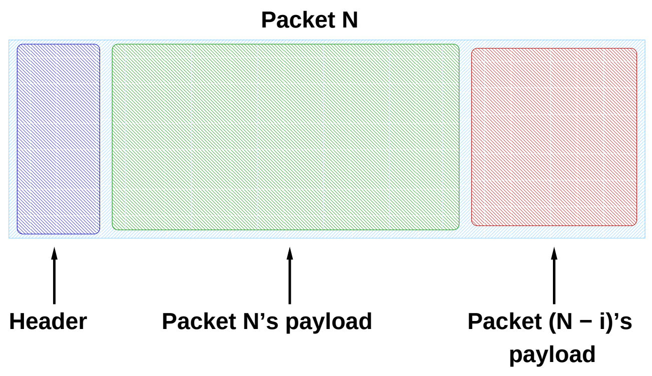


Figure 3.4: Media–Dependent FEC: a packet carrying its own payload, plus a compressed copy of a previous packet's payload. If packet $N - i$ is lost, it can be reconstructed from packet N.

3.3.2.2 Interleaving

Interleaving allows to disperse the effects of a packet loss over the stream, in order to make them less noticeable. The idea is to re–order the samples (be it audio or video) in such a way that consecutive samples end up in different packets. Thus, if a packet is lost, instead of a big gap in the stream, we get several smaller gaps which, if they are small enough (for instance, shorter than a phoneme for VoIP traffic), are not easy to notice. In order to better explain the concept of interleaving, allow us to provide an example for voice traffic. Assume that we are using 20ms packets, in which 4 samples of 5ms each are encoded. In this case, the four samples in each packet are spread over four different packets, as shown in Figure 3.6.

[3]Actually, both copies are compressed, since the original has already been processed by a codec. The redundant copy is usually compressed at a lower quality, in order to diminish bandwidth consumption.

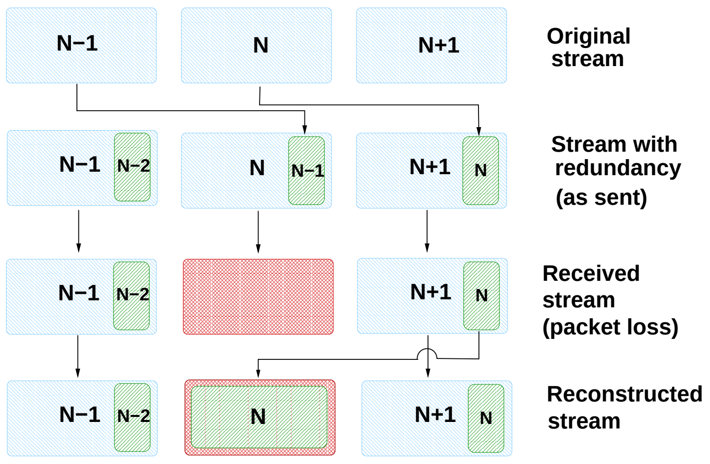


Figure 3.5: Media–dependent FEC in action. When packet N is lost, the stream is reconstructed by using the compressed version carried by packet $N + 1$.

Using interleaving in this example's context, a flow with the following packets / sample structure

```
(1,2,3,4),(5,6,7,8)...
```

results in the following flow:

```
(1,5,9,13),(2,6,10,14)....
```

Packets are "filled" sequentially in the following way. Let S_i be the sequence of all samples in the ordered stream, and P_j the sequence of packets in which the samples will be stored in the interleaved stream. We proceed by placing each sample S_i in the next available slot in the next non–full packet P_j such that $i \mod 4 = j \mod 4$. The position of S_i in P_j is equal to $P_k^0 \mod 4$, where P_k^0 is the packet that held S_i in the ordered stream. The de–interleaving process is analogous.

As with FEC, the use of interleaving increases the mouth–to–ear delay, since in order to re–order a packet's samples into their original state, we need to wait for several packets to arrive. Depending on the packetization interval used, and how many packets are used to spread the samples, this extra delay could be quite high. In the example discussed above, we would induce an extra 60ms delay (which is needed to reconstruct every $(4i + 1)^{\text{th}}$ packet), which is why it might not be always useful for interactive applications.

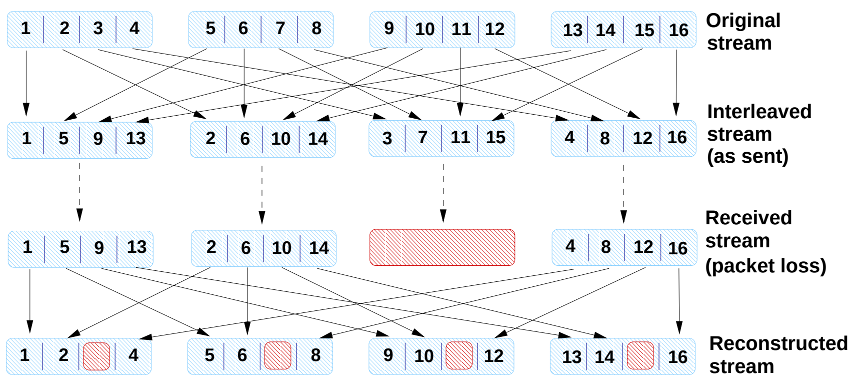


Figure 3.6: Interleaving example from [106]. Consecutive samples are spread over several packets to minimize the impact of a lost packet on the perceived quality. When a packet is lost, the reconstructed stream has several (very) small gaps which are more difficult to detect than a single large one.

3.3.3 Improving Quality at the Receiver's End

Besides the techniques mentioned above that require the cooperation of the sender, several methods exist to improve the perceived quality which can be implemented at the receiver alone. This section provides a short overview of the most commonly used ones.

3.3.3.1 Buffering

Buffers are used at the receiver as a means to diminish the impact of jitter on quality. In most applications, packets need to be fed to the decoder at a constant rate. However, one of the problems with packet networks such as the Internet, is that the end–to–end delay is quite variable. Therefore, if no buffer is present at the receiver, packets might arrive later than expected, forcing to stop playback, or sooner than expected, in which case they would have to be dropped.

A dejittering buffer helps to solve this problem (or at least reduce its influence on the perceived quality), by absorbing variations in delay between packets. Of course, the price for this improvement is an increase of the delay, since packets sit at the buffer for some time before being consumed by the decoder. A commonly found addition to dejittering buffer implementations is the ability to dynamically adjust their size, in order to minimize the extra delay by just having enough buffering time so as to compensate for the jitter at any given moment [21, 95, 101, 108].

3.3.3.2 Loss Concealment

In [59, 106] the authors present several techniques used to “conceal” packet losses in audio streams when they are not recoverable. Similar techniques exist for video streams [22, 36] as well. We briefly describe some of these techniques below.

We find three main kinds of loss concealment techniques, namely:

- Insertion–based, which add data to the received stream in order to fill the gap caused by the loss (see below).
- Interpolation–based, which use data surrounding the gap in order to create a substitute for the missing data.
- Regeneration–based, which rely on surrounding packets and decoder state in order to regenerate the missing data. This is particularly useful for predictive (hybrid) coding schemes. With these coding schemes, new samples are derived from previous ones and the codec state. This allows some level of recovery if a packet is lost, since the receiver has the previous ones, and knows the codec state.

Insertion–based techniques

According to [106], insertion based techniques are the easiest to implement, but they perform poorly. Some examples of this type of concealment technique are:

Splicing consists of simply ignoring the missing packet, and playing the surrounding packets consecutively. This mechanism has a big impact on the perceived quality, due to the clipping it produces. Besides, if the loss rate is too high, it can cause a buffer under-run, since packets are consumed faster than they should be.

Silence substitution is a technique that replaces missing audio samples with silence. It is only useful for short lengths amounts of audio, up to about 15ms. This means that it is not very useful for traditional VoIP applications (which tend to use packetization intervals of 20ms and more), but it may be used for instance along with interleaving. Even if this technique does not provide good quality improvements, it is widely implemented due to its sheer simplicity.

Noise substitution replaces the missing audio with white noise, which allows the listener’s brain to restore the missing data via a phenomenon known as *phonemic restoration*. This provides a better perceived quality than silence substitution [59, 106].

Repetition consists of substituting the missing data for a copy of the preceding packet's data. This mechanism presents the advantage of being very easy to implement and providing good improvements in the perceived quality. This technique is prescribed in the GSM standard.

Some of the preceding techniques may also be adapted for video streams, especially repetition, which is usually accompanied by some form of motion compensation calculated from the blocks surrounding the missing one.

Interpolation-based techniques

Interpolation–based techniques try to approximate the missing data based on the surrounding data. These techniques tend to yield better quality than the substitution–based ones [106], since they take the changing nature of the signal into account. Some examples of interpolation techniques are:

Waveform substitution uses audio data preceding (and also after, if need be) the missing data in order to find a suitable signal to replace the lost data. This technique uses templates to locate suitable pitch patterns on either side of the gap [106]. Interpolation effectively occurs in the case when both surrounding packets are used. In the one–sided case, the pitch pattern is repeated. The two–sided scheme provides a more significant quality improvement.

Pitch waveform replication is a refinement of waveform substitution used for voice streams, which falls back on packet repetition if the loss happens during an unvoiced speech [121] segment. For voiced speech segments, it repeats a waveform of appropriate pitch length. This technique works slightly better than waveform substitution.

Time scale modification stretches the audio samples surrounding the missing data, in order to cover the gap. The idea is to make each of the surrounding samples last longer, so that they play over the time slot of the missing data. This technique, while computationally complex, yields better results than the preceding two.

Hybrid temporal and spatial concealment is a technique used for video streams[22], which replaces a missing block by a new one calculated from the blocks surrounding it, and information from the previous frame. In this way, the missing block can be quite accurately estimated, even taking motion into account.

Regeneration–based techniques

These mechanisms use knowledge of the compression algorithm, and of the decoder state to synthesize the missing data from the codec parameters. Of course, the performance of these mechanisms is largely codec–dependent, but they perform generally well (and at a somewhat high computational cost).

3.3.3.3 Layered coding

Another technique to improve the perceived quality, is to use a layered coding scheme [15, 110]. The idea is that the sender transmit several flows containing each a different layer of the stream. The receiver can choose which of the layers to receive, based on network conditions. Depending on the application, the different layers may be selected by subscribing to different multicast groups, or by establishing more connections to the server, for example. Typically, there is a layer that provides a basic quality level, and one or more higher quality layers.

This technique is particularly useful in multicast environments, where scaling issues prevent from sending feedback to the server for it to adjust the stream's quality at the source. Moreover, if the network does provide some form of QoS provisioning, such as DiffServ, different quality settings can be used for the different layers, thus ensuring that at least basic QoS levels are met.

3.3.3.4 Recovering Corrupted Data

Not all packet losses are due to congestion in the network. Some packets are discarded because they have corruption problems. This is particularly true in error prone networks, like wireless LANs (WLANs) or cellular networks. When packets have erroneous bits, they are generally discarded by the network stack of the receiving host, since the checksum controls (be it at the MAC or transport layer) will not be correct.

Hammer et al. [57] show, however, that if the header data is intact, speech can sometimes be recovered from corrupted packets. Their approach makes use of UDP–Lite [94], which allows the UDP checksum to cover an arbitrary portion of the packet. If the checksum is only applied to the headers, errors in the payload would not cause the packet to be dropped. In their study, they used the ARM codec [2], which was designed for use in the Universal Mobile Telecommunications System (UMTS), for the 3^{rd} generation of mobile telephones. The ARM codec was conceived with robustness in mind, which allows for better recovery of corrupted data. Their results show a significant improvement in perceived quality when corrupt packets are recovered, especially for higher bit error rates (in the order of 10^{-3}). A higher improvement was observed when header compression was used, which is to be expected since the reduction of header sizes diminishes the probability of header corruption.

For video, a similar approach has been taken by Masala et al. in [97]. In their proposal, they deal with the transmission of real–time video over IEEE 802.11 networks. They study the use of partial checksums for the 802.11 MAC layer, coupled with UDP–Lite at the transport layer. As with speech, there is a marked improvement on the video quality when the corrupted data is recovered instead of being dropped.

Chapter 4

Multimedia Quality Assessment: State of the Art

In this chapter we present the issues associated with the assessment of the perceived quality of multimedia streams. We also discuss the evaluation mechanisms currently available in the literature, and why they don't necessarily fulfill the current needs in terms of quality assessment.

4.1 Subjective Quality Assessment

Subjective quality assessment methods measure the overall perceived media quality, and as implied by their name, they are carried out by human subjects. They are basically lab experiences in which users are confronted with degraded[1] audio or video samples, and they have to rate their quality in one or more ways. The process of subjectively assessing the quality of a media sample (be it networked or local) is a complex and time–consuming one. There are several types of subjective assessment techniques, and the one to be used depends on the kind of application to assess, and what quality aspects need to be evaluated.

Two main classes of subjective assessment can be found, namely

- Qualitative assessment, and
- Quantitative assessment.

Qualitative assessment [18, 19] is based on descriptions of the end–user perception of quality, which do not necessarily translate well into numeric scales. This kind of assessment is useful in several contexts, for instance, when evaluating pricing schemes for a multimedia service, or how different people react to variations of the perceived quality. These methods are more suited to the "sociologic" aspects of quality than to the "technical" ones, since they tend

[1]The samples are degraded with respect to their original version be it by the use of a codec, or by the transmission over a network, etc.

to be too vague to extract useful information on how the different factors affecting quality do actually affect it.

Quantitative techniques, on the other hand, provide a more concise approach to quality assessment, and are more widely used in the literature than qualitative ones. The main idea behind this kind of tests is to have a (sufficiently large) group of people assess media samples and grade them according to a predefined scale. The results of these tests are normally presented in the form of a Mean Opinion Score (MOS) which summarizes the group's assessment of the perceived quality. There exist several ways of subjectively assessing the quality of media streams. The most widely used ones are those found in ITU standards such as ITU-T P.800 [85] for voice, ITU-T P.920 [88] for interactive multimedia, and ITU-R BT.500–10 [80] for video (TV) quality.

Within these standards, sometimes several procedures are proposed, depending on the purpose of the assessment. One of the most commonly used is Absolute Category Rating (ACR), in which the subjects grade the quality of samples within a given scale (such as the ones shown in Tables 4.1 and 4.2), as they see fit. Sometimes it is desirable to measure the degradation caused by some encoding or transmission scheme, in which case a Degradation Category Rating (DCR) is more appropriate. In this kind of test, users are asked to rate the perceived degradation of the sample with respect to a high–quality reference sample. For cases in which quality–enhancing techniques are used, a Comparison Category Rating may be used. In this case, the subjects are given an original sample and a processed one, and they have to rate the quality of the latter with respect to the former. Sometimes, more complex setups are used, such as MUSHRA (MUlti Stimuli with Hidden Reference and Anchor points) [29, 79], which provides hidden references to allow for calibration and grade anchoring.

Numerical Score	Corresponding Quality
5	Excellent
4	Good
3	Fair (Toll Quality)
2	Poor
1	Bad

Table 4.1: ITU 5–point scale.

Depending on the kind of test to be carried out, the number of subjects needs to be between 4 and 40, although most tests require from 10 to 20 subjects. The subjects should not be familiar with the domain[2], and they should not be given any information on the factors being considered in the experiment.

The standards also define guidelines for sample length, and types of samples to be used. For

[2]Sometimes, tests may be carried out by experts [88], which by definition are familiar with the domain, of course.

Numerical Score	Corresponding Quality
8, 9	Excellent
6, 7	Good
4, 5	Fair (Toll Quality)
2, 3	Poor
1	Bad

Table 4.2: ITU 9–point scale.

instance, for voice, it is desirable to have 50% of male speakers and 50% of female speakers, and the samples should be between 5s and 10s long. For video, different samples should have different properties in terms of movement, color, etc. The length of the assessment sessions is also limited by the standards, and it should not go beyond 30 minutes. If need be, the test can be divided into several sessions. Each session should start with some *warm–up* samples, which should not be taken into account in the results. Their purpose is to train the subjects so that they provide more accurate results. Furthermore, hidden reference samples can be used during the tests in order to determine the reliability of the subjects.

Once the test is carried out, the scores given by the subjects are averaged, and a statistical screening is done in order to detect subjects who could not conduct the test properly. What follows the description of one of the methods proposed for this screening in the ITU-R BT.500–10 recommendation [80]. The first step of the screening process is to determine whether the scores have a normal distribution or not. We do this by using the β_2 test as follows: let

$$\beta_2 = \frac{m_4}{(m_2)^2},$$

where

$$m_x = \frac{\sum_{i=1}^{N}(u_{ijkr} - \bar{u}_{jkr})^x}{N}.$$

In this expression, $\bar{u}_{jkr}$ is the mean of all the evaluations u_{ijkr} done by all users $i = 1 \ldots N$, under condition $j = 1, \cdots, J$, for sequence $k = 1, \cdots, K$ in the r^{th} repetition[3] $r = 1, \cdots, R$. If $2 \leq \beta_2 \leq 4$ then the scores' distribution may be taken to be normal.

We then define two counters P_i and Q_i for each observer i, and update them as shown in Algorithm 1, where J is the number of conditions, K is the number of samples, R is the number of repetitions, and S_{jkr} is the standard deviation of the scores for each tuple (sample,condition,repetition).

[3]In the context of this dissertation, we will consider overall scores for each condition, not taking into account each individual sample, and not doing any repetitions, so in our case, u_{ijkr} becomes u_{ij}. We will, however, describe the screening process for the general case.

Algorithm 1 Update rules for P and Q.

```
$P_i \leftarrow 0$
$Q_i \leftarrow 0$
**for** $j, k, r = 1, 1, 1$ to $J, K, R$ **do**
  **if** $2 \leq \beta_{2jkr} \leq 4$ **then**
    **if** $u_{ijkr} \geq \bar{u}_{jkr} + 2 \times S_{jkr}$ **then**
      $P_i \leftarrow P_i + 1$
    **end if**
    **if** $u_{ijkr} \leq \bar{u}_{jkr} - 2 \times S_{jkr}$ **then**
      $Q_i \leftarrow Q_i + 1$
    **end if**
  **else**
    **if** $u_{ijkr} \geq \bar{u}_{jkr} + \sqrt{20} \times S_{jkr}$ **then**
      $P_i \leftarrow P_i + 1$
    **end if**
    **if** $u_{ijkr} \leq \bar{u}_{jkr} - \sqrt{20} \times S_{jkr}$ **then**
      $Q_i \leftarrow Q_i + 1$
    **end if**
  **end if**
**end for**
```

We can then reject any observer i such that

$$\frac{P_i + Q_i}{J.K.R} > 0.05 \text{ and } \left| \frac{P_i - Q_i}{P_i + Q_i} \right| < 0.3$$

This test should only be run once on the data set, and once the results have been screened, a new mean score should be calculated on the resulting data set. This new mean is the Mean Opinion Score[4].

One common characteristic of all subjective assessment methods is that they are very time–consuming and difficult to set up. If one is to follow the recommendations in a strict manner, then numerous parameters ranging from room size to equipment calibration should be taken into account and precisely controlled. Moreover, since they involve an important number of people and are time–consuming, they are very expensive to carry out, and by definition, they cannot be automated, which rules out their use for real–time quality control, or billing.

Some authors [131, 132] also argue that the assessment protocols, as defined in the ITU recommendations, are not well adapted to multimedia streams sent over a best–effort network, since they were designed for applications which normally have better quality levels. Others [58, 123] suggest that the use of short samples, separating audio and video, are not the most appropriate way to measure the perceived quality of a stream, since it is not possible to capture

[4]The MOS is sometimes called DMOS or CMOS if the test was a Degradation Category Rating or Comparison Category Rating, respectively.

the semantic factors that come into play during normal viewing/listening. For instance, the relative weight of audio and video in a mixed stream's quality strongly depends on the contents of the stream. If it is, say, a news anchor giving a report, it is very likely that audio will be more important than video in the overall quality. For a sports event, it could be the other way around.

4.2 Objective Quality Assessment

The difficulties, costs and limitations associated with subjective quality assessment have motivated the development of *objective* quality assessment mechanisms. These exist for audio, speech and video, and for the most part, they base their estimation of the perceived quality on a certain notion of *distance* between the original signal, and the degraded one. It is therefore necessary to have both streams at hand when performing the assessment, which limits their use to non–real time applications. The complexity of these techniques is quite variable, from very simple measures such as Signal to Noise Ratio (SNR), to very complex metrics that take into account models of human vision or hearing.

The performance of these objective metrics is generally emasured in terms of their correlation with subjective scores. The correlation coefficient of two random variables X and Y, which we will denote by r_{XY}, provides a measure of the linear relationship between X and Y. An ideal quality metric should have a correlation of 1 with subjective scores. This coefficient is calculated as the ratio between the covariance of X and Y and the product of their respective standard deviations.

$$r_{XY} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_y}$$

In this section, we will briefly cover the main objective techniques found in the literature for both speech and video assessment.

4.2.1 Speech Quality

Objective speech quality assessment techniques can be classified into three groups, each having a certain correspondence with the three coding techniques mentioned in Section 3.3.1.1.

Time domain metrics are mostly intended for coding / transmission systems which try to reproduce the original waveform (such as waveform coding techniques), and require a very precise time–alignment of the streams in order to compare both waveforms.

Spectral domain metrics are usually applied to short segments, typically about 15 to 30ms long. They are less sensitive to time–alignment problems than time domain metrics, and generally provide better results. They are, however, very dependent on the codec used (which limits their generality) and not suitable for use in networked systems, which may modify some characteristics of the stream. These metrics are well suited for assessment of coding impairments, in particular for source coding techniques.

Perceptual domain metrics are based on some form of psycho–acoustic model. They are usually more accurate than the other kinds mentioned above. One weak point of these metrics is that they are usually optimized for a certain kind of speech, so they may not provide accurate assessments with other kinds of speech data.

We list some of the most prominent objective assessment techniques for speech quality. Among the objective speech quality metrics found in the literature, the most relevant are:

- Signal-to-Noise Ratio (SNR)
- Segmental SNR (SNRseg)
- Bark Spectral Distortion (BSD) [129]
- Enhanced Modified Bark Spectral Distortion (EMBSD) [136]
- Perceptual Analysis Measurement System (PAMS) [111]
- Perceptual Speech Quality Measure (PSQM) [12]
- PSQM+ [10]
- Measuring Normalizing Blocks (MNB) [128]
- Perceptual Evaluation of Speech Quality (PESQ) [87, 112]
- ITU E–model [81]

The following subsections provide a short overview of each of these metrics.

4.2.1.1 Signal to Noise Ratio

The signal to noise ratio (SNR) is perhaps the simplest objective speech quality metric, and it is based on a simple comparison between the original stream and the degraded one. The formula for SNR is as follows:

$$\mathrm{SNR} = 10\log_{10}\frac{\sum_{i=1}^{N} o(i)^2}{\sum_{i=1}^{N}(o(i)-d(i))^2}$$

where N is the number of samples in the stream, $o(i)$ is the i^{th} sample of the original stream and $d(i)$ is the i^{th} sample in the degraded stream.

For SNR to provide any meaningful results, both streams have to be properly time–aligned, since a slight shift (for instance caused by missing data due to a packet loss), can severely affect the results.

Segmental SNR (SNRseg) is a variation of SNR that works on short (15 to 20ms) segments of the stream. This allows for easier time–alignment, and it provides results which are slightly better (in terms of correlation with subjective assessment) than those of SNR. It is calculated as

$$\mathrm{SNRseg} = \frac{10}{S} \sum_{s=0}^{S-1} \log_{10} \sum_{j=Ls}^{Ls+L-1} \left(\frac{\sum_{i=1}^{L} o_j(i)^2}{\sum_{i=1}^{L} (o_j(i) - d_j(i))^2} \right)$$

with S being the number of segments into which the stream is divided, L being the number of samples in each segment, and $o_j(i)$ and $d_j(i)$ being the i^{th} sample of the j^{th} segment for the original and degraded stream respectively.

4.2.1.2 Bark Spectral Distortion

The Bark Spectral Distortion (BSD) metric [129] is based on a psychoacoustic model. BSD assumes that a speech sample's quality is directly related to its loudness. The loudness is defined as the magnitude of auditory sensation. BSD uses a psychoacoustic model to calculate the loudness of the original and degraded samples. It then calculates the mean Euclidean distance between their respective loudness vectors, and uses the result as a measure of distortion. It should be noted that this approach only works for voiced speech, so the quality of the estimation is dependent on the speech samples themselves, and of course, it needs a mechanism to detect voiced speech within the sample.

4.2.1.3 Enhanced Modified Bark Spectral Distortion

This metric [136] is based on BSD. In EMBSD, the speech is decomposed into overlapping frames. Each frame exceeding a certain energy threshold is transformed into the spectral domain, and the Bark coefficients are transformed into a loudness measure (in dB). The first 15 coefficients of each frame are used to calculate the difference in loudness between the original stream and the distorted one. If a large enough difference is maintained during at least 10 frames (or 200ms), it is considered as distortion. An average of these differences is later calculated to give the final distortion measure. The quality of an EMBSD measurement is strongly dependent on the synchronization of both speech samples.

4.2.1.4 Perceptual Speech Quality Measure

The Perceptual Speech Quality Measure (PSQM) [12, 86] was the former ITU standard for voice assessment. It was withdrawn in 2001, giving way to the newer Perceptual Evaluation of Speech Quality (PESQ) [87] (see Section 4.2.1.8).

PSQM was derived from an audio quality metric (Perceptual Audio Quality Metric, or PAQM [11]), and it provides a good performance for measuring coding distortions. The PSQM algorithm maps the signal into the loudness domain, and then applies a (nonlinear) scaling factor to the loudness vector. It then calculates the *noise disturbance* as the difference between the scaled loudness vectors of each frame in both the original and the degraded signals. A weighted average is then performed on the noise disturbances of all the frames in order to get a measure of distortion. Silent periods are considered with a very small weight, or may also be discarded.

4.2.1.5 PSQM+

PSQM+ [10] is, as its name suggests, an improved version of PSQM. The main difference with PSQM is the introduction of a scaling factor when calculating the overall distortion. This provides a better performance when loud distortions (such as those produced by temporal clipping) are found.

4.2.1.6 Perceptual Analysis Measurement System

The Perceptual Analysis Measurement System (PAMS) [111, 113] was developed by British Telecom in order to assess the quality of voice streams (in particular those carried over a PSTN, though it can also be used for VoIP). PAMS uses an auditory model combining a psychoacoustic model with a technique that takes into account the subjectivity of errors in the degraded speech.

The original and degraded speech samples are time–aligned, equalized, and subjected to an *auditory transform* that maps them into a time–frequency loudness representation. The difference between both signals in this domain yields an error surface, which is then interpreted by the *perceptual layer*, which predicts the perceived quality.

4.2.1.7 Measuring Normalizing Blocks

The Measuring Normalizing Blocks (MNB) metric [128] provides quality estimations for narrowband (4kHz) speech, for a variety of codec types, allowing for bit–errors, frame drops, and constant delay.

It performs frequency warping and logarithmic scaling on speech signals in order to map them into an approximate loudness domain. The algorithm takes into account the human sensitivity to the distribution of distortion within the speech signal. An approximate loudness vector is generated for each frame, and two measurements are done:

Time Measuring Normalizing Blocks (TMNB), which integrate over frequency scales and measure differences over time periods, and

Frequency Measuring Normalizing Blocks (FMNB), which integrate over time intervals and measure differences over frequency scales.

Once these MNB have been calculated for several frames, they are linearly combined to provide an estimation of the speech distortion.

4.2.1.8 Perceptual Evaluation of Speech Quality

The Perceptual Evaluation Speech Quality metric (PESQ) [87] is the latest objective quality measurement scheme from the ITU. It was designed to provide good assessments for both PSTN and packet–based networks. It can take several quality–affecting factors into account, including coding distortions, bit–level errors, packet losses, delay, jitter and analog filters in older PSTNs.

The following is an outline of the steps performed by the PESQ algorithm.

Level alignment – both the original and degraded samples are aligned to a constant power level.

Input filtering – it is needed to compensate for the filtering that occurs in the network and on handsets.

Time alignment – it is used to compensate for variable delay or losses in the transmission, which may modify the timescale of the degraded signal. PESQ uses a modified version of the time alignment algorithm used in PAMS.

Auditory transform – both signals are processed by a psycho–acoustic model, and their inaudible parts are removed.

Disturbance processing – non linear averages are used to calculate two different disturbances:

- the absolute disturbance, which is a measure of the absolute audible error, and
- the additive disturbance, which is a measure of errors much louder than the original signal.

PESQ is considered to be state the art in objective speech quality assessment as of this writing. We will discuss more on its performance and how it compares to the other objective metrics and to our approach in Section 5.5.

4.2.1.9 The ITU E–Model

From the available objective speech quality measures in the literature, only the ITU E-model does not need the access to the original signal to compute a quality estimation. Therefore, it is the only available measure which is computationally simple [55] and can be used in real–time applications. However, the E-model was not designed as a quality assessment tool (as clearly stated in the recommendation defining it), but as a network planning tool, in particular for “the conversational quality of 3.1 kHz telephony”.

The E-model provides the means to take several quality–affecting parameters into account when designing a telephony network, most of which are more related to the field of signal processing than to that of computer networks. For example, impairments due to packet losses have only been explicitly included into the E-Model in the latest revision of the recommendation (dated from March 2003), and then only a uniform loss distribution is considered, which does not provide an accurate model of packet loss on a big network like the Internet. Therefore, the assessments obtained by means of the E-Model are to be considered in the appropriate context, and it is not surprising that they are not as close to MOS tests as one would want. Indeed, from the results reported in [55], the correlation coefficient with subjective results when using it to evaluate the quality variation produced by both encoding and network impairments is about 0.70 in average (the results obtained are highly dependent on the codec used).

In addition, in [28, 35] the authors argue that the E-model is not yet suitable to correctly consider the impairments caused by the transmission over a best–effort network and that more research is needed to adapt it to this function.

The performance problems of the E-model (see Section 5.5 for more details on this) are probably due to its conceptual simplicity, since the quality is supposed to be affected linearly by some specific parameters. The basis for this assumption is that "psychological factors on the psychological scale are additive" [4]. However, the ITU-T recommendation G.107, which defines the E-Model, states that this assumption has not been verified in a satisfying way.

According to the E–model, the quality of a stream is: $R = R_0 - I_s - I_d - I_e + A$, (called the *transmission rating factor*). The R_0 factor is called the *basic signal–to–noise ratio*, and corresponds to the sound quality of speech only distorted by room and circuit noise, with no distortion from coding or network factors. The I_s factor (called *simultaneous impairment factor*) corresponds to the degradation of the speech signal itself, for instance due to excessive loudness, sidetone or other impairments which may occur simultaneously with speech. Delay–related impairments, such as echoes and the end–to–end delay itself are considered in the I_d factor (*delay impairment factor*). The distortions caused by the use of codecs, and network impairments such as packet loss are combined into the I_e factor, which is called the *equipment impairment factor*. The impairments caused by different codecs at different rates are tabulated, and the tables are based on subjective tests done for each codec. Loss-related impairments used to be tabulated as well, but the last revision of the E–model provides a way to calculate them based on the loss rate (assuming losses are independent). A is the *advantage factor*, which expresses the decrease in the rating R that a user is willing to tolerate because of the access advantage that certain systems (e.g. cellular networks) have over traditional wire-bound telephony. For more details about the E-model see [28, 35, 81].

4.2.2 Limitations of Objective Speech / Audio Assessment in a Network Context

Many of the objective metrics described above were originally developed to measure the degradation suffered by speech when it is encoded. Therefore, they do not generally take into account network factors, and their performance (in terms of correlation with subjective assessment results) is affected when they are used to assess VoIP streams, for instance. Besides, except for the ITU E–model, all these metrics propose different ways to compare the received signal *with* *the original one*. This precludes their use in real–time, which is essential for quality–control applications.

The E–model allows to obtain an approximation of the perceived quality as a function of several ambient, coding and network parameters, which are measurable in real–time. It is therefore able to provide real–time estimations of the perceived quality. However, as stated in [55] and even in its specification [81], this tool was conceived to be used for network capacity planning, and its results do not correlate well with subjective assessments.

4.2.3 Video Quality

As with speech, objective video quality metrics range from the very simple to the very complex. In particular, those based on *human vision systems* (HVS) tend to be very complex, so that even when both the original and the degraded signals are available, the assessment cannot be done in real–time. Moreover, *none* of the available techniques can be used without the original signal, as the E–model does for audio, so performing dynamic control of the quality of a video stream based on the current quality, remains unfeasible with traditional mechanisms.

In this section we briefly present some of the most widely used objective video assessment techniques found in the literature, namely:

- Peak Signal–to–Noise Ratio (PSNR)
- Moving Picture Quality Metric (MPQM)
- Color Moving Picture Quality Metric (CMPQM)
- Normalization Video Fidelity Metric (NVFM)
- ITS Video Quality Metric

4.2.3.1 Peak Signal–to–Noise Ratio

The PSNR metric is one of the most widely ones used for objective video quality assessment, due to its simplicity and its low computational requirements. It calculates the ratio between the maximum value of a signal and the background noise. If we consider video clips consisting of K frames of $M \times N$ pixels each, we can define the Mean Square Error (MSE) as :

$$\text{MSE} = \frac{1}{K.M.N} \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{n=1}^{N} [o_k(m,n) - d_k(m,n)]^2$$

where $o_k(m,n)$ and $d_k(m,n)$ are the pixels in position (m,n) in the k^{th} frame of the original and degraded stream, respectively.

We then define the PSNR as

$$\mathrm{PSNR} = 10.\log_{10} \frac{\max_k}{\mathrm{MSE}}$$

with $\max_k$ being the *maximal value* in frame k. Normally, $\max_k$ is taken to be the maximal luminance value in the frame (normally 255), but PNSR can also be calculated with chrominance values, too. In any case, the performance of this metric is quite poor, as it usually does not correlate well with subjective scores. In spite of this, its sheer simplicity makes for its widespread usage in the literature.

4.2.3.2 Moving Picture Quality Metric

The Moving Picture Quality Metric [126] is based on a basic HVS model. In a first phase, it estimates the distortion as the difference between the original clip and the degraded one. It then decomposes both clips into perceptual channels. A channel–based distortion measure is then computed while accounting for contrast sensitivity and masking. Finally, the data is pooled over all the channels to compute the distortion (E), which can then be used to compute a Masked PSNR value:

$$\mathrm{MPSNR} = 10\log_{10} \frac{255^2}{E^2}$$

or it can be mapped into a MOS scale:

$$\mathrm{MOS} = \frac{5}{1+\gamma E}$$

where γ needs to be experimentally determined (this is useful if some subjective assessments are available for one of the degraded clips).

4.2.3.3 Color Moving Picture Quality Metric

The Color Moving Picture Quality Metric [125] is an extension to MPQM that takes chrominance values into account, as opposed to MPQM, which only uses luminance values. The color values of the clip are converted to RGB values, which are linear with luminance. Then, these values are converted to coordinate pairs, corresponding to luminance, red–green, and blue–yellow channels. Then each component of both clips is filtered (in a way similar to that of MPQM, except less filters are used for the color channels). The rest of the algorithm is the same as in MPQM.

Both MPQM and CMPQM provide good correlation values with subjective assessment for high–quality video streams. However, as mentioned in [126], they are less suited for very low bit rate codecs such as the ones normally used in video–conferencing applications, for instance.

4.2.3.4 Normalization Video Fidelity Metric

This metric is based on visibility prediction [126]. The perceptual decomposition is performed on the original clips, without using Fourier transforms. Each image is decomposed into spatial frequency and orientation bands. Both components are then filtered, and an *excitatory-inhibitory* stage is used to measure their energy. The output of this stage is then normalized, and the distortion is calculated as a squared error norm.

4.2.3.5 ITS Video Quality Metric

The Institute for Telecommunication Sciences' Video Quality Metric (VQM) [134] is based on the extraction and statistical manipulation of scalar spatio-temporal features from the original and degraded video sequences to obtain a single measure of distortion. The extracted features closely mimic the effect of various distortions introduced from the encoder and the transmission. These distortions may appear in both the spatial and the temporal dimension.

The ITS model works on edge-enhanced versions of the original and distorted sequences. Gradient operators are used on the pixel domain (as in NVFM, no Fourier transform is applied to the clip) to measure the magnitude and direction of the spatial gradient within spatio–temporal regions, of six frames of 8x8 pixels. These gradients give an indication of the type and size of the distortion. The perceptual impairments within each spatio–temporal region are calculated using functions that model human visual masking. These impairments are then pooled using spatial (average of worst 5% of the measured distortions) and then temporal collapsing (averaged over the clip's duration, which is normally 8-10 sec) to obtain a single quality score for the video clip.

Chapter 5

PSQA – Pseudo–Subjective Quality Assessment

5.1 Motivation

In Chapter 4 we have presented several methods for multimedia quality assessment. Each type of method, and more specifically each individual technique has its own pros and cons, and is more or less suitable for use in different scenarios. However, with the exception of the E-model, none of them can be used in a real–time, networked context since they require both the original and the received signal to estimate the quality. The ability to perform the assessment in real–time is crucial when considering control applications and / or live quality monitoring.

What we need is an approach able to give results that resemble those given by humans and can work without access to the original signal, since this is generally not available in real–time. For this, we propose the use of a hybrid assessment technique. The main idea is to capture the relation between certain parameters that affect the perceived quality, such as network losses, encoding schemes, etc. and the perceived quality itself. This relation is a function of all the *quality–affecting parameters*, whose expression we don't know. Each parameter affects quality differently based on the type of application considered, the contents of the stream, and the values of the other quality–affecting parameters. The technique presented here was first developed in the doctoral work of Samir Mohamed [98], and is based in the use of a statistical estimator, trained with the results of a subjective test, in order to capture the function that ties the parameters considered and the perceived quality. Previous work by Nielsen [103] used Artificial Neural Networks to estimate audio quality from signal characteristics. That work does not deal with the same problems that we (Mohamed's work included) approach, since it uses the ANN to map signal parameters to quality, whereas here we are concerned with application and network–level factors.

5.2 General outline

In this Section, we describe the overall steps needed to build a tool such as the one described in Section 5.1. The statistical estimator used is a Random Neural Network (RNN) [42, 53]. Other tools can be used (cf. Section 5.7), but we found that in our context RNNs outperform them.

We start by choosing a set of *a priori* quality-affecting parameters corresponding to the application considered and to the network supporting the transmission. Examples are the bit rate of the source, the parameters corresponding to the redundancy used, the loss rate in the network, etc. Then, for each parameter π we select a range, that is, a minimal value $\pi_{\min}$, and a maximal one, $\pi_{\max}$. We also choose a set of representative, or significant values in the interval $[\pi_{\min}, \pi_{\max}]$. For example, we will choose the loss rate as one of the key network parameters. If its value is expected to vary from 0 to 10%, and if the typical observed values are between 0 and 5%, then we may use 0, 1, 2, 5, and 10% as selected loss rate values.

Let us denote by $\mathcal{P} = \{\pi_1, \cdots, \pi_P\}$ the set of selected parameters. Let us then denote by $\{p_{i1}, \cdots, p_{iH_i}\}$ the set of possible values chosen for parameter π_i, with $p_{i1} < p_{i2} < \cdots < p_{iH_i}$. We call *configuration* of the set of quality-affecting parameters, any set of possible values for each one. That is, a configuration $\vec{\gamma}$ is a vector $\vec{\gamma} = (\gamma_1, \cdots, \gamma_p)$, where $\gamma_i \in \{p_{i1}, \cdots, p_{iH_i}\}$ The total number of possible configurations (that is, $\prod_{i=1}^{P} H_i$) is usually large. For this reason, we select only a part of this large cardinality set of configurations, which will be used as (part of) the input data for the neural network in the learning phase.

