DELAY ASPECTS IN INTERNET TELEPHONY

THÈSE N^{O} 2715 (2002)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Catherine BOUTREMANS

ingénieur civil électricien, Faculté Polytechnique de Mons, Belgique
et de nationalité belge

acceptée sur proposition du jury:

Prof. J.-Y. Le Boudec, directeur de thèse
Dr C.Diot, rapporteur
Prof. H.Schulzrinne, rapporteur
Prof. P.Thiran, rapporteur

Lausanne, EPFL
2003

For Claudine, Robert & Ed.
For Fabrice.

Acknowledgements

This work is the outcome of a wonderful four-year collaboration with my excellent PhD advisor, Prof. Jean-Yves Le Boudec. I'm deeply grateful to him for the time he spent contributing to this work and for his guidance in structuring technical writing. I greatly value his patience, his perseverance and his inspiration.

I am particularly thankful for the fruitful discussions, the suggestions and encouragements from Prof. Patrick Thiran, who proved to be a wonderful friend and office-neighbor.

Everyday life would have been boring without my cheerful and sociable colleagues in the Laboratory for Computer Communications and Applications (LCA). My gratitude goes to those with whom I had the pleasure of sharing an office, namely Dr. Olivier Verscheure, Milan Vojnovic, Dr. Carmen Mas, Dr. Ljubica Blazevic and Mathilde Durvy. I thank equally my colleagues, past and present, from the coffee-breaks' clique: Olivier Dousse alias *Soft*, our *horoscope addict*, Hung Nguyen, Philippe Deleval, Sylviane Dal Mas, our *guardian angel*, Angela Devenoge alias *Angie*, our *LCA super woman*, Bozidar Radunovic alias *Boki*, Dr. Silvia Giordano, Dr. Paul Hurley and all the other people who made social life very attractive at EPFL.

I am grateful to the secretarial and system manager staff, in particular Danielle Alvarez, Holly Cogliati, Angela Devenoge, Jean-Pierre Dupertuis and Marc-André Lüthi. They were always helpful and made my life much easier.

This work was supported by a PhD fellowship from the Communication Systems department, EPFL, which I gratefully acknowledge. I would also like to thank all members of the jury for having accepted to review this thesis and for the interest they demonstrated in respect to my work.

I appeciate the generosity of Sprint ATL who provided funding for the last year of this work and offered me the opportunity of an internship. I had a great time there and had the pleasure of working with a wonderful team. I'm more than grateful to Dr. Christophe Diot for his support and encouragement and for his incomparable sense of humour. A special thanks goes to Dr. Gianluca Iannacone, the best *little boss* I have ever had, and with whom I really enjoyed working. I can not evoke this internship without winking at Dr. Dina Papagiannaki who turned out to be one of the most marvelous friends one could dream of.

I'm indebted to Joerg Widmer, *my best partner*, with whom I had the privilege of working during the last months of the PhD. I had a wonderful time collaborating with him and I greatly appreciate the fruitful and interesting discussions that we had about the TCP sometimes-not-so-friendly algorithms. I'm deeply grateful to Joerg who spent the

last long hours of writing this thesis at my side.

Since life is not only work, I owe a special thanks to my precious friends who always supported me and shared all the good and the bad moments. Distance is an excellent filter for friendship and I would like to express all my gratitude to Caroline Hayot, Juliette Trivier and Adrien Trivier who have always remained faithful friends. I thank Laurent Letier Lacaze alias *L3 le meuble* for his support and for having kept me company during long evenings when I was writing the thesis.

I could not have succeeded in finishing this thesis without the help, support and love from Fabrice Serey alias *Fabro* who patiently encouraged me and stayed beside me during the most crucial moments.

Finally, my gratitude goes to my beloved family. *Au clan Bout, merci pour votre soutien, vos encouragements et votre amour inconditionnel. A Dine et Albert, merci de nous avoir appris à penser et de nous avoir constamment poussés à aller au bout des choses. A Edward, simplement merci d'être et de rester un véritable frère.*

Abstract

In this work, we address the transport of high quality voice over the Internet with a particular concern for delays. Transport of interactive audio over IP networks often suffers from packet loss and variations in the network delay (jitter). Forward Error Correction (FEC) mitigates the impact of packet loss at the expense of an increase of the end-to-end delay and the bit rate requirement of an audio source. Furthermore, adaptive playout buffer algorithms at the receiver compensate for jitter, but again this may come at the expense of additional delay. As a consequence, existing error control and playout adjustment schemes often have end-to-end delays exceeding 150 ms, which significantly impairs the perceived quality, while it would be more important to keep delay low and accept some small loss.