In order to generate a media database composed of samples corresponding to different configurations of the selected parameters, a simulation environment or a testbed must be implemented. This is used to send media samples from the source to the destination and to control the underlying packet network. Every configuration in the defined input data must be mapped into the system composed of the network, the source and the receiver. For example, when working with IP networks and speech streams, the source controls for instance the bit rate, the packetization interval and the encoding algorithm; the routers' behavior contribute to the loss rate and the loss distribution, together with the traffic conditions in the network. The destination stores the transmitted audio sample and collects the corresponding values of the parameters (that is, the configuration). Then, by running the testbed or by using simulations, we produce and store a set of distorted samples along with their associated configurations.

Formally, we select a media sample σ, a set of S configurations denoted by $\{\gamma_1, \cdots, \gamma_S\}$ where $\gamma_s = (v_{s1}, \cdots, v_{sP})$, v_{sp} being the value of parameter π_p in configuration γ_s. From sample σ, we build a set $\mathcal{T} = \{\sigma_1, \cdots, \sigma_S\}$ of degraded samples that have encountered varied conditions when transmitted. That is, the clip σ_s corresponds to the result of sending σ through the source-network system where the selected P parameters have the values of configuration γ_s.

After building these samples, a subjective quality test (for example, one of those mentioned in Section 4.1) must be carried out. The average of the scores given by the human subjects (after the statistical filter described in Section 4.1 has been applied) to each sequence is computed;

the average corresponding to sequence σ_s is denoted here by ψ_s.

In the third and last step, a suitable RNN architecture and a training algorithm must be selected. The samples database is divided into two parts: one of them is used to train the RNN and the other one to test its accuracy (in the *validation* phase of the process). Once trained and validated, the RNN will be able to provide an estimation of the perceived quality for any given values of the parameters. Formally, the trained RNN is seen as a function f having P real variables and with values in $[M_{\min}, M_{\max}]$, such that

(i) for any sample $\sigma_s \in \mathcal{T}$, $f(v_{1s}, \cdots, v_{P,s}) \approx \psi_s$,

(ii) and such that for *any other* vector of parameter values $(v_1, \cdots, v_P)$, $f(v_1, \cdots, v_P)$ is close to the MOS that would be associated in a subjective evaluation with a media sample for which the selected parameters had those specific values $v_1, \cdots, v_P$.

The validation phase is intended to check that the critical condition expressed in (ii) holds. This is done by comparing the output of the RNN to the expected MOS values for a set of configurations which were not used during training. If the validation tests were to fail (which has never occurred yet in the tests we have carried out), it could be due to several reasons. It could be, for example, due to an insufficient number of training samples, which would mean that more subjective tests are needed. Another cause could be low quality subjective results, although the use of statistical screening makes this option less likely. Finally, the problem could be related to the sampling performed on the configuration space when selecting the configurations to be used. This last option would also imply that more subjective tests need to be done.

Once the three steps described above are completed successfully, we implement the final tool, which is composed of two modules: the first one collects the values of the selected quality-affecting parameters (based on the network state, the codec parameters, etc.). These values are fed into the second one, which is the trained RNN that will take the given values of the quality-affecting parameters and correspondingly computes the MOS estimation.

5.3 On the RNN Model

Let us briefly describe the mathematical structure of the RNN model [42, 50, 51, 53]. An RNN is an open Markovian queuing network with positive and negative customers. These models are also called G-networks [42, 43, 45]. RNNs have been successfully applied in many fields (see [9] for a survey). In particular, we can cite [30, 31, 49] for applications related to networking, or the more recent developments about cognitive packet networks as in [46–48, 52]. The RNN we use here is a particular case of a G-network. Each neuron behaves as a $./M/1$ queue processing units in, say, FIFO order (even if this is not actually essential). The service rate at neuron i is denoted by μ_i and, after leaving neuron (queue) i, a customer either leaves the network with probability d_i, goes to queue j as a positive customer with probability

p^+_{ij} or as a negative customer with probability p^-_{ij}. Customers arrive from the environment at neuron i as positive ones, according to a Poisson process with rate λ^+_i. In our RNN, no negative customers arrive to the network. If a customer goes from i to j as a negative one, when it arrives at queue j it kills the last customer at the queue (if any), and it kills itself in all cases. Transfers between queues are, as usual with queuing network models, instantaneous. The previously described dynamics imply that at any point in time, there are only positive customers in the queues. It has then been shown in [43, 45] that when the associated vector Markov process $\vec{N}_t = (N^1_t, \cdots, N^M_t)$, where N^i_t is the number of customers in queue i at time t and M is the number of nodes, is stable, its distribution is of the product-form type: in equilibrium, that is, in the stationary case,

$$\Pr(\vec{N}_t = (k_1, \cdots, k_M)) = \prod_{i=1}^{M} (1 - \varrho_i)\varrho_i^{k_i}.$$

Factors ϱ_i are, as usual, loads in the queuing terminology: ϱ_i is the probability that queue i is not empty (under the stationary distribution). When the system is seen as a Neural Network, the backlog N^i_t of queue i at time t is called the *potential* of neuron i at t.

We use the network as a statistical estimator (it "learns" how to evaluate the quality of audio signals as humans do). As an estimator, its input is the vector $\vec{\lambda} = (\lambda^+_1, \cdots, \lambda^+_M)$ and the output is the vector $\vec{\varrho} = (\varrho_1, \cdots, \varrho_M)$, or just some elements of this vector, as we will see later. It is important not to confuse the output of the RNN as a learning tool (the vector $\vec{\varrho}$), with the output flow of customers of the G-network. That is, as an open queuing network, we would usually consider the output process as the flows of customers going out of the set of nodes; for instance, the mean rate of customers leaving the network from neuron i is $\varrho_i \mu_i d_i$.

The loads are obtained by solving a non-linear system of equations built using the data composed of the service rates and the routing probabilities. If we denote $w^+_{ij} = \mu_i p^+_{ij}$ and $w^-_{ij} = \mu_i p^-_{ij}$, the non-linear equations to solve are

$$\varrho_i = \frac{\lambda^+_i + \sum_{j=1}^{M} \varrho_j w^+_{ji}}{\mu_i + \sum_{j=1}^{M} \varrho_i w^-_{ij}}, \quad i = 1, \cdots, M \tag{5.1}$$

The w^*_* factors are called *weights* as in the standard neural network terminology, and they play the same role. For our application, we need the RNN to output a single scalar value, which corresponds to the perceived quality for any given parameter configuration. Let us denote the quality value for configuration γ_i as ψ_i. We will only consider one output neuron, o, and so the perceived quality will be given by its load ϱ_o. More specifically, our training data is composed of the set of K pairs $(\vec{v_s}, \psi_s)_{s=1,\cdots,K}$, with $\vec{v_s} = (v_{1s}, \cdots, v_{I,s})$. What we want is that the G-network behave in the following way: it must have I input neurons, one output neuron, and when the rate of the arrival flow (of positive customers) at neuron i is $\lambda^+_i = v_{i,s}$, then the occupation rate of neuron o is $\varrho_o \approx \psi_s$. To do this, consider the output ϱ_o as a function of the set of weights, that we denote here by $\vec{w}$, and of the input rates $\vec{\lambda^+}$, and write $\varrho_o(\vec{w}, \vec{\lambda^+})$. Then, look for a set of weights $\vec{w}_0$ defined by

$$\vec{w}_0 = \text{argmin}_{\vec{w} \geq \vec{0}} \; \frac{1}{2} \sum_{s=1}^{S} (\varrho_o(\vec{w}, \vec{v_s}) - \mu_s)^2$$

the minimization taking place on the set of positive multidimensional vectors $\vec{w}_*^*$. This optimization problem can be solved used standard techniques as gradient descent[1]. See Section 5.3.1 for details on this training algorithm.

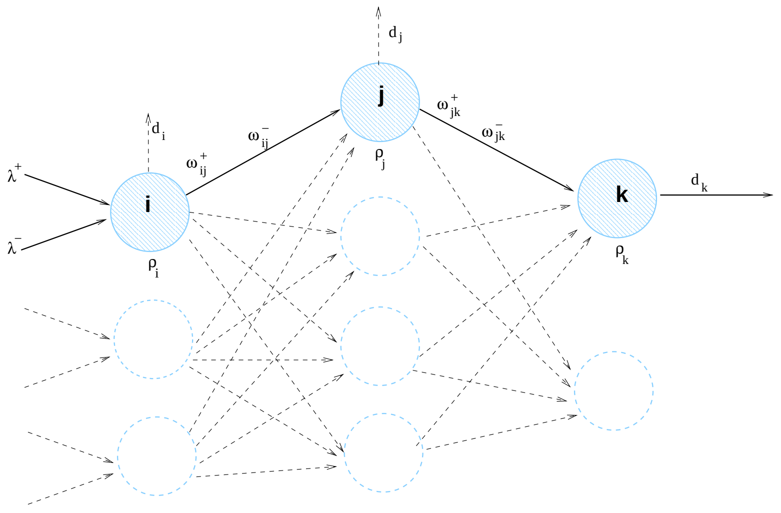


Figure 5.1: Components of a Random Neural Network. The schematics show a portion of a 3–layer feed-forward network.

An interesting observation for the reader familiar with standard Neural Networks, is that the "knowledge" of the RNN, once trained, is stored in the set of weights, as in the standard case, but here the weights are mean frequencies of the signals sent by a neuron to the others, which appears to be closer to the functioning of real nervous systems.

In learning applications such as ours, it is usual to use *feed–forward* topologies (in the associated G-network, a customer cannot visit more than once any given queue; see Figures 5.1 and 5.2 for examples of this kind of architecture). We use this class of model, for which the output is a rational fraction in the input variables (and also in the weights), that is, it is the ratio between two multi-variate polynomials in the variables $\lambda_1^+, \ldots, \lambda_M^+$. This means that its computation is straightforward, and the same for its derivatives. This in turn leads to efficient

[1]Observe that we are able to compute any partial derivative of the output, using the non-linear system of equations satisfied by the occupation rates.

learning procedures. For details, see [42–45, 50].

In our RNN there are I neurons receiving positive customers from the environment. Let us call $\mathcal{I}$ this set of neurons. Each external flow corresponds to one of the parameters selected for being part of the analysis, and λ_i^+ is the i^{th} parameter's value. There is only one neuron, denoted here by o, sending customers out of the network. Between this output element and the set of input neurons $\mathcal{I}$, there is a set of H neurons, called the *hidden layer*, receiving flows from the set $\mathcal{I}$ and sending customers to neuron o. Finally, there is no connection between $\mathcal{I}$ and o. We have that for all $i \in \mathcal{I}$, $\varrho_i = \lambda_i^+/\mu_i$, then, for all $h \in \mathcal{H}$,

$$\varrho_h = \frac{\displaystyle\sum_{i\in\mathcal{I}} \frac{\lambda_i^+}{\mu_i} w_{ih}^+}{\displaystyle\mu_h + \sum_{i\in\mathcal{I}} \frac{\lambda_i^+}{\mu_i} w_{ih}^-}$$

and finally

$$\varrho_o = \frac{\displaystyle\sum_{h\in\mathcal{H}} \varrho_h w_{ho}^+}{\displaystyle\mu_o + \sum_{h\in\mathcal{H}} \varrho_h w_{ho}^-},$$

the stability conditions being that for every neuron j we have $\varrho_j < 1$. Without loss of generality, we scale so that for every $i \in \mathcal{I}$ we have $0 \le \lambda_i^+ \le 1$. Then, a sufficient condition for stability is that for any input neuron $i \in \mathcal{I}$, $\lambda_i^+ < \mu_i$, that for any hidden neuron $h \in \mathcal{H}$ we have $\mu_h \ge I$ and, finally, that we have $\mu_o \ge I$. In practice, we observe that the number of iterations needed in the learning phase is sensitive to the effective values of these parameters, and that often much smaller values are used. As a last observation, observe that once trained, the computation of the network output is a fast process: it involves exactly $2HI + 3H + I + 1$ products and sums.

5.3.1 Training the RNN

The method we used to train our RNN is based on a gradient descent algorithm, and is described by Gelenbe in [44]. This algorithm can be used with any RNN, not only the feed–forward type described above with only one output neuron and limited exchange of signals with the environment. In this general context, Equation (5.1) can be written as

$$\varrho_i = \frac{N_i}{D_i}, \ i = 1, \cdots, M, \tag{5.2}$$

where

$$N_i = \lambda_i^+ + \sum_{j=1}^{M} \varrho_j w_{ji}^+,$$

and

$$D_i = \lambda_i^- + \mu_i + \sum_{j=1}^{M} \varrho_j w_{ji}^-$$

The training proceeds as follows. Let L be a set of K input / output pairs $(\vec{\iota}, \vec{\psi})$ that will be used to train the RNN. The set of inputs is $\vec{\iota} = \iota_1, \cdots, \iota_K$, with $\iota_k = (\vec{\lambda}^{+(k)}, \vec{\lambda}^{-(k)})$, and $\vec{\lambda}^{+(k)}$ and $\vec{\lambda}^{-(k)}$ being respectively the sets of positive and negative signal flow rates arriving at each of the M neurons, for input k from the environment. The set of outputs are the vectors $\vec{\psi} = (\psi_1, \cdots, \psi_K)$, with $\psi_k = (\psi_1^{(k)}, \cdots, \psi_M^{(k)})$. Each $\psi_i^{(k)} \in [0,1]$ corresponds to the output value of each neuron. We want the RNN to estimate the desired outputs in such a way that minimizes the error function

$$E_k = \frac{1}{2}\sum_{i=1}^{M} a_i(\varrho_i - \psi_i^{(k)})^2, \;\; a_i \geq 0$$

We use a gradient descent algorithm to compute, for each input $\vec{\iota}$, new values for two weight matrices $W^{+(k)} = w_{ij}^{+(k)}$ and $W^{-(k)} = w_{ij}^{-(k)}$. The value of each weight $w_*^{*(k)}$ needs to be positive. The rule used to update the weights is as follows:

$$w_{uv}^{*(k)} = w_{uv}^{*(k-1)} - \eta \sum_{i=1}^{M} a_i(\varrho_i^{(k)} - \psi_i^{(k)})[\partial\varrho_i/\partial w_{uv}^*]_k, \tag{5.3}$$

where η is an arbitrary constant, sometimes called the *learning rate*, and $\varrho_i^{(k)}$ is calculated as in Equation (5.2), with $\lambda_i^* = \vec{\lambda}_i^{*(k)}$, and $w_{uv}^* = w_{uv}^{*(k-1)}$.

The partial derivatives $[\partial\varrho_i/\partial w_{uv}^*]_k$ are evaluated with $\varrho_i = \varrho_i^{(k)}$ and $w_{uv}^* = w_{uv}^{*(k-1)}$.

In order to compute $[\partial\varrho_i/\partial w_{uv}^*]_k$ we derive from Equation (5.2):

$$\begin{aligned} \partial\varrho_i/\partial w_{uv}^+ &= \sum_j \partial\varrho_i/\partial w_{uv}[w_{ji}^+ - w_{ji}^-\varrho_i]/D_i \\ &\quad -1[u=i]\varrho_i/D_i + \varrho_u/D_i \\ \partial\varrho_i/\partial w_{uv}^- &= \sum_j \partial\varrho_i/\partial w_{uv}[w_{ji}^+ - w_{ji}^-\varrho_i]/D_i \\ &\quad -1[u=i]\varrho_i/D_i - \varrho_i\varrho_u/D_i \end{aligned}$$

Let $\vec{\varrho} = (\varrho_1, \cdots, \varrho_M)$, and let

$$W = [w_{ij}^+ - w_{ij}^-\varrho_i]/D_j, \;\; i,j = 1,\ldots,M$$

be a M matrix. We can then write the following vector equations:

$$\begin{aligned}
\partial\vec{\varrho}/\partial w^+_{uv} &= \partial\vec{\varrho}/\partial w^+_{uv} W + \vec{\varphi}^+_{uv}\varrho_u \\
\partial\vec{\varrho}/\partial w^-_{uv} &= \partial\vec{\varrho}/\partial w^-_{uv} W + \vec{\varphi}^-_{uv}\varrho_u
\end{aligned}$$

where $\vec{\varphi}^+ = (\varphi^+_{1\,uv}, \cdots, \varphi^+_{M uv})$ and $\vec{\varphi}^- = (\varphi^-_{1\,uv}, \cdots, \varphi^-_{M uv})$ are defined as follows:

$$\begin{aligned}
\varphi^+_{i\ uv} &= \begin{cases} -1/D_i & \text{if } \ u = i, \quad v \neq i, \\ 1/D_i & \text{if } \ u \neq i, \quad v = i, \\ 0 & \text{otherwise;} \end{cases} \\
\varphi^-_{i\ uv} &= \begin{cases} -(1+\varrho_i)/D_i & \text{if } \ u = i, \quad v = i, \\ -1/D_i & \text{if} u = i, \quad v \neq i, \\ -\varrho_i/D_i & \text{if } \ u \neq i, \quad v = i, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

We then have

$$\begin{aligned}
\partial\vec{\varrho}/\partial w^+_{uv} &= \vec{\varphi}^+_{uv}\varrho_u[I - W]^{-1} && (5.4) \\
\partial\vec{\varrho}/\partial w^-_{uv} &= \vec{\varphi}^-_{uv}\varrho_u[I - W]^{-1} && (5.5)
\end{aligned}$$

The learning algorithm is then:

1. Initialize the weight matrices W^+_0 and W^+_0, for example with random (positive) values, or with more appropriate ones if they are available. Choose a value for the learning step η.
2. For each $k \in 1, \ldots, K$ set the input values to $\iota_k = (\vec{\lambda}^+_k, \vec{\lambda}^-_k)$, and repeat the following three steps.
3. Calculate $\vec{\varrho}$ as per Equation (5.2)
4. Use the results of the previous step to solve the system in Equation (5.4)
5. Use the results of the previous steps in Equation 5.3 to update the weight matrices. If any updated weight has a negative value, then
   - set the weight to zero, and stop iterating on it for step k, *or*
   - reset it to its previous value, and repeat the last step with a smaller value of η

The algorithm can either be stopped after an arbitrary number of iterations, or when the measured error is smaller than a certain threshold.

As a final note, recall that in the general case,

$$\sum_j (w^+_{ij} + w^-_{ij}) = \sum_j \mu_i(p^+_{ij} + p^-_{ij}) = \mu_i(1 - d_i)$$

which leaves us with an extra degree of freedom. For our application, we consider that all the input and hidden neurons do not emit clients to the environment (hence $\sum_j (w_{ij}^+ + w_{ij}^-) = \mu_i,\ i \in \mathcal{I} \cup \mathcal{H}$), and the output neuron emits all of its clients to the environment (thus we have $d_o = 1$ and μ_o is fixed to an arbitrary value).

5.4 Performance of PSQA using RNN

In this section we present performance results for PSQA in the context of one of the testing campaigns we performed for VoIP. The experiments themselves are described in detail in Section 6.1.1. Our performance evaluation is based on the subjective scores we obtained for the 112 configurations used for this test. We considered six input parameters, and produced an MOS estimation based on them.

We experimented with several neural network architectures, and with different sizes of training databases. The architectures used for the networks ranged from classical 3–layer designs (Figure 5.2) to more complex ones with several hidden layers and feedback between neurons, to extremely simple ones, with only an input layer and an output neuron (see Section 5.4.1 below for more on this). We didn't observe significant variations in performance by modifying the architecture. However, the complex architectures showed a small advantage in convergence over the simpler ones during the first hundred iterations of the training process (using the algorithm described earlier in Section 5.3.1). Figure 5.3 shows an example of these differences. Convergence time is not really important for this type of application, as the training is only done once. We don't rule out, however, that better performances (both in the learning process and in the estimation) might be obtained by taking into account the possible relations between different parameters (e.g. loss rate and loss mean burst size) when designing and training the neural network. The performance results presented here are for a 3–layer feed–forward design, with six input neurons, 13 neurons in the hidden layer, and one neuron in the output layer (see Figure 5.2).

As usual when working with neural networks, the available data is divided into two subsets, one of which is used to train the network, and the other one is used to validate it. We studied the performance of the RNN when trained and validated with different fractions of the available data. Figure 5.4 shows the results given by a 3–layer network trained with 92 configurations when faced with the 20 configurations reserved for its validation phase. We also compared the results obtained to some of the human subjects' evaluations, and found the RNN estimation's distance[2] from the MOS to be generally smaller than those of the subjects.

We obtained similar results when using only 56 configurations to train the network, and another 56 to test it. The mean square error for that case was very similar ($\approx$0.23), but there were some outliers.

[2]Measured as the mean square error.

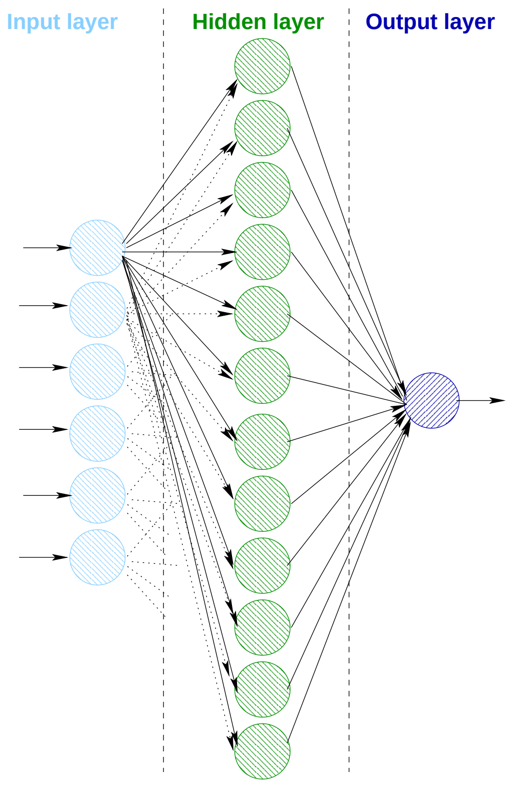


Figure 5.2: Example of a 3–layer RNN architecture (simplified schematics).

As for the correlation coefficient, we found it to be dependent on the number of configurations used for the testing, and it was in the 0.71–0.93 range (the lowest value being for a 56–configuration training set, and the highest one for a 102–configuration one). We found that the performance of the RNN was acceptable (correlation coefficients of about 0.75–0.8 on average) even when using a relatively small number of training configurations, which is interesting, since the main cost of our approach resides in the number of subjective evaluations to be made. In order to give an idea of this cost, it is worth noting that this subjective campaign implied having each subject test 115 configurations, at four 10–second samples per configuration. Several 30 to 40–minute sessions were needed for each subject to perform the assessment. Besides the assessment itself, the test setup and generation of the samples took about one week of work. For other kinds of subjective assessment, for instance in the case of interactive streams, the assessment itself is more complex to set up and perform.

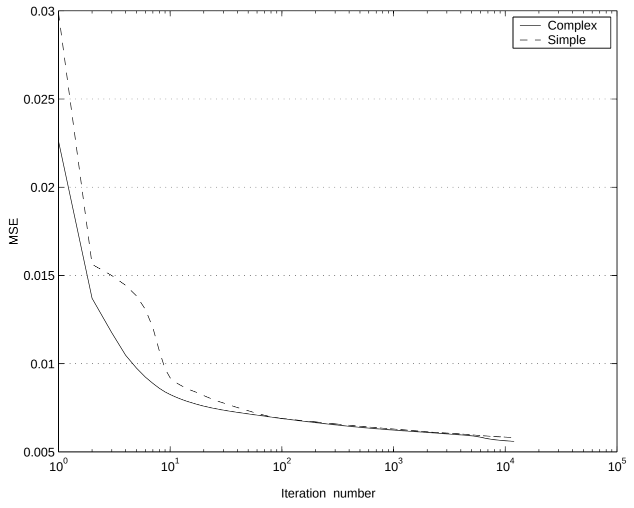


Figure 5.3: Convergence speed for two different neural network architectures. The simple one is a 3–layer (6/13/1) feed–forward RNN, and the complex one is a 3-layer RNN with a fully recurrent hidden layer.

5.4.1 Going Minimal: a 2–Layer RNN Architecture

As mentioned above, among the different RNN architectures that we tried, there was a two–layer design consisting only of six input neurons, and one output neuron, as shown in Figure 5.5. Surprisingly, we didn't find any significant difference with the performances of the other architectures we used for our application. We find this very interesting, since it allows to represent the quality of the speech stream with a relatively simple formula, obtained by substituting our coefficients in the RNN general expression. Figure 5.6 shows its performance when estimating MOS values for the validation database, and the difference with a 3–layer RNN's estimation.

In Section 7.2.3 we use this RNN architecture to derive a simple closed–form expression for the perceived quality as a function of network load and capacity.

5.4.2 Robustness of the Estimation

One important aspect when analyzing the performance of PSQA is whether the results it provides can be used in a wide range of conditions, even those that were not foreseen when designing an instance of the tool. For example, is it right to assume that subjective assessments performed in one location are sufficiently general so that the application of a PSQA–based tool trained with them will be also valid in another location? Or, does the use of a different codec, not previously considered, have a significant impact on the way the perceived quality varies with the other parameters?

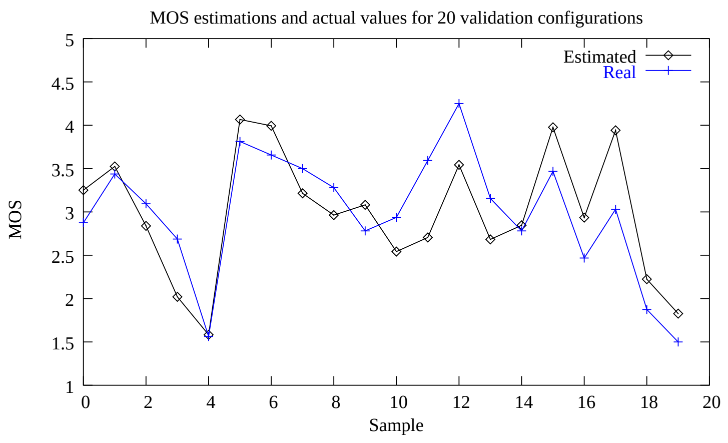


Figure 5.4: Estimations for 20 validation (randomly chosen) configurations. Note than an error of less than 0.5 MOS points is barely perceptible, and unlikely to get noticed by users.

We would like to have a tool as generic as possible, so that it can be applied in a wide variety of contexts without needing to perform further subjective assessments, which imply an increased cost.

We have tried to answer these questions with the data we had available from previous experiments, in which different parameters, with values in different ranges were used. We also performed a new subjective assessment for the samples in our campaign, performed by only one listener who had previous experience, and knew the details about the experiment. In this section we present the results from both of these performance evaluations.

5.4.2.1 Cross–validation with Data from a Previous Campaign

We used a RNN trained with results from our VoIP testing campaign to estimate MOS scores for configurations from a previous one, done by Mohamed for [98]. The parameters considered in the old campaign can be found in Table 5.1.With these parameters, the network effects were simulated on encoded files, and the resulting degraded files were used to conduct subjective evaluations (as specified in [85]) in three languages (French, Spanish and Arabic). Once the MOS results were screened, the PSQA technique was applied as described in Section 5.2, resulting in three trained RNN, one for each language considered.

For the second set of tests, we refined our network model, substituting the simple loss model used in the first campaign (the one suggested in [58], which considers loss bursts of constant size) with a simplified Gilbert one, as suggested in [17, 135]. Also, instead of simulating the network impairments as in the previous experiments, the distorted speech samples were generated on a live network (see Section 6.1.1 for details on the experimental setup). The parameters used for our experiment are listed on table 5.2.

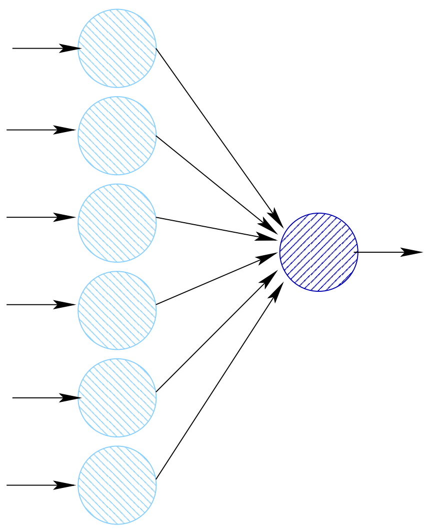


Figure 5.5: The simplest RNN architecture for PSQA: six input neurons and one output neuron, with no hidden layer.

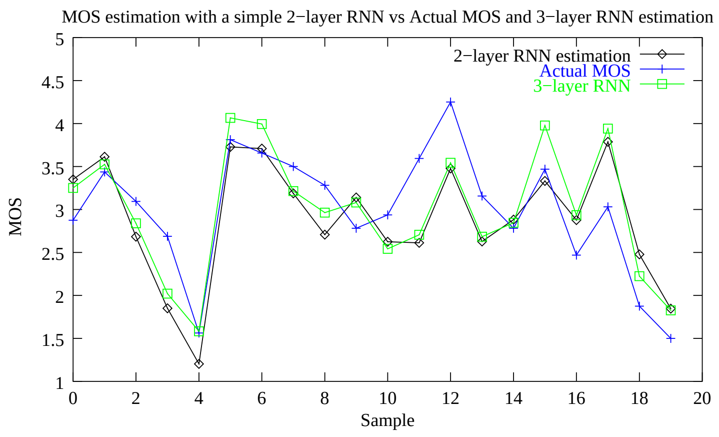


Figure 5.6: Performance of the 2–layer RNN for 20 validation configurations. Despite its simplicity, it provides a very good approximation of the MOS.

Parameter	**Values**
Loss rate	0%...40%
Loss burst size (constant, as in [58])	1...5
Codec	PCM Linear-8, ADPCM (G.726) and GSM
Packetization interval	20, 40, 60 and 80ms

Table 5.1: Network and encoding parameters and values used for the first test set

Parameter	**Values**
Loss rate	0%...15%
Mean loss burst size	1...2.5
Codec	PCM Linear 16 bits, GSM
FEC	ON(GSM)/OFF
FEC offset	1...3
Packetization interval	20, 40, and 80ms

Table 5.2: Network and encoding parameters and values used for the second test set

Both testing campaigns have several differences, including the handling of network impairments, the ranges used for loss rate and mean loss burst sizes, the codecs used, the people performing the subjective assessments, and the tools used. As seen in both Tables 5.1 and 5.2, the ranges for both the loss rate and the mean loss burst size were significantly shorter in the second campaign. We have therefore decided to perform this cross–validation study in two separate phases. In the first one, we only tested configurations that fell within the ranges considered for the second campaign, so that the RNN would not need to extrapolate outside its training space. As in the first campaign FEC had not been considered as a parameter, we set the FEC inputs of the RNN to indicate that FEC is not being used. As for the codecs, our RNN did not know about ADPCM, but being that it is in the same family of codecs as PCM, we set the codec input to PCM for configurations that used ADPCM.

Figures 5.7 through 5.10 show the comparison between the cross–estimated MOS, the actual MOS and the MOS estimated by the original RNN in the first campaign for the PCM and GSM codecs. The results indicate that the RNN trained during the second campaign is a bit pessimistic with respect to the actual scores and the original prediction. However, it is interesting to see that the curves behave very similarly, and that a simple adjustment (for instance, adding between 0.5 and 1.0 MOS points to the estimation) could be enough to obtain a much better fit. Even more interesting are the results for the ADPCM codec, which was unknown to the second RNN. Figures 5.11 and 5.12 show that, while the estimation is rougher, the RNN from the second campaign still shows a similar behavior, and a similar adjustment to be made.

For the second phase of the cross-validation, we did not limit the ranges for the parameters' values, and so the RNN from the second campaign had to extrapolate MOS values for configurations well beyond its training space. Figures 5.13 through 5.18 show the performance of

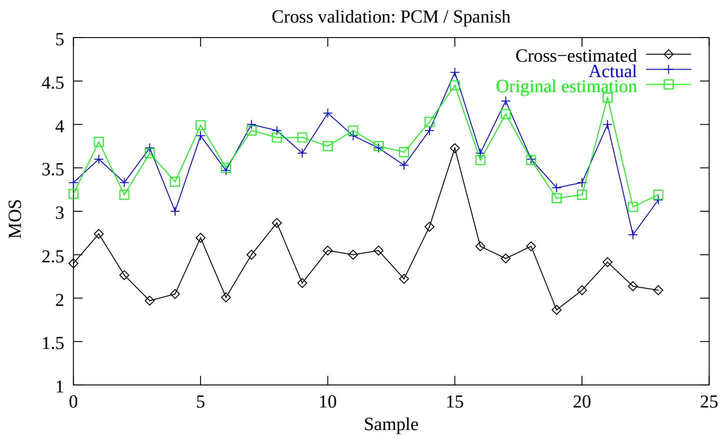


Figure 5.7: First phase of cross–validation (configurations within the same ranges) for the PCM codec, Spanish language samples.

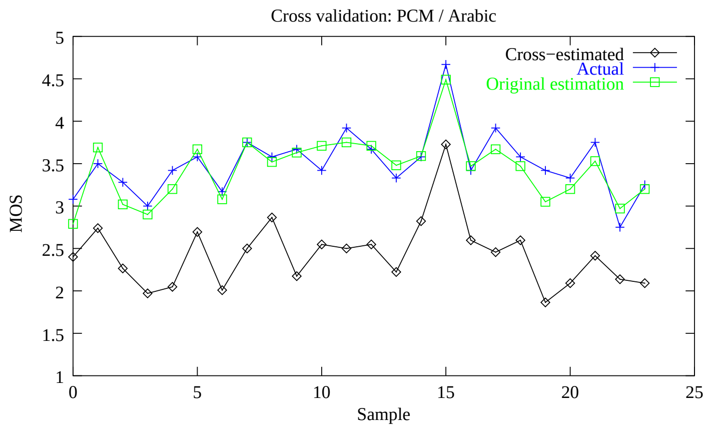


Figure 5.8: First phase of cross–validation (configurations within the same ranges) for the PCM codec, Arabic language samples.

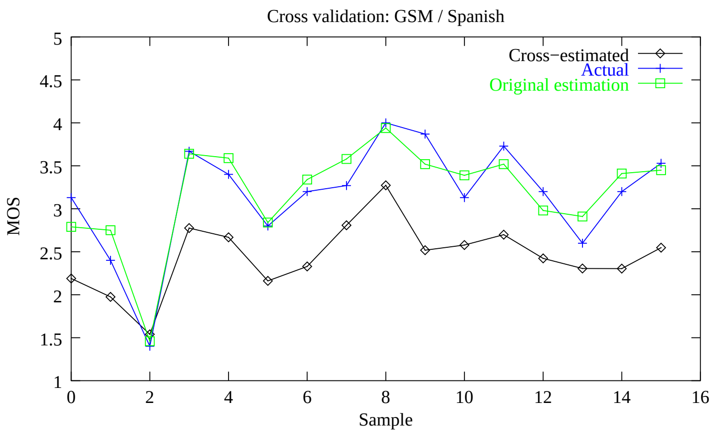


Figure 5.9: First phase of cross–validation (configurations within the same ranges) for the GSM codec, Spanish language samples.

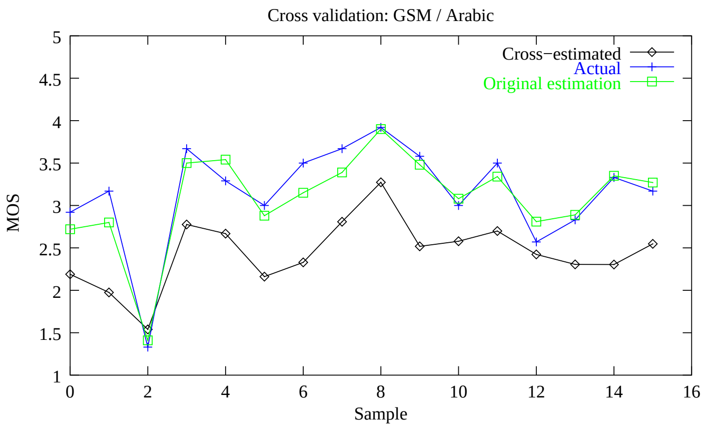


Figure 5.10: First phase of cross–validation (configurations within the same ranges) for the GSM codec, Arabic language samples.

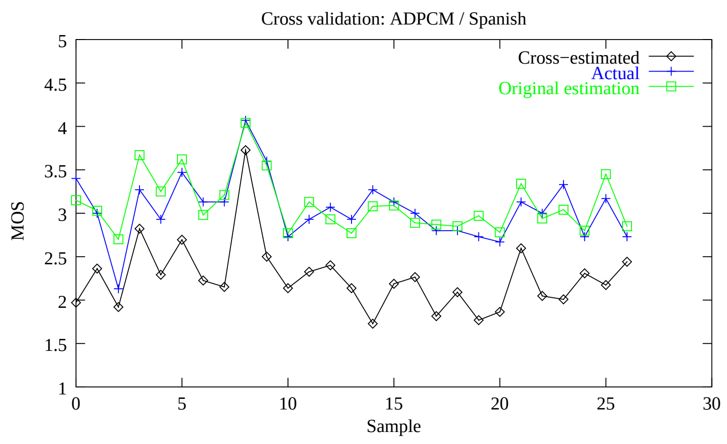


Figure 5.11: First phase of cross–validation (configurations within the same ranges) for the ADPCM codec, Spanish language samples.

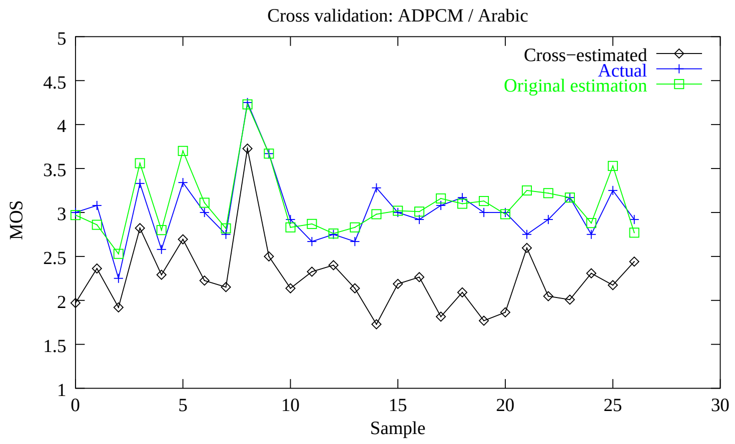


Figure 5.12: First phase of cross–validation (configurations within the same ranges) for the ADPCM codec, Arabic language samples.

the RNN for the second campaign for this phase. As expected, its performance is not as good as for the first phase, since the RNN has to extrapolate MOS values for loss rates well outside its training space. This is especially noticeable for the PCM codec. For GSM and ADPCM, though, the cross–estimation is still quite good.

Although no conclusions can be drawn yet for the generality of the PSQA approach, the results obtained as of this writing are promising, since it would seem that the differences found could be solved with a simple shifting and / or scaling factor. Depending on the context in which the application was used, input from the user could be used to *tweak* the output of the RNN, in order to provide a finer tuning of the tool. In other contexts, the tuning could be performed by an expert listener, instead of performing an entirely new subjective assessment campaign.

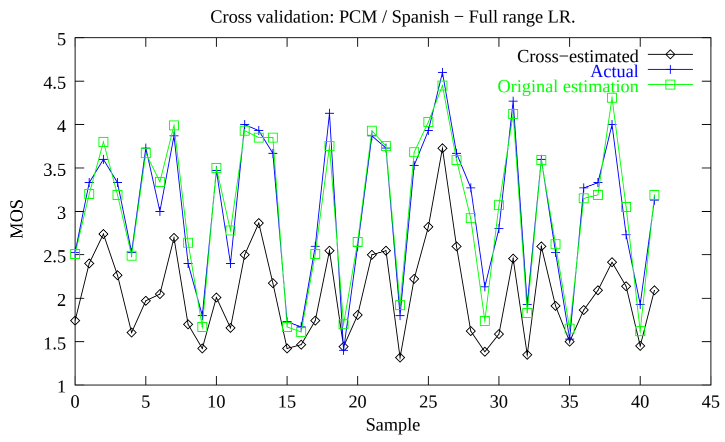


Figure 5.13: Second phase of cross–validation (values from the full ranges considered during the first campaign) for the PCM codec, Spanish language samples.