We develop a joint playout buffer and FEC adjustment scheme for Internet Telephony that incorporates the impact of end-to-end delay on perceived audio quality. To this end, we take a utility function approach. We represent the perceived audio quality as a function of both the end-to-end delay and the distortion of the voice signal. We develop a joint rate/error/playout delay control algorithm which optimizes this measure of quality and is TCP-Friendly. It uses a channel model for both loss and delay. We validate our approach by simulation and show that (1) our scheme allows a source to increase its utility by avoiding increasing the playout delay when it is not really necessary and (2) it provides better quality than the adjustment schemes for playout and FEC that were previously published.

We use this scheme in the framework of *non-elevated* services which allow applications to select a service class with reduced end-to-end delay at the expense of a higher loss rate. The tradeoff between delay and loss is not straightforward since audio sources may be forced to compensate the additional losses by more FEC and hence more delay. We show that the use of non-elevated services can lead to quality improvements, but that the choice of service depends on network conditions and on the importance that users attach to delay. Based on this observation, we propose an adaptive service choosing algorithm that allows audio sources to choose in real-time the service providing the highest audio quality.

In addition, when used over the standard IP best effort service, an audio source should also control its rate in order to react to network congestion and to share the bandwidth in a fair way. Current congestion control mechanisms are based on packets (i.e., they aim to reduce or increase the number of packets sent per time interval to adjust to the current level of congestion in the network). However, voice is an inelastic traffic where

packets are generated at regular intervals but packet size varies with the codec that is used. Therefore, standard congestion control is not directly applicable to this type of traffic. We present three alternative modifications to equation based congestion control protocols and evaluate them through mathematical analysis and network simulation.

Résumé

Dans ce travail, nous nous penchons sur la transmission de voix haute fidélité dans le réseau Internet, avec un souci particulier pour les problèmes liés au temps de transmission de bout en bout. La qualité des conversations interactives par Internet souffre de la perte de paquets et de la variation de leurs temps de parcours (gigue). L'effet des pertes peut être atténuée par des mécanismes de correction d'erreur en boucle ouverte (*Forward Error Correction* - FEC), mais ceci au prix d'un délai supplémentaire et d'une augmentation du débit à la source. La gigue est absorbée par un tampon de lissage (*playout buffer*) placé à la destination, dont la taille est contrôlée de manière adaptative. Mais ceci s'effectue à nouveau au prix d'un retard additionnel. En conséquence, les algorithmes de contrôle d'erreur et d'ajustement du tampon de lissage entraînent souvent des délais de bout en bout qui dépassent 150 ms, ce qui dégrade considérablement la qualité de la conversation, alors qu' il serait préférable de conserver un retard un peu moins important, au prix de quelques pertes supplémentaires.

Dans ce travail de thèse, nous proposons un algorithme d'adaptation conjointe de la FEC et du tampon de lissage pour la téléphonie sur Internet qui tient compte de l'influence du délai sur la qualité des conversations. Notre approche est basée sur une fonction d'utilité. Nous définissons la qualité perçue comme une fonction de la distorsion du signal et du retard total subi par le signal. Nous développons un algorithme de contrôle de conjoint des trois paramètres: débit de la source, stratégie de correction d'erreur et tampon de lissage qui maximise cette mesure de la qualité et qui soit *TCP-Friendly*. Cet algorithme est basé sur un modèle de canal qui considère les pertes et les délais de transmission. Nous prouvons le bien-fondé de notre approche par simulations et démontrons que (1) notre système permet à une source d'obtenir une utilité supérieure en lui évitant de d'augmenter le retard à la restitution lorsque ce n'est pas vraiment nécessaire et (2) il permet d'obtenir une qualité supérieure à celle obtenue avec d'autres mécanismes de contrôle d'erreur et de tampon de lissage publiés précédemment.

Nous utilisons notre algorithme de contrôle dans le cadre des services *non-elevated* qui permettent aux applications de sélectionner un service à délai de transmission réduit au prix de taux de pertes de paquets plus élevés. Le compromis entre délai et pertes n'est pas limpide dans le cas des sources audio car celles-ci pourraient être forcées de compenser les pertes supplémentaires par un augmentation de la redondance (FEC) ce qui entraînerait un accroissement du délai de bout en bout. Nous montrons que l'utilisation d'un service *non-elevated* permet d'améliorer la qualité, mais que le choix de la classe de service optimale dépend des conditions dans lesquelles se trouvent le réseau et de

l'importance que l'utilisateur attache au retard. Sur base de ces observations, nous proposons un algorithme de sélection de service qui permet à une source de déterminer en temps réel le service procurant la meilleure qualité.

De plus, une source audio qui utilise le service *best-effort* (meilleur service possible) de l'Internet est tenue de contrôler son débit de manière a réagir aux congestions dans le réseau et à partager la bande passante de manière équitable avec les autres connexions. Les mécanismes de contrôle de congestion actuels tendent a contrôler le nombre de paquets émis par les sources de manière à ajuster le débit en fonction du niveau de congestion dans le réseau. Toutefois, la voix constitue un trafic "inélastique" où les paquets sont émis à intervalles réguliers mais où la taille de paquet peut varier en fonction du codeur utilisé. Par conséquent, les mécanismes standards de contrôle de congestion ne sont pas applicables à ce type de trafic. Nous présentons un choix de trois modifications possibles pouvant être apportées aux protocoles de contrôle de congestion, que nous évaluons au moyen d'une analyse mathématique et de simulations de réseaux.

Contents

Chapter 1

Introduction

1.1 Dissertation Overview

Internet Telephony has evolved during the past decades from “a toy for long distance lovers” to a key application viewed as an important source of revenue by major providers and constructors. Packet-based telephony integrates both data and real-time voice traffic on the same infrastructure and hence allows easier introduction of new multimedia services and integrated business solutions. In spite of this increased flexibility, in order for Internet to constitute an attractive alternative to traditional Public Switched Telephone Network (PSTN), it must provide high quality Voice over IP service (VoIP).

In this dissertation, we are interested in the provisioning of toll quality Voice over IP service. As one of the key features of such a service is to enable natural interactions between users, our main concerns are problems related to network delay in Internet Telephony.

Transport of real-time, interactive audio over IP networks often suffers from packet loss and variations in the network delay (jitter), which calls for two types of corrective actions. First, Forward Error Correction (FEC) [92] is an efficient way to mitigate the impact of packet losses. However, it results in increasing the end-to-end delay, since the destination has to wait for the redundant packet(s) to be received in order to repair packet losses. Moreover, FEC increases the bit rate requirement of an audio source. Second, adaptive playout buffer algorithms at the receiver compensate for jitter. Again, this may come at the expense of some additional delay.

When used over the standard IP best effort service, an audio source should also control its rate in order to react to network congestion and to share the bandwidth in a fair way [103, 93, 21]. Bolot et al [22] proposed an adaptive rate/error control which optimizes a subjective measure of quality and incorporates a rate control. Their algorithm supposes

that the destination plays the best copy received of a given packet. They neither consider losses due to playout buffer overflow nor try to optimize the overall end-to-end delay. In particular, they do not manage the additional delay due to FEC.

However, it is recognized that the end-to-end delay has an impact on the perceived quality of interactive conversations, with a threshold effect around 150 ms [11, 10] for strongly interactive conversations, whereas many voice coders can tolerate some small loss without severe penalty. As a result, the FEC scheme in [22] may in some cases increase the delay when it would be more important to keep delay low while accepting some small loss. This is our main motivation for developing adaptive FEC and playout adjustment schemes that are delay aware (i.e., that take into account the impact of delay in their design) and that include a TCP-Friendly rate control.

1.1.1 A Delay Aware FEC scheme

In **Chapter 4**, we start by proposing a first adaptive error control scheme for real time audio over best effort networks which is *delay aware*, namely, which chooses the FEC according to its impact on the end-to-end delay. We consider the perceived audio quality as a function of the audio reconstructed rate at the destination *and* of the end-to-end delay. Then, we formulate our control problem as an optimization problem, and solve it numerically and theoretically. This first formulation of the optimization problem is based on the assumptions that (1) losses can be modeled with a Gilbert process, (2) the playout delay is equal to the delay that would be used if FEC were absent plus an additional delay to be able to use FEC (as in *rat* and *freephone* [6, 2]) and (3) a *play best* strategy is used at the destination, namely that the destination plays the best copy received before the playout time. This scheme also incorporates a TCP-Friendly rate control module (as the one discussed in Chapter 7) in order to ensure that the sending rate conforms with current practice for the Internet. This gives the basis for a first rate/error/delay control algorithm which (1) optimizes our delay-aware measure of audio quality and (2) is TCP-Friendly. The scheme is implemented in ns2 [4]. We show by simulation that the delay aware scheme increases the utility of a source by avoiding that it wastes delay on the FEC when it is not necessary.