5.4.2.2 Cross–validation with a One–man Subjective Assessment

Given that the main cost of the PSQA technique resides in the subjective assessment that needs to be done, one issue that we find interesting is whether this assessment can be done with less people. This would allow to decrease the cost of each campaign, and also to do them more quickly.

We have performed a one–person subjective assessment of the degraded VoIP samples, and trained a new RNN with those results. The person who performed the assessments, while by no means a *golden ear*, was very familiar with the domain, and with the experimental setup. We have compared the results of this new RNN with the actual MOS values obtained in a large testing campaign, and with those provided by another RNN, trained with those subjective re-

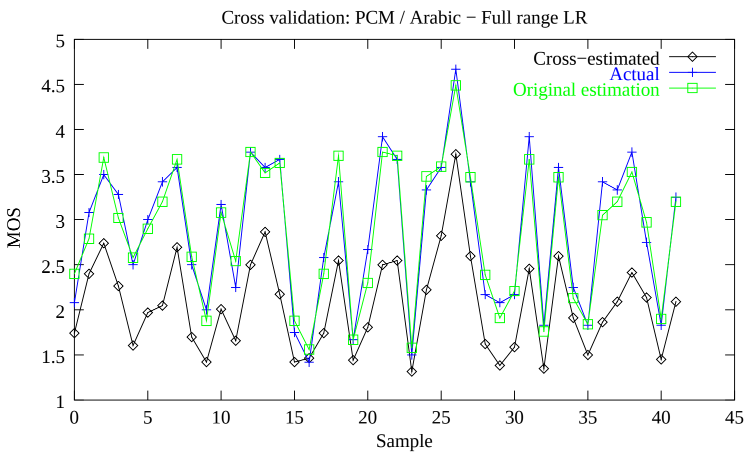


Figure 5.14: Second phase of cross–validation (values from the full ranges considered during the first campaign) for the PCM codec, Arabic language samples.

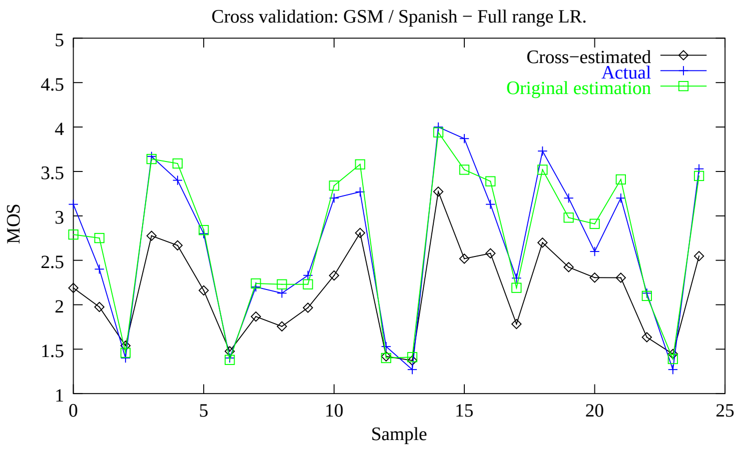


Figure 5.15: Second phase of cross–validation (values from the full ranges considered during the first campaign) for the GSM codec, Spanish language samples.

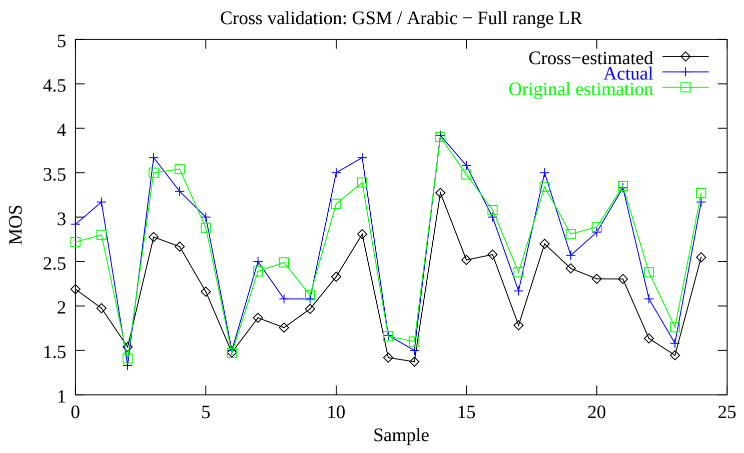


Figure 5.16: Second phase of cross–validation (values from the full ranges considered during the first campaign) for the GSM codec, Arabic language samples.

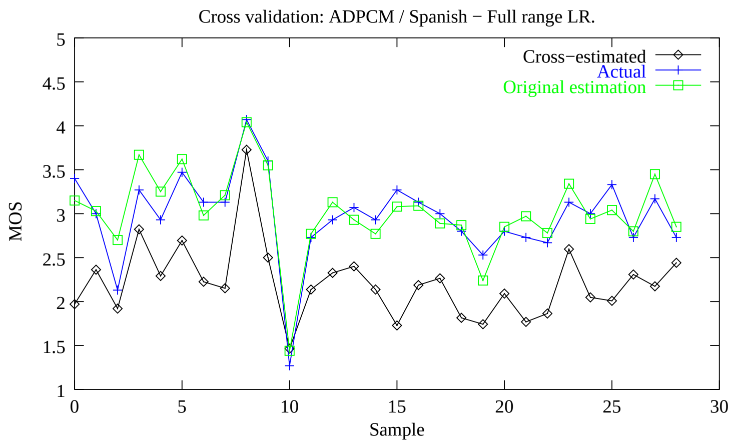


Figure 5.17: Second phase of cross–validation (values from the full ranges considered during the first campaign) for the ADPCM codec, Spanish language samples.

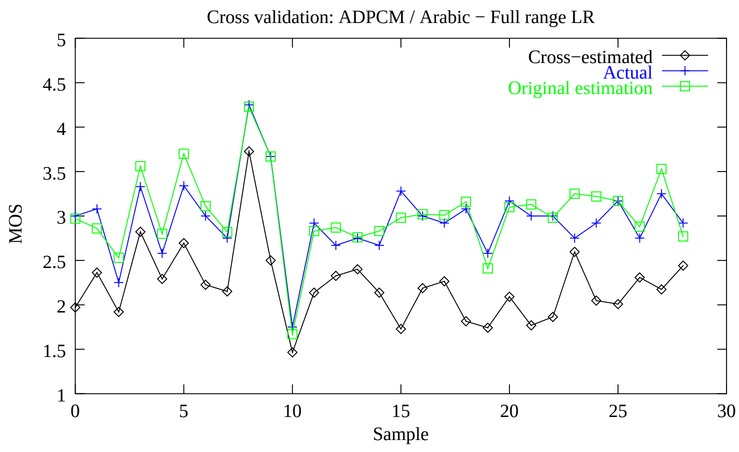


Figure 5.18: Second phase of cross–validation (values from the full ranges considered during the first campaign) for the ADPCM codec, Arabic language samples.

sults.

The comparison of the estimations given by both RNNs and the actual MOS values can be seen in Figure 5.19. As in the previous cross–validation experiment, there is a more or less constant shift in the predicted values, but otherwise, the curves behave in a very similar way. In this case, the shift can be explained by the familiarity of the listener with this kind of tests. While the tests were performed in a double–blind fashion as prescribed in the ITU recommendations, a listener who knows what to listen for can detect more impairments than another one who is not experienced.

The results obtained in both cross-validation experiences are, if not conclusive, very encouraging, since they show that

1. the results provided by PSQA are not strictly limited to situations identical to those considered when performing the initial calibration, and
2. one well–trained observer might be enough to produce a good estimation of the perceived quality, which can be used to fine–tune a previously trained RNN, to adapt it to new situations, or to perform preliminary tests in new environments.

5.5 Performance of other Objective Speech Quality Measures

In order to have an idea about the performance of the known objective speech quality measures (such as those that have been introduced in Section 4.2.1), we have collected some data and figures from the literature. We present these results in two parts: the first one shows the

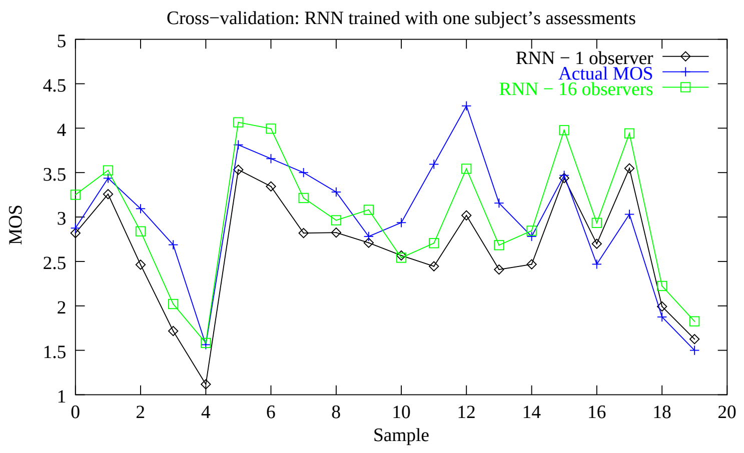


Figure 5.19: Cross validation of a RNN trained with subjective assessments from only one observer. We can see that while the estimations are more pessimistic of those of the original RNN trained with MOS values from a larger test, both curves behave in a very similar way.

performance of these metrics when assessing speech quality with only encoding impairments considered; the second one when they are used with both encoding and network impairments. Depending on the available data, the compared metrics are SNR, SNRseg, BSD, MBSD, EMBSD, PSQM, PSQM+, MNB(1,2), E-model, PESQ and PAMS.

Table 5.3 shows the correlation coefficients for the metrics considered (except PESQ) when assessing the impact of encoding, and MOS. It is easy to see that the simplest metrics like SNR and SNRseg yield relatively poor correlation with subjective quality tests (ranging from 0.22 to 0.52 for SNR and from 0.22 to 0.52 for SNRseg). BSD, MBSD and EMBSD give better results than SNR or SNRseg.

PSQM and its enhanced version PSQM+, exhibit better performances, comparable to that of PAMS and MNB. The correlation coefficient can reach up to 0.98 for certain metrics. The variation in the correlation values is due to different levels of distortion being considered.

However, when networking parameters are also taken into account (cf.Table 5.4), the performance of these metrics suffers a sometimes very significant drop. The most comprehensive comparative study of the performance of objective assessment metrics we found is the one in [136]. Note, however, that the authors did not disclose all the measured performances explicitly. This does not hinder our study, since we are interested in knowing the best performances for the other metrics, and so these values are sufficient. We can see that there is a general drop in performance when some form of network impairment is considered, even if these impairments are not necessarily those specific to VoIP applications (the impairments considered were temporal shifting, front clipping, bit errors, frame erasures and level variations, which

Objective Measures	Correlation with Subjective Measure (MOS)
SNR	0.22-0.52
SNRseg	0.22-0.52
BSD	0.36-0.91
PSQM	0.83-0.98
PSQM+	0.87-0.98
MNB2	0.74-0.98
PAMS	0.64-0.89
MBSD	0.76
EMBSD	0.87

Table 5.3: Correlation coefficient of some existing objective speech quality measures with MOS. Results are taken from the literature. Only the encoding impairments are considered. Sources are [136], [128], [113] and [111].

correspond better to wireless networks, such as GSM or CDMA). It should be noted that the correlation coefficients were calculated for regression curves used to fit the results and not for the metrics themselves, which may lead to a slightly higher correlation than that of the actual output of the metrics and MOS values.

Objective Measures	Correlation with Subjective Measure (MOS)
A	0.87
B	0.85
C	0.56
D	0.86
E	0.90
F	0.86
MBSD	0.24
EMBSD	0.54

Table 5.4: Correlation coefficient of some existing objective speech quality measures with MOS for encoding impairments and some network impairments which can be found on GSM or CDMA networks, and some of the effects found on IP networks. Source is [136]. The letters A to F represent some of the objective quality measures mentioned in Sec. 5.5. Note that not all the metrics evaluated were explicitly named in the study.

Table 5.5 shows results taken from [136] and [55], which give the correlation coefficients obtained for the listed metrics and MOS values when used with real VoIP traffic (the E-model values are taken from [55], since the E-model is not considered in [136]). The higher values correspond to the results obtained in [136]. The authors did not specify the network conditions used, but only stated that it was voice traffic recorded over a real network. In [55], the network parameters considered are loss rate with values 0%, 1% and 5% (the distribution of losses

Objective Measures	Correlation with Subjective Measure (MOS)
EMBSD	0.39 – 0.87
MNB1	0.61 – 0.83
MNB2	0.63 – 0.74
E-model	0.62 – 0.86

Table 5.5: Correlation coefficients for EMBSD, MNB(1 & 2) and E-model with MOS for VoIP impairments, taken from [136] and [55].

within the streams was not specified), and jitter values of 0, 50 and 100ms. We believe that the difference in performance of the different metrics for both tests is due to differences in network conditions (which may have been harsher in [55]), resulting in more damaged samples, which may have affected the metrics' performance.

In Table 5.6, we present results from [107], which compare the performance of PESQ, PAMS, PSQM, PSQM+ and MNB. These results show average and worst–case correlation coefficients for three different scenarios, namely a mobile and fixed telephone networks, and for VoIP. We can see that PESQ outperforms all of the other objective metrics, and that its worst–case correlation coefficient for VoIP is even a bit higher than the worst case we found for PSQA (which was of 0.71). However, PSQA does obtain even better correlation values (up to 0.99 in the first testing campaign, and up to 0.93 in the second one), and without needing the original sample to perform the estimation. Another interesting observation that can be made from Table 5.6, is the difference in the performance of these metrics when used in a telephone network, and on a VoIP context.

Network Type	PESQ	PAMS	PSQM	PSQM+	MNB
Mobile	0.905 – 0.962	0.895 – 0.954	0.843 – 0.924	0.859 – 0.935	0.731 – 0.884
Fixed	0.902 – 0.942	0.805 – 0.936	0.657 – 0.881	0.652 – 0.897	0.596 – 0.801
VoIP	0.810 – 0.918	0.758 – 0.916	0.260 – 0.674	0.469 – 0.726	0.363 – 0.690

Table 5.6: Correlation coefficients (worst–case – average) for five objective metrics. Taken from [107].

From the objective speech quality measures considered, only the ITU E-model does not need access to the original signal to compute the quality. Thus, it is the only available measure which is computationally simple [55] and can be used in real-time applications. However, as stated in its specification [81], and from the results reported in [55], the E-model was designed as a network planning tool, and not as a precise quality metric.

As a final example of the performance problems currently found in objective quality metrics, in Figures 5.20 and 5.21 we present two scatter plots (taken from [55]) for MNB2 and the E-model. In these plots, there is a very significant number of values that are clearly inconsistent

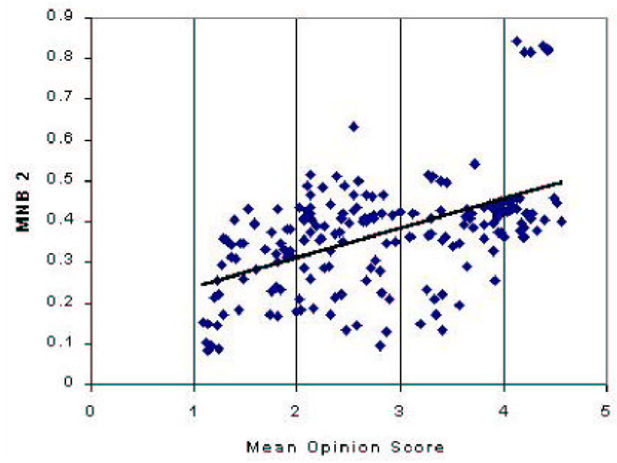


Figure 5.20: Scatter plot: MNB2 results versus actual MOS values on a set of speech samples distorted by both encoding and network impairments (from [55]).

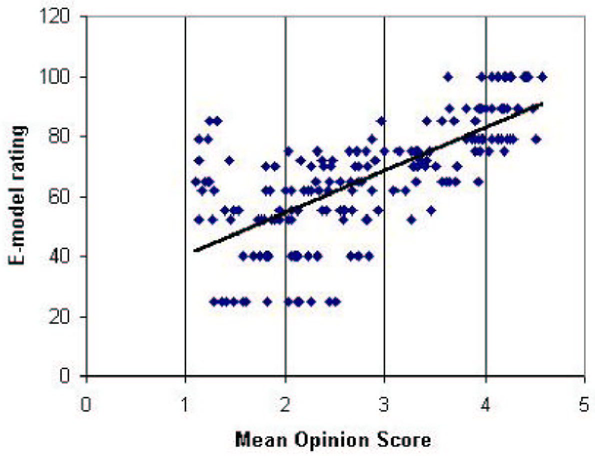


Figure 5.21: Scatter plot: E–model results versus actual MOS values on a set of speech samples distorted by both encoding and network impairments (from [55]).

with MOS results. For example, there are points that correspond to the same values of MNB or the E-model R rating, but have very different MOS values (there are differences of 2 and 3 MOS points, which are highly significant), and vice-versa.

5.6 Comparing PSQA to the Other Approaches

In this section we present the results we obtained with our approach for two different VoIP test campaigns we have performed. For the first battery of tests [98] the parameters considered were as listed in Table 5.1. The results obtained were very good, with correlation coefficients of 0.99 for Spanish and Arabic, and 0.98 for French (using only validation data). Figures 5.22 through 5.24 show scatter plots for these tests.

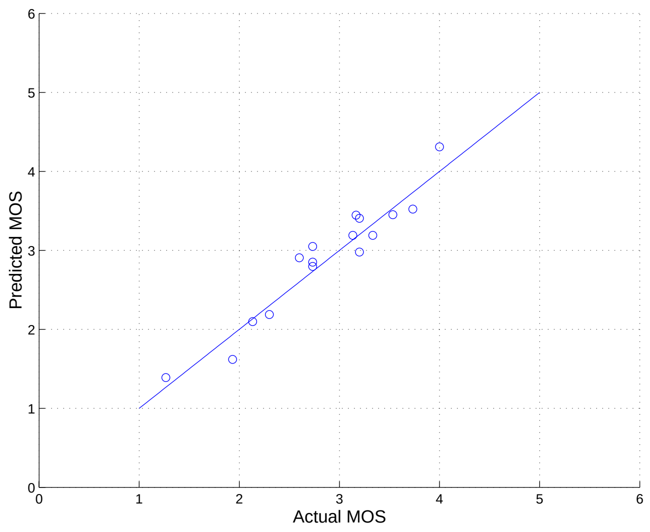


Figure 5.22: Scatter plot: (Spanish samples – Correlation Coefficient = 0.99). Estimations are for validation data (never seen before by the RNN).

For the second VoIP campaign, the results obtained varied with the different sizes of training/validation databases, and yielded correlation coefficients between 0.73 and 0.93 with actual MOS values. It is interesting to see that even when using relatively small sets of training samples, good results can still be obtained, and this allows for a trade-off between cost and performance for our method (since its main cost is that of performing the subjective tests to train the RNN). Figure 5.25 shows a scatter plot for the validation data of the second set of tests.

The results produced by PSQA are better (in terms of correlation with subjective assessments) than most of those produced by the objective metrics found in the literature. This is, by itself, a good reason to prefer this pseudo–subjective approach to at least some of the objective

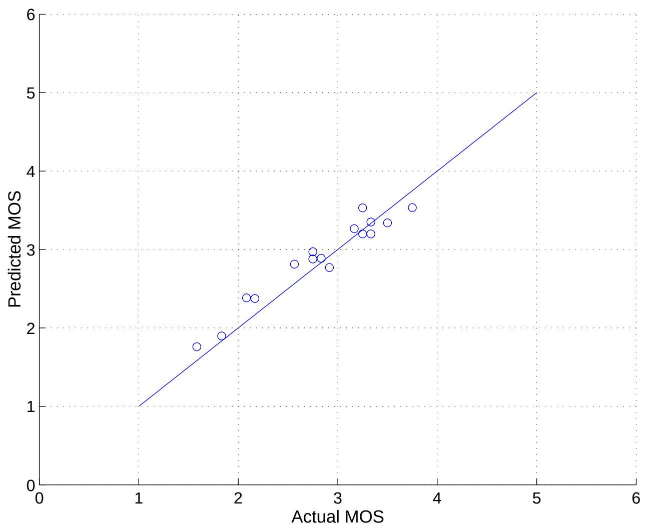


Figure 5.23: Scatter plot: (Arabic samples – Correlation Coefficient = 0.99). Estimations are for validation data (never seen before by the RNN).

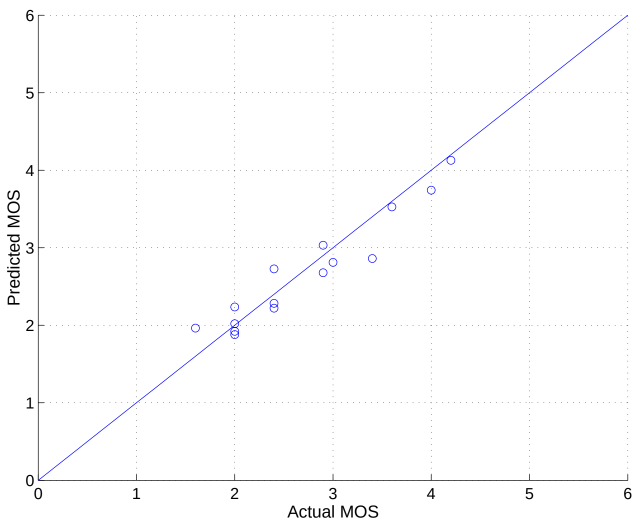


Figure 5.24: Scatter plot: (French samples – Correlation Coefficient = 0.98). Estimations are for validation data (never seen before by the RNN).

ones. However, what is even better, is that these results are provided without needing to access the original signal.

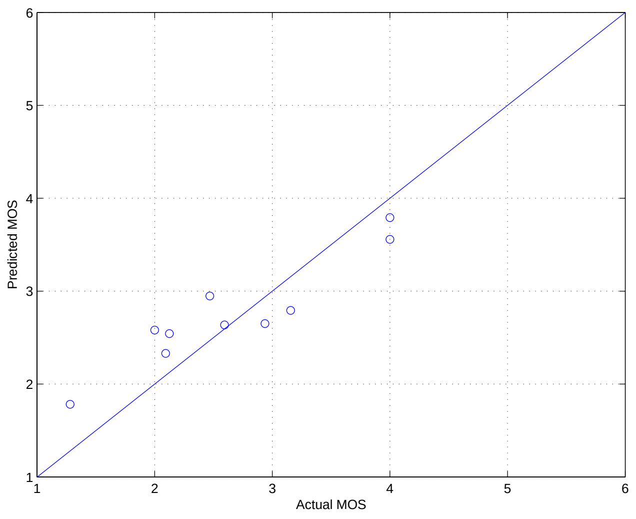


Figure 5.25: Scatter plot for the second series of tests – Correlation Coefficient = 0.93. Estimations are for validation data (never seen before by the RNN).

5.7 Other Statistical Learning Tools that Might Be Used

As mentioned in Section 5.1, PSQA needs a statistical estimator in order to work. The choice of the Random Neural Network model to fulfill this need was not arbitrary. In this section we briefly discuss two other possible tools to be used with PSQA, and their performances.

5.7.1 Artificial Neural Networks

The first work on what came to be PSQA, done by Mohamed [98], was based on Artificial Neural Networks. The results obtained with ANN were relatively good, but two problems were noticed (see Section 4.5.1 in [98] for a detailed performance comparison between ANN and RNN in this context):

- they have a tendency to suffer from over-training, i.e. they learn *too well* their training set, so their ability to generalize is impaired, and
- the results they provide depend heavily on the architecture used, which makes it necessary to try and find an optimal architecture for each set of data. This is not an easy task, and needs to be done by trial and error.

Figures 5.26 and 5.27 (taken from [98]) show examples of these problems. In Figure 5.26, we can see the effect of over-training an ANN, which leads to an uneven quality of the estimation, since points not in the training database (that is to say, most points that will be evaluated during actual use of the tool) will be badly estimated.

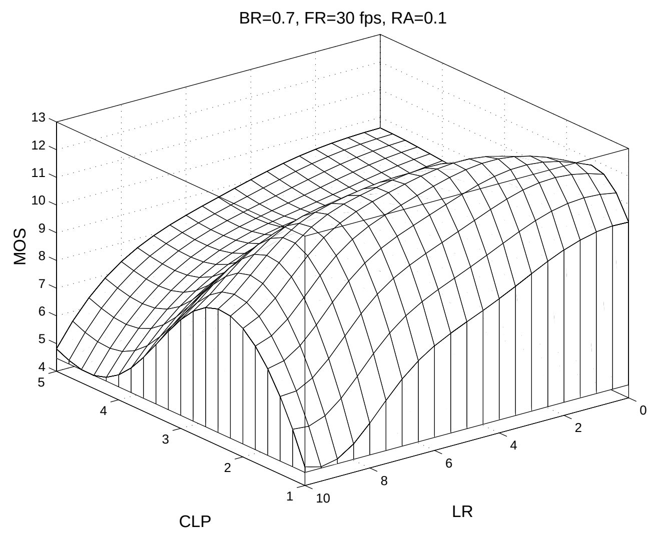


Figure 5.26: Quality of a video stream as a function of loss rate (LR) and mean loss burst size (CLP) as predicted by an over-trained ANN [98].

As for the second problem, Figure 5.27 shows the variations in the estimated quality when the number of neurons in the hidden layer of a 3–layer feed–forward ANN is modified.

5.7.2 Naive Bayesian Classifiers

We also tried using naive Bayesian classifiers in order to provide MOS estimations for PSQA. This was meant primarily to test their accuracy more than for production use, since two reasons make them less desirable than RNN:

- they require a larger amount of training data, and
- they only provide classification into discrete values, whereas RNNs provide a continuous, differentiable function.

However, this kind of classifier is easier to implement than RNNs, and is less computationally–intensive, which could be useful in certain contexts where computing power or memory are very limited.

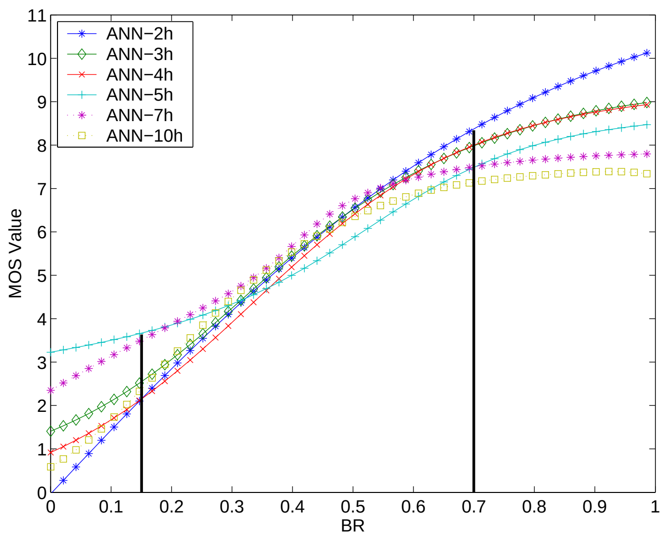


Figure 5.27: Quality of a video stream as a function of bit rate (BR, normalized values) for several ANN with different numbers of neurons in the hidden layer [98]. Note the significant differences in the estimated quality, as opposed to the small variations found for RNN, e.g. in Figure 5.6.

Naive Bayesian classification is based on the observation of a sufficiently large amount of data, and the assumption that those observations are statistically independent. We perform the classification by counting, for a large number of assessments, the number of times each score happens for each value of each parameter considered. This allows us to know, for each value of each parameter, which are the probabilities of each score happening (i.e. the conditional probability of a score being s given that value $v_i = \eta_i$, for each parameter in $1 \cdots P$). Assuming the parameters' values, and their relation with the scores, to be independent, we can then, for a configuration $\gamma = (v_1, \cdots, v_P)$, calculate the probabilities for each score s under that configuration, as:

$$\Pr(s) = \frac{\prod_{i=1}^{P} \Pr(s|v_i = \eta_i)}{\prod_{i=1}^{P} \Pr(s|v_i = \eta_i) + \prod_{i=1}^{P}(1 - \Pr(s|v_i = \eta_i))}$$

We can thus, given a configuration, find the score which is most likely to be associated with it.

As stated above, this approach needs a large body of data to be trained. As not enough subjective assessments were available to train the classifier, we generated a large set of assessments (covering the whole parameter space) with a previously trained RNN, and used them instead. We performed several tests, using over 20,000 configurations to train the classifier. For the first rounds of testing, we trained the classifier to grade configurations in a nine–point scale (from 1 to 5, with a 0.5 MOS point appreciation). We also used a coarser five–point scale in later experiments. The results given for validation data (which was a 500–item random subset of the training space) were reasonable ones, with 96% of the predicted scores being within 0.5 MOS points of the real score and 57% correct estimations for the nine–point classifier, and 82% of correct scores for the five–point one. However, when used to assess configurations for which we had actual MOS values from subjective assessments, the performance of the naive Bayesian estimators was clearly lower than that of RNN.

Among the possible explanations for this bad performance, the foremost is that the quality–affecting parameters, and their relation to MOS scores are not actually independent. Given the results obtained, it is reasonable to think that they are too correlated for the assumption of statistical independence to hold.

As mentioned before, even if this approach did perform well, it does not offer the advantages of using a RNN, and in any case, it needs the RNN (or another trained estimator) in order to be trained.

5.8 Summary

In this chapter we have described a novel real–time quality assessment technique for multimedia streams. We have gone beyond the original proposal by Mohamed [98], and performed an in–depth study of its performance in different situations.

We have shown that, while not completely universal, a well trained RNN–based PSQA tool can be used beyond the domain for which it was originally conceived, and still produce usable estimations. We have experimented with different RNN architectures, one of which simple enough that a short closed–form expression for quality can be derived from it, as described later, in Chapter 7.

The performance of PSQA has been compared (for voice streams) to that of several objective speech quality metrics, and the results show that PSQA outperforms most of them in the assessment of VoIP streams, and it gives estimations of similar quality to those of the best objective metric, PESQ. Moreover, it is able to do this estimation without needing to access the original signal, which allows it to perform real–time assessment of a stream. This is a very significant result, since the only objective metric that can perform real–time assessment is the E–model, whose results do not correlate well with subjective ones.

Finally, other statistical estimators which could be used instead of the RNNs have been tested, and the results obtained indicate that they do not provide estimations with the same level of quality.

Chapter 6

PSQA in action: Studying the Effects of Several Parameters on the Perceived Quality of VoIP

In Chapter 5 we have presented an accurate mechanism for assessing the quality of multimedia streams. In this chapter we present an analysis of the influence of several applicative and network parameters on the quality of VoIP streams, both one–way and interactive. To this end, we follow the PSQA approach, and use two RNNs respectively trained for one–way and two–way scenarios.

In Section 6.1, we present the analysis of one-way VoIP flows quality under a variety of conditions. We study how the different quality–affecting parameters influence the perceived sound quality. Section 6.2 presents preliminary results for interactive VoIP quality assessment. We are currently working on completing the subjective testing phase (which is much more complex than for one–way streams, from a logistics point of view), and more work needs to be done in order to produce a more complete analysis.

6.1 One–Way VoIP Streams

The assessment of non–interactive streams is generally focused on the sound (or video) quality itself, and not on what is generally called the *conversational quality*, which includes other factors, such as the difficulty for interrupting the other speaker, the perceived delay and cross–talk, etc. It is therefore much simpler to perform, and it is easier to interpret the results obtained this way. In fact, with the exception of the E–model, all of the objective quality metrics discussed in Chapter 4 assess the quality of the stream itself, and not the conversational quality.

Despite its more limited range, this kind of study is very useful, since it provides good insight into several performance issues, and allows to easily assess the impact of different factors on the stream itself. In a way, it provides a “lower–level” view of the perceived quality as the one given in conversational quality assessment.

6.1.1 Description of the Experiments

Following the PSQA methodology, we have developed a database of subjective test results (MOS values) for different speech samples transmitted over a network under different conditions. The distorted samples were built using the Robust Audio Tool (RAT) [96], over a LAN in which we generated losses according to our loss model (c.f. Section 3.2.1). Using a LAN context as a testbed to generate the distorted samples allowed us to control precisely the network parameters, which would have been much harder, or even impossible, in a larger network. RAT has been developed as part of a research project on audio and video–conferencing, and has proved very useful in the experiments undertaken in this thesis. It has many encoding options, and it is based on the Mbus architecture [1, 75]. This allowed us to automate the generation of the degraded sequences (over 400 of them) by means of a small Mbus application written in Java and a few Ruby scripts, without needing to modify a single line of RAT's code.

The losses in the network were generated with a modified version of Orion Hodson's packet reflector [62], which is an application-layer proxy that was developed for use with RAT. We modified it to implement the simplified Gilbert model. As this software can be controlled remotely, we had the same application that drove RAT set the loss parameters for the reflector.

We considered six parameters which affect the perceived speech quality, two of them concerning the network, and four others concerning the encoding schemes used. The network parameters that we selected were the loss rate (LR), and the mean size of loss bursts (MLBS, for Mean Loss Burst Size). These parameters are enough to set those of the loss model used. As for delay and jitter, their effects can be considered as packet losses, since RAT implements a dejittering buffer. As a side note, both delay and jitter were negligible in our testbed, and so it is safe to say that the only network losses seen by RAT after the dejittering process were those that had actually happened at the network level[1]. We used four values for LR (2, 7, 10 and 15%) and three values for MLBS (1, 1.7 and 2.5 packets).

For the encoding, we considered the following parameters:

Codec – the primary codec (16 bit linear PCM, and GSM),

Redundancy (FEC) – the secondary codec used for FEC (GSM), if any,

Redundancy offset – the offset, in packets, of the redundant data (we used offsets of 1 and 3 packets for the FEC scheme described in Section 3.3.2.1, which is the default in RAT),

Packetization Interval (PI) – the length (in milliseconds) of audio contained in each packet (we considered packets containing 20, 40 and 80ms).

[1]That is, there were no lost packets due to the jitter.

These six parameters give place to many possible configurations, even when using just a few values for each one. We chose the values used so as to have a reasonable number of sequences to assess, but at the same time, to have enough data to train the neural networks. Of the 216 possible valid configurations for those values, we chose about one half, concentrating on higher quality encodings (Linear-16 PCM as the primary codec) and small (20ms) packets, and ended up with 112 configurations to test.

Twelve original samples were used to generate 112 four–sequence groups, one for each of the configurations considered. This is in accordance with [85]. The original sequences were sampled at 8kHz (16 bit, mono) and their contents were unrelated. Half of them were male voices and the other half were female. The four samples in each group were chosen randomly between the original ones, and then transmitted with RAT over our test network in order to affect their quality as it would be affected by normal usage.Finally, three hidden reference groups (original samples and very degraded ones) were added to the test, in order to add dynamism to the test's scale (that is, to expose the subjects to the full range of quality variations that they are likely to encounter during the tests), and to help detect subjects who couldn't conduct proper evaluations.

The results obtained were screened to eliminate subjects that produced erroneous values, and so from the 17 subjects that originally performed the test, 16 results were used to calculate the MOS, as one of the listeners was rejected during the screening process. The screening method used was the β_2 test described in Section 4.1.

We then used randomly chosen subsets of the screened results to train several neural networks, and the rest of the results to validate the trained networks, as described in Chapter 5.

In the following sections we analyze the perceived quality of the VoIP flows as a function of the parameters considered.

6.1.2 Loss Rate and Mean Loss Burst Size

Let us start by looking at the quality of the received voice stream when the loss rate and the mean loss burst size vary, and the other parameters remain constant. We considered LR values ranging from 1% to 20%, and MLBSs ranging from 1 (i.e. mostly isolated losses) to 3. The codec used was PCM linear-16, and packet size was 20ms. Figures 6.1 and 6.2 show the evolution of quality as a function of the network loss rate and the MLBS, with and without FEC, respectively.

We observed that the use of FEC produced an improvement of about one MOS point in the perceived quality, which is significant. We also verified that when there is no error correction, an increase of the MLBS has a slightly positive effect on quality, since there are fewer loss episodes (this had already been noted in [58]). As expected, the FEC scheme used suffers a significant degradation when the MLBS is increased, since in this case, more redundant packets

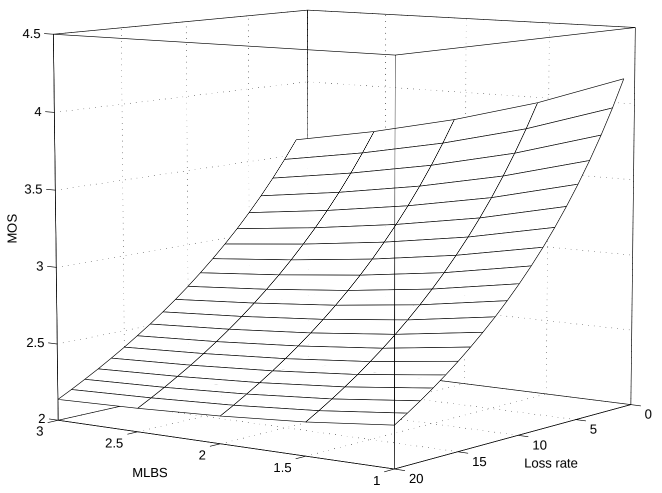


Figure 6.1: Perceived quality as a function of LR and MLBS, with FEC (GSM, 3–packet offset).

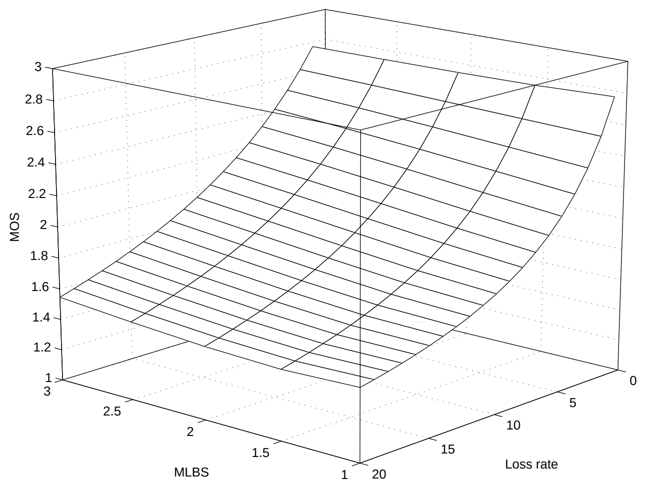


Figure 6.2: Perceived quality as a function of LR and MLBS, without FEC.

are lost in the same loss episode that the original ones.

6.1.3 Mean Loss Burst Size and Redundancy Offset

Having seen how the MLBS affects the performance of the FEC scheme used, it is interesting to see how much does modifying the offset of the secondary encoding improve the perceived quality. For that, we studied the evolution of quality as we varied the MLBS and offset parameters. The results are shown in Figures 6.3 and 6.4.

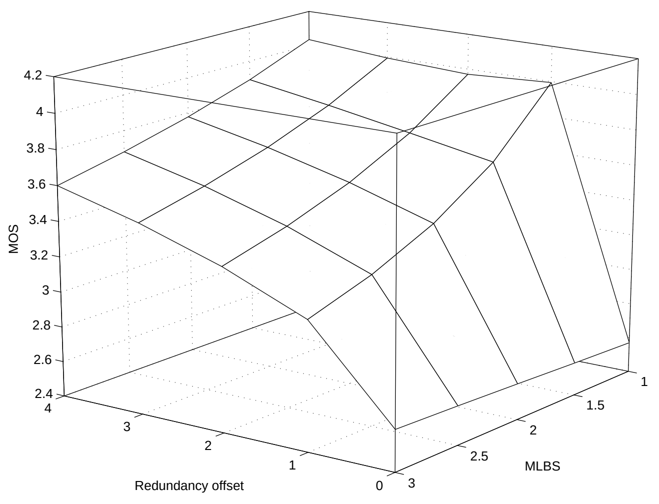


Figure 6.3: Variation of perceived quality as a function of MLBS and FEC offset (an offset of 0 means that no FEC was used), for a loss rate of 2%.