1.1.2 Joint FEC and Playout Delay Adjustment for Internet Telephony

Our first error control scheme is based on the philosophy that if some redundancy (FEC) was added by the source then the destination should wait for the redundant information to arrive. This philosophy is challenged by the work of Rosenberg et al [99], who tackle the problem of the delay introduced by FEC, in the case of non-delay-aware FEC. They

point out that waiting for all the redundant information is inappropriate when the loss rate is low and propose a number of playout adaptation algorithms that provide a coupling between FEC and playout buffer adaptation. These algorithms suppose that a *play first* strategy is used at the destination, namely, that the destination plays the first copy of a given packet it received correctly. These coupled playout algorithms improve the delay performance for non delay aware FEC schemes, since they do not require a destination to wait for the FEC packets that are not needed. However, this is less clear in the case of delay aware FEC, since the source sends FEC packets only when necessary. This is one of the questions addressed in Chapter 5.

More generally, we consider the problem of joint FEC and playout adjustment at source and destination of an interactive audio source, while accounting for the impact of delay in the perceived utility. We propose and analyze two solution methods of increasing complexity that solve the joint problem. The first one ("partial" method, called *N1* in this dissertation) adjusts the playout buffer at destination using the coupled algorithms from [99]; the FEC scheme (at the source) is aware of the playout delay computed at the destination and adjusts redundancy as a function of network characteristics and of playout delay, so as to maximize the perceived audio quality. Method N1 supposes that the destination uses a *play first* strategy (to be consistent with the use of a coupled playout algorithm). A second, more elaborate method ("complete" method, called *N2*), jointly chooses both the playout delay and the FEC scheme such that the perceived audio quality is maximized. In the latter method, the playout delay and the FEC scheme are parameters of the optimization problem, while in the former, the playout delay is adapted separately and the best FEC scheme is chosen given this playout delay. In method N2, we consider the use of both play first and play best strategies at the destination.

As a basis for comparison, we also use two combinations of existing FEC and playout adjustment schemes. The first one (method *O1*) uses the first delay aware FEC adjustment method proposed above (together with the classical playout adaptation, plus enough delay to wait for all redundant information to be received at destination). The second one (method *O2*) combines the non delay aware FEC scheme proposed by Bolot [22] with the coupled playout delay adjustment in [99].

In addition to the development of methods N1 and N2, we address the following questions:

1. Are there significant quality improvements by using the complete joint method N2 ? More generally, how do the different methods compare ?

2. Is it worth using a joint FEC/playout adaptation scheme when delay aware FEC is used or, as mentioned earlier, can we rely on the source to send only those FEC packets that are necessary and wait for all FEC to be received ?

3. With the complete method N2, is it preferable to adopt a *play first* or a *play best* strategy ?

Our technical approach for designing N1 and N2 is as follows. We start by considering the perceived audio quality as a function of (1) the audio reconstructed rate at the destination, (2) the packet loss rate *and* (3) the end-to-end delay. We model the channel by a (1) packet loss process that we assume to be a Gilbert loss process and (2) a stationary delay process (not necessarily i.i.d.). We assume that the network delivers packets in sequence, a reasonable assumption since audio applications have a relatively small rate. We show that, given this assumption, our method needs only to know the marginal distribution of delays (i.e. the latter hypothesis encompasses all the needed information on the joint delay distribution). We further suppose that (3) delay and losses are stochastically independent, an assumption which makes sense if packet losses are due to active queue management schemes such as RED. Then we write our FEC and playout control problem as an optimization problem, and solve it numerically. We also incorporate a TCP-friendly constraint into our design.

We designed and implemented methods N1, N2, O1 and O2 and found by simulation that:

1. Our complete scheme N2 performs better than the partial scheme N1 and the combinations of existing schemes O1 and O2, in the cases where end-to-end delay is important.