We can see that the curves are quite similar regardless of the loss rate, though there is a one point overall difference for the two loss rates considered. As in Fig. 6.1 we can see that the quality degrades as we increase the MLBS. However, we can achieve a significant improvement in quality by increasing the offset of the FEC payload. Again, we observe that when no FEC is used, increasing the MLBS produces a slight improvement in quality (as seen in Figure 6.2). It should be noted that although using high values for the redundant data offset counters the effects of high MLBSs, it also introduces more delay to the transmission, which may decrease the perceived quality for interactive applications, even if the sound quality is better.

6.1.4 Mean Loss Burst Size and Packetization Interval

Another interesting aspect to consider is how does the packetization interval affect the quality of the speech streams, under various MLBSs. We found that increasing the PI produces an im-

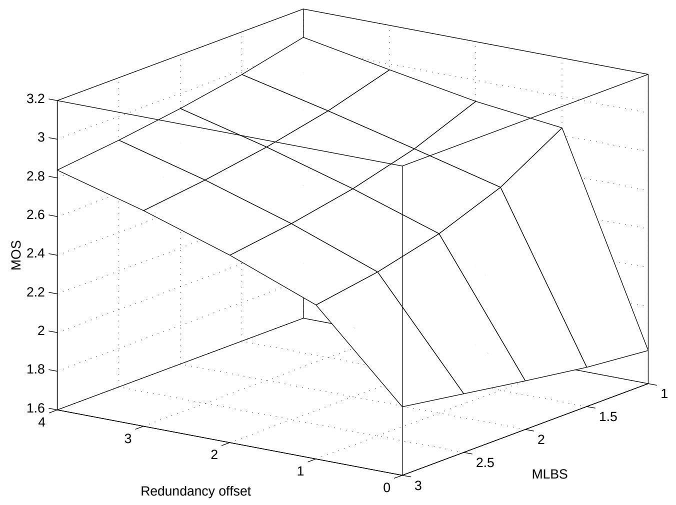


Figure 6.4: Variation of perceived quality as a function of MLBS and FEC offset (an offset of 0 means that no FEC was used), for a loss rate of 10%.

provement in quality, whether FEC is being used or not, and independently of the MLBS. We haven't found an explanation for this phenomenon in the literature, but we believe that it is due to a lower number of loss episodes being produced. This is reasonable, since using longer packets implies sending fewer packets, and thus, fewer loss episodes. This is interesting, because it allows to have an acceptable quality without adding redundant data, even with relatively high values of LR and MLBS. Increasing the PI also produces a better network utilization, since there is less header overhead, and it doesn't affect the performance of the FEC scheme used. Figures 6.5 and 6.6 show the results we obtained.

This may not be applicable under certain circumstances, for example, in applications where minimizing the delay is critical, or where receiver based loss concealment techniques such as waveform interpolation are used, in which case a smaller value of PI may be preferred.

6.1.5 Loss Rate and Packetization Interval

To further understand the effect of packetization interval on speech quality, we studied the quality variation as a function of PI and LR, in order to determine in which situations an increase in PI is a good option for improving quality[2]. We found that we can get near–toll quality without FEC at 5% of packet losses, which we believe to be interesting. Figures 6.7 and 6.8 show the results obtained.

We can see that the influence of PI is stronger for low values of LR (almost one MOS point for LR = 1%), and it decreases as the loss rate increases. At LR = 8%, the quality is about 2.5

[2]We must still consider the remarks made about delay on the previous section.

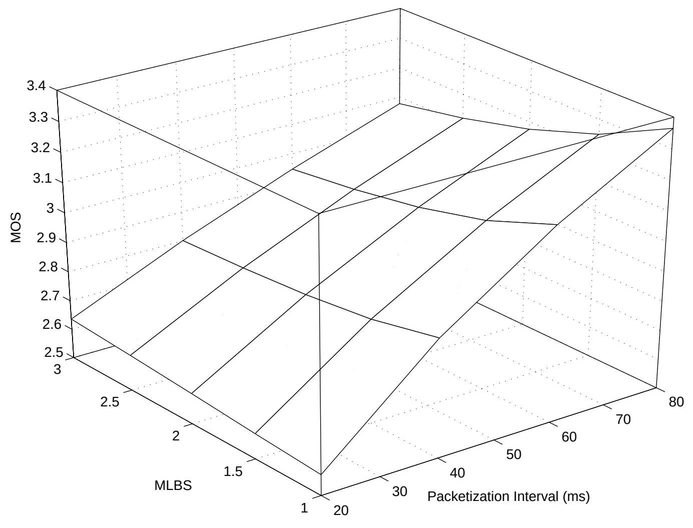


Figure 6.5: Variation of the perceived quality as a function of MLBS and PI, without FEC. Loss rate is 2%.

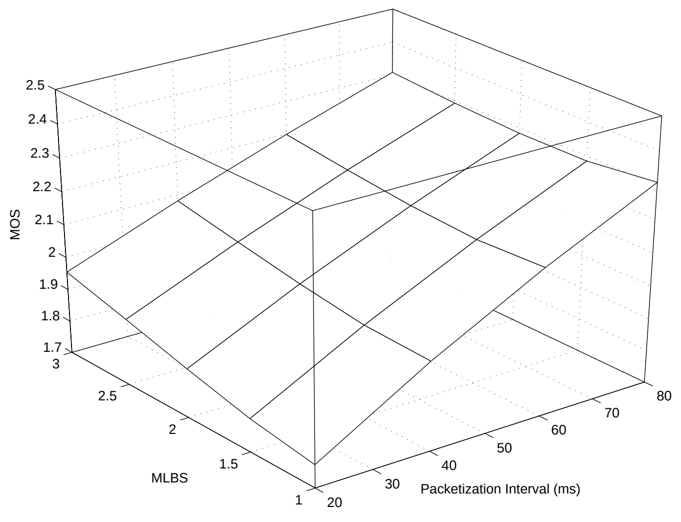


Figure 6.6: Variation of the perceived quality as a function of MLBS and PI, without FEC. Loss rate is 10%.

MOS points, which is not bad considering no error correction is being done. When using FEC, on the other hand, the improvement brought by the increase of PI is not so evident, but it is still there.

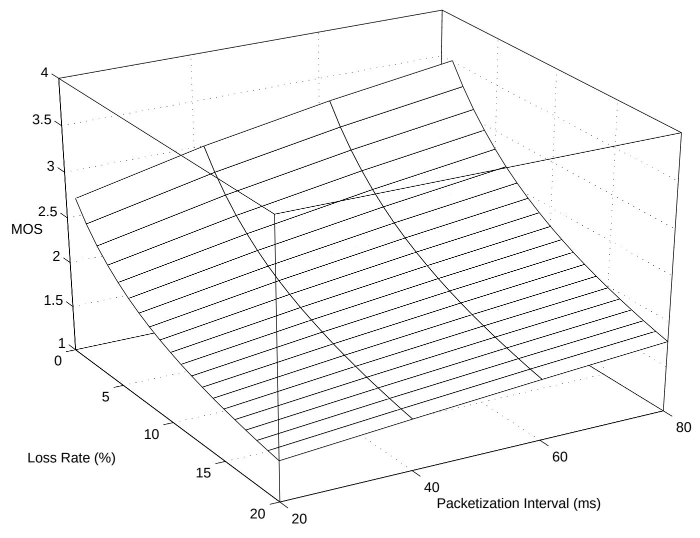


Figure 6.7: Variation of the perceived quality as a function of LR and PI (MLBS = 1.3 packets, no FEC used).

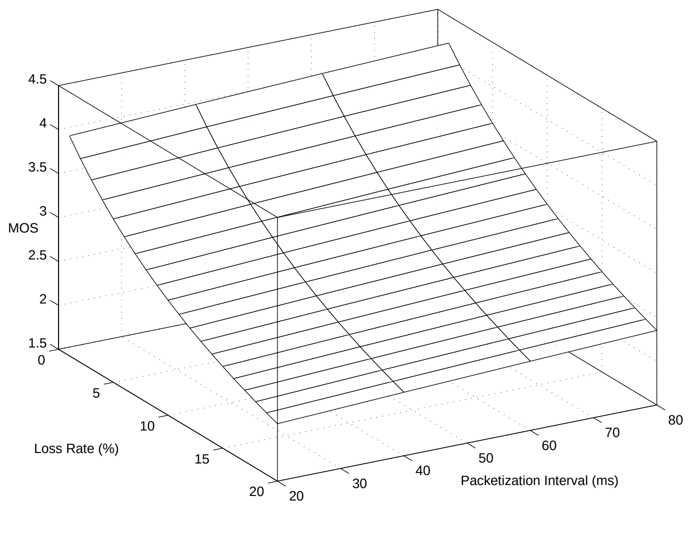


Figure 6.8: Variation of the perceived quality as a function of LR and PI (MLBS = 1.3 packets, FEC = GSM, with a 2–packet offset).

6.1.6 Codecs and Speech Quality

While the purpose of our study was not the comparison of the codecs' performances *per se*, we did study whether the codec used had an important influence on the way the other parameters affect quality. We found that the evolution of quality as a function of the network and redundancy parameters is similar for both GSM and PCM Linear–16, but the relative performances vary with the use of FEC.

In Figures 6.9 and 6.10, we observe that the perceived quality degrades significantly as the loss rate increases. In both cases, both curves are similar, however, we can see that the PCM codec seems to suffer more than GSM when no FEC is present. For small loss rates, PCM is still as good as GSM, but once the losses are more significant, the gap between both curves widens (although it is negligible for the average user). This is explained by GSM being a predictive codec, and PCM being sample based. GSM can recover more gracefully from isolated losses. Moreover, the GSM standard specifies that at least packet repetition is to be used to fill in for missing packets.

When FEC is used, on the other hand, the expected difference in quality between PCM and GSM is evident. It is interesting to note that in the context considered, if no FEC is being used, then maybe it is not worth using PCM. Indeed, according to the results obtained, the improvement in the encoding quality does not make up for the degradation perceived even for low loss rates.

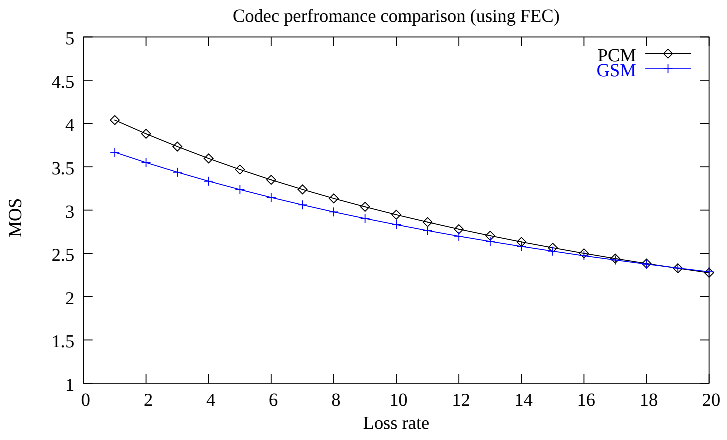


Figure 6.9: The perceived quality as a function of loss rate, for both PCM and GSM, with FEC set to GSM, 2–packet offset.

6.2 Interactive VoIP Streams

Assessing the quality of interactive VoIP streams is a much more difficult task than that of assessing one–way streams. Not only more parameters need to be considered, such as the delay and jitter for example, but there are other factors, such as the type of conversation (e.g. is it very interactive, or is one of the speakers monopolizing the conversation?), the intelligibility problems that may arise due to double–talk, etc. To the best of our knowledge, no objective

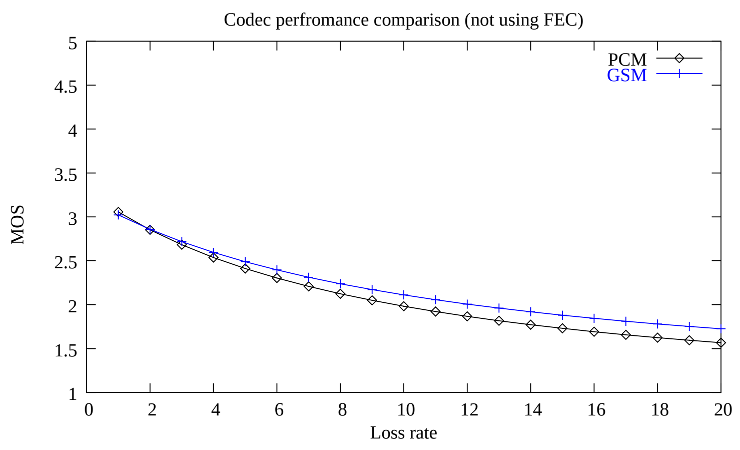


Figure 6.10: The perceived quality as a function of loss rate, for both PCM and GSM, with no FEC

metrics exist in the literature that can accurately assess the quality of a voice conversation. As for subjective assessment, it is also more difficult to set up than for one–way streams, since the subjects need to interact, and there are several restrictions on how that interaction should be carried out. The methodology for carrying subjective evaluations of conversational quality is specified in the ITU–T recommendation P.920 [88].

Besides the technical considerations concerning the conditions on which the tests are to be carried out (room and equipment characteristics, etc.), the P.920 recommendation specifies what kind of scenarios should be used, how to manage the interactivity of the session, etc. Some of the restrictions imposed by the standard are not adequate for our needs, since this kind of test is usually done on a small number of configurations, which suffice to test equipment quality, or the performance of a certain coding technique. Therefore, such requirements as having each conversation last at least five minutes, were overlooked, since testing 120 different configurations at five minutes each would only produce bad assessments due to fatigue. Others, such as demanding that subjects do not know each other were not respected in this preliminary study for practical reasons. The most serious deviation from the recommendation that we incurred in for this first approach to conversational assessment, is that we only used two subjects, who where very familiar with the topic. The P.920 recommendation states that in the case of using expert subjects, at least six should be used. We are currently working toward a more complete subjective assessment campaign in order to obtain more reliable results.

6.2.1 Description of the Experiments

For this experiment, we also considered six parameters, mentioned in Table 6.1. Instead of RAT, we used a tool called VivaVoz, developed at the Federal University of Rio de Janeiro.

VivaVoz is an experimental audio–conference application which was originally developed to test the performance of a new media–independent FEC scheme. Besides this new FEC scheme, it also proposes the use of the Speex codec [124], which is a new (and open) codec specifically designed for voice, and which provides good quality at relatively low bit rates. Having access to the source code, the tool was modified in order to be adapted to the testing conditions required (e.g. eliminating visual cues of the bit rate or FEC level used). Another modification done to VivaVoz was the addition of a PSQA module, which allows to obtain real–time quality estimations when using the tool, and will be used in a later stage for control purposes (cf. Chapter 9).

Parameter	**Values**
Loss rate	0%…60%
Mean loss burst size	1…5
Mean One–way Delay	0ms…250ms
Jitter (as a fraction of the delay)	0%…40%
Bit rate (Speex codec)	2.4kbps…24.8kbps
FEC (media–independent scheme, proposed in [37])	off, 1:3, 1:3::2:6

Table 6.1: Network and encoding parameters used for the interactivity tests, and their values.

As interactive testing requires a live network, we needed to recreate the desired network conditions for each configuration in our testing environment. To this end, we used a Linux host acting as a router between the two hosts running VivaVoz. We used a modified version of the NetEm [60] Linux kernel module in order to recreate network conditions similar to those found on the Internet. NetEm is a Linux router queuing discipline based on the NIST Net [104] wide–area network emulator. This module allows to inject packet losses, variable delays and generate packet reordering in streams traversing the router in which it is running.

The network architecture of our testbed is depicted in Figure 6.11

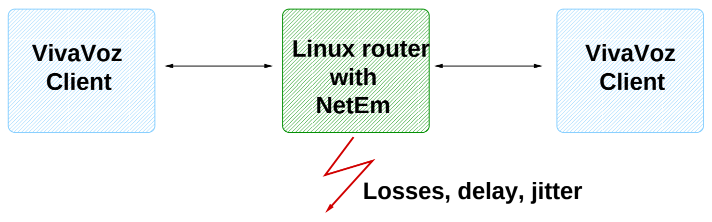


Figure 6.11: The network testbed used for interactive tests. The flows between the clients pass through a Linux–based router in which the NetEm module is running.

We considered a very wide range of network conditions, with loss rates of up to 60%, and one–way delays of up to 250ms. While such extreme conditions are not common for daily use, it is useful to train the RNN with some of these unlikely configurations in order for it to better estimate the quality for more "normal" configurations. We believe that, in a way, the assessment of the border configurations allows the RNN to provide a better fit for points inside the configuration space. The use of large values for the borders makes it more unlikely that the RNN will need to extrapolate outside the training space. Of the 120 configurations used to train the RNN, 48 corresponded to the border conditions, and the rest were randomly chosen, with a bias toward what we consider normal operating conditions, namely: low loss rates and mean loss burst sizes, low delays and jitter values, and medium to high bit-rates.

As mentioned before, the results from this testing campaign are preliminary, since only two (knowledgeable) people participated in the assessment. We tested 120 different configurations, and for each configuration, two tasks were performed:

- taking turns counting up to 20 as quickly as possible, and
- a conversation, which would be based on a scenario, on a picture, or free.

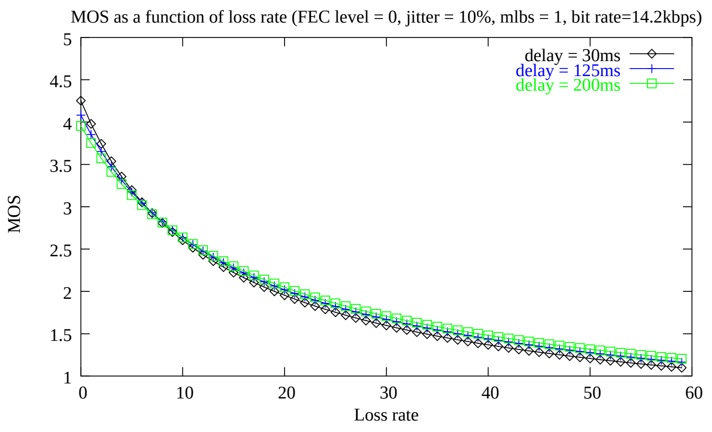
The idea behind the different tasks is to have conversations at different levels of interactivity, which may affect the way the delay impacts the perceived conversational quality.

We found that the interactive subjective assessment is far more tiring than the assessment of one–way pre–recorded samples (which is already quite tiring). This poses an extra problem, since fatigue settles in quite quickly, and so in order to obtain reliable measurements, the assessment needs to be spread over several sessions. For a larger campaign, we are considering using more people, and having each pair of listeners assess fewer configurations, so as to avoid fatigue. The set of configurations that we need to assess would be spread over overlapping subsets, and each subset would be tested by at least two pairs of subjects. Further analysis, and maybe more *in–house*, smaller scale campaigns, need to be done in before a large–scale campaign is carried out.

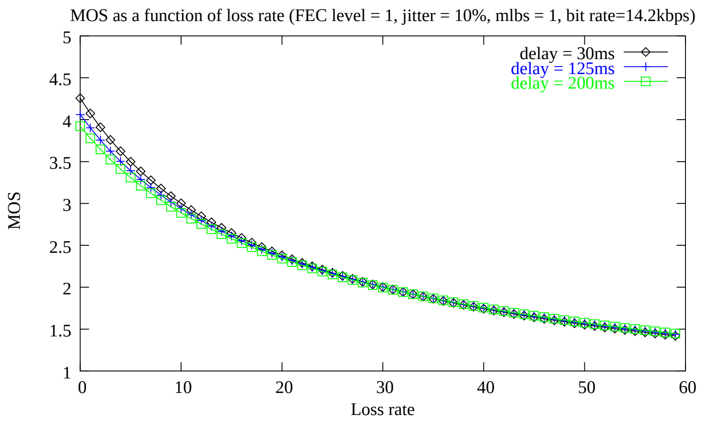
6.2.2 Preliminary Conversational Quality Analysis

In this section we present a first analysis of the results obtained from the interactive testing campaign. While not definitive, these results show an interesting fact, which is in accordance with the subjective appreciation of the testers: poor sound quality has a greater impact on conversational quality than high delay values. It is widely stated in the literature that delays higher than 150ms significantly impact the perceived conversational quality. In our experience, while there is an impact in low-loss conditions, it is not a very significant one, as depicted in Figure 6.12. It is easy to see that the delay only makes for a small variation of MOS for very low loss rates. Afterward, the quality is mainly driven by the loss rate, and the impact of delay seems to become irrelevant.

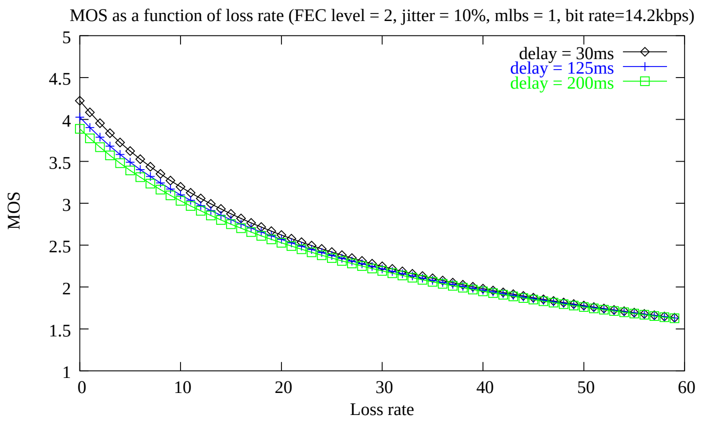
VivaVoz uses a media–independent scheme [37] based on XOR operations among packet groups of different lengths. This novel scheme works by dividing the packets into windows of



(a)



(b)



(c)

Figure 6.12: The perceived quality as a function of loss rate, for different delay values. (a) No FEC. (b) FEC level = 1 (1:2). (c) FEC level = 2 (1:2::3:6).

a given length l, and dividing each window into s non–overlapping subsets. An XOR operation is then performed among packets in each subset, and the results are piggy–backed on the first s packets of the next window (these configurations are noted as $s : l$). In Figure 6.13 we see an instance of this scheme for a window of six packets, and two subsets of three packets each (2:6). More than one of these schemes can be overlapped in order to obtain better protection against losses. A detailed analysis of the performance of this scheme is found in [37]. Figure 6.14 shows the impact of this FEC scheme on the perceived conversational quality. The scores show that there is a significant improvement when FEC is used, in particular for loss rates of up to 15 to 20%, where it yields almost acceptable values of quality. We can see that there is a slight decrease in performance for high delay values. This could be due to the increase in delay caused by the use of FEC. At 20ms per packet (the packetization interval used by VivaVoz), the FEC scheme may induce an extra delay of up to 120ms, which is a nuisance if the network delay was already high (even considering the relatively low impact of delay mentioned above).

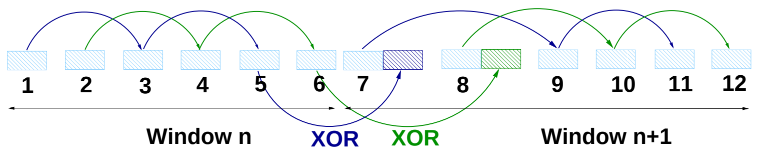
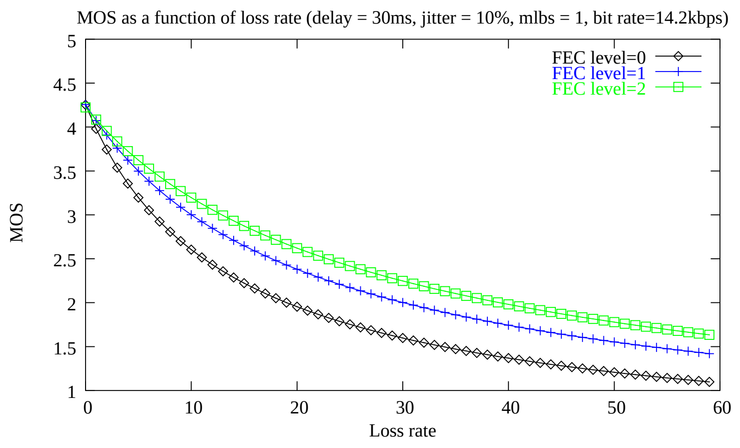


Figure 6.13: An instance of the FEC scheme presented in [37]. The figure shows how the redundancy is added, for a 6–packet window divided into two 3–packet subsets (2:6). (From [37])

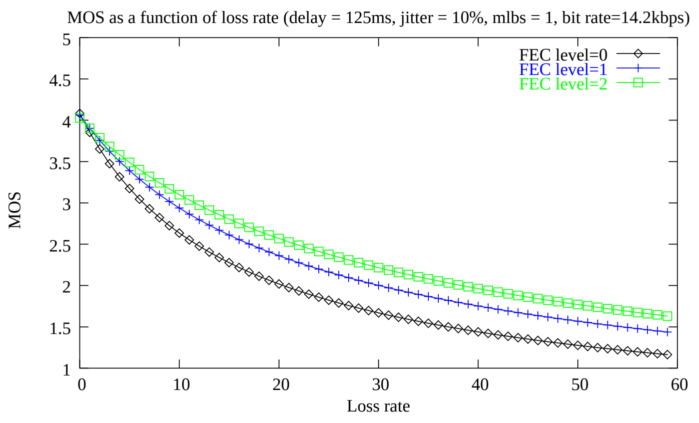
As for the Speex codec, we can see that it performs quite well even when no FEC is used. For instance, in Figure 6.15 (a), where no FEC is used, MOS values of up to 2.5 are attainable at high bit rates even for loss rates of up to 15%. While a MOS of 2.5 is not acceptable quality, streams at that level are generally intelligible. At lower loss rates (5%), even relatively low bit rates of about 8kpbs yield acceptable quality levels. When using FEC (Figure 6.15 (b)), a MOS of about 3 can be observed when using bit rates between 14.2 and 18.4kbps at a loss rate of 15%. For the highest bit rate used (24.4kbps), we observe acceptable quality levels at 20% loss rate, which is indeed a very good performance.

As mentioned above, more of these subjective tests are needed in order to get a better assessment of the conversational quality. As of this writing, there are two points that are not very clear in the current analysis. The first one concerns the impact of delay on the conversational quality. Based on what is found in the literature, the conversational quality should greatly degrade for high delay values, which does not seem to be the case here. Several explanations are possible for this:

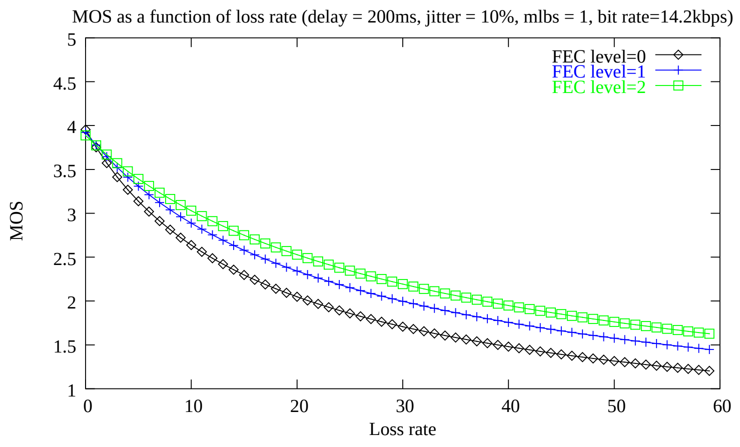
- the fact that the listeners were knowledgeable on the subject, may have helped them adapt their interactivity to the increased delay,
- the differences in sound quality between high–loss and low–loss scenarios may have a greater impact than the loss of interactivity due to the delay. The fact that the config-



(a)

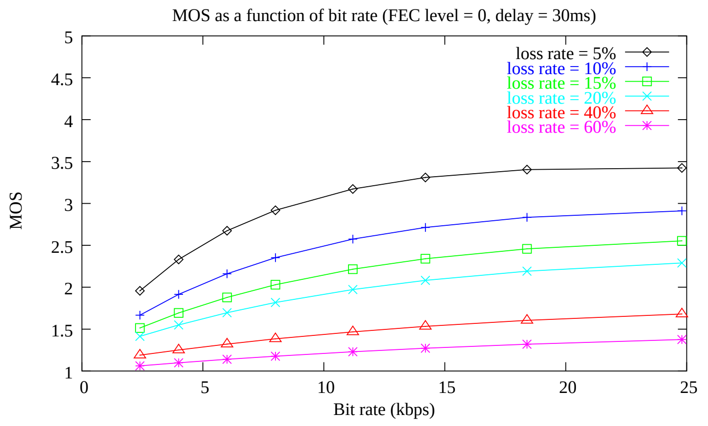


(b)

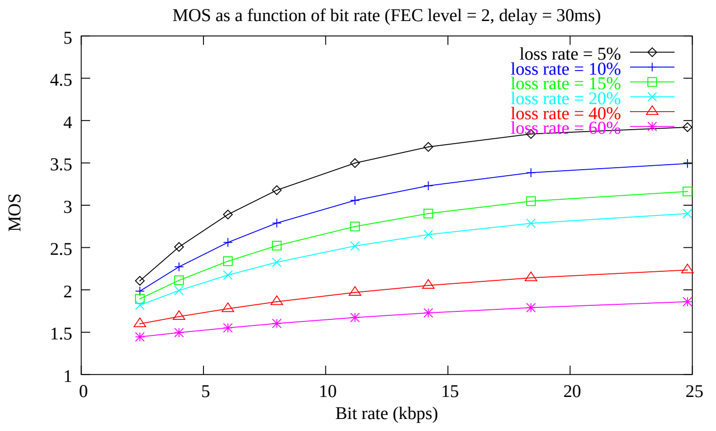


(c)

Figure 6.14: The perceived quality as a function of loss rate, for different mean delay values. (a) Delay = 30ms. (b) Delay = 125ms. (c) Delay = 200ms.



(a)



(b)

Figure 6.15: The perceived quality as a function of the bit rate, for several loss rate values (delay = 30ms, jitter = 10%, mlbs = 1). (a) FEC level = 0. (b) FEC level = 2.

urations were assessed in random order may have amplified this effect (for instance, if low–delay, high–loss configurations were next to high–delay, low-loss ones),

- previous studies on delay have analyzed its effects in isolation, whereas its impact is relatively small next to that of the loss rate, or

- a combination of all of the above.

The second issue with the current results concerns the impact of the mean loss burst size in this context. The results we have obtained seem to indicate that, unlike in the one–way case, its impact is almost nil for the configurations considered. One reason for this may lie with the FEC scheme used. In the one–way scenario presented in Section 6.1, the media-dependent FEC scheme used was very sensitive to the mean loss burst size, and its performance depended also on the FEC offset used. The FEC scheme used for the interactive tests (at least in the

1:2::3:6 configuration) allows for a greater spacing of the redundant data, which may alleviate the quality drop caused by large MLBS values. In any case, no further conclusions may be drawn until a more extensive subjective assessment campaign has been carried out.

If, however, MLBS (or some other parameter) had no (significant) impact on quality, the PSQA results would reflect this and the parameter could be ignored or replaced by a more relevant one in future studies.

6.3 Summary

In this chapter we have studied how different network and applicative parameters affect the quality of VoIP flows. In the first section we focus on one–way (that is, non–interactive) streams, and we analyse the perceived voice quality as a function of all the six parameters considered. The use of PSQA makes this easy, and allows to observe how the different parameters interact with the quality. We have observed and explained how some parameters interfere with others (e.g. how the mean loss burst size may affect the peroformance of FEC), which can be used for control purposes (e.g. increasing the FEC offset to compensate for an increase in the MLBS).

The second part of this chapter deals with interactive VoIP streams, and the measurement of conversational quality. The results presented here are preliminary, since this work is still in progress. The difficulties involved in assessing the conversational quality are greater than those of assessing the voice quality in one–way streams. The automated measurement of conversational quality (which we have integrated into the tool used for these tests) with as many parameters as we consider, has not, to the best of our knowledge, been done before. We have studied, as in the one–way case, the influence of several parameters (both network and application–level) on the conversational quality as perceived by the end–users. As a part of future work on real–time dynamic control of the conversational quality, we have implemented a PSQA module into the tool used for the tests, which allows to obtain a real-time estimation of quality during the conversation.

In the next chapter, we will show how to integrate the kind of analysis presented in this chapter, with traditional performance evaluation techniques, such as queuing models or simulations.

Chapter 7

Integrating PSQA and Traditional Performance Evaluation

In this chapter we propose mechanisms to take advantage of PSQA and its ability to provide accurate quality estimations for media streams, in the context of a more traditional approach to performance evaluation. Currently, if one wants to take some notion of perceived quality into account when dimensioning or designing a network, one could use the E–model. However, that approach is limited to voice streams (whereas PSQA can be used for other media types), and as seen in Chapter 5, it is not very accurate.

We describe how to integrate PSQA with network models, and use this approach to assess the influence of FEC on the perceived quality of media streams, by means of a simple network model to take into account the impact of the added redundancy on the network state.

7.1 Motivation

Let us consider the following situation: we need to analyze the effect of a quality–affecting parameter (such as the offset of FEC data or the variations in loss rate) on the perceived quality of a VoIP application. However, we don't necessarily have access to the actual network (maybe it is still in its design phase, for example), so we use a model, as for instance in [6, 23, 24].

What is usually done in this case is to build the model (generally, some queuing system capturing the behavior of the network) which allows us to obtain performance metrics as a function of specific input parameters. Let us take, for instance, a simple $M/M/1/W$ model for the network (representing the whole network or its bottleneck by a single queue is a simplification commonly found in the literature). The inputs are the arrival rate (or offered mean traffic) λ in packets per second (pps), the transmission rate μ (also in pps) that the network offers to the packets and the maximum number of packets that the network can hold, represented globally here by the storage capacity W of the queue. This implies that these parameters can be estimated from the network characteristics and from the expected behavior of the offered traffic.

The loss probability p_L of this simple $M/M/1/W$ queue is given by

$$p_L = \frac{\rho^W(1-\rho)}{1-\rho^{W+1}} \tag{7.1}$$

with

$$\rho = \frac{\lambda}{\mu} \neq 1.$$

If $\rho = 1$ then

$$p_L = \frac{1}{W+1}.$$

At this point, some *utility functions* can be used to take into account the way users react. These functions are mappings between the performance metrics and some abstract concept of quality. In many cases, these mappings are not defined, and the analysis, the dimensioning tasks, etc. are done based only on such metrics as the loss rate, or the delay. This, of course, provides a much rougher answer to the questions that a network designer needs to have answered. The problem here is how to define the utility functions, that is to say, how to define the way a user values a service. One way to do so, is to base the valuation on the perceived quality of that service. It is in this context that the use of PSQA can be helpful.

For instance, in the example shown below, we will provide a function mapping the load ρ, the network memory capacity W, the loss rate, the redundancy scheme used, the type of codec and the packetization interval into quality, for a voice stream. The method we propose provides a function $Q = f(x_1, \cdots, x_P)$, where the different input parameters x_i are both application–level ones, such as the codec used, and network metrics, such as the loss rate, which can be derived from the model, and the output is the perceived quality, for example in a standard MOS scale.

A Note on the Sample Model Used

In order to simplify the example, we will use the simple $M/M/1/W$ queue mentioned above as our network model. This is a well known (and basic) model, and it is simple enough to allow us to keep the focus on the method. However, it should be clear that the model to choose depends on the needed accuracy, the available data, the questions to answer, the available computer power, etc. For instance, if we needed to estimate the performance of some UDP-based application using a TCP-friendly control algorithm and we had decided to use some specific model to capture the aspect of the system behavior that interests us, we would have to find a way to derive the metrics that interest us from that model. For example, if instead of an $M/M/1/W$ queue, we had chosen a $ON–OFF/G/1/W$ one, we would have more parameters to consider, such as the ON–OFF customer arrival rates, the mean sojourn times in each state, the mean service rate for the server, and its variability. From these parameters, we would have to derive the actual metrics that we are concerned with, such as the loss rate, the loss distribution, or the delay.

If the chosen model is accurate (that is, if the metrics we obtain from it are close to those found in a real network), then Q will provide us with a good estimate of the perceived quality, directly as a function of the available data. We could also decide to represent the system by a complex Markovian queuing network having many "low level" input parameters (instead of just, say, ρ and W as in the example above), which can only be solved numerically. In that case, $f()$ will be accessible through a numerical procedure, but again, the result will be an accurate estimation of the quality as perceived by the end–user of the application, whose accuracy will be essentially proportional to that of the model used.

7.2 Our approach at work

In this section we describe in detail the application of the approach we propose to a specific type of situation and we provide some numerical results to illustrate the method and its use. Our choice is thus to study the performance of a one–way voice stream transmitted over a best–effort network.

7.2.1 Modeling

We resume here the example discussed above in Section 7.1. Assume that we want to predict the behavior of some audio streaming application over an IP network (for instance, an audio web-cast for a news site). As already explained, our approach has two parts: a modeling component for performance evaluation purposes, and a PSQA module that will receive inputs from the former. We will consider the whole network represented by an $M/M/1/W$ model as discussed above. The input parameters of the model are its load ρ and its storage capacity W. We also want to take into account several application–level parameters, such as the codec used, and the possibility of using FEC. These parameters are specific to the application. We will consider that, as is the case in the current Internet, audio traffic represents a very small fraction of the total traffic, and therefore the impact of changing codecs, or using more or less redundancy in the flow is negligible on a global scale. From the modeling point of view, we will just focus on losses. Of course, more parameters can be taken into account, as long as they can be derived from the chosen model. For example, if we were to consider interactive applications as opposed to one-way streaming, network delay and jitter would need to be considered too.

As mentioned before, we know that the loss probability or loss rate is related to the input variables ρ and W through the expression $p_L = \rho^W(1-\rho)/(1-\rho^{W+1})$ if $\rho \neq 1$, and $p_L = 1/(W+1)$ if $\rho = 1$. As seen in Sections 6.1.2 and 6.1.3, the loss rate alone is too poor to capture the way losses have an impact on quality. In many cases, the perceived qualities of different samples are quite different even if the loss rate in the network is the same, as seen for instance in Figures 6.1, 6.3 or 6.4. For this reason, we chose to also include the mean loss burst size (MLBS) as a parameter.

In our model choice, we need to compute MLBS as a function of ρ and W, which is a simple task. The probability that a burst of losses has size j, $j \geq 1$, is equal to $p^{j-1}q$ where $p = \rho/(1+\rho)$ and $q = 1 - p = 1/(1+\rho)$. This is because a burst has size j if, and only if, after a first loss, the *next* packet is lost (probability p), and the next one, and so on exactly $j-1$ times, and the following packet is not lost, which happens with probability q. The expectation MLBS is then

$$\mathrm{MLBS} = \sum_{j \geq 1} j p^{j-1} q = \frac{1}{q} = 1 + \rho.$$

As mentioned in Section 6.1.1, we chose to limit the analysis to some specific ranges according to the goals of the study. For the two network parameters, the loss rate p_L will be considered in the range $[0, p_0]$ (for instance, $p_0 = 0.15$, since higher loss rates normally imply that applications see their quality degraded below acceptable levels). In the same way, we consider $\mathrm{MLBS} \in [0, \mathrm{MLBS}_0]$ (for instance, $\mathrm{MLBS}_0 = 4$ following the observation of traces in the Internet [135]). Of course, this must be translated into our final input parameters, which for the network part of the analysis are ρ and W. Since

$$\mathrm{MLBS} = 1 + \rho,$$

we have some constraints, which we detail below. These restrictions on ρ and W are somehow a price to pay for using a specific network model. A different model would yield a completely new set of constraints.

7.2.2 Restrictions on the values of ρ and W

Let us assume that the maximal loss rate that we consider is $p_0 \in (0, 1)$, and that the maximal MLBS value is $\mathrm{MLBS}_0 \geq 1$. In other words, we assume that

$$0 \leq p \leq p_0 \qquad (7.2)$$

$$1 \leq \mathrm{MLBS} \leq \mathrm{MLBS}_0 \qquad (7.3)$$

We want to know the restrictions that Equations (7.2) and (7.3) induce on the load ρ and the network capacity (buffer size) W. The first immediate condition comes from the expression $\mathrm{MLBS} = 1 + \rho$, which leads to

$$\rho \leq \mathrm{MLBS} - 1 \qquad (7.4)$$

For every $W \in \mathrm{N}$, $\quad W \geq 1$, we define the function $f_W()$ on the positive reals by

$$f_W(x) = \begin{cases} \dfrac{1-x}{1-x^{W+1}} x^W & \text{if} \quad x \neq 1, \\ \dfrac{1}{W+1} & \text{if} \quad x = 1. \end{cases}$$

We can check that $f_W()$ is continuous. When the buffer size is W and the load is ρ, the loss probability is $f_W(\rho)$. Function $f_W()$ is increasing for every $W \geq 1$.

Case 1: $\rho < 1$

$$p = \frac{1-\rho}{1-\rho^{W+1}}\rho^W \leq p_0$$

Let us set $\rho^W = x < 1$. We have

$$\frac{(1-\rho)x}{1-\rho x} \leq p_0. \tag{7.5}$$

Since $1 - \rho x > 0$,

$$\begin{aligned}(7.5) \quad &\Leftrightarrow \quad (1-\rho)x \leq p_0(1-\rho x)\\ &\Leftrightarrow \quad x \leq \frac{p_0}{1-\rho(1-p_0)}\end{aligned}$$

From

$$\rho^W \leq \frac{p_0}{1-\rho(1-p_0)},$$

taking logs,

$$W \ln \rho \leq \ln \frac{p_0}{1-\rho(1-p_0)} \Leftrightarrow W \geq W_0 = \left\lceil \frac{\ln \frac{p_0}{1-\rho(1-p_0)}}{\ln \rho} \right\rceil$$

Case 2: $\rho > 1$

With the same notation as above, we have

$$\frac{(\rho-1)x}{\rho x - 1} \leq p_0, \tag{7.6}$$

where now $x = \rho^W > 1$. Since $\rho x - 1 > 0$,

$$\begin{aligned}(7.6) \quad &\Leftrightarrow \quad (\rho-1)x \leq p_0(\rho x - 1)\\ &\Leftrightarrow \quad [1-\rho(1-p_0)]\, x \geq p_0.\end{aligned}$$

Now,

$$1-\rho(1-p_0) \gtrless 0 \Leftrightarrow \frac{\rho}{1-p_0} \gtrless \rho.$$

Case 2a: $1 < \rho < \dfrac{1}{1-p_0}$**.** We have $1-\rho(1-p_0) > 0$, so

$$\rho^W \geq \frac{p_0}{1-\rho(1-p_0)} \Leftrightarrow W \geq W_0 = \left\lceil \frac{\ln \frac{p_0}{1-\rho(1-p_0)}}{\ln \rho} \right\rceil$$

Case 2b: $\rho > \dfrac{1}{1-p_0}$ This case is impossible, since $\rho > 1/(1-p_0)$ implies that $p > p_0$: since $f_W()$ is increasing,

$$\rho > \frac{1}{1-p_0} \Rightarrow f_W(\rho) > f_W(\frac{1}{1-p_0}),$$

that is,

$$p > \frac{\left(\dfrac{1}{1-p_0} - 1\right)x}{\dfrac{1}{1-p_0}x - 1} = \frac{p_0 x}{x-(1-p_0)} > p_0 \quad (\text{since } 1-p_0 > 0)$$

Case 2c: $\rho = \dfrac{1}{1-p_0}$ This case is also impossible, since we must have

$$[1-\rho(1-p_0)]x \geq p_0$$

and if $\rho = 1/(1-p_0)$,

$$[1-\rho(1-p_0)]x = 0.$$

Case 3: $\rho = 1$

Here,

$$\frac{1}{W+1} \leq p_0 \Leftrightarrow W \geq W_0 = \left\lceil \frac{1-p_0}{p_0} \right\rceil$$

Summary

Let us define the function $\varphi()$ over the real set $[0, 1/(1-p_0))$ by

$$\varphi(y) = \begin{cases} \dfrac{\ln \dfrac{p_0}{1-(1-p_0)y}}{\ln y} & \text{if } y \neq 0,\ y \neq 1 \\ \dfrac{1-p_0}{p_0} & \text{if } y = 1 \\ 0 & \text{if } y = 0 \end{cases}$$

We can verify that $\varphi()$ is well defined and that it is a continuous function. We also have

$$\lim_{y \to 1/(1-p_0)} \varphi(y) = \infty.$$

We had that ρ must satisfy $\rho \leq \mathrm{MLBS}_0 - 1$ and we also found that $\rho < 1/(1-p_0)$. This means there are two possible cases to consider.

Case (i): $\dfrac{1}{1-p_0} \leq \mathrm{MLBS}_0 - 1.$ Here, $\rho \in (0, 1/(1-p_0))$ and $W \geq W_0 = \lceil \varphi(\rho) \rceil$.

Case (ii) $\text{MLBS}_0 - 1 < \dfrac{1}{1-p_0}$**.** In this case, $\rho \in (0, \text{MLBS}_0 - 1]$, and once again $W \geq W_0 = \lceil \varphi(\rho) \rceil$.

Final Remarks In case (i), when ρ goes from 0 to $(1-p_0)^{-1}$, W goes from 1 to ∞. In case (ii), when ρ goes from 0 to MLBS_0^{-1}, W goes from 1 to

$$\lceil \varphi(\text{MLBS}_0^{-1}) \rceil = \left\lceil \frac{\ln \dfrac{p_0}{1-(1-p_0)\text{MLBS}_0^{-1}}}{\ln \text{MLBS}_0^{-1}} \right\rceil .$$

7.2.3 Assessing Quality: PSQA in a Closed Form Expression

In Section 5.4.1 we discussed the performance of PSQA when using a very simple, two–layer, feed–forward RNN architecture. An interesting observation was that the results provided by that minimal RNN were good (almost as good, in fact, as that of a more complex three–layer one). This allows us to provide a simple closed form expression for quality as a function of network load and capacity; see below for details.

As the RNN is feed-forward, the calculations are particularly simple: let us denote by 1 to n the entry neurons and by $n+1$ the only output one. We have that for all $i = 1, \cdots, n$,

$$\varrho_i = \frac{\lambda_i^+}{\mu_i} = \frac{\lambda_i^+}{w_{i,n+1}^+ + w_{i,n+1}^-},$$

and

$$\varrho_{n+1} = \frac{\sum_{j=1}^{n} \rho_j w_{j,n+1}^+}{\mu_{n+1} + \sum_{j=1}^{n} \rho_j w_{j,n+1}^-}.$$

In our experiments, where $n = 6$, we decided to keep constant the rate of neuron 7 (the value we used was $\mu_7 = 0.01$). Once trained, the values of the weights are given in Tables 7.1 and 7.2.

Observe that the preceding discussion leads to the following expression for the quality of the stream:

$$Q = \varrho_7 = \frac{\sum_{i=1}^{6} a_i \lambda_i^+}{0.01 + \sum_{i=1}^{6} b_i \lambda_i^+} \tag{7.7}$$

where

$$a_i = \frac{w_{i,7}^+}{w_{i,7}^+ + w_{i,7}^-} \quad \text{and} \quad b_i = \frac{w_{i,7}^-}{w_{i,7}^+ + w_{i,7}^-}.$$

Such a simple and explicit expression allows to easily analyze the variation of the quality with respect to specific parameters. Formally, setting $a_0 = 0$ and $b_0 = 0.01$,

$$\frac{\partial Q}{\partial \lambda_k^+} = \frac{c_0 + \sum_{i=1}^{6} c_i \lambda_i^+}{(b_0 + \sum_{i=1}^{6} b_i \lambda_i^+)^2}$$

where

$$c_i = \begin{vmatrix} a_k & a_i \\ b_k & b_i \end{vmatrix} = a_k b_i - b_k a_i \text{ (in particular, } c_k = 0).$$

For example, in Figure 7.6 we show the sensitivity of the quality with respect to the load for two values of the capacity.

Let us now give an example of a closed form expression for quality, as a function of the parameters ρ and W. In Table 5.2 we have presented the list of the 6 inputs for the neural network. Respecting the same ordering, let $x_1, \cdots, x_6$ be the normalized values of:

- the loss rate (going from 0% to 15%)
- the mean loss burst size (going from 1 to 2.5)
- the codec used (PCM or GSM)
- the FEC state (ON or OFF)
- the FEC offset (1, 2 or 3 packets)
- the packetization interval (going from 20ms to 80ms)

The RNN provides us with the function $f(x_1, \cdots, x_6)$ mapping these 6 variables into a MOS quality metric. Assume that we fix variables x_3 to x_6 to some specific (for instance, typical) values, and that we build a new function having as input variables $x_3, \cdots, x_6$, plus W (the network capacity) and ρ, the *global* load of the network. To this end, we use the expressions of x_1 (the loss probability), and x_2 (the normalized mean burst loss rate), as a function of these two new input variables W and ρ, provided by the analysis of the associated performance model (the $M/M/1/W$ queue in our example). Recall that the limited range of the loss rate and the mean loss burst size lead to corresponding specific ranges to the load and the buffer size, namely $\rho \leq \rho_0$ and $W \geq W_0$ (see Section 7.2.2).
Function $f(x_1, \cdots, x_6)$ is rational with known coefficients (as previously described). Once variables x_3 to x_6 fixed to specific values, and x_1 (resp. x_2) replaced by $x_1 = [(1-\rho)\rho^W]/[1-\rho^{(W+1)}]$ (resp. $x_2 = 1 + \rho$), we obtain, after some algebra,

$$Q = \frac{\alpha + \beta\rho + \gamma\rho^W - (\gamma + \alpha)\rho^{W+1} - \beta\rho^{W+2}}{\alpha' + \beta'\rho + \gamma'\rho^W - (\gamma' + \alpha')\rho^{W+1} - \beta'\rho^{W+2}}. \quad (7.8)$$

The parameters α, β, γ, α', β' and γ' are functions of the encoding ones. As an example, assume we chose the PCM codec, FEC with an offset of 1 and a packetization interval equal to 20 ms. We obtain $\alpha = 0.1326$, $\beta = 0.0201$, $\gamma = 0.0674$, $\alpha' = 0.1659$, $\beta' = 0.0399$, $\gamma' = 0.9326$. (Figure 7.5 shows the perceived quality as a function of the load for two values of W, using these coefficients.) The quality function becomes (after multiplying numerator and denominator by 100 for typographical purposes)

$$Q = \frac{13.26 + 2.01\rho + 6.74\rho^W - 20\rho^{W+1} - 2.01\rho^{W+2}}{16.59 + 3.99\rho + 93.26\rho^W - 109.8\rho^{W+1} - 3.99\rho^{W+2}}.$$

The next subsection illustrates the use of this analytical expression of the perceived quality.

Parameter	$w^+_{i,7}$
Codec	0.831879
FEC	1.53147
FEC Offset	1.08491
Loss rate	0.193885
Mean burst size	1.12165
Packetization interval	1.50425

Table 7.1: Weights of positive connexions in the expression of quality (see relation (7.7))

Parameter	$w^-_{i,7}$
Codec	1.50221
FEC	1.64266
FEC Offset	1.36289
Loss rate	2.68204
Mean burst size	2.23472
Packetization interval	1.59028

Table 7.2: Weights of negative connexions in the expression of quality (see relation (7.7))

7.3 Sample Results

Figures 7.1 through 7.6 illustrate the kind of insight that a network designer can gain from using the approach proposed to know how quality reacts to network impairments. Currently, networks tend to be over–dimensioned if some level of performance is to be attained. With this approach, it should be possible to tune the network more finely, obtaining the desired QoS levels without wasting resources. For example, knowing (approximately) the expected traffic levels, and the fraction of that traffic which is expected to be voice flows, we could adjust the needed capacity (and thus the expected load), so as to maintain voice quality over a certain threshold. Moreover, one could even evaluate the utility (or lack thereof) of using FEC to protect those flows, depending on the expected increase in load that it would generate, and its impact on quality. The same idea is applicable to any other application for which PSQA can be used, such as video, for example.

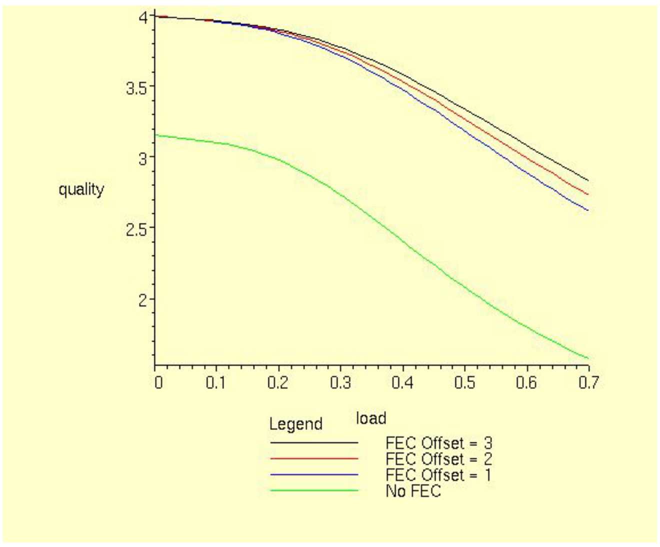


Figure 7.1: Quality as a function of the load, for different FEC settings. W = 3

In Figures 7.1 and 7.2, we can see how different FEC settings provide different qualities under an increasing load, for two values of network capacity. We can see that the efficiency of the FEC varies with W, and also what is the gain obtained. Recall that a quality (MOS) value of 3 is considered to be acceptable [85]. It is easy to see that an increase in W allows to have an acceptable quality with a quite higher load. This qualitative behavior is obviously expected, but the important point is that we can know *just how much* higher the perceived quality will be for bigger values of W, and then make a design decision based on the QoS levels we want to attain, and the costs associated with the higher capacity. It is also interesting to see that this kind of curve is a more accurate representation of the FEC performance than the utility functions used for example in [7, 17], which were chosen somewhat arbitrarily.

Figures 7.3 and 7.4 present the perceived quality as a function of both network load and capacity, with and without FEC, respectively. It should be noted that in the model we chose, not every point in the (ρ, W) plane is feasible. The pair (ρ, W) must satisfy the restrictions given in Section 7.2.2 (see the summary in page 128). Besides the 1 MOS point difference between the surfaces, we also see that when not using FEC, the variations in quality are more pronounced than when a FEC scheme is present. This kind of plot allows to measure the benefits of using FEC or not, depending on what the expected amount of VoIP traffic is in our

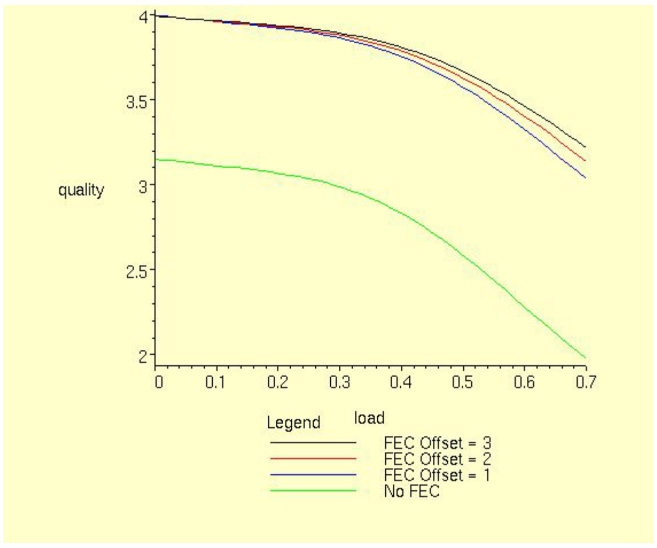


Figure 7.2: Quality as a function of the load, for different FEC settings. W = 5

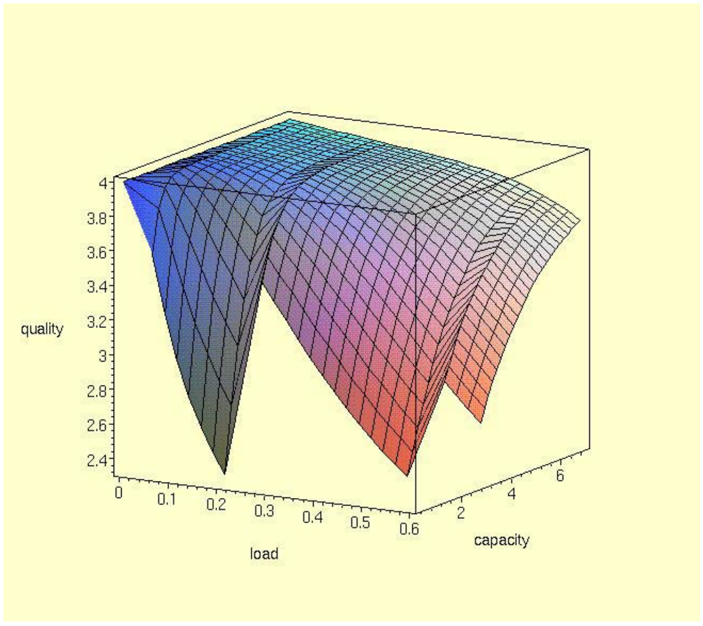


Figure 7.3: Quality as a function of the load and network capacity, with FEC (offset = 1).

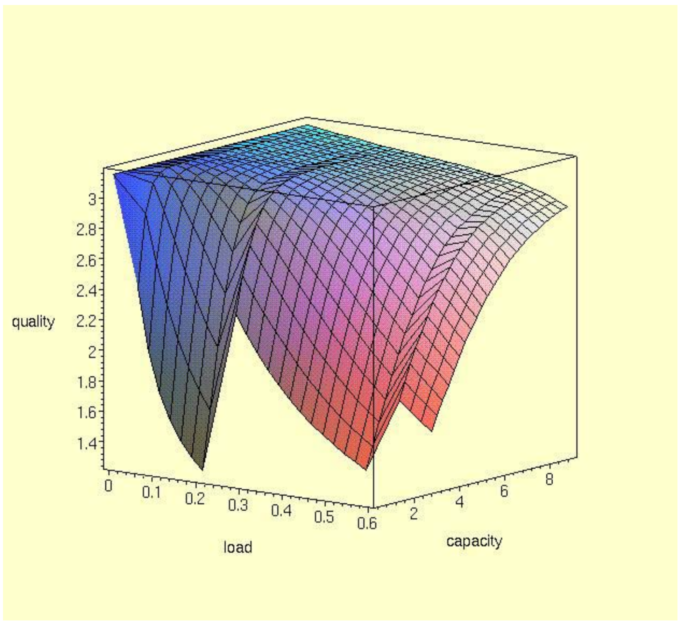


Figure 7.4: Quality as a function of the load and network capacity, without FEC

network. For example, if voice traffic is predominant in the network, and we know that using FEC increases the global load by a factor x, we can assess whether it will be useful or not to enable it, or even if we could get away with a smaller value of W and no FEC (this doesn't seem to be the case here, but it may well be the case for other kinds of applications).

Figure 7.5 shows two "cuts" of figure 7.3, which show the quality as a function of load for two values of W. We can see that a bigger buffer allows for a significant increase in load before the quality degrades as much as when using a smaller buffer. Furthermore, the difference in quality is more pronounced as the load increases.

Finally, Figure 7.6 illustrates the differences in the variation of the derivative of the quality with respect to the load, for two different network capacities.

7.4 Application of the Technique: Studying the Effects of Media–dependent FEC on VoIP Quality

In this section we will use the approach described above to study the impact of FEC on the perceived quality for a VoIP stream. In order to do this, there are two things that we must study, namely:

- how does the increase in bit rate induced by the addition of FEC affect the network load, and

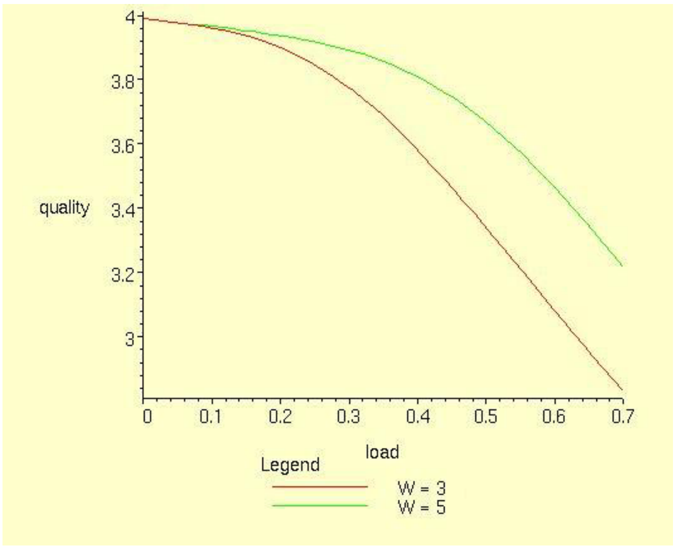


Figure 7.5: Quality as a function of network load, for two values of W, and with FEC (offset = 1).

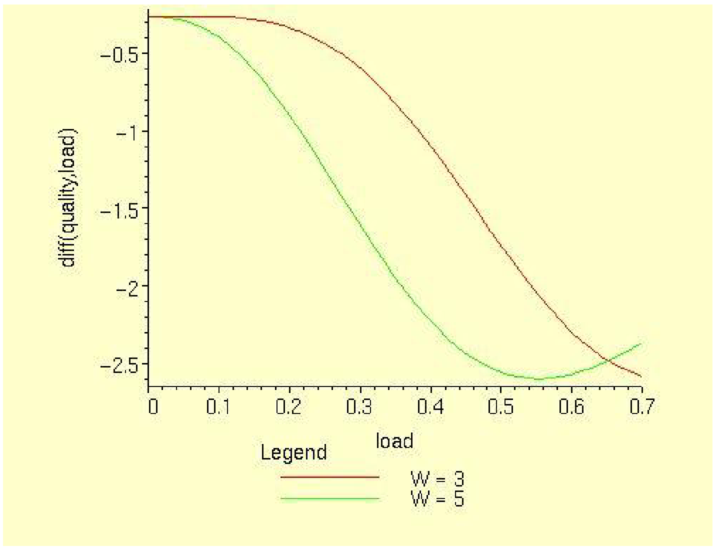


Figure 7.6: $\partial Q/\partial \rho$ as function of ρ

- how is the perceived quality affected by the addition of FEC and the new network state.

To this end we will consider, for the time being, two classes of packets: audio packets and background traffic. Audio packets can have FEC or not, but we will consider that if FEC is on, then all flows are using it. Our bottleneck router will have a drop–tail buffer policy, which is a common practice in current best–effort networks.

7.4.1 Impact of FEC on the Network Load

We start by evaluating the impact of the use of FEC in the global load of the network. This is achieved by studying, for the same proportion of voice traffic, what are the loss rates and mean loss burst sizes without FEC, and with FEC.

7.4.1.1 Transmission Without FEC

First, consider the case of audio without FEC. We will take the standard environment in the $M/M/1/W$ case: Poisson arrivals, exponential services, and the usual independence assumptions. Note that while VoIP traffic normally has a constant rate, we consider the aggregate of voice streams to behave as a Poisson source. The arrival rates of class i units is λ_i pps (packets per second) and the link has a transmission capacity of c bps. The average packet length for class i packets is B_i bits. In order to be able to use analytical expressions, we consider in the model the global average length of the packets sharing the link, B, given by $B = \alpha_1 B_1 + \alpha_2 B_2$, where $\alpha_i = \lambda_i/\lambda$, with $\lambda = \lambda_1 + \lambda_2$. The service rate of the link *in pps* is then $\mu = c/B$.

Let us assume that the buffer associated with the link has capacity equal to N bits. Then, in packets, its capacity will be taken equal to $W = N/B$. To simplify the analysis, we will use the expressions for the performance metrics of the $M/M/1/W$ models even if W is not an integer; this does not significantly change the results and the exposition is considerably simpler.

Recall from Equation (7.1) that the packet loss probability p_L is $p_L = \rho^W(1-\rho)/(1-\rho^{W+1})$ (assuming $\rho \neq 1$). We also need to compute the mean size of loss bursts for audio packets, since the correlation between losses influences the speech quality. Here, we must discuss different possible definitions for the concept of loss burst, because of the multi-class context. To this end, let us adopt the following code for the examples we will use in the discussion: a chunk of the arrival stream at the link will be denoted by "$\ldots, x, y, z, \ldots$" where symbols $x, y, z, \ldots$ are equal to i if a class i packet arrives and is accepted (the queue was not full) and to $\bar{i}$ if the arrival is of class i and it is rejected (because there was no room for it).

Assume that a packet of class 1 (which we will consider to be audio packets) arrives at a full queue and is lost, and assume that the previous class 1 packet that arrived was not lost. Then, a burst of class 1 losses starts. Strictly speaking, the burst is composed of a set of consecutive audio packets all lost, whatever happens to the class 2 packets arriving between them.

For instance, in the path "$\ldots, 1, 2, \overline{1}, \overline{1}, 2, 2, \overline{1}, \overline{2}, 2, \overline{2}, 1, \ldots$" there is a class 1 burst with loss burst size three. If we use this definition of burst, when audio packets are a small fraction of the total offered traffic, this definition can exaggerate the effective impact of correlation (since it can "group" several congestion episodes into one). Even in the case of many audio packets in the global arrival process, allowing the class 2 packets to merge inside audio loss bursts can be excessive. On the other extreme, we can define an audio loss burst as a consecutive set of class 1 arrivals finding the buffer full, without class 2 packets between them. In the path shown before, if we consider this burst definition, there is a first burst of class 2 losses with size 2, then a second one composed of only one packet. Consider now the chunk "$\ldots, 1, 2, 2, \overline{1}, \overline{1}, \overline{1}, \overline{2}, \overline{2}, \overline{1}, 2, 2, 2, \overline{1}, \overline{1}, 1, \ldots$". An intermediate definition consists of considering that we accept class 2 packets inside the same audio burst only if they are also lost (because this corresponds basically to the same congestion period). In the last example, this means that we have a loss burst with size 4. We will keep this last definition for our analysis.

Let us denote by MLBS the mean loss burst size (recall that we focus only on class 1 units). The probability that a burst has size strictly greater than n is the probability that, after a class 1 loss, the following n class 1 arrivals are losses, accepting between them class 2 losses in any number. This means

$$\Pr(LBS > n) = p^n,$$

where LBS is the random variable *Loss Burst Size*, and

$$p = \sum_{k \geq 0} \left(\frac{\lambda_2}{\lambda_1 + \lambda_2 + \mu} \right)^k \frac{\lambda_1}{\lambda_1 + \lambda_2 + \mu} = \frac{\lambda_1}{\lambda_1 + \mu}.$$

The last relationship comes from the fact that we allow any number of class 2 units to arrive as far as their are lost, between two class 1 losses (that is, while the burst is not finished, no departure from the queue is allowed). Using the value of p, we have

$$\mathrm{E}(LBS) = \sum_{n \geq 0} \Pr(\mathrm{LBS} > n) = \sum_{n \geq 0} \left(\frac{\lambda_1}{\lambda_1 + \mu} \right)^n = 1 + \frac{\lambda_1}{\mu}.$$

7.4.1.2 Transmission With FEC

If FEC is used, each audio packet has supplementary data, and we denote this overhead in the following way: If B_1' is the mean audio packet length when FEC is used, then $B_1' = B_1(1+r)$, with $r > 0$. The rest of the parameters are computed as before. We have $B' = \alpha_1 B_1' + \alpha_2 B_2$, $\mu' = c/B'$, $W' = N/B'$ and $\rho' = \lambda/\mu'$. This leads to the corresponding expression for the loss probability $p' = \rho'^{W'}(1 - \rho')/(1 - \rho'^{W'+1})$ and $\mathrm{E}(LBS') = 1 + \lambda_1/\mu'$, with LBS' representing the loss burst size in this case.

7.4.2 The Impact of FEC on VoIP Flows

The main idea of using FEC is that as the real–time requirements of VoIP make it impossible to retransmit lost packets, it is possible to mitigate the effect of the losses by transmitting the information contained in each packet more than once. To this end, we send one (maybe more) copy of packet n's contents in packet $n + i$ ($n + i + 1$, and so on if several copies are sent), i being variable. The extra copies are normally compressed at a lower bit rate, so as to minimize the increase in bandwidth usage. When a packet is lost, if the packet containing the redundant copy of the lost one arrives in a timely fashion, the receiver can use said copy to recover from the loss, in a way that minimizes the perceived quality degradation. The variability of i is due to the bursty nature of packet losses on IP networks [58, 135], and it allows to improve the performance of the FEC scheme described above by avoiding the loss of all copies of a given voice packet in the same loss burst. However, i should remain as close to 1 as possible, in order to minimize the increase of the end–to–end delay, which is an unwanted effect.

In this study we focus on the perceived audio quality, and so we do not consider the effects of FEC on the end–to–end delay, since this affects the conversational quality, but not the voice quality itself.

In order to assess the impact on FEC at the network loss level, we must consider two factors:

- the amount of redundancy introduced by the use of FEC (and therefore the increase in packet size for the flows using FEC),
- the proportion of flows using FEC, which allows us to derive the new network load.

While the relative increase in size is likely to be small, more or less equal for different applications, and certainly bounded by 1, the amount of flows using FEC is very hard to determine. Estimations of the number of VoIP flows on the Internet are not readily available in the literature, but there are some estimations [39] that say that UDP traffic only accounts for about 15 to 20% of the total network traffic. Even then, VoIP is probably still a modest fraction of that UDP traffic (some studies [40] suggest that about 6% of traffic corresponds to streaming applications). However, being that VoIP applications have a growing user base, it seems reasonable that in some time they may account for a higher fraction of the total Internet traffic.

We studied different scenarios, with increasing volumes of VoIP traffic in order to assess the impact of FEC in the quality perceived by the end–user. For simplicity's sake, we assumed that if FEC is in use, then all VoIP flows are using it.

Some studies [26] show that an important percentage (about 60%) of packets have a size of about 44B, and that about 50% of the volume of bytes transferred is transferred on 1500B or higher packet sizes. Table 7.3 shows the distribution we chose for our background traffic packet sizes, which yields an average packet size B_2 of about 600B.

For our example, we will consider a T3–type line which runs at 45Mbps (actual speeds are not really relevant, since we will concern ourselves with the load of the network – numbers are used to better set the example). In practice, buffer space in core routers is not limited by physical memory, but rather by the administrators, who may wish to minimize delays

Size (Bytes)	**Probability**
40	50
512	0.25
1500	0.24
9180	0.01

Table 7.3: Packet size distribution

(while this implies that the loss rate is higher than it could be, it is important to make TCP and other congestion–responsive flows behave properly). This time is normally limited to a few milliseconds [127]. We will use a value of 200ms (which requires a buffer space of $N = 45\text{Mbps} * 0.2\text{s} = 9\text{Mbits}$, or about 1965 packets of 600B), which is on par with current practices.

We considered that network load varies between 0.75 and 1.15. Smaller values were not considered, since in the model we used, they result in low loss rates that result, in turn, in a relatively good quality. Higher load values result in loss rates above 15%, which typically yield unacceptable quality levels. It is between the chosen values that the quality shows an inflection point.

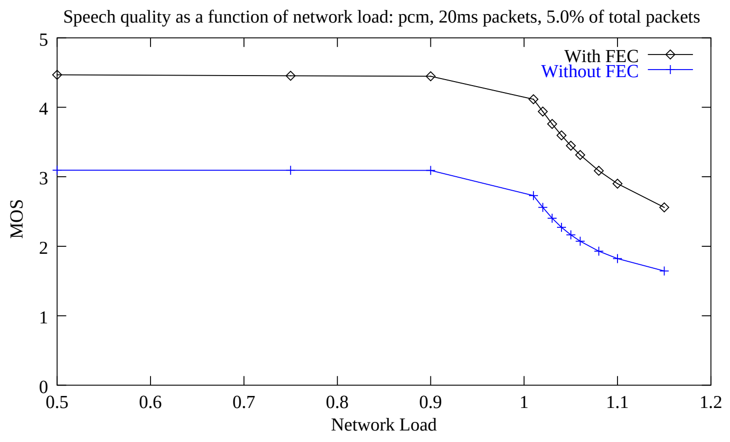
As for the fraction of VoIP traffic, we studied different values, between 5% and 50% of the total packet count. Granted, the higher values are currently unrealistic, but they may make sense in a near future if more telephony providers move their services toward IP platforms, and other VoIP applications gain more acceptance.

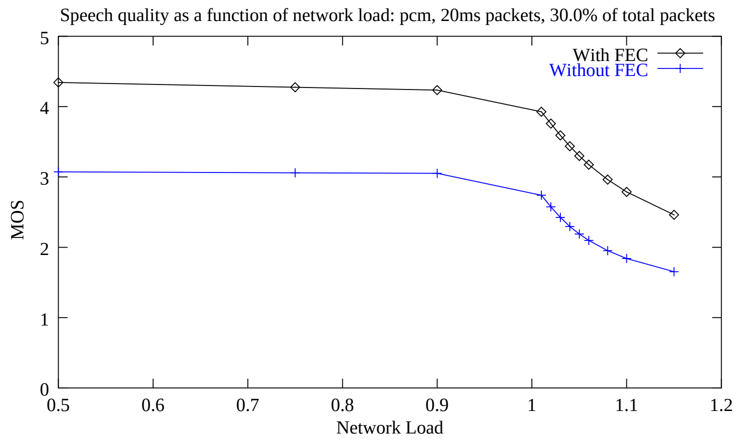
We chose to use PCM audio with GSM encoding for FEC, and an offset of one packet for the redundant data. This is a bit pessimistic, since sample–based codecs are not very efficient for network transmission, but it allows us to get a worst–case scenario of sorts (since increasing the fraction of audio traffic does indeed increase network load). In this case, the increase in packet size is of about 10% when using FEC, which gives payloads of 353B, against 320B of PCM–only traffic for 20ms packets. We also tried GSM/GSM, which results in a much smaller packet size (66B and 33B with and without FEC respectively), but found that the results are qualitatively similar to those of PCM, and so we will discuss only the PCM ones.

7.4.2.1 Assessing the Impact of FEC on the Perceived VoIP Quality

We present the results obtained as a series of curves, plotting the estimated MOS values against network load, and for different values of the proportion of voice packets.

As can be seen in the curves in Figure 7.7, using FEC is beneficial for the perceived quality in all the conditions considered. It can be seen that when the proportion of voice traffic becomes important ($> 30\%$) the performance of the FEC protection decreases. We believe





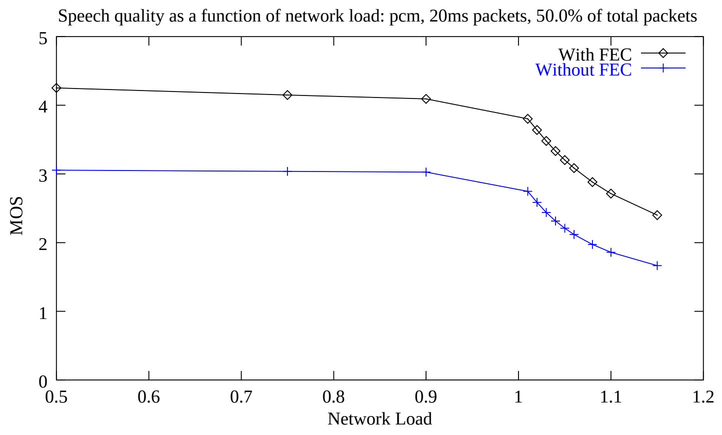


Figure 7.7: MOS as a function of network load for different fractions of voice traffic, with and without FEC (20ms packets).

that this is related to the fact that a higher proportion of voice packets implies a higher mean loss burst size (which is still smaller than 2 in our model for the conditions considered), and being that we chose an offset of 1 packet for the FEC, it is logical to see a slight decrease in performance. We did not find, however, a negative impact of FEC on the perceived quality, as predicted in [6, 7]. Even when most of the packets carry voice with FEC, we found that it is better to use FEC than not to use it. It might be that for extremely high values of the network load, this will not hold, but in that case quality will be already well below acceptable values, FEC or no FEC, so it doesn't really matter.

The second strong fact coming from these numerical values is that the qualitative behavior of perceived quality when load increases is basically the same in all cases, with a significant degradation when load approaches 1 and beyond.

We also found that using a larger packetization interval can help improve the perceived quality. We tried doubling the packet size, obtaining 40ms packets (20ms and 40ms packets are commonly used in telephony applications), and obtained a slight increase in quality even for higher proportions of audio packets (25% of audio packets, which corresponds to the same number of flows of a 50% 20ms audio packet proportion). This can be seen in Figure 7.8. While increasing the packetization interval is beneficial in one–way streams, it should be studied whether the higher delay values that this creates do not counter these benefits.

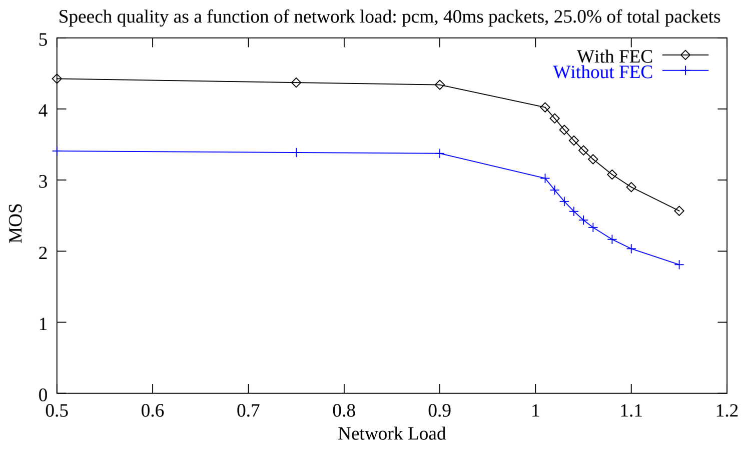


Figure 7.8: MOS as a function of network load for 25% of voice traffic, with and without FEC (40ms packets)

7.5 Summary

We have presented a method for integrating the PSQA approach with traditional performance evaluation techniques such as queuing analysis. This integration provides useful insight when designing and dimensioning a network, since it allows to obtain accurate estimations of what the perceived quality of media streams will be like in the network considered. We have also provided an example application of the proposed technique, studying the effects of network load and capacity on VoIP quality for a simple $M/M/1/W$ model. Moreover, we have provided a closed–form expression for the perceived quality in this model, based on the simplified RNN architecture described in Section 5.4.1. Note that this illustrates an important difference between our approach and standard performance evaluation analysis using models: for instance, instead of dimensioning the system in order to keep, say, the loss probability or the delay below a certain threshold, we dimension in order to keep the *perceived quality* above a minimal (acceptable) value.

We have then studied the effects of FEC on voice streams' quality, using a multi–class extension of the $M/M/1/W$ model mentioned above. We have performed an analysis similar to those found in [6, 7]. Those studies used arbitrary utility functions in order to assess the influence of FEC on the perceived quality. The results presented therein conclude that the usefulness of FEC is highly dependent on the utility functions used, and that it can even have a negative impact on quality. Our work provides a more accurate assessment by using PSQA instead of utility functions, and shows that under reasonable load conditions, and using a widely–deployed FEC scheme, the use of FEC is always beneficial to the perceived quality.

In the next chapter, we will use the technique described in order to assess the quality of VoIP streams in a wireless networking context. However, as the network model proposed is quite complex to be treated analytically, we use simulation techniques to derive the necessary network performance metrics.