2. Contrary to some intuition, it is worth using a joint playout adjustment algorithm when a delay aware FEC scheme is used. Classical playout delay adjustments as used in *rat* and *freephone* lead to excessive playout delay.

3. When used with the joint adaptation scheme N2, the *play first* and *play best* strategies have similar performance.

The Joint FEC and Playout Delay Adjustment scheme for Internet Telephony is presented in **Chapter 5**.

1.1.3 Audio on Non Elevated Services

Another motivation to develop delay aware schemes for FEC and playout adjustment is the emergence of a number of proposals for Internet differentiated services which offer a tradeoff differentiation between delay and packet drop probability (or throughput) [66, 50]. These differentiated services are called *non-elevated* services. With non-elevated services, the low delay class is likely to receive more packet loss.

In **Chapter 6**, we address the following question: does it make sense, for an audio application, to use this kind of non-elevated service, since it may be forced to compensate for the additional loss by more FEC, and thus more end-to-end delay?

To answer this question, we perform simulations where audio sources with the delay-aware error and playout control scheme make use of the Alternative Best Effort (ABE) service [66]. We show that the use of such a service can lead to quality improvement but that the choice of service depends on both (1) the network load and (2) the importance that users attach to delay. In the light of these results, we propose an adaptive service choosing algorithm that allows the source to choose in real time the service leading to the highest quality. Finally, using the optimal error and service control scheme, we evaluate the benefit brought by a non-elevated service compared to a single-class best effort service.

1.1.4 End-to-end congestion Control for Variable Packet Sizes

In **Chapter 7**, we address problems related to the design of end-to-end congestion control protocols for applications, such as VoIP, that need to send packets of different size than the maximum transmission unit (MTU).

The goal of end-to-end congestion control is to adapt the resource usage of a flow to the available resources in the network, and to do so in a way that resources are shared fairly between competing flows. Usually, the limited resource is either bandwidth or the packet rate. When the period of time it takes to handle a packet is mainly composed of the serialization delay, bandwidth will be the limiting factor. If this time interval is mainly composed of the packet processing time at the router, the packet rate will be limited.

Current congestion control mechanisms are based on packets (i.e., they aim to reduce or increase the number of packets sent per time interval to adjust to the current level of congestion of the network). This is the correct behavior in a packet rate limited environment. Furthermore, if all flows use the same packet size, for example the MTU, then bandwidth will be shared fairly among them also in the case of a bandwidth limited bottleneck. In an environment where the bottleneck is bandwidth limited but flows may use packets of different sizes, however, resources will not be shared fairly among flows but the resource usage will depend on the packet size. A packet size different from the MTU may be necessary for applications such as VoIP, where a high packet rate needs to be maintained in order to provide audio transmission with a sufficiently low delay.

In this dissertation, we study the impact of variations in the packet size on equation based congestion control. Ultimately, it is desirable to design a congestion control mechanism that ensures a fair sharing of resources independently of the packet size of a flow. To this end, it is necessary to determine:

- a rate that is fair to competing flows with different packet sizes with respect to the limited resource and

- how to calculate this rate, given that a different number of packets per time interval over which the loss event rate is measured will introduce a bias in the measured loss event rate.

We base our analysis on the unicast TFRC protocol [56] and the multicast TFMCC protocol [111]. In TCP-friendly equation based congestion control, the sending rate of a flow is adapted to the rate a TCP flow would achieve under the same network conditions. This rate is derived using a model for TCP throughput based on the current round-trip time (RTT) and loss event rate. TFRC and TFMCC are very similar in the way a fair sending rate is calculated and hence the considerations in this dissertation apply to both protocols, unless stated otherwise.

We give a quantitative analysis of the bias introduced into the loss measurement process when sampling the bottleneck at different rates. Consequently, three alternative modifications to the loss measurement mechanism of equation-based congestion control protocols are presented that remove this bias. These mechanisms are evaluated in detail through mathematical analysis and network simulation. Modifications to the loss event rate measurement mechanisms must ensure that a flow sending small packets does not achieve a higher throughput than a flow sending larger packets. At the same time, a flow sending small packets should not achieve a much smaller throughput than what is justifiable given the actual resources used by that flow. When the bottleneck is bandwidth limited, the presented mechanisms aim to achieve a sending rate comparable to that of a TCP flow with path MTU discovery under similar network conditions. Hence, for a fair sharing of resources, it is no longer necessary that competing flows have the same packet size.