Chapter 8

VoIP Quality in a Wireless Context

In this chapter we consider a wireless home networking scenario which we believe will be common in a near future. We propose a model for the wireless network and use the technique introduced in Chapter 7 to predict the performance for voice and video streams in the proposed scenario.

8.1 Introduction

It is well known that most of the QoS–related problems are currently found in the access networks, and not in the network core or in the local area networks. The recent explosion in the number of homes with broadband connections, coupled with the development of wireless technologies such as IEEE 802.11, and new services being currently marketed might soon change that situation. As more and more devices are networked, we may see a new bottleneck, namely the wireless LAN (WLAN). The most widely deployed wireless technologies for home networks are currently IEEE 802.11b and 802.11g, which provide nominal top bandwidths of 11 and 54Mbps respectively. To a lesser extent, there are also Bluetooth connected devices, which can work at about 2Mbps in good conditions. In practice, the throughputs observed in *wi–fi* networks range from 4–5Mbps for 802.11b to 20-25Mbps to 802.11g. Besides, the network performance varies depending on the topology of the network, environmental parameters, and interference from other equipment. These performances are low when compared to current Ethernet LANs (100Mbps – 1Gbps), but the lack of cabling makes wireless networking an attractive option for home use.

As new devices, applications and services continue to be developed, it is very likely that the load imposed on the network will increase, shifting the bottleneck from the access network to the wireless links. As of this writing, there are already several ISP offering TV channels over DSL lines, and VoIP services, for what can be considered very cheap prices. It is reasonable to assume that this will further increase the adoption of broadband and of these services. The next logical step is to allow for wireless TV sets, music players, and the like to tap into the home network in order to get their data. Considering that a single TV–quality stream may require between 1.5 and 10Mbps of bandwidth, depending on encoding and quality parameters (and that's just TV, not HDTV), the 5 to 25Mbps (shared by all stations for down–link, up–link and local traffic) offered by the current crop of *wi–fi* networks suddenly seem a bit limiting. Furthermore, one cannot forget other uses of the network, such as mail, WWW, file transfers (notably P2P, which is quite demanding in terms of bandwidth), gaming, etc.

Besides the limited bandwidth available in *wi–fi*, the nature of wireless networks is such that their performance is very variable. Impairments due to the position of the hosts, multi-path propagation, hosts with weaker signal strength, etc. might drastically decrease the network's performance at any time. This in turn implies a strong variability in the perceived quality of real–time applications such as voice streams, which may have a big impact on the adoption of these new technologies. In fact, as these services are not generally free, their perceived quality should be at least comparable to that of their traditional counterparts, which means at least *toll quality*[1] for VoIP.

Recent results [90] show that VoIP performance over wireless networks is quite limited. In that paper, the authors used the network only for VoIP flows (which are not especially bandwidth–hungry), and showed that the quality was severely degraded even for a relatively low number of flows. When considering other applications which may be found in a media–centered home network and physical aspects such as the spatial distribution of the hosts, it is clear that VoIP quality will decrease. It is therefore important to know how to protect these flows in order to preserve quality.

In this chapter we study the networking context presented above, and based on that, we present a stochastic model for the wireless networking context we consider. This model allows us to simulate the variations in network performance, and see how they affect the perceived quality of the streaming applications. We then use PSQA on the simulation traces (using in fact the method described in Chapter 7) to provide a view of the evolution of the perceived quality with time. We also evaluate the performance and shortcomings of media–dependent FEC in this context.

8.2 The Network Model

As mentioned above, we place ourselves in a home network context, where a high–speed xDSL or cable connection is available, and a wireless router provides access to it for different kinds of devices. Although in this study the actual speeds are not as relevant as the load of the network, we will use values similar to those typically found in an 802.11b *wi–fi* network, which might include the upcoming 802.11e [66] QoS mechanisms. In our model, several devices such as PCs, PDAs, gaming consoles, TVs, etc. are connected to the wireless network in order to access the Internet, and to have connectivity among them. Some of these devices run "standard" TCP or UDP applications, such as file transfers, e–mail, P2P, etc. while others run real–time applica-

[1]That is, the quality to which the user is used to, which for telephony is the one provided by the PSTN.

tions (normally RTP over UDP traffic), such as VoIP, video on demand, video-conference, or multi-player gaming. We assume that most of these applications are working over the Internet, and therefore the bulk of the traffic has to pass through a unique *wi–fi* router / access point (AP) and share the radio channel.

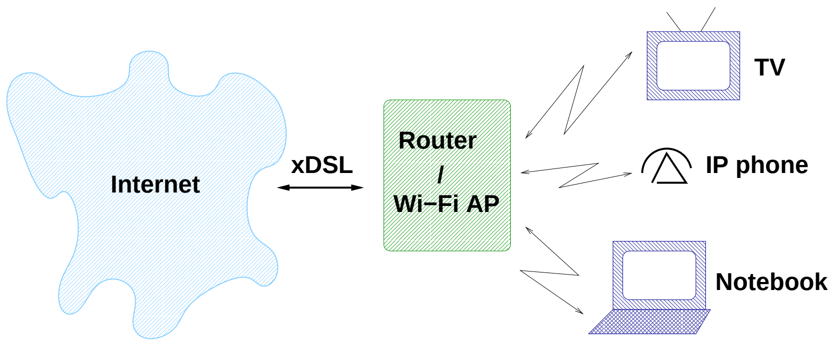


Figure 8.1: Simplified view of the networking context considered.

The wireless network's performance is very variable. Factors such as signal–to–noise ratio (SNR) variations due to internal and external interferences (which in turn depend on the radio frequency used – 2.4GHz or 5GHz), signal fading, access contention mechanisms, and multi-path propagation all contribute to the degradation of the actual network's performance.

Variations in the SNR often imply a need to adapt to a fluctuating radio channel (Link Adaptation Mechanism) by means of more robust coding. This comes at a price, namely a drop in the effective bandwidth. The standards allow for several functioning modes, each one resulting in a smaller effective bandwidth, so globally, the available bandwidth of a *wi–fi* network varies in a discrete fashion among several levels. For instance, for 802.11b, there are four operation modes [102], as shown in Table 8.1.

Mode	Capacity	Theoretical throughput
1 Mbps	0.93	0.93 Mbps
2 Mbps	0.86	1.72 Mbps
5.5 Mbps	0.73	4 Mbps
11 Mbps	0.63	6.38 Mbps

Table 8.1: 802.11b operation modes and maximal theoretical speeds

When too many losses are detected, or when the SNR is too low, the network will go into the next lower–bandwidth mode. This behavior makes for violent variations in the available bandwidth, which greatly impact the performance of real–time applications. Other factors can also degrade network performance. For example, if a host is having a weak signal, it will use a low–bandwidth mode, in turn decreasing the effective bandwidth for the other hosts [61]. This happens because the contention avoidance mechanism will give all stations the same probability of access to the radio channel, and stations with a lower rate mode need more time to send the same amount of data. The transmission can also experience short loss bursts [114] due to temporary radio perturbations. These loss bursts induce delay spikes due to contention at the MAC level, which in turn may induce losses at the IP level, since the queues at the AP may overflow.

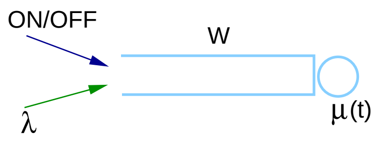


Figure 8.2: The queuing model used. The service rate varies over time, as the available bandwidth in a *wi–fi* network. The buffer capacity (in packets) is W, and traffic is Poisson for real–time flows (we consider the aggregate of those streams), and ON/OF for background traffic.

For our experiments, we used a simplified network model, like the one depicted in Figure 8.2. We represent the network bottleneck (the AP in this case) by means of a multi-class $./M(t)/1/W$ queue, in which the server capacity varies with time in a way similar to that of a *wi–fi* network. The server capacity hops among four different levels, the time spent on each one being an exponentially distributed random variable. After that, it either goes up (or stays at the maximum) or down (or stays at the minimum) one level. Our traffic is split in two classes, background and real–time. Our background traffic represents file transfers (web or FTP), P2P applications, e–mail, etc. This traffic is modeled using an ON/OFF process, which allows to control its burstiness. The real–time traffic is an aggregate of audio, video, voice, game streams, etc. This kind of traffic is less bursty, so we modeled it as Poisson process. We considered that real–time traffic accounted for 50% of the packets transmitted.

As in Chapter 7, and based on the data used in [78], we chose an average packet size of about 600B, which we used to calculate the different service rates of the AP. The queue size of the router/AP was set to 250 packets, and the scheduling policy used was FIFO. In order to keep the model as simple as possible, we did not represent the aforementioned short loss spikes on the wireless link. This should not be a problem, since at the load levels we considered, most of the losses will be due to IP–level congestion at the router/AP. In the situation considered, there are very few losses caused by delay spikes at the MAC level, compared to those caused by the bandwidth limitations of the WLAN.

Figure 8.3 shows a bandwidth trace for a 200s period using the model described above. In Figure 8.4 we can see the loss rates for real–time traffic (averaged every 5s) over the same

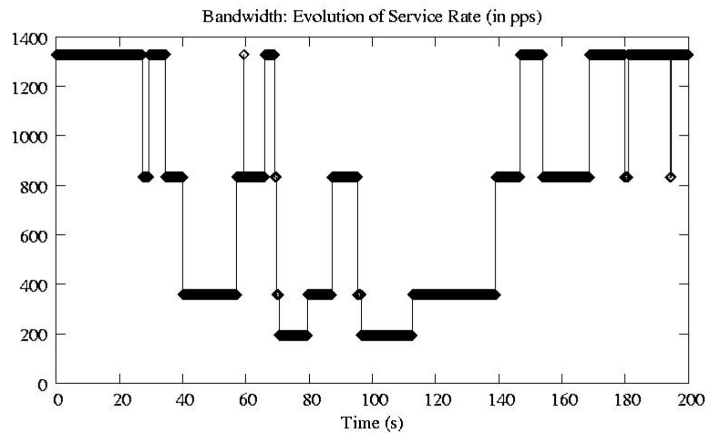


Figure 8.3: Sample 200s bandwidth trace. The mean throughput was set to 800pps for this simulation.

200s period and for a medium load case (about 800 pps of global offered traffic). It is easy to see how the loss rate varies with the available bandwidth. Normally, voice streams are quite resilient to network losses, but with values as high as the ones obtained, there will be very noticeable drops in the perceived quality. Furthermore, the loss process is such that relatively long bursts are common. In Figure 8.5, we can observe mean loss burst size (MLBS) values as high as 5, which have a clearly negative impact on the effectiveness of the FEC scheme studied.

8.2.1 Quality Assessment of the Simulation Results

In this section we present some VoIP and video[2] PSQA results for some traces representative of our simulation results. It can be seen that the perceived quality is quite poor in the conditions we considered. Figure 8.6 shows the perceived quality of a VoIP flow for the bandwidth variation pattern seen in Figure 8.3.

Recall that in this scale, an MOS score of three points is considered acceptable. Also, with the coding parameters used (codec, forward error correction, packetization interval, etc.), the top quality that we could expect is slightly above four points (the optimal value corresponding to the MOS value associated with the codecs used). We can see that in this case the quality is sub–par, except for relatively calm periods at the beginning and at the end of the simulation. The high loss rates observed, together with an increase in the mean loss burst size during those congestion periods render FEC almost useless in this circumstances. Although it does slightly improve the perceived quality with respect to non–protected flows, it cannot provide enough protection to achieve acceptable quality levels in this context. Besides the low quality scores,

[2]The RNN used for the video quality assessment was trained by Mohamed for [98].

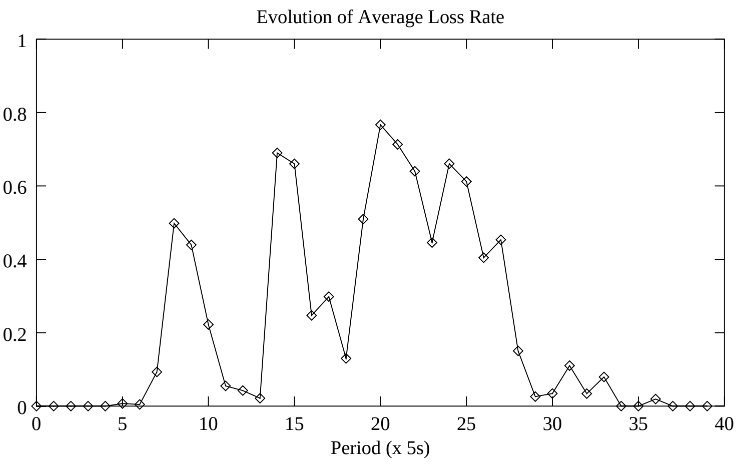


Figure 8.4: 5–second average loss rates (real–time traffic) for the same 200s period

the high quality variability with time makes for an even worse impression with the end–user.

The next figures show VoIP PSQA results for different bandwidth and traffic patterns. Note that each run had a different traffic pattern, so similar bandwidths do not necessarily translate into similar MOS values. As in the previous case, the perceived quality is well below acceptable levels most of the time.

Figures 8.11 and 8.12 respectively show the bandwidth evolution and QCIF video PSQA scores for another 200–second simulation. Here we also found that the quality was poor, as expected.

It is easy to see that the congestion at the AP is generating very high loss rates and long loss bursts, which translate into bad quality scores. Loss rate traces for the second, third and fourth examples behave in a similar fashion of those for the first example (cf. Figure 8.4). In Chapter 9 we explore some ways to mitigate this problem and to improve the perceived quality whenever possible.

8.2.2 The Influence of FEC on the Perceived Quality

As seen in Chapter 7, the impact of FEC in the overall network load is negligible for "reasonable" load levels (reasonable meaning that MOS scores were at least close to toll quality). The media–dependent FEC scheme used produces a 10% overhead for voice flows, which on themselves contribute a small part of the total load. Moreover, we found that for the same reasonable load values, using FEC was always worthwhile ([116]). Therefore, for this analysis we did not take into account the difference in overhead between FEC and non-FEC packets, and

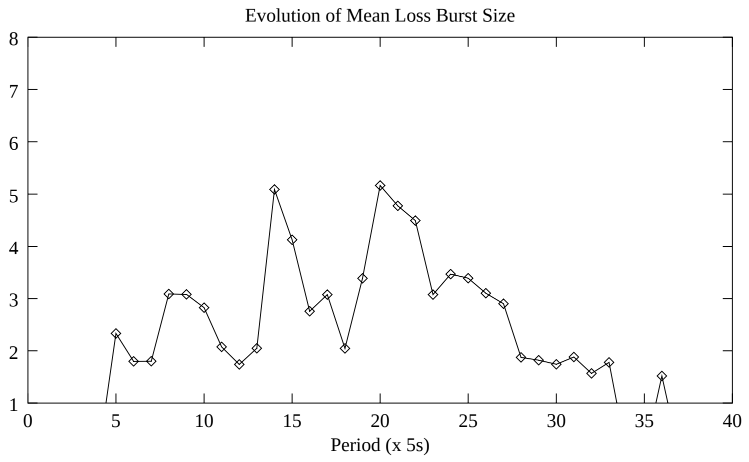


Figure 8.5: Mean loss burst sizes (sampled every 5 seconds for real–time traffic) for the same 200s period

we bundled them all together with the rest of the real–time traffic (such as video and on-line games).

In this section we present quality assessment results for several simulation runs. As the network performance is very variable, we do not present the results as in Chapter 7, where we plotted PSQA scores versus network load values. Instead, we chose several low–to–medium mean load values for our simulations, and run those in 200–second intervals, to understand how the quality evolves with time.

Figure 8.13 shows MOS estimations for a voice stream transmitted during the 200s period shown in Figures 8.3, 8.4, and 8.5. For this simulation, we chose a mean throughput of 800 pps. It can be seen that most of the time, the quality is lower than 3, which would be unacceptable to most users. As observed in our previous study of FEC [116] (cf. Section 7.4, on page 134), the FEC–protected flow has a higher quality than the unprotected one (it is important to note that a difference of less than 0.5 MOS points is very difficult to appreciate for most people, so this gives an idea of the improvement obtained when using FEC). However, the difference in quality is generally smaller and much more variable in this wireless context than in the wired one. This fact can be explained by the higher loss rate, and especially the higher MLBS values. In wired networks, losses seem to be less correlated, and the most commonly observed values for MLBS vary between 1 and 2.5 [17, 135].

Figures 8.14 and 8.15 show PSQA scores for two other 200-second simulation runs. The mean throughput for those cases was of 875 and 450pps respectively. It can be seen that although the network load variation was important, and in general well below nominal capacity,

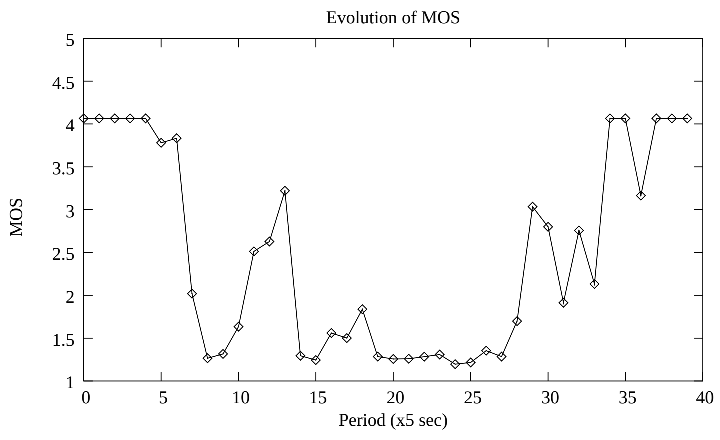


Figure 8.6: PSQA values for a VoIP flow and the bandwidth variation pattern seen in Figure 8.3

the sheer variability of the available bandwidth and the bursty nature of traffic can produce a severe degradation of the perceived quality.

In order to show the influence of the higher MLBS on quality, we also used a higher value for the FEC offset (3 packets instead of only one), which allows for a better perceived quality under the same circumstances (cf. Figures 8.13 through 8.15) It may be tempting to abuse this technique in order to compensate for the higher MLBS and loss rates, but it should be noted that increasing the FEC offset does also increase the end–to–end delay, so until a clearer understanding of how this affects interactivity, we cannot be sure of the real benefits of increasing the offset too much.

Although there is a visible improvement in the perceived quality when using a higher FEC offset, there are still significant periods of time when the MOS scores fall below toll quality, and this even in cases where the overall network load is not very high. This indicates that the use of FEC alone is not sufficient in this context. There are several other methods for improving the quality, ranging from application–level ones such as loss concealment or sample interleaving [106] to network–level approaches such as DiffServ.

8.3 Summary

In this chapter we have presented a likely scenario for future home networks, in which several multimedia devices (PCs, notebooks, PDAs, TV sets, IP phones, etc.) share a wireless link in order to obtain the data they need. We have then proposed a stochastic network model for the wireless network, that mimics the behavior of current *wi–fi* technologies, which are the norm for home WLANs. Based on performance metrics obtained via simulations over this network

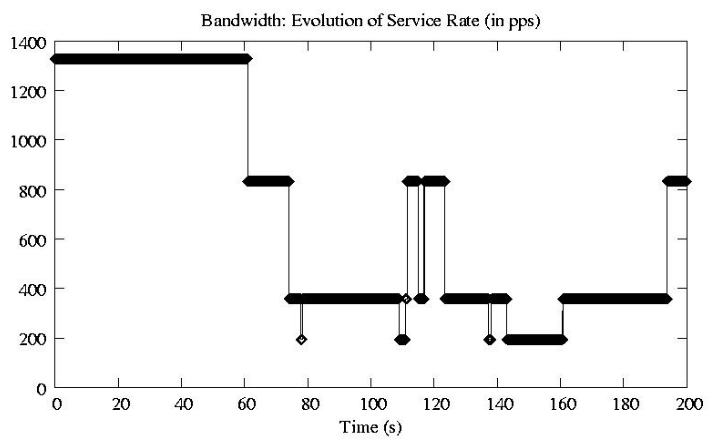


Figure 8.7: Second 200–second example: evolution of bandwidth.

model, we have used PSQA to study the performance of media streams in the scenario considered.

We have studied the performance of VoIP streams, and to a lesser extent, of QCIF video streams, which are commonly used for video–conferencing. The performance results that we have obtained clearly show that the quality levels obtained are not acceptable. Even the use of forward error correction, which normally produces a significant improvement in the perceived quality falls short in this wireless context.

It is therefore interesting to study other ways of improving the quality of the media streams, such as IP level priority management, for example. In the next chapter we propose a proof–of–concept IP–level priority scheme for the WLAN, and show how it would work in the context considered. We also show that as such a scheme may result in starvation for other traffic, it should be controlled dynamically, so as give priority only to those flows that really need it, and when they need it. This can be done by monitoring the perceived quality in real–time with PSQA.

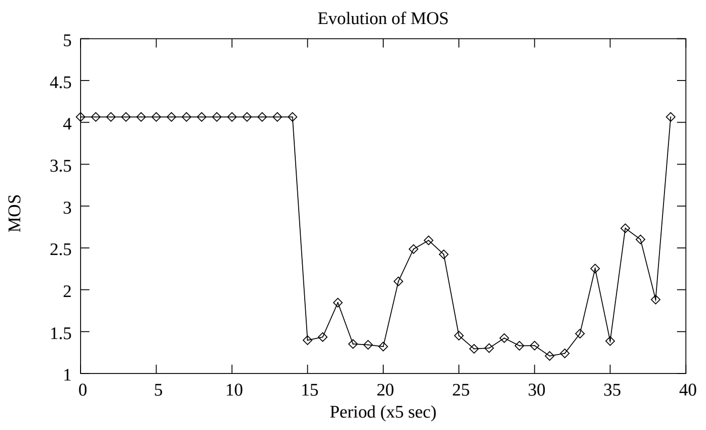


Figure 8.8: Second 200–second example: MOS variation for VoIP traffic.

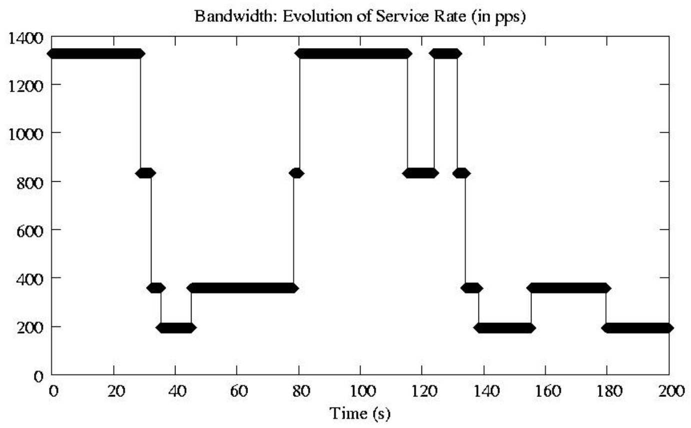


Figure 8.9: Third 200–second example: evolution of bandwidth.

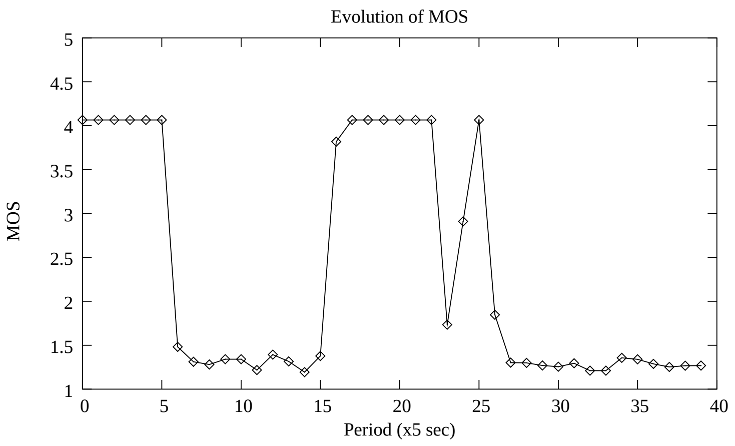


Figure 8.10: Third 200–second example: MOS variation for VoIP traffic.

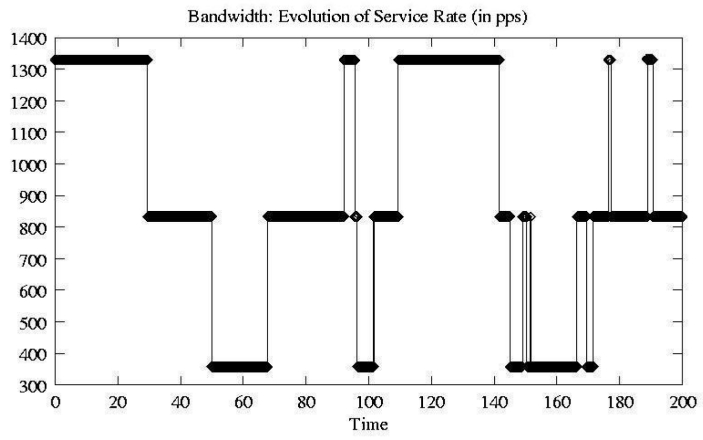


Figure 8.11: Fourth 200–second example: evolution of bandwidth.

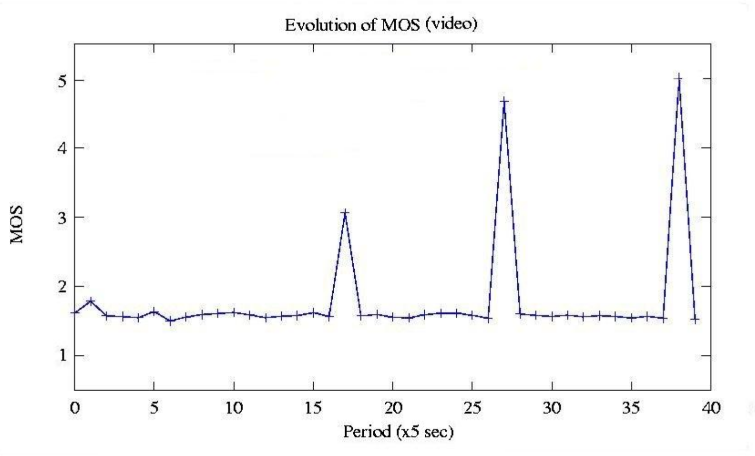


Figure 8.12: Fourth 200–second example: MOS variation for QCIF video traffic

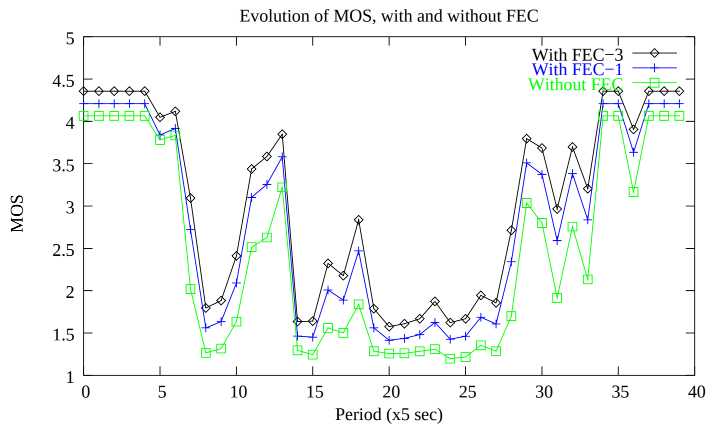


Figure 8.13: PSQA results (sampled every 5 seconds) for the same 200s period as in Figure 8.3. FEC–protected streams are using offsets of 1 and 3 packets respectively for FEC data.

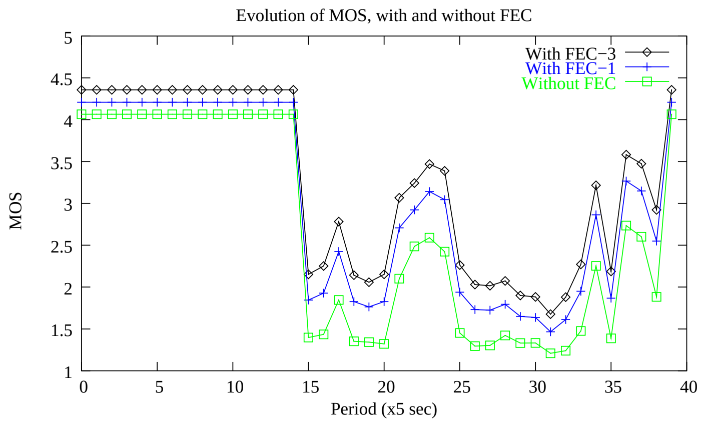


Figure 8.14: PSQA results (sampled every 5 seconds) for the same 200s period as in Figure 8.7. FEC–protected streams are using offsets of 1 and 3 packets respectively for FEC data.

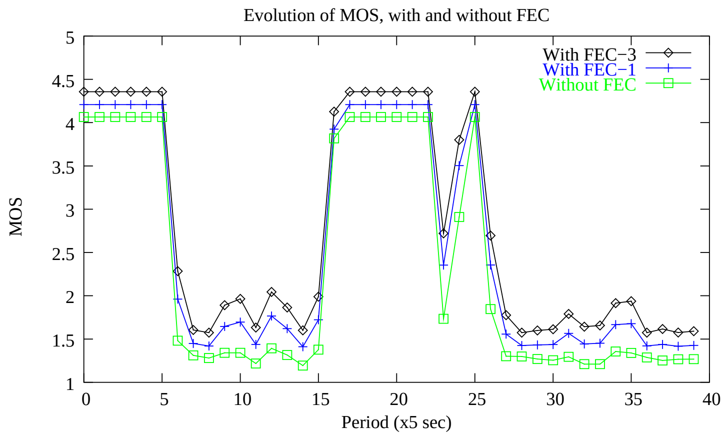


Figure 8.15: PSQA results (sampled every 5 seconds) for the same 200s period as in Figure 8.9. FEC–protected streams are using offsets of 1 and 3 packets respectively for FEC data.

Chapter 9

Control Issues

In previous chapters we have presented and analyzed the characteristics of a tool which is able to provide a good insight into what defines the quality of a multimedia stream. We have also presented an example of its use for understanding the behavior of VoIP streams' quality in several networking contexts. More precisely, this tool allows to analyze, in real–time, how the different parameters having an impact on the perceived quality do affect it. We have also described different contexts and situations for which the perceived quality is not as good as we would like it to be. In this chapter, we provide a first approach to quality–driven control mechanisms using our quality assessment technique. It is worth noting that while there exist dynamic quality control algorithms in the literature, they are generally based on simple network metrics, and therefore do not explicitly take the perceived quality into account. Our aim in this chapter is to show that PSQA can indeed be used to improve the perceived quality of a stream, or to keep it within certain bounds in order to help control resource allocation if need be. Note that this makes explicit use of PSQA's ability to provide an accurate QoS estimate in real–time.

In Section 9.1 we present two simple real–time control algorithms for one–way VoIP streams, and we provide an analysis of their performance, by comparing the results obtained to the ones obtained when no dynamic control is used. We also consider possible extensions to these simple schemes to be used for two–way VoIP flows.

Section 9.2 deals with the wireless scenario presented in Chapter 8. We discuss what could be done to improve the perceived quality at different levels of the network / application stack, and we present a simple DiffServ–like priority scheme for real–time flows within the WLAN. An assessment of its performance shows how it improves the perceived quality, and why some form of dynamic control over which packets are marked is needed, lest other flows be starved.

9.1 Proof–of–Concept Dynamic VoIP Quality Control

In Section 6.1 we have studied the impact of several parameters on the perceived quality of voice streams. Of the six parameters we considered (cf. Table 5.2 on page 86), four were application–level parameters, which means they can be easily controlled from within the application itself. The remaining two concerned the network, and in a best effort context, they are not available for control purposes. If some form of QoS provisioning is implemented in the network, they might be (indirectly) controlled, either via packet marking, or by modifying the network's QoS parameters.

Here we will focus only on the application–level parameters, which are easily accessible and modifiable. The algorithms described in this section are designed for one–way flows, but it should be fairly easy to adapt them for use in an interactive context, as discussed below.

9.1.1 A Naive Approach

The first algorithm we propose is a very simple one, which tries to improve quality by manipulating the codec, packetization interval (PI), and FEC settings. It makes a sparing use of FEC, only when the quality is dropping below a certain acceptable threshold. When the quality reaches a specific hysteresis threshold, the FEC is removed (if, that is, removing it does not cause the quality to drop below the acceptable threshold). We have based our implementation on the parameters described in Table 5.2, and so we have two codecs, and several possible values for the packetization interval (ranging from 20 to 80ms) and FEC settings. Of course, the algorithm could be easily adapted to include other parameters. The algorithm proposed is given in Algorithm 2.

We start by checking if a change of codec and / or packetization interval would increase quality, and make the respective adjustments if necessary. Then we check, if we are above the hysteresis threshold, if we can safely turn off the FEC without getting the perceived quality below the acceptable threshold. Finally, if the quality is below this acceptable threshold, we turn on the FEC. Further tuning could be added by manipulating the FEC offset, if needed. We chose to set it to three packets in this case, which should be enough to withstand MLBSs found in normal usage. The curves showing the performance of this algorithm were created using values of 3.1 and 3.8 MOS points for the acceptable and hysteresis thresholds, respectively. While we have not yet performed an in–depth analysis of these parameters, it is easy to see that they shouldn't be chosen neither too close (as turning off the FEC would likely make the quality drop below acceptable values, and the hysteresis control would therefore be useless) nor too separated, in order to avoid violent variations in the perceived quality, which are easily noticed by the user. The quality assessment and control were performed once per second, and the network metrics were calculated with a moving average for a five–second window.

Figures 9.1 and 9.5 show the performance of this algorithm under five different sets of network conditions. The three plots shown in Figure 9.1 were generated using Sue Moon's traces (cf. [135]), which exhibit small loss rates and MLBS values. The starting configurations used the PCM codec, a packetization interval of 20ms and no FEC. We can see that in this case, the algorithm performs almost as well as the "FEC always ON" scenario. However, the debugging output from the control algorithm (see Figures 9.2 through 9.4) show that FEC was not always

Algorithm 2 A naive algorithm for dynamic VoIP quality control.

```
1: get parameters from application
2: get network metrics
3: if getMOS(currentCodec) < getMOS(alternateCodec) then
4:     currentCodec ←alternateCodec
5: end if
6: if currentPI < maxPI then
7:     mosPI–up ←getMOS(currentPI×2)
8: end if
9: if currentPI > minPI then
10:    mosPI–down ←getMOS(currentPI/2)
11: end if
12: if mosPI–up > mosPI–down then
13:    alternatePI ←mosPI–up
14: else
15:    alternatePI ←mosPI–down
16: end if
17: if getMOS(currentPI) < getMOS(alternatePI) then
18:    currentPI ←alternatePI
19: end if
20: currentMOS ←getMOS(currentConfig)
21: if (currentFecState = ON) and (currentMOS >= HYSTERESIS–THRESHOLD) and (get-
    MOS(noFEC) >= ACCEPTABLE–THRESHOLD) then
22:    currentFecState ←OFF
23: else if currentMOS < ACCEPTABLE–THRESHOLD then
24:    currentFecState ←ON
25: end if
```

used, which helps reduce delay, and consumes less resources.

The debugging logs indicate that for the cases considered, increasing the packetization interval always increases the perceived quality. This would not necessarily be the case if the delay was being considered, since the increased length of the packets imply a longer delay, which in turn should have a negative impact on the quality. A similar reasoning can be applied to the use FEC, and the values of the redundancy offset. It is interesting to see that only minor modifications are needed to adapt the algorithm for use in an interactive scenario. It would just be necessary to consider the added delay when using larger PIs or when adding FEC. A more advanced interactive algorithm could also adjust the dejittering buffer size based on the current jitter.

In Figure 9.5 we see that under more severe network conditions the control algorithm offers a better quality than static parameters. Even in these cases, FEC is not always used, as shown in the debugging logs (Figures 9.6 and 9.7). In both sets of scenarios, the perceived quality remains between both thresholds, and while it varies, the variation is relatively small, and therefore unlikely to be a nuisance to the user.

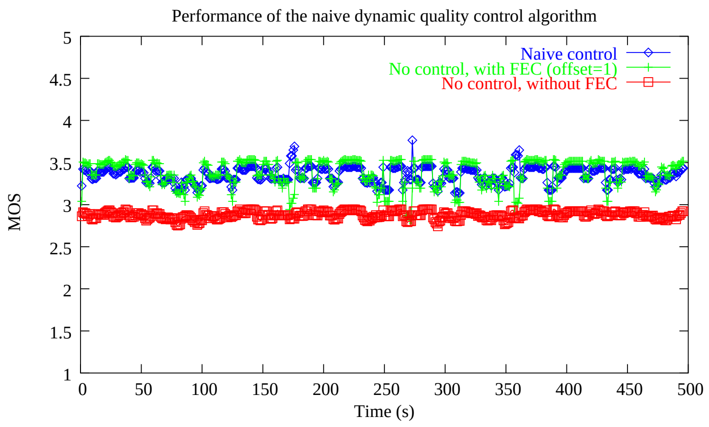
9.1.2 Bandwidth–aware Quality Control

In the previous section we have presented a first approach to dynamic quality control. Looking at the debugging output, we can see that once FEC is used, the algorithm will most likely set the codec to PCM (which, as seen in Figure 6.9, provides better quality than GSM when FEC is used). While this is good from a pure quality standpoint, it is worth noting that this PCM codec has a high bit rate, which is further increased if FEC is used. GSM, on the other hand, consumes only a small fraction of the bandwidth required by the PCM codec. It is therefore interesting, for example in a wireless context for which we want to minimize the bandwidth consumption[1] to use GSM as much as possible.

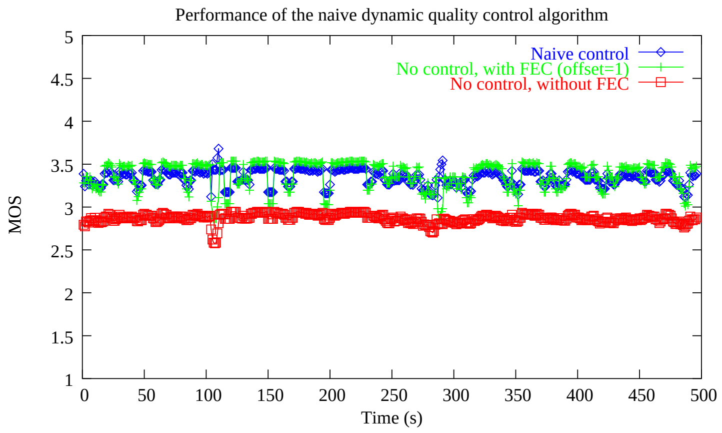
Seeing that in any case using GSM with FEC consumes less bandwidth than using PCM, we propose a second algorithm which favors the use of GSM, with FEC if necessary, instead of using PCM. We only switch to PCM if using GSM with FEC is not enough to attain acceptable quality levels. As we did for FEC in the first “naive” method, we switch back to GSM (and turn off FEC) whenever possible. The new procedure is described in Algorithm 3.

Figures 9.8 and 9.9 show the performance of this algorithm for the same network conditions as in Section 9.1.1. Unlike in the previous case, we start by using GSM instead of PCM (the static assessments, however, are done for configurations using PCM). We see that when using dynamic quality control, the quality is at least as good as when no control is used, and as the network conditions degrade, the difference in performance becomes significant (about 0.5

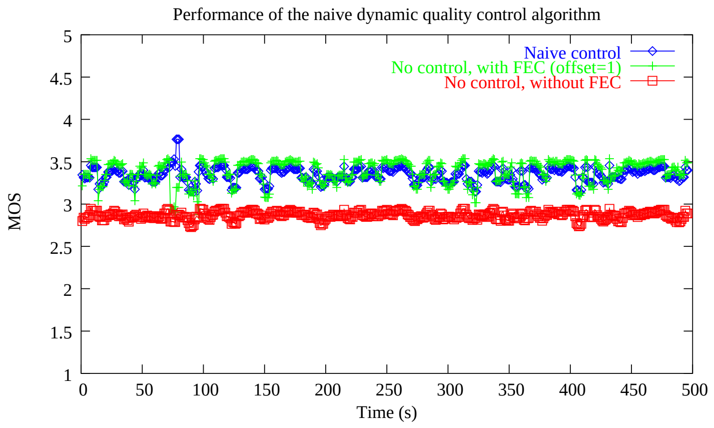
[1]Of course, one could argue that in such a context, PCM would be a bad choice, and indeed it would be. However, the same reasoning applies to other, less bandwidth–hungry codecs.



(a)



(b)



(c)

Figure 9.1: Performance of the naive control algorithm. The network parameters found in these traces are as follows: (a) LR = 1.69%, MLBS = 1.21 packets, (b) LR = 3.37%, MLBS = 1.23 packets, and (c) LR = 2.22%, MLBS = 1.18 packets. Note that the control algorithm only uses FEC when needed, and yet reaches almost the same quality levels achieved when FEC is always used, while consuming less network resources.

```
[1] Changed codec (now:1)  orig: 0.572489  modif: 0.584105
[1] Changed PI (now:0.5)  orig: 0.584105  modif: 0.598312
[1] Below acceptable threshold, setting FEC  orig: 0.598312  modif: 0.644674
[2] Changed codec (now:0)  orig: 0.70532  modif: 0.777398
[2] Changed PI (now:1.0)  orig: 0.777398  modif: 0.800642
[2] Over hysteresis threshold, disabling FEC  orig: 0.800642  modif: 0.684392
[172] Below acceptable threshold, setting FEC  orig: 0.619798  modif: 0.69801
[177] Over hysteresis threshold, disabling FEC  orig: 0.798925  modif: 0.682185
[268] Below acceptable threshold, setting FEC  orig: 0.607697  modif: 0.67694
[274] Over hysteresis threshold, disabling FEC  orig: 0.802365  modif: 0.686614
[357] Below acceptable threshold, setting FEC  orig: 0.611572  modif: 0.676396
[362] Over hysteresis threshold, disabling FEC  orig: 0.774469  modif: 0.66363
```

Figure 9.2: Debugging output of the naive algorithm for the trace used in Figure 9.1 (a). Note that FEC is disabled during a significant fraction of the time, which results in a marginally better network utilization.

```
[1] Changed codec (now:1)  orig: 0.558658  modif: 0.571016
[1] Changed PI (now:0.5)  orig: 0.571016  modif: 0.59358
[1] Below acceptable threshold, setting FEC  orig: 0.59358  modif: 0.677446
[2] Changed codec (now:0)  orig: 0.676932  modif: 0.74034
[2] Changed PI (now:1.0)  orig: 0.74034  modif: 0.764277
[2] Over hysteresis threshold, disabling FEC  orig: 0.764277  modif: 0.648149
[105] Below acceptable threshold, setting FEC  orig: 0.596912  modif: 0.700124
[111] Over hysteresis threshold, disabling FEC  orig: 0.800642  modif: 0.684392
[288] Below acceptable threshold, setting FEC  orig: 0.605913  modif: 0.670509
[292] Over hysteresis threshold, disabling FEC  orig: 0.774469  modif: 0.66363
```

Figure 9.3: Debugging output of the naive algorithm for the trace used in Figure 9.1 (b). Note that FEC is disabled during a significant fraction of the time, which results in a marginally better network utilization.

Algorithm 3 A bandwidth–aware algorithm for dynamic VoIP quality control.

```
1: get parameters from application
2: get network metrics
3: if currentPI < maxPI then
4:    mosPI–up ←getMOS(currentPI$\times$2)
5: end if
6: if currentPI > minPI then
7:    mosPI–down ←getMOS(currentPI/2)
8: end if
9: if mosPI–up > mosPI–down then
10:    alternatePI ←mosPI–up
11: else
12:    alternatePI ←mosPI–down
13: end if
14: if getMOS(currentPI) < getMOS(alternatePI) then
15:    currentPI ←alternatePI
16: end if
17: if getMOS(currentConfig) < ACCEPTABLE–THRESHOLD then
18:    currentFecState ←ON
19: end if
20: if currentCodec = PCM then
21:    if getMOS(GSM) > ACCEPTABLE–THRESHOLD then
22:       currentCodec ←GSM
23:    end if
24: else
25:    if getMOS(currentConfig) < ACCEPTABLE–THRESHOLD then
26:       currentCodec ←PCM
27:    end if
28: end if
29: if (getMOS(currentConfig) >= HYSTERESIS–THRESHOLD) and (getMOS(noFEC) >=
    ACCEPTABLE–THRESHOLD) then
30:    currentFecState ←OFF
31: end if
```

```
[1] Changed codec (now:1)  orig: 0.559976  modif: 0.572339
[1] Changed PI (now:0.5)  orig: 0.572339  modif: 0.593011
[1] Below acceptable threshold, setting FEC  orig: 0.593011  modif: 0.669141
[2] Changed codec (now:0)  orig: 0.686699  modif: 0.753409
[2] Changed PI (now:1.0)  orig: 0.753409  modif: 0.777107
[2] Over hysteresis threshold, disabling FEC  orig: 0.777107  modif: 0.662596
[73] Below acceptable threshold, setting FEC  orig: 0.616453  modif: 0.698861
[81] Over hysteresis threshold, disabling FEC  orig: 0.774469  modif: 0.66363
[96] Below acceptable threshold, setting FEC  orig: 0.611572  modif: 0.676396
[97] Over hysteresis threshold, disabling FEC  orig: 0.805832  modif: 0.691108
```

Figure 9.4: Debugging output of the naive algorithm for the trace used in Figure 9.1 (c). Note that FEC is disabled during a significant fraction of the time, which results in a marginally better network utilization.

MOS point).

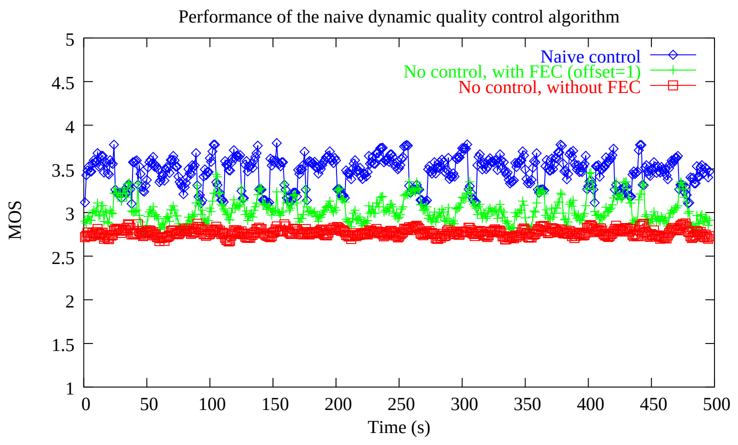
We have also compared the naive and the bandwidth–aware control algorithms, and we found that their relative performance depends on the network conditions (cf. Figure 9.10). This suggests that the control algorithms could be dynamically adapted based on current network conditions, in order to obtain better quality levels.

9.1.3 Dynamic Quality Control for Interactive VoIP

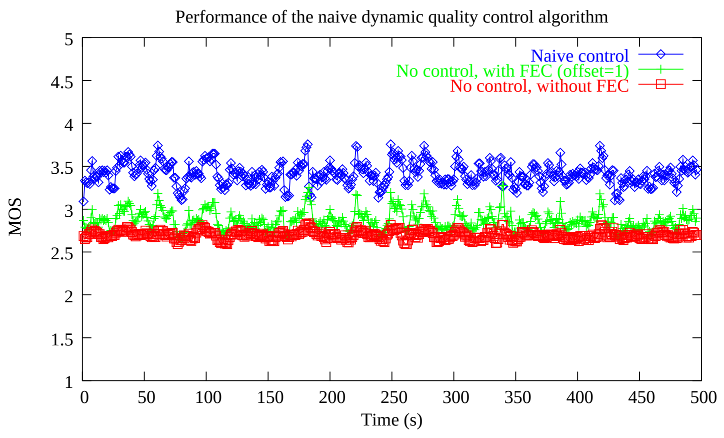
As mentioned in Section 9.1.1, extending the algorithms described above to consider interactivity factors should not be difficult. Three extra parameters could be considered in this context, namely:

Delay: in order to take into account the effects of this parameter, the extra delay induced by the use of FEC or a larger packetization interval should be calculated and fed to the RNN when predicting the resulting MOS, to assess whether the modification being currently considered is actually worth doing.

Jitter: while none of the changes that we can do at the application level is likely to modify the jitter, its value should be taken into account when estimating the MOS, and dynamic resizing of the dejittering buffer could be performed, as described in the literature (cf. [20, 41, 115, 122]). Also, the delay and jitter values may influence the choice of values for other parameters, such as those concerning FEC or the packetization interval.



(a)



(b)

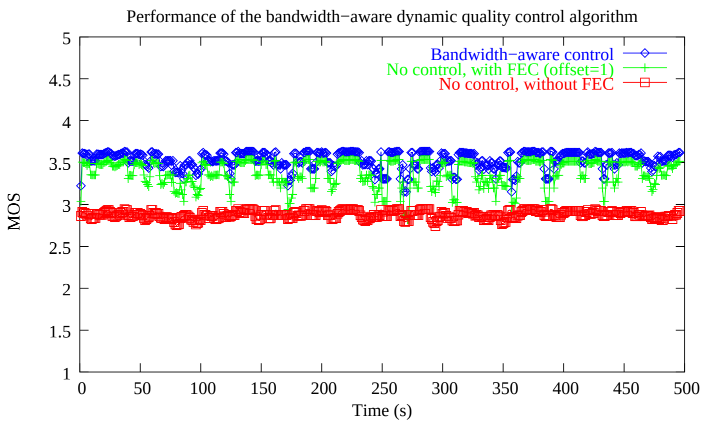
Figure 9.5: Performance of the naive control algorithm. The network parameters found in these traces are as follows: (a) LR = 7.02%, MLBS = 1.75 packets, and (b) LR = 12.47%, MLBS = 2.13 packets. Under these network conditions, the control algorithm offers a better quality than when no control is used.

```
[1] Changed codec (now:1)  orig: 0.543642  modif: 0.558892
[1] Changed PI (now:0.5)  orig: 0.558892  modif: 0.573139
[1] Below acceptable threshold, setting FEC  orig: 0.573139  modif: 0.623173
[2] Changed codec (now:0)  orig: 0.619211  modif: 0.663438
[2] Changed PI (now:1.0)  orig: 0.663438  modif: 0.685612
[25] Over hysteresis threshold, disabling FEC  orig: 0.765082  modif: 0.652353
[39] Below acceptable threshold, setting FEC  orig: 0.619893  modif: 0.715344
[43] Over hysteresis threshold, disabling FEC  orig: 0.774469  modif: 0.66363
[44] Below acceptable threshold, setting FEC  orig: 0.619185  modif: 0.706086
[90] Over hysteresis threshold, disabling FEC  orig: 0.771284  modif: 0.662454
[95] Below acceptable threshold, setting FEC  orig: 0.604052  modif: 0.682425
[105] Over hysteresis threshold, disabling FEC  orig: 0.788768  modif: 0.669276
[112] Below acceptable threshold, setting FEC  orig: 0.616262  modif: 0.718014
[125] Over hysteresis threshold, disabling FEC  orig: 0.764778  modif: 0.649389
[128] Below acceptable threshold, setting FEC  orig: 0.61342  modif: 0.709758
[139] Over hysteresis threshold, disabling FEC  orig: 0.76355  modif: 0.653231
[148] Below acceptable threshold, setting FEC  orig: 0.619637  modif: 0.723544
[159] Over hysteresis threshold, disabling FEC  orig: 0.774469  modif: 0.66363
[170] Below acceptable threshold, setting FEC  orig: 0.614595  modif: 0.697689
[176] Over hysteresis threshold, disabling FEC  orig: 0.765082  modif: 0.652353
[178] Below acceptable threshold, setting FEC  orig: 0.619893  modif: 0.715344
[201] Over hysteresis threshold, disabling FEC  orig: 0.76184  modif: 0.653075
[208] Below acceptable threshold, setting FEC  orig: 0.60923  modif: 0.696526
[258] Over hysteresis threshold, disabling FEC  orig: 0.776921  modif: 0.661541
[273] Below acceptable threshold, setting FEC  orig: 0.611686  modif: 0.698893
[305] Over hysteresis threshold, disabling FEC  orig: 0.76355  modif: 0.653231
[312] Below acceptable threshold, setting FEC  orig: 0.618159  modif: 0.708429
[360] Over hysteresis threshold, disabling FEC  orig: 0.765082  modif: 0.652353
[366] Below acceptable threshold, setting FEC  orig: 0.619185  modif: 0.706086
[399] Over hysteresis threshold, disabling FEC  orig: 0.76355  modif: 0.653231
[406] Below acceptable threshold, setting FEC  orig: 0.619185  modif: 0.706086
[421] Over hysteresis threshold, disabling FEC  orig: 0.765082  modif: 0.652353
[435] Below acceptable threshold, setting FEC  orig: 0.605277  modif: 0.687051
[443] Over hysteresis threshold, disabling FEC  orig: 0.774469  modif: 0.66363
[445] Below acceptable threshold, setting FEC  orig: 0.616664  modif: 0.703673
[472] Over hysteresis threshold, disabling FEC  orig: 0.76458  modif: 0.652959
[481] Below acceptable threshold, setting FEC  orig: 0.600662  modif: 0.68057
```

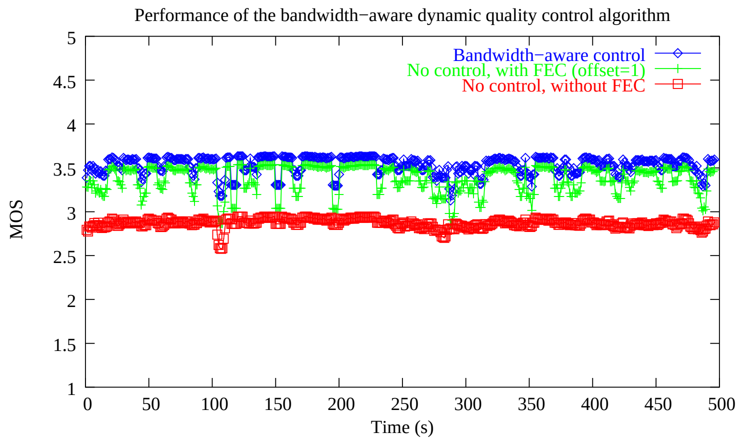
Figure 9.6: Debugging output of the naive algorithm for the trace used in Figure 9.5 (a). Note that FEC is disabled during a significant fraction of the time, which results in a marginally better network utilization.

```
[1] Changed codec (now:1)  orig: 0.536942  modif: 0.552983
[1] Changed PI (now:0.5)  orig: 0.552983  modif: 0.567223
[1] Below acceptable threshold, setting FEC  orig: 0.567223  modif: 0.618023
[2] Changed codec (now:0)  orig: 0.605195  modif: 0.644531
[2] Changed PI (now:1.0)  orig: 0.644531  modif: 0.666536
[183] Over hysteresis threshold, disabling FEC  orig: 0.76184  modif: 0.653075
[186] Below acceptable threshold, setting FEC  orig: 0.605277  modif: 0.687051
[339] Over hysteresis threshold, disabling FEC  orig: 0.76458  modif: 0.652959
[341] Below acceptable threshold, setting FEC  orig: 0.610158  modif: 0.702207
```

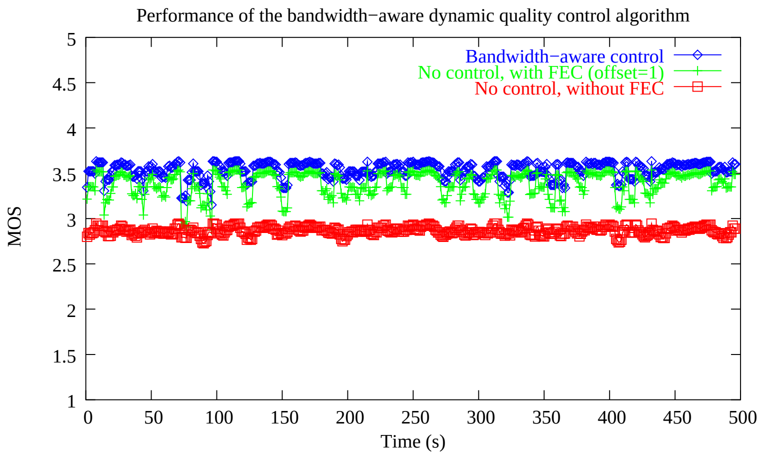
Figure 9.7: Debugging output of the naive algorithm for the trace used in Figure 9.5 (b). In this case, due to the heavier losses, FEC is needed most of the time.



(a)

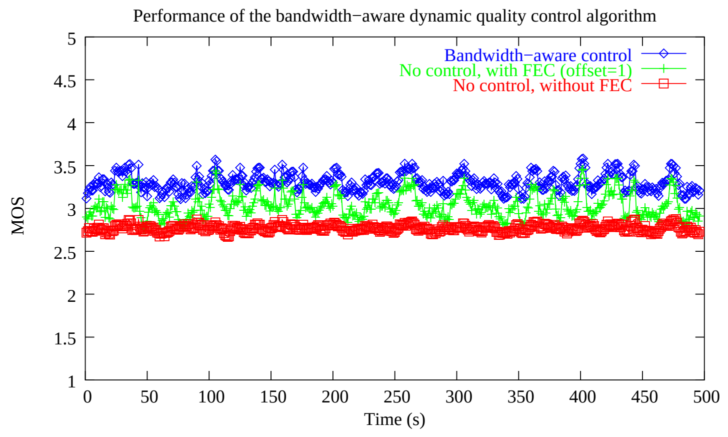


(b)

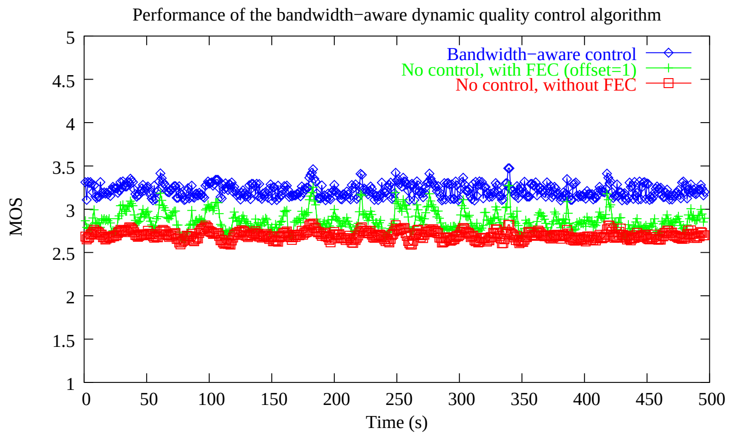


(c)

Figure 9.8: Performance of the bandwidth–aware control algorithm. The network parameters found in these traces are as follows: (a) LR = 1.69%, MLBS = 1.21 packets, (b) LR = 3.37%, MLBS = 1.23 packets, and (c) LR = 2.22%, MLBS = 1.18 packets. Note that the control algorithm only uses FEC when needed, and chooses GSM over PCM whenever possible. In this way, it reaches almost the same quality levels achieved when FEC and PCM are always used, but with a much lower bandwidth consumption.

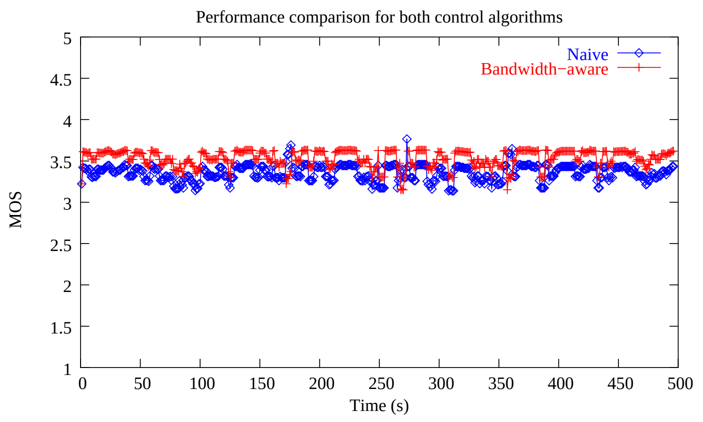


(a)

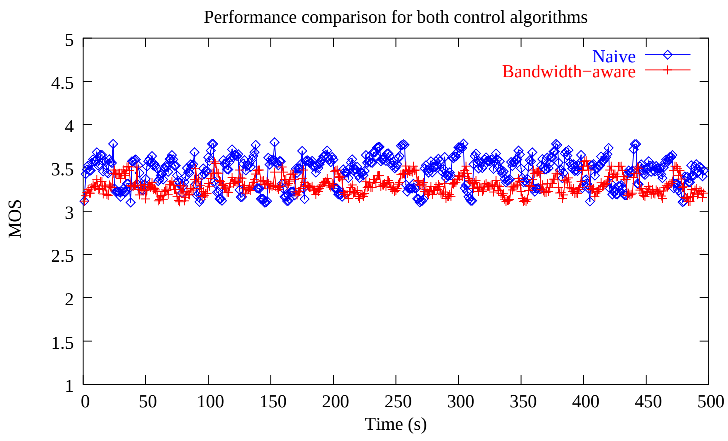


(b)

Figure 9.9: Performance of the bandwidth–aware control algorithm. The network parameters found in these traces are as follows: (a) LR = 7.02%, MLBS = 1.75 packets, and (b) LR = 12.47%, MLBS = 2.13 packets. Under these network conditions, the control algorithm offers a better quality than when no control is used.



(a)



(b)

Figure 9.10: Performance comparison for both control algorithms. The network parameters found in these traces are as follows: (a) LR = 1.69%, MLBS = 1.21 packets, (b) LR = 7.02%, MLBS = 1.75 packets. We can see that the relative performance of both algorithms varies with the network conditions.

Interactivity: as mentioned in Section 6.2, some conversational interactivity metric (like the conversational temperature proposed in [56]) could also be taken into account when assessing quality. This could give a better insight into the actual impact of delay in the current conversation, which would be useful when deciding whether modifications to the FEC or the PI should be done. For instance, if the conversation is not very interactive, a slight increase in delay will have likely have a lesser impact on quality than if it were a very interactive conversation.

9.2 QoS Control in a Wireless Context

In this section we continue to work with the scenario described in Chapter 8. There, we have established that the perceived quality is generally too low during extended periods of time, even if FEC is used, and that other mechanisms were needed to improve it. This section presents some of the methods that could be used to obtain a better quality.

9.2.1 Improving the Perceived Quality

When considering multimedia applications in the networking context considered, there are three levels on which we can try to improve the perceived quality:

- the application layer,
- the Medium Access Control (MAC) layer, and
- the IP layer.

9.2.1.1 The Application Level

From an applicative point of view, there are two main problems to deal with, namely lost packets, and late packets, which in many cases count as lost (if they arrive too late to be played). In order to deal with delay and jitter, there is little to do at this level, except to use an adequate dejittering buffer, which allows to absorb the variation in delay experienced by packets, and thus mitigates undesirable effects such as clipping. For interactive applications, however, the buffer must be as small as possible, in order to minimize the end–to–end delay, which is an important quality–affecting factor. In this study we do not concern ourselves with delay issues, but concentrate instead on the impact of the loss rate and loss distribution on the perceived quality.

In order to overcome the loss of packets in the network, there are several techniques available, following the type of stream (video or audio, coding mechanism, etc.), as discussed in Chapter 3. Roughly, we can either conceal the loss (for instance by repeating the last packet in audio, or not changing the affected image region for video), or try to recover the lost data with some form of redundancy (e.g. FEC). However, as seen in Section 8.2.2 the violent variations in loss rate and distribution we found in our simulations require more than FEC to overcome them. Indeed, as can be seen in Figures 8.13 through 8.15 for VoIP streams, the use of FEC is not enough to obtain acceptable quality levels when the network conditions are severe. Moreover, the addition of FEC (in particular for video streams), implies at least a slightly increased load in the network. We have studied this issue for VoIP over wired networks (cf. Chapter 7, [116]), where it does not pose a real problem, since the overhead incurred is relatively small. However, in a different context such as the one we consider here, this might not be always the case. Furthermore, in the case of video streams, it may imply a more significant increase in bit rate than in the VoIP case. Video FEC can need up to a 25% increase in bandwidth consumption which, given the bit rates needed for video, may imply an increase of over 1Mbps in bit rate.

For bandwidth–intensive applications such as video it may also be possible to change, if the application is able to, the coding scheme used in order to decrease bandwidth consumption in the hope of diminishing the congestion. Rate control algorithms are also available [109], which allow video streams to back down when network congestion is detected. Layered approaches [110] may also be used, which allow a host to improve video quality if it has the necessary bandwidth, or to degrade it gracefully if not.

9.2.1.2 The MAC Level

Barring a change of the physical medium (cabling, new wireless standards, etc.), there is not much to do at the physical layer of the network. At the MAC level, however, some basic QoS provisioning mechanisms are becoming available.

In [33] the authors suggest that hosts using real–time services should work in 802.11's Point Coordination Function (PCF) mode, instead of the normally used Distributed Coordination Function (DCF). The results they present are promising, except for the fact that PCF is optional to implementers, and it is very hard to implement. As a consequence, it is not currently deployed.

However, it is interesting to point out some upcoming improvements to *wi–fi* networks. The 802.11e QoS extensions [66, 114] will allow to define different Access Categories in order to differentiate several service levels during the MAC contention process. It will be also possible to omit the acknowledgments and disable retransmission. In case of packets lost at the physical level, this will produce a loss at the IP level, but no extra delay will be added, which might be useful for interactive applications if the loss rate is within the bounds in which FEC is still effective. In order to better tune the 802.11e QoS parameters (e.g. set priorities among flows, set the number of retransmissions to be used, etc.), it would be very useful to understand how they affect the perceived quality (via the performance modifications they induce at the IP level). The use of PSQA may greatly help to this end, since it allows us to see how each network parameter affects the perceived quality. As mentioned in Chapter 6, ongoing research in our team aims to assess the impact of interactivity factors such as the delay and the jitter, so that they can be also considered when performing this kind of network tuning.

9.2.1.3 The IP Level

At the IP level, it is possible to use a Differentiated Services architecture to give real–time streams a higher priority. This would allow to reduce both the loss rates, and the delay for the protected streams. Another possibility, if layered streams are used, is to filter the higher levels at the AP, which might help reduce the network congestion. The filtering would be done via DiffServ, e.g. marking higher levels as low–priority, or by simply dropping them with a firewall–based filter. This is actually a hybrid application/IP level approach, since it must be coordinated by both the application and the AP.

9.2.1.4 A Simple Proof–of–Concept Example at the IP Level

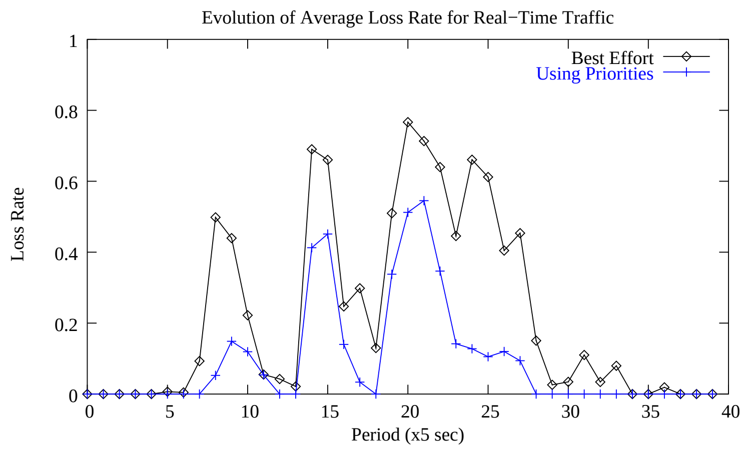
In [137] the authors propose an AQM scheme to be used at the AP so as to improve TCP performance. In a similar spirit, but biased toward real–time streams which are basically non-responsive to congestion, we have added a very simple priority scheme to our WLAN model which allows us to see how such an architecture might affect the perceived quality. Note that we only consider the wireless link, since it is the only part of the network over which the user may have some form of control. While there exist end–to–end QoS schemes such as DiffServ, they are not widely deployed, nor easily adapted to micro–management such as the one we need. Furthermore, even if they were readily available, other complex issues such as pricing should be taken into account. For simplicity's sake, we consider a simple priority scheme with few parameters needed to calibrate it.

In our model, which is loosely based on RED, the real–time traffic is always accepted if the queue is not full. For the background traffic, we define two thresholds, σ_1 and σ_2, with $\sigma_1 < \sigma_2$. If the number of packets in the queue is lower than σ_1, the packet is accepted. If the number of queued packets is between σ_1 and σ_2, the packet is accepted with probability p. Finally, if the number of packets in the queue exceeds σ_2, the packet is dropped. For our simulations, we used the values found in Table 9.1. It is worth noting that this is just a first approach to a priority scheme for the WLAN, and that as such, we have not delved into issues such as optimal threshold values, or more refined backlog metrics such as those found in RED.

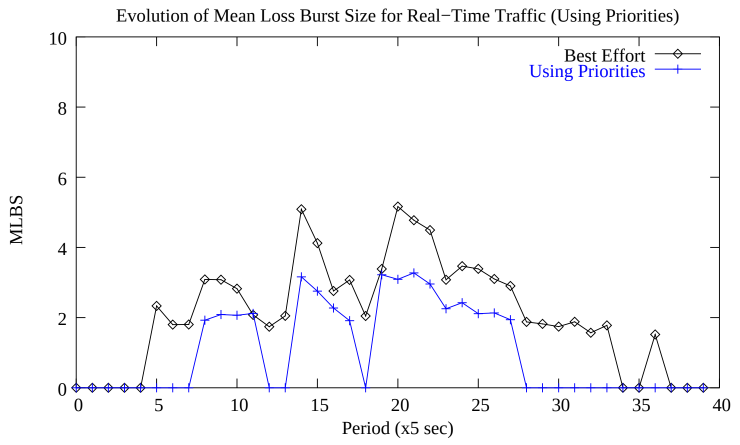
Parameter	Value
σ_1	$0.3W$
σ_2	$0.6W$
p	0.5

Table 9.1: Thresholds used in our simulations. W is the capacity of the AP queue.

Figures 9.11 through 9.13 show the loss rates and mean loss burst sizes (averaged every 5 seconds) for real–time flows in the three examples shown above, both for best effort and the priority scheme proposed above. It can be seen that, even though there are still some serious loss episodes, the situation is vastly improved.



(a)

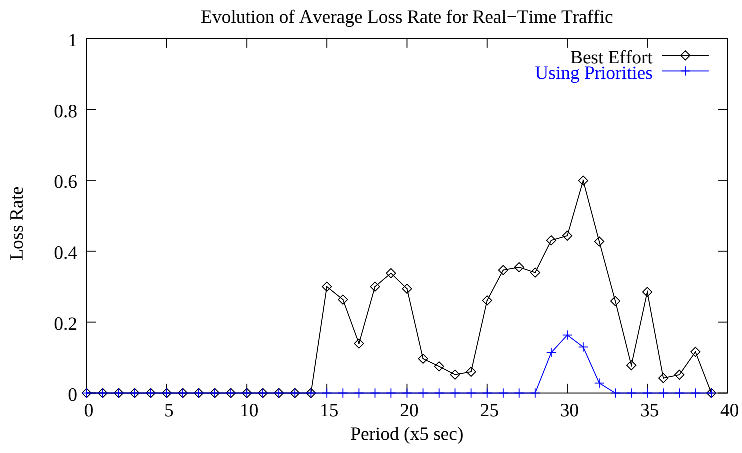


(b)

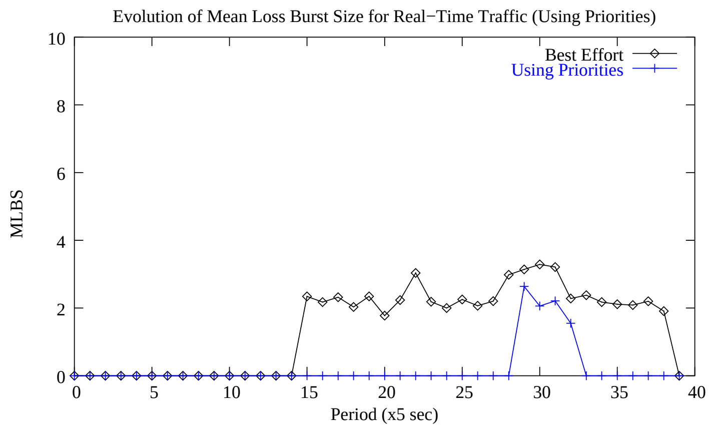
Figure 9.11: Loss rates (a) and mean loss burst sizes (b) for real–time flows, with and without service differentiation (first example).

In Figure 9.14, we can see how the protection of multimedia flows translates into perceived quality for VoIP, while in Figure 9.15, we see the improvement for a video stream. It can be seen that even when using priorities, the quality does sometimes drop below acceptable levels. The addition of FEC helps to obtain better quality values, as expected. For instance, for all three cases shown in Figure 9.14, several periods of time where speech was barely intelligible (MOS values of about 2 points), had their quality increased to almost acceptable levels by the addition of FEC. However, the wireless network shows its limitations, since there are periods for which even the combination of FEC and priority treatment for audio packets is not enough to obtain acceptable quality levels.

It is clear that the perceived VoIP quality is much better when the real–time packets are

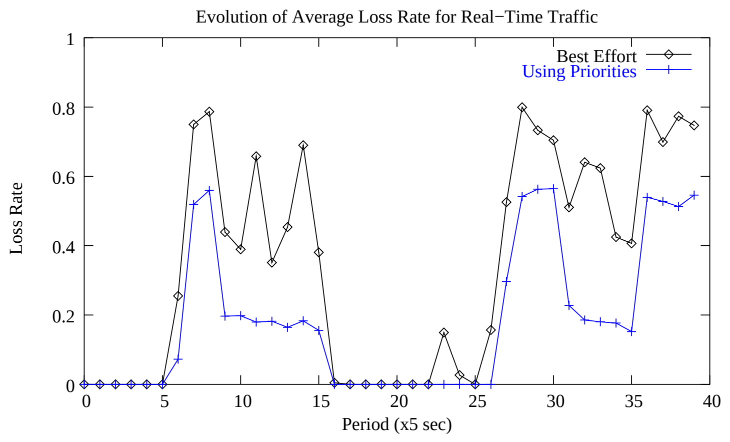


(a)

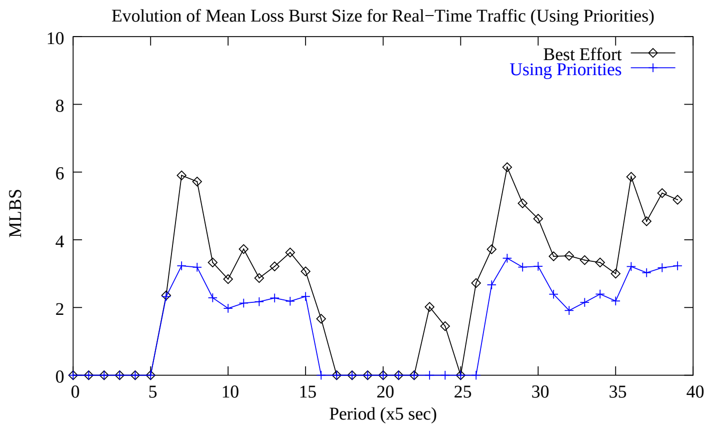


(b)

Figure 9.12: Loss rates (a) and mean loss burst sizes (b) for real–time flows, with and without service differentiation (second example).

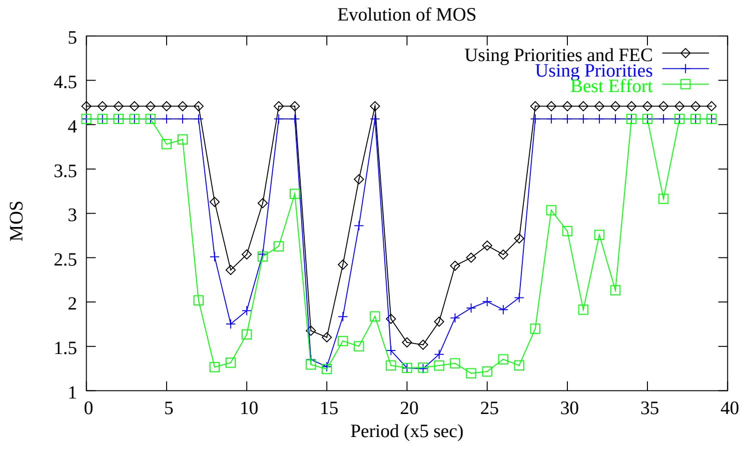


(a)

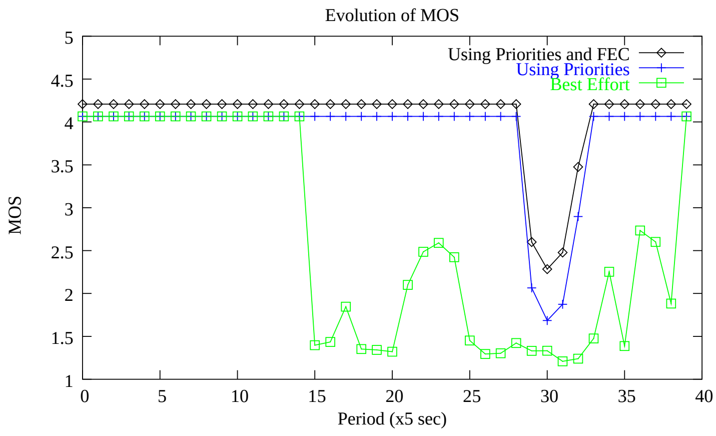


(b)

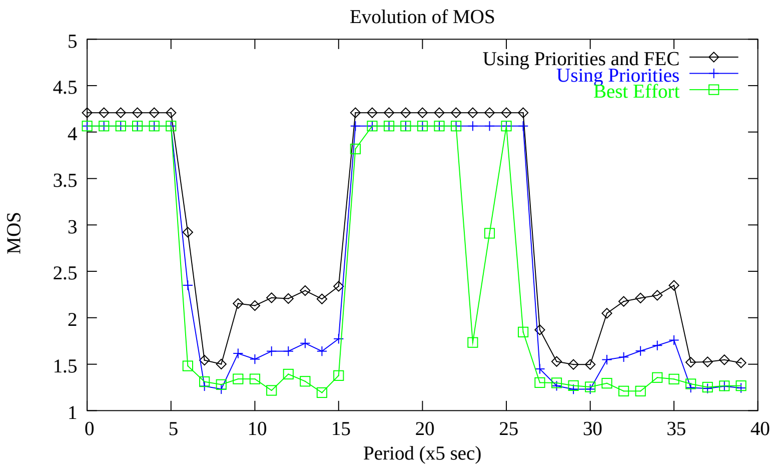
Figure 9.13: Loss rates (a) and mean loss burst sizes (b) for real–time flows, with and without service differentiation (third example).



(a)



(b)



(c)

Figure 9.14: PSQA scores for VoIP flows, for both our service differentiation scheme and best effort service. (a) First example. (b) Second example.(c) Third example. We can see the improvement due to the priority scheme used, and also to the use of FEC.