1.1.5 VoIP Measurements

The first chapters of this thesis aim at providing potential solutions to improve the quality (and delays) of VoIP when the network fails to provide a high quality delivery service. In **Appendix A**, we consider all possible causes of quality degradation of a VoIP service and we assess the feasibility of the deployment of a toll quality VoIP service over an IP backbone.

Recently, tier-1 Internet Service Providers (ISPs) have shown an ever increasing interest in providing voice and telephone services over their current Internet infrastructures. Voice-over-IP (VoIP) appears to be a very cost effective solution to provide alternative services to the traditional telephone networks.

However, ISPs need to provide a comparable quality both in terms of voice quality and availability of the service. We can identify three major causes of potential degradation of performance for telephone services over the Internet: network congestion, link failures

and routing instabilities. Our goal is to study the frequency of these events and to assess their impact on VoIP performance.

We use passive monitoring of backbone links to evaluate the occurrence and impact of network congestion on data traffic. Passive measurements carried over different locations in the U.S. Sprint IP backbone allow us to study the transmission delay of voice packets and to evaluate the degree of congestion. However, this kind of measurement cannot provide any information related to link failures or routing instabilities.

For this purpose, we have deployed an active measurement infrastructure in two locations well connected to the backbone. We capture and timestamp the probe packets at both ends to quantify losses and observe the impact of route changes on the voice traffic. We performed many week-long experiments in order to observe different link failure scenarios.

Given that all our measurements take place in the same Autonomous System (AS) we also complement our data with IS-IS routing information [31] collected in one of the backbone Points of Presence (POPs). This additional information give us a fairly complete view of the events that occur during our experiments. Indeed, active probes and routing information give us the capability of identifying precisely the links, the routers and even the interfaces that are responsible for failures or instabilities in the network.

Our findings indicate that the Sprint IP backbone network is ready to provide a toll-quality voice service. The level of congestion in the backbone is always negligible and has no impact on the voice quality.

On the other hand, link failures can impact the availability of VoIP services. We discovered that link failures may be followed by long periods of routing instability, during which packets can be dropped because forwarded along invalid paths. Such instabilities can last for tens of minutes resulting in the loss of reachability of a large set of end-hosts.

1.1.6 Fairness of TCP Vegas

Finally, an auxiliary part of the thesis is devoted to the study of the fairness of TCP Vegas [28] (**Appendix B**). Our objective in this chapter is to better understand the dynamics of the congestion control algorithm implemented in TCP Vegas and to better assess its possible use in the context of TCP friendly applications.

In contrast to the TCP Reno, which induces congestion to learn the available network capacity, TCP Vegas uses measures of round trip time (RTT) as congestion feedback. A Vegas source anticipates the onset of congestion by monitoring the difference between the rate it is expecting to see and the rate it is actually realizing. The actual and expected rates in a connection are evaluated using, respectively, the actual value of the RTT and the minimum value of all RTTs ever measured in this connection (this is the estimation of the propagation delay). Vegas' strategy is to adjust the congestion window size in an attempt

to keep the difference between these two rates between two parameters, α and β.

Several drawbacks of TCP Vegas have been pointed out in the literature. First, because of the default values of α and β in its implementation [28, 15], TCP Vegas does not share the total bandwidth among connections in a fair way [24, 81]. Secondly, the same unfairness can be observed in case of an inaccurate estimation of the propagation delay.

This led Hasegawa [63, 64] to propose an enhanced Vegas in which $\alpha = \beta$. He shows that this setting leads to oscillations in the window size values, and claims that their amplitude can provide an accurate estimate of the propagation delay. This is crucial as even in this setting, the accurate estimate is necessary to ensure a fair sharing of the bandwidth.

In this chapter, we first check the fairness of the enhanced TCP Vegas proposed by Hasegawa. In particular, we address the problem of the propagation delay estimation. This leads us to review the case where $\alpha < \beta$. Our main results follow.

Under the $\alpha = \beta$ setting, we find that the rate oscillations, which we show increase with the rate value, do not allow a connection to accurately estimate the propagation delay. Also, any over-estimation of the propagation delay of a given connection will increase its rate, an increase that becomes more pronounced with the over-estimation factor.