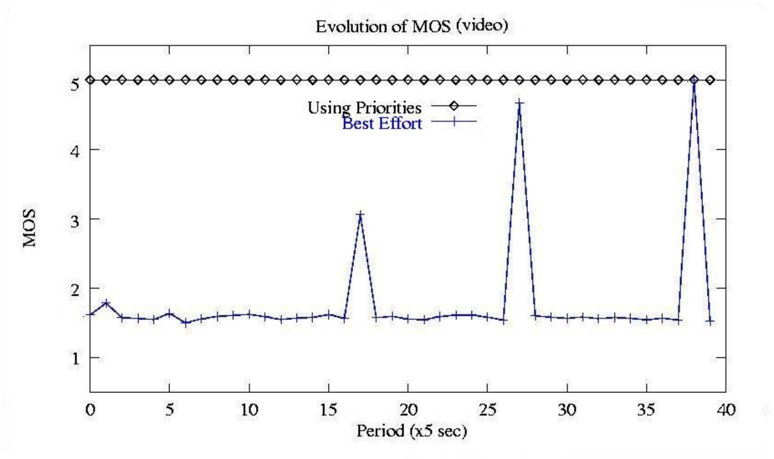


Figure 9.15: PSQA scores for a video stream for both our service differentiation scheme and best effort service (fourth example).

given a higher priority. There is, however, one problem with this approach. The improvement of real–time flows may imply a complete starvation of the background traffic, as can be seen in Figure 9.16, which somewhat limits the utility of the priority scheme.

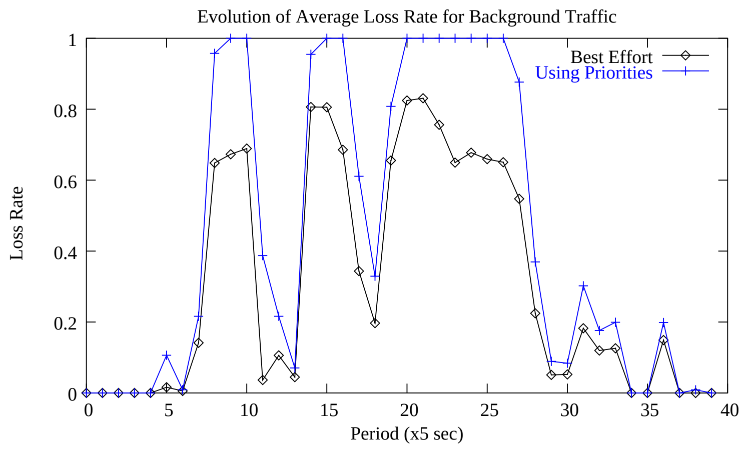
9.2.2 Controlling the Perceived Quality

As discussed above, it is possible to improve, at least to a certain extent, the perceived quality of multimedia applications in the networking context we defined. From application–level parameters to network QoS schemes, several tools are available that allow the user to make the best use of his network. However, blindly implementing all of these measures is not enough. Priority mechanisms, for instance, should only be used when needed, so as not to punish other applications if it's not necessary.

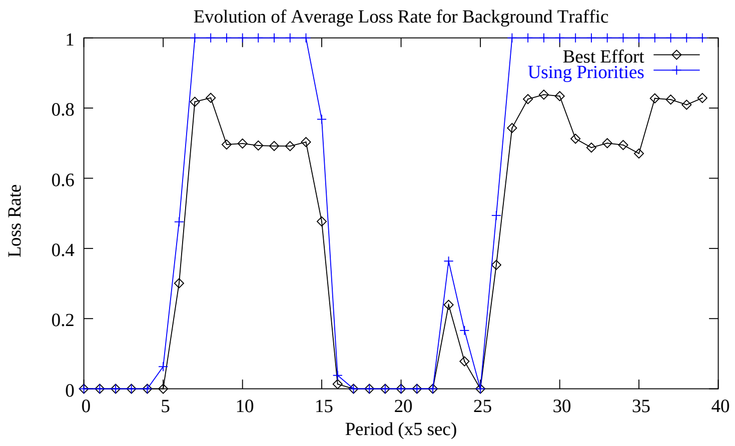
Being able to accurately assess the perceived quality in real time with tools such as PSQA, makes it possible to react to network impairments as they happen, without sacrificing other applications' performance unless really needed. Among the options available to perform QoS control in the context we described, there are:

- **Access control** – It is possible to prevent certain flows from being created if the network conditions are not appropriate, e.g. if the predicted PSQA scores are too low. The user might then manually choose to shut down other applications which are less important to him, such as P2P software, for instance. This could be also done automatically via a centralized server, in which low–priority applications would be identified to be throttled back if necessary.

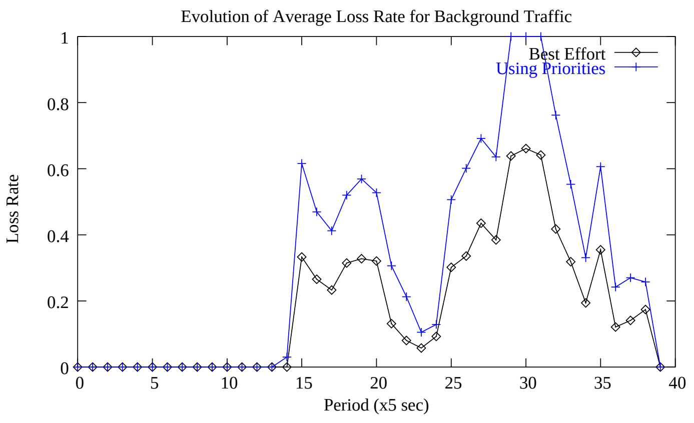
- **Application–level adjustments** – From changing the FEC offset to changing the codec



(a)



(b)



(c)

Figure 9.16: Loss rates for background traffic, with and without service differentiation. (a) First example. (b) Second example. (c) Third example.

used, many parameters are easily adjustable at the application level. Even if by themselves these changes may not be enough, they certainly help, and they might even help reduce congestion.

- **IP–level filtering** – If multi–layer streams are being used, the AP could filter the upper layers in order to avoid congestion. The perceived quality would then degrade gracefully, and in conjunction with other measures (such as ECN for competing TCP flows) the loss spikes can be avoided or at least reduced.

- **DiffServ schemes** – As we have previously seen, this kind of approach can greatly improve the perceived quality even when the network conditions are very bad. It would be interesting to use the quality feedback obtained by means of PSQA to fine–tune the parameters of these mechanisms, and to mark packets only when necessary, so as not to punish the background traffic if not needed. This should be also used in conjunction with the QoS provisions of 802.11e, if available, so as to allow marked flows a prioritized access to the medium.

9.3 Summary

In this chapter we have discussed several issues related with real–time dynamic control of the perceived quality under different networking contexts.

We started by considering a one–way VoIP scenario over a wired IP network. We proposed two simple proof of concept algorithms which provide a noticeable improvement on the perceived quality. The first algorithm uses a naive approach to quality improvement, by modifying codec, packetization interval, and FEC settings to improve quality whenever possible. Certain bounds are defined, which allow to diminish the variations in the perceived quality (very noticeable variations in quality are a nuisance to the user).

The second control algorithm strives to minimize bandwidth consumption while maintaining acceptable quality levels. This kind of approach is useful in contexts where the available resources are limited, so the user is therefore willing to accept a trade–off between "extra" quality and being able to distribute the available resources more fairly among all the applications competing for them. To this end, we have proposed the use of a lower bit rate codec whenever possible, only switching to a higher bit rate one when the quality drops below acceptable values.

We have presented performance results for both algorithms, comparing them to each other, and to static configurations under several network conditions. A description of what is needed in order to adapt these algorithms to interactive VoIP streams was also given.

The second part of the chapter deals with quality control in the wireless networking context introduced in Chapter 8. We discussed the different levels of the application / network stack, and what can be done to improve quality at each one. At the IP level, we have proposed a simple priority scheme to be used within the WLAN, and shown how it improves the quality of real–time applications such as VoIP and video. Finally, we concluded with a discussion of what would be needed to control the quality of media streams in this context.

Chapter 10

General Conclusions and Perspectives

10.1 Conclusions

In this dissertation, we have proposed several contributions in the domain of multimedia quality assessment and control. We have built upon the real–time automatic quality assessment approach (which we call Pseudo–Subjective Quality Assessment, or PSQA) proposed by Mohamed [98], and used it to better understand the quality of media streams transmitted over a network such as the Internet. We have been particularly interested in Voice over IP applications, which have greatly increased in popularity during the last few years. Several traditional telephony operators are starting to provide VoIP services, and a number of Internet telephony software packages have been released and widely adopted by users worldwide. For example, a recently launched Internet telephony tool, Skype^{TM}, has been downloaded over 140 million times, and has an installed base of several million users.

Our first contribution has been an in–depth study of PSQA's performance, which aimed at

1. validate the approach itself,
2. compare it with other quality assessment mechanisms, and
3. test its robustness and generality.

It is worth noting that the choice of RNNs as the statistical learning tool to use with PSQA has played an important role in the quality of the assessments obtained, as illustrated by Mohamed's performance comparison of RNNs and ANNs, and our own comparison of RNNs and Naive Bayesian classifiers.

The results of these analysis are presented in Chapter 5.

The second contribution of our work has been an extensive analysis of the quality of VoIP streams (made possible by the use of PSQA), and the ways in which it is affected by various factors concerning both the network and the application. We have studied VoIP performance both for one–way and interactive applications, and for wired as well as wireless networks. We have payed particular attention to the performance of a widely used forward error correction scheme, which could be especially useful in quality–control applications. To the best of our knowledge, no results currently found in the literature are comparable, in terms of detail and the amount of information conveyed, to those presented in Chapters 6, 7, and 8. The results obtained and the insight they provide into the way the different factors considered affect the perceived quality, pave the way for improvements to that quality, for example in the form of better real–time control mechanisms for multimedia applications.

The third contribution presented in this dissertation is a method to integrate PSQA with other performance evaluation tools. This allows to use PSQA during early phases of network design and / or dimensioning, and to complement the more traditional metrics with a higher–level view of QoS. These results are presented in Chapter 7, and used throughout Chapters 8 and 9, where they have allowed us to consider a wide range of network conditions without needing to implement a real testbed.

Our fourth contribution concerns the control of the perceived quality of multimedia applications. Our goals were to exploit the ability of PSQA to provide accurate quality assessments in real–time, in order to dynamically adjust applicative parameters (and network QoS parameters too, if they are available). We have proposed two quality–driven control algorithms, which use PSQA to dynamically adjust application–level parameters. These algorithms were designed to control one–way voice streams, but the extensions needed to use them in an interactive context are also given. We have also proposed quality–enhancing mechanisms which could be used to control quality in a wireless context. In particular, we have proposed a simple priority scheme for packets in the wireless link, and studied its effectiveness for voice and video flows.

10.2 Perspectives

Several extensions to the work presented herein are possible, and desirable. We are particularly interested in further studying the interactivity aspects of VoIP quality. A full–fledged interactive testing campaign, to be carried out in cooperation with the LAND team at the Federal University of Rio de Janeiro, is already underway. Further tests will also be performed at our lab. Once the tests have been carried out, and an RNN has been trained with them, the quality control algorithms proposed in this work will be extended and integrated into VivaVoz (the VoIP tool we are using for these experiments), so that we can test their performance for interactive VoIP.

A second extension on which we are working as of this writing is in the domain of wireless networking, as described in Chapter 8. We are currently developing a more detailed simulation environment (based on NS–2, instead of high–level stochastic models), which will allow us to better understand the network dynamics. Furthermore, we expect to be able to work across several layers, at the MAC, IP and application levels. Again, the ultimate goal of this experience is the development of control mechanisms which would help us overcome the problems exposed in Chapter 8. This could also lead to an extension of our work to 3rd generation mobile telephony systems.

Concerning PSQA itself, several research issues remain to be explored. It would be useful to do further studies on the generality of each instance of the tool, by performing more cross–validation studies. This is interesting, since the main cost of the method is related to the subjective tests needed to train the RNN. The results presented in this work show that it is indeed possible to perform estimations well outside the parameter domain originally considered when training the RNN. Further work on this area should focus on ways to build mappings to correct cross-estimations, and on studying the use of experts for the execution of subjective tests, as a way to reduce the number of subjects needed to perform them.

Another domain we have worked on, but for which we do not yet have all the desired results, is the assessment of high–quality audio streams. We have worked in cooperation with France Telecom R&D on the comparison of the Windows Media 9 and Real Audio 9 codecs [29], but for the time being we only have subjective scores for some points in the configuration space. Future work in this direction should concentrate on reviewing the subjective testing scheme used (which differs from the other ones we have worked with in that it uses a continuous scale), performing more subjective tests, and analyzing the results obtained via PSQA.

Finally, it would be useful to extend the work done by Mohamed in the domain of low bit rate video streams [98, 100] to cover high quality video streams, and also to mixed audio / video streams. These mixed streams might need more than a simple combination of audio and video–trained RNNs, since other issues, such as synchronization, the combined effects of the common parameters (loss rate, delay, etc.) need to be taken into account.

Another aspect which would be interesting to study both in this mixed–stream context and also for simple audio, video and voice streams is the the effect of the semantic contents in the perceived quality of the stream. For instance, it is reasonable to presume that the distortions induced by coding will not have the same impact on quality for metal rock than for classical music. Likewise, the effects of losses on a video stream should not affect a more or less static scene like, say, a news anchor speaking, in the same way as they would affect a dynamic scene like, say, a tennis match. For mixed streams, it would be interesting to know, depending on the type of scene, which media (audio or video) has a greater impact on quality, or how the presence and quality of each affects the quality of the other (along the same lines as the work described in [123]).

Bibliography

[1] Mbus web site. http://www.mbus.org.

[2] 3rd Generation Partnership Project. AMR Speech Codec; General Description (TS 26.071), June 2002.

[3] B. Ahlgren, A. Andersson, O. Hagsand, and I. Marsh. Dimensioning Links for IP Telephony. Technical Report T2000–09, Swedish Institute of Computer Science (SICS), 2000.

[4] J. Allnatt. Subjective Rating and Apparent Magnitude. *International Journal of Man-Machine Studies*, 7(6):801–816, nov 1975.

[5] E. Altman, O. Ait-Hellal, A. Jean-Marie, and I. Kurkova. On Loss Probabilities in Presence of Redundant Packets and Several Traffic Sources. *Performance Evaluation*, 36-37:486–518, August 1999.

[6] E. Altman, C. Barakat, and V. M. Ramos R. On the Utility of FEC Mechanisms for Audio Applications. *Lecture Notes in Computer Science*, 2156, 2001.

[7] E. Altman, C. Barakat, and V. M. Ramos R. Queueing Analysis of Simple FEC Schemes for IP Telephony. In *Proceedings of INFOCOM '01*, pages 796–804, 2001.

[8] ITU-T Recommendation G.729 Annex B.

[9] H. Bakircioglu and T. Kocak. Survey of Random Neural Network Applications. *European Journal of Operational Research*, 126(2):319–330, 2000.

[10] J. Beerends. Improvement of the P.861 Perceptual Speech Quality Measure. ITU-T SG12 COM-34E, December 1997.

[11] J. Beerends and J. Stemerdink. A perceptual audio quality measure based on a psychoacoustic sound representation. *Journal of Audio Eng. Soc.*, 40:963–978, September 1992.

[12] J. Beerends and J. Stemerdink. A Perceptual Speech Quality Measure Based on a Psychoacoustic Sound Representation. *Journal of Audio Eng. Soc.*, 42:115–123, December 1994.

[13] J-C. Bolot. Characterizing End-to-End Packet Delay and Loss in the Internet. *Journal of High-Speed Networks*, 2(3):305–323, December 1993.

[14] J-C. Bolot, S. Fosse-Parisis, and D.F. Towsley. Adaptive FEC–Based Error Control for Internet Telephony. In *Proceedings of INFOCOM '99*, pages 1453–1460, New York, NY, USA, March 1999.

[15] J-C. Bolot and T. Turletti. Experience with Rate Control Mechanisms for Packet Video in the Internet. In *Computer Communication Review*, January 1998.

[16] J-C. Bolot and A. Vega Garcia. Control Mechanisms for Packet Audio in the Internet. In *Proc. IEEE Infocom '96*, San Fransisco, CA, March 1996.

[17] J-C. Bolot and A. Vega Garcia. The case for FEC-based error control for packet audio in the Internet. In *ACM Multimedia Systems*, 1996.

[18] A. Bouch and M. A. Sasse. Why Value is Everything: a User-Centered Approach to Internet Quality of Service and Pricing. *Lecture Notes in Computer Science*, 2092, 2001.

[19] A. Bouch, M. A. Sasse, and H. DeMeer. Of Packets and People: a User-centered Approach to Quality of Service. *Proceedings of IWQoS2000*, pages 189–197, 2000.

[20] C. Boutremans. *Delay Aspects in Internet Telephony* . PhD thesis, EPFL, Lausanne, Switzerland, 2002.

[21] C. Boutremans and J-Y. Le Boudec. Adaptive delay aware error control for internet telephony. In *2nd IPTelephony workshop*, Columbia, NY, USA, 2001.

[22] P. Ramanathan C. Dovrolis, D. Tull. Hybrid Spatial/Temporal Loss Concealment for Packet Video. In *9th International Packet Video Workshop*, NY, USA, 1999.

[23] Y. Calas and A. Jean-Marie. On the Efficiency of Forward Error Correction at the Packet Level. In *Proceedings of CIC'04*, 2004.

[24] Y. Calas and A. Jean-Marie. Audio Quality for a Simple Forward Error Correcting Code. In *Proceedings of TSI'05*, 2005.

[25] A. Choi and A. Constantinides. Effect of packet loss on 3 toll quality speech coders. In *Second IEE National Conference on Telecommunications*, pages 380–385, York, UK, April 1989.

[26] K. Claffy, G. Miller, and K. Thompson. The nature of the Beast: Recent Traffic Measurements from an Internet Backbone. In *INET '98*, Geneva, Switzerland, 1998.

[27] M. Claypool and J. Tanner. The Effects of Jitter on the Perceptual Quality of Video. In *Proceedings of ACM Multimedia Conference*, 1999.

[28] R. Cole and J. Rosenbluth. Voice over IP Performance Monitoring. *ACM Computer Communication Review*, 31(2):9–24, April 2001.

[29] C. Colomes, M. Varela, and J-C. Gicquel. Subjective Audio Tests: Quality of Some Codecs When Used in IP Networks. In *Proceedings of the Measurement of Speech and Audio Quality in Networks workshop (MESAQIN'04)*, Prague, Czech Republic, June 2004.

[30] C. Cramer and E. Gelenbe. Video Quality and Traffic QoS in Learning-Based Sub-Sampled and Receiver-Interpolated Video Sequences. *IEEE Journal on Selected Areas in Communications*, 18(2):150–167, 2000.

[31] C. Cramer, E. Gelenbe, and H. Bakircioglu. Low bit rate video compression with neural networks and temporal sub-sampling. *Proceedings of the IEEE*, 84(10):1529–1543, 1996.

[32] A. Cray. Voice Over IP: Hear's How. *Data Communications International*, 27(5):44–59, April 1998.

[33] B. Crow, I. Widjaja, J. Geun Kim, and P. Sakai. Investigation of the IEEE 802.11 Medium Access Control (MAC) Sublayer Functions. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 126. IEEE Computer Society, 1997.

[34] P. Denisowski. How Does it Sound? *IEEE Spectrum*, pages 60–64, February 2001. Introductory paper for speech quality assessment.

[35] A. Estepa, R. Estepa, and J. Vozmediano. On the Suitability of the E-Model to VoIP Networks. In *Seventh International Symposium on Computers and Communications, IEEE ISCC 2002*, pages 511–516, Taormina, Italy, July 2002.

[36] N. Feamster and H. Balakrishnan. Packet Loss Recovery for Streaming Video. In *12th International Packet Video Workshop*, April 2002.

[37] D. Ratton Figueiredo and E. de Souza e Silva. Efficient Mechanisms for Recovering Voice Packets in the Internet. In *Globecom'99*, 1999.

[38] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[39] M. Fomenkov, K. Keys, D. Moore, and K. Claffy. Longitudinal Study of Internet Traffic in 1998-2003, 2003. CAIDA, San Diego Super Computing Center, University of California San Diego.

[40] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Kahn, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 17(6):6–17, 2003.

[41] D. Frankowski and J. Riedl. Hiding Jitter in an Audio Stream. Technical Report TR–93–50, University of Minnesota Department of Computer Science, 1993.

[42] E. Gelenbe. Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Computation*, 1(4):502–511, 1989.

[43] E. Gelenbe. Stability of the Random Neural Network Model. In *Proc. of Neural Computation Workshop*, pages 56–68, Berlin, West Germany, February 1990.

[44] E. Gelenbe. Learning in the Recurrent Random Neural Network. *Neural Computation*, 5(1):154–511, 1993.

[45] E. Gelenbe. G-Networks: new Queueing Models with Additional Control Capabilities. In *Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 58–59, Ottawa, Ontario, Canada, 1995.

[46] E. Gelenbe. Towards Networks with Cognitive Packets. In *Proc. IEEE MASCOTS Conference*, pages 3–12, San Francisco, CA, 2000.

[47] E. Gelenbe. Reliable networks with cognitive packets. In *International Symposium on Design of Reliable Communication Networks*, pages 28–35, Budapest, Hungary, October 2001. Invited Keynote Paper.

[48] E. Gelenbe. Cognitive packet networks: QoS and performance. In *Proc. IEEE Computer Society MASCOTS02 Symposium*, pages 3–12, Fort Worth, TX, October 2002. Opening Keynote.

[49] E. Gelenbe, C. Cramer, M. Sungur, and P. Gelenbe. Traffic and Video Quality in Adaptive Neural Compression. *Multimedia Systems*, 4(6):357–369, 1996.

[50] E. Gelenbe and J.M. Fourneau. Random Neural Networks with Multiple Classes of Signals. *Neural Computation*, 11(3):953–963, 1999.

[51] E. Gelenbe and K. Hussain. Learning in the Multiple Class Random Neural Network. *IEEE Trans. on Neural Networks*, 13(6):1257–1267, 2002.

[52] E. Gelenbe, R. Lent, and Z. Xu. Design and Performance of Cognitive Packet Networks. *Performance Evaluation*, 46:155–176, 2001.

[53] E. Gelenbe, Z.-H. Mao, and Y.-D. Li. Function Approximation with Spiked Random Networks. *IEEE Trans. on Neural Networks*, 10(1):3–9, 1999.

[54] E. Gilbert. Capacity of a Burst–loss Channel. *Bell Systems Technical Journal*, 5(39), September 1960.

[55] T. A. Hall. Objective Speech Quality Measures for Internet Telephony. In *Voice over IP (VoIP) Technology, Proceedings of SPIE*, volume 4522, pages 128–136, Denver, CO, USA, August 2001.

[56] F. Hammer and P. Reichl. Hot discussions and Frosty Dialogues: Towards a Temperature Metric for Conversational Interactivity. In *8th International Conference on Spoken Language Processing (ICSLP/INTERSPEECH 2004)*, Jeju Island, Korea, October 2004.

[57] F. Hammer, P. Reichl, T. Nordström, and G. Kubin. Corrupted Speech Data Considered Useful. In *First ISCA International Tutorial and Research Workshop on Auditory Quality of Systems*, Mont-Cenis, Germany, April 2003.

[58] D. Hands and M. Wilkins. A Study of the Impact of Network Loss and Burst Size on Video Streaming Quality and Acceptability. In *Interactive Distributed Multimedia Systems and Telecommunication Services Workshop*, October 1999.

[59] M. Hassan, A. Nayandoro, and M. Atiquzzaman. Internet Telephony: Services, Technical Challenges, and Products. *IEEE Communications Magazine*, 38(4):96–103, April 2000.

[60] S. Hemminger. Netem website. http://developer.osdl.org/shemminger/netem/.

[61] M. Heusse, F. Rousseau, and A. Duda. Performance Anomaly of 802.11b. In *Infocom'03*, March 2003.

[62] O. Hodson. Packet Reflector. http://www.cs.ucl.ac.uk/staff/O.Hodson/misc/reflector.tar.gz.

[63] P. Hurley, J-Y. Le Boudec, and P. Thiran. The Asymmetric Best-Effort Service. In *Proceedings of IEEE Globecom 1999*, Rio de Janeiro, Brazil, December 1999.

[64] P. Hurley, M. Kara, J-Y. Le Boudec, and P. Thiran. A Novel Scheduler for a Low Delay Service Within Best-Effort. In *Proceedings of IwQoS 2001*, Karlsruhe, June 2001.

[65] P. Hurley, M. Kara, J-Y. Le Boudec, and P. Thiran. ABE: Providing a Low-Delay Service within Best Effort. *IEEE Network Magazine*, 15(3), May 2001.

[66] IEEE 802.11 Working Group. Draft amendment to standard for information technology - telecommunications and information exchange between systems - lan/man specific requirements part 11 wireless medium access control (mac) and physical layer (phy) specifications: Amendment 7: Medium access control (mac) quality of service (qos) enhancements, 2004.

[67] IETF Network Working Group. RTP: A Protocol for Real–Time Applications (RFC 1889), 1996.

[68] IETF Network Working Group. General Characterization Parameters for Integrated Service Network Elements (RFC 2215), 1997.

[69] IETF Network Working Group. Integrated Services in the Internet Architecture: an Overview (RFC 1633), 1997.

[70] IETF Network Working Group. Resource ReSerVation Protocol (RSVP) (RFC 2205), 1997.

[71] IETF Network Working Group. An Architecture for Differentiated Services (RFC 2475), 1998.

[72] IETF Network Working Group. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 3246), 1998.

[73] IETF Network Working Group. A Proposal to Add Explicit Congestion Notification (ECN) to IP (RFC 2481), 1999.

[74] IETF Network Working Group. Assured Forwarding PHB Group (RFC 2597), 1999.

[75] IETF Network Working Group. A Message Bus for Local Coordination (RFC 3259), April 2002.

[76] IETF Network Working Group. Assured Forwarding PHB Group (RFC 3246), 2002.

[77] IETF Network Working Group. SIP: Session Initiation Protocol (RFC 3261), 2002.

[78] J. Incera and G. Rubino. Bit-level and packet-level, or pollaczec-khintine formulae revisited. In *QEST'04*, 2004.

[79] ITU-R Recommendation BS.1534. Method for the Subjective Assessment of Intermediate Quality Level of Coding Systems, 2001.

[80] ITU-R Recommendation BT.500-10. Methodology for the subjective assessment of the quality of television pictures, March 2000.

[81] ITU-T Recommendation G.107. The E-model, a Computational Model for Use in Transmission Planning, 2003.

[82] ITU-T Recommendation G.114. One-way transmission time, 2003.

[83] ITU-T Recommendation G.711. Pulse Code Modulation (PCM) of Voice Frequencies, 1988.

[84] ITU-T Recommendation H.323 (version 5). Packet-based Multimedia Communications Systems, 2003.

[85] ITU-T Recommendation P.800. Methods for Subjective Determination of Transmission Quality, 1996.

[86] ITU-T Recommendation P.861. Ojective quality assessment of telephone-band (300-3400 Hz) speech codecs, 1998.

[87] ITU-T Recommendation P.862. Perceptual Evaluation of Speech Quality (Pesq), an Objective Method for End-To-End Speech Quality Assessment of Narrowband Telephone Networks and Speech Codecs, 2001.

[88] ITU-T Recommendation P.920. Interactive test methods for audiovisual communications, 2000.

[89] W. Jiang and H. Schulzrinne. Analysis of On-Off Patterns in VoIP and their Effect on Voice Traffic Aggregation. In *The 9th IEEE International Conference on Computer Communication Networks*, 2000.

[90] M. Keppes and S. Garg. Can I Add a VoIP Call? In *ICC'03*, 2003.

[91] N. Kitawaki and K. Itoh. Pure Delay Effects on Speech Quality in Telecommunications. *IEEE Journal on Selected Areas in Communications*, 9(4):586–593, May 1991.

[92] T. Kostas, M. Borella, I. Sidhu, G. Schuster, J. Grabiec, and J. Mahler. Real–time Voice Over Packet Switched Networks. *IEEE Network*, 12(1):12–27, February 1998.

[93] I. Kouvelas, O. Hodson, V. Hardman, and J. Crowcroft. Redundancy Control in Real-time Internet Audio Conferencing. In *AVSPN'97*, Aberdeen,Scotland, September 1997.

[94] Larzon, L–A. and Degermark, M. and Pink, S. The UDP–Lite Protocol (INTERNET DRAFT), 2002.

[95] F. Liu, J. Won Kim, and C.C. Jay Kuo. Adaptive Delay Concealment for Internet Voice Applications with Packet-Based Time-Scale Modification. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing ICASSP*, Salt Lake City, May 2001.

[96] University College London. Robust Audio Tool website. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/index.html.

[97] E. Masala, M. Bottero, and J.C. De Martin. Link-Level Partial Checksum for Real-Time Video Transmission over 802.11 Wireless Networks. In *14th International Packet Video Workshop (PVW)*, Irvine, CA, December 2004.

[98] S. Mohamed. *Automatic Evaluation of Real-Time Multimedia Quality: a Neural Network Approach*. PhD thesis, INRIA/IRISA, Univ. Rennes I, Rennes, France, jan 2003.

[99] S. Mohamed, F. Cervantes, and H. Afifi. Integrating Networks Measurements and Speech Quality Subjective Scores for Control Purposes. In *Proceedings of IEEE INFOCOM'01*, pages 641–649, Anchorage, AK, USA, April 2001.

[100] S. Mohamed and G. Rubino. A Study of Real–time Packet Video Quality Using Random Neural Networks. *IEEE Transactions On Circuits and Systems for Video Technology*, 12(12):1071 –1083, December 2002.

[101] S. Moon, J. Kurose, and D. Towsley. Packet Audio Playout Delay Adjustment Algorithms: Performance Bounds and Algorithms. *ACM/Springer Multimedia Systems*, 6:17–28, January 1998.

[102] P. Mülhethaler. *WiFi 802.11 et les Réseaux Sans Fil*. Eyrolles, 2002.

[103] L. Nielsen. A Neural Network Model for Prediction of Sound Quality. Technical report, Technical University of Denmark, 1993.

[104] NIST. NIST Net website. http://www-x.antd.nist.gov/nistnet/.

[105] J. Orozco. *Quality of Service Management of Multimedia Flows Over DiffServ IP Networks*. PhD thesis, INRIA/IRISA, univ. Rennes I, Rennes, France, March 2005.

[106] C. Perkins, O. Hodson, and V. Hardman. A Survey of Packet-Loss Recovery for Streaming Audio. *IEEE Network*, 12(5):40–48, September 1998.

[107] Psytechnics Ltd. PESQ: an Introduction. `http://www.psytechnics.com`, September 2001.

[108] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks. In *IEEE Infocom'94*, pages 680–688, 1994.

[109] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for Congestion Controlled Video Playback over the Internet. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 189–200. ACM Press, 1999.

[110] R. Rejaie, M. Handley, and D. Estrin. Layered Quality Adaptation for Internet Video Streaming, 2000.

[111] A. Rix. Advances in Objective Quality Assessment of Speech over Analogue and Packet-based Networks. In *the IEEE Data Compression Colloquium*, London, UK, November 1999.

[112] A. Rix, J. Beerends, M. Hollier, and A. Hekstra. Perceptual Evaluation of Speech Quality (PESQ) - a New Method for Speech Quality Assessment of Telephone Networks and Codecs. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–752, Salt Lake City, Utah, USA, May 2001.

[113] A. Rix and M. Hollier. The Perceptual Analysis Measurement System for Robust End-to-End Speech Assessment. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing – ICASSP*, pages 1515–1518, Istanbul, Turkey, June 2000.

[114] R. Rollet and C. Mangin. IEEE 802.11a, 802.11e and Hiperlan/2 Goodput Performance Comparison in Real Radio Conditions. In *GlobeCom'03*, 2003.

[115] J. Rosenberg and H. Schulzrinne. Integrating Packet FEC into Adaptive Voice Playout Buffer Algorithms on the Internet. In *Proceedings of the IEEE Infocom*, pages 1705–1714, March 2001.

[116] G. Rubino and M. Varela. Evaluating the utility of media–dependent FEC in VoIP flows. In *LNCS 3266: Proceedings of the Fifth International Workshop on Quality of future Internet Services (QofIS'04)*, Barcelona, Spain, September 2004.

[117] G. Rubino, M. Varela, and S. Mohamed. Performance evaluation of real-time speech through a packet network: a random neural networks-based approach. *Performance Evaluation*, 57(2):141–162, May 2004.

[118] H. Sanneck, G. Carle, and R. Koodli. A Framework Model for Packet Loss Metrics Based on Loss Runlengths. In *Proceedings of the SPIA/ACM SIGMM Multimedia Computing and Networking Conference*, pages 177–187, San Jose, CA, January 2000.

[119] H. Sanneck, N. Le, and A. Wolisz. Intraflow Loss Recovery and Control for VoIP. In *ACM MULTIMEDIA '01*, Ottawa, Canada, 2001.

[120] H. Schulzrinne, J. F. Kurose, and D. Towsley. Distribution of the Loss Period for Some Queues in Continuous and Discrete Time. Technical report, University of Massachusetts; Amherst, MA, 1991.

[121] D. Snider. *Speech Synthesis Using an Aeroacoustic Fricative Model*. PhD thesis, Rutgers University, 1999.

[122] L. Sun and E. Ifeachor. Perceived Speech Quality Prediction for Voice over IP-based Networks. In *Proceedings of IEEE International Conference on Communications (IEEE ICC'02)*, pages 2573–2577, New York, USA, April 2002.

[123] A. Vahedian, M.R. Frater, and J.F. Arnold. Impact of Audio on Subjective Assessment of Video Quality. In *Proceedings of International Conference on Image Processing*, volume 2, pages 367–370, Kobe, Japan, October 1999.

[124] J-M. Valin. Speex website. http://www.speex.org.

[125] C.J. van den Branden Lambrecht. Color Moving Picture Quality Metric. In *Proceedings of the IEEE International Conference on Image Processing*, September 1996.

[126] C.J. van den Branden Lambrecht. *Perceptual Models and Architectures for Video Coding Applications*. PhD thesis, EPFL, Lausanne, Swiss, 1996.

[127] Various Authors. Discussion on the e2e mailing list: "Queue size for routers?". `http://www.postel.org/pipermail/end2end-interest/2003-January/002643.`

[128] S. Voran. Estimation of Perceived Speech Quality Using Measuring Normalizing Blocks. In *IEEE Workshop on Speech Coding For Telecommunications Proceeding*, pages 83–84, Pocono Manor, PA, USA, September 1997.

[129] S. Wang, A. Sekey, and A. Gersho. An Objective Measure for Predicting Subjective Quality of Speech Coders. *IEEE Journal on Selected Areas in Communications*, 10(5):819–829, 1992.

[130] Y. Wang, M. Claypool, and Z. Zuo. An Empirical Study of RealVideo Performance Across the Internet. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pages 295–309, San Francisco, California, USA, November 2001.

[131] A. Watson and M.A. Sasse. Evaluating Audio and Video Quality in Low-Cost Multimedia Conferencing Systems. *ACM Interacting with Computers Journal*, 8(3):255–275, 1996.

[132] A. Watson and M.A. Sasse. Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications. In *Proceedings of ACM Multimedia'98*, pages 55–60, Bristol, UK, September 1998.

[133] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[134] S. Wolf, M. Pinson, S. Voran, and A. Webster. Objective Quality Assessment of Digitally Transmitted Video. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 477–483, 1991.

[135] M. Yajnik, S. Moon, J.F. Kurose, and D.F. Towsley. Measurement and Modeling of the Temporal Dependence in Packet Loss. In *Proccedings of IEEE INFOCOM '99*, pages 345–352, 1999.

[136] W. Yang. *Enhanced Modified Bark Spectral Distortion (EMBSD): an Objective Speech Quality Measrure Based on Audible Distortion and Cognition Model*. PhD thesis, Temple University Graduate Board, May 1999.

[137] S. Yi, M. Keppes, S. Garg, X. Deng, G. Kesidis, and G. Das. Proxy-RED: An AQM Scheme for Wireless Local Area Networks. In *IEEE ICCCN'04*, 2004.

List of Figures

Abstract

The fast growth of the Internet over the last few years has spurred the development of new applications, in particular multimedia services such as voice over IP (VoIP), music streaming, video on demand (VoD), video–conferencing, etc. However, the Internet infrastructure was not designed with these applications in mind, and thus their performances are not as good as they should be. It is therefore necessary to develop mechanisms able to improve the quality of these services, and there is indeed an important ongoing world–wide research effort in this direction. In order to effectively implement the aforementioned mechanisms, it is important to understand how the perceived quality of each application is affected by network conditions and applicative parameters.

In this dissertation, we develop on a recent quality assessment technique, which we call Pseudo–Subjective Quality Assessment (PSQA), which is able to accurately estimate the quality as perceived by the application users. Moreover, it can provide precise estimations of quality in real–time, which no other technique currently found in the literature can do.

Our first contribution concerns an in–depth study of PSQA and its performance, and how it compares to the other available assessment mechanisms. We then go on to provide a detailed analysis of the quality of VoIP as a function of several parameters, both network and application–level. We study both one–way and interactive flows, and consider wired and wireless networking contexts. The third contribution of this dissertation is a technique to integrate PSQA with other performance evaluation techniques, such as queuing models. This approach allows network designers to obtain a high–level view of the quality of service of the network being designed or dimensioned. Our fourth contribution concerns the development of quality–driven dynamic control mechanisms for multimedia applications, based on the real–time assessment capabilities of PSQA. We propose two proof–of–concept algorithms for control of one–way VoIP flows, and show how to extend them for interactive VoIP. We also propose quality–enhancing techniques for VoIP in a wireless context, including a simple priority scheme for the wireless link. We study their performance and how they might be used for dynamic control of the perceived quality.

Résumé

Au cours de ces dernières années, la croissance de l'Internet a donnée lieu à une nouvelle génération d'applications multimédia qui l'utilisent comme moyen de diffusion. De telles applications, par exemple la voix sur IP (VoIP), de la vidéo en temps-réel et sur demande (VoD), ou la téléconférence. Or, l'architecture de l'Internet n'a pas été conçue pour supporter ces types d'applications, et de ce fait, leur performance n'est pas aussi bonne que l'on souhaiterait. Il est donc nécessaire de développer de mécanismes capables d'améliorer la qualité de ces nouveaux services, et pour cela, il est impératif de comprendre comment cette qualité est affectée par les différents paramètres applicatifs et du réseau.

Dans cette thèse, nous travaillons sur une technique d'évaluation de la qualité des flux multimédia qui a été récemment développée. Cette technique, que nous appelons Évaluation Pseudo–Subjective de la Qualité (PSQA pour sa sigle en Anglais), permet d'obtenir des estimations très fines de la qualité d'un flux multimédia tel que perçue par les utilisateurs. De plus, elle permet d'obtenir ces estimations précises en temps-réel, ce que aucune des autres techniques disponibles dans la littérature peut faire.

Notre première contribution est donc une étude approfondie de PSQA et de sa performance. Nous présentons aussi une comparaison de la performance de PSQA et celle des autres mécanismes d'évaluation de la qualité les plus répandus. Par la suite, notre deuxième contribution est une analyse détaillée de la qualité des flux VoIP en fonction de plusieurs paramètres, tant à niveau de la couche applicative comme à niveau de la couche réseau. Nous étudions des flux unidirectionnels et interactifs, et des contextes réseau filaires et sans fil. Notre troisième contribution est la proposition d'une technique pour intégrer PSQA avec d'autres techniques d'évaluation de performance, telles que la modélisation avec files d'attente. Ceci permet aux concepteurs de réseaux d'avoir une vue "haut–niveau" de la qualité de service dans le réseau lors de sa conception. Notre quatrième et dernière contribution est dans le domaine du contrôle dynamique des applications multimédias, basé sur des estimations en temps–réel de la qualité perçue. En particulier nous montrons, via deux algorithmes simples pour le contrôle des flux VoIP unidirectionnels, qu'il est possible d'obtenir des améliorations de la qualité de cette manière. Nous présentons aussi des extensions nécessaires pour adapter ces algorithmes aux flux interactifs. Finalement, nous proposons des mécanismes pour améliorer la qualité de la VoIP dans des réseaux sans fil, et discutons comment ils peuvent être intégrés dans des mécanismes de contrôle de la qualité.