When $\alpha < \beta$, we show that the rate of a connection converges to a stable value that depends on the arrival order of all connections. As a result, the first connections to be established will be favored when the propagation delays are properly estimated. Yet, in later connections the propagation delays are overestimated and so their rates are greater than what they should be. These two effects tend to counterbalance each other but the second tends to dominate.

We conclude that the use of TCP Vegas in the future should rely on the setting $\alpha = \beta$ but requires the propagation delay to be correctly estimated. Since this may be difficult to achieve in today's Internet during periods of congestion, it is preferable not to deploy TCP Vegas at present to avoid (un)fairness problems.

1.2 Dissertation outline

This thesis is organized as follows. Chapter 2 gives an overview of the state of the art of VoIP. Chapter 3 discusses the importance of delay in interactive communications. A first step towards a delay-aware error control is introduced in Chapter 4. Chapters 5, 6 and 7 constitute the main contributions of the dissertation. Chapter 5 presents the core of the delay-aware solution, namely the adaptive joint playout buffer and FEC adjustment scheme for Internet Telephony. In Chapter 6, which is an important application of results found in Chapter 5, we evaluate the benefit for audio sources to use a non-elevated service and we present our service choosing algorithm. Chapter 7 is devoted to important results

on end-to-end congestion control for variable packet sizes. In Appendix A, we assess the feasibility of the deployment of a VoIP service over the Sprint IP backbone. The study of the fairness of TCP Vegas is given in the Appendix B. Finally, Chapter 9 concludes this dissertation.

1.3 Contributions

We can summarize contributions of this thesis as follows:

- we designed a joint FEC and playout adjustment scheme that is delay aware and showed that it performs better than any combination of existing FEC and playout adjustment scheme in the case where delay is important to the user.
- we used this delay-aware error control and playout adjustment scheme to evaluate the benefits, for an interactive audio source, to use a non-elevated service such as ABE and showed that the use of such a service can lead to quality improvement but that the choice of service depends on both (1) network load conditions and (2) the importance that users attach to delay. We proposed an adaptive service choosing algorithm that allows the source to choose in real time the service leading to the highest quality.
- we analyzed the bias introduced into the loss measurement process of existing equation based congestion control protocols when sampling a bottleneck at different rates. We presented three alternative modifications to the loss measurement process of these protocols that remove this bias.
- within the Sprint VoIP measurement project, we assessed the feasibility of a toll quality VoIP service over the Sprint IP backbone. We showed that congestion was not a hindrance to VoIP on the backbone but that link failures are the major cause of quality degradation on the backbone.
- we studied the fairness of TCP Vegas and showed that even in the case where $\alpha = \beta$, the problem of over-estimation of the propagation delay leads to an unfair distribution of bandwidth among users. We also showed that the problem of over-estimation is partly compensated for by the unfairness introduced by setting $\alpha < \beta$, but that the two effects do not cancel out completely.

Chapter 2

State of the Art

In this Chapter, we start by giving the 'Big Picture' of the VoIP scenario considered in this dissertation. Then, we provide background information on

- Error Recovery Techniques (Section 2.2) to cope with packet losses,
- how to model Packet Loss process of audio traffic (Section 2.3)?
- Playout Adjustment techniques (Section 2.4) to absorb delay variations,
- Congestion Control (Section 2.5). In particular, we introduce the constraints imposed when designing a TCP Friendly rate control for audio; the hardest constraint being to keep a fixed time interval between audio packets. And finally,
- the E-model (Section 2.6), a quality measure that was standardized by the International Telecommunications Union,
- Quality of service in IP Networks (Section 2.7). In particular, we introduce Non-Elevated services, a new generation of enhanced best effort service that offer a tradeoff differentiation between delay and losses.

2.1 A VoIP system : The Big picture

Figure 2.1 shows the big picture of the VoIP system considered in this thesis. The first component is the encoder which periodically samples the original voice signal and assigns a (usually fixed) number of bits to each sample, creating a constant bit rate stream. The traditional sample-based encoder G.711 uses Pulse Code Modulation (PCM) to generate 8-bits samples per 0.125 ms, leading to a data rate of 64 kb/s. In the same family of sample based encoders, G.726 uses ADPCM to achieve 16-40 kb/s. Recent frame-based encoders provide drastic rate reduction (e.g., 8 kb/s for G.729, 5.3 and 6.3 kb/s for G.723.1) at the expense of additional complexity and encoding delay as well as lower quality. Further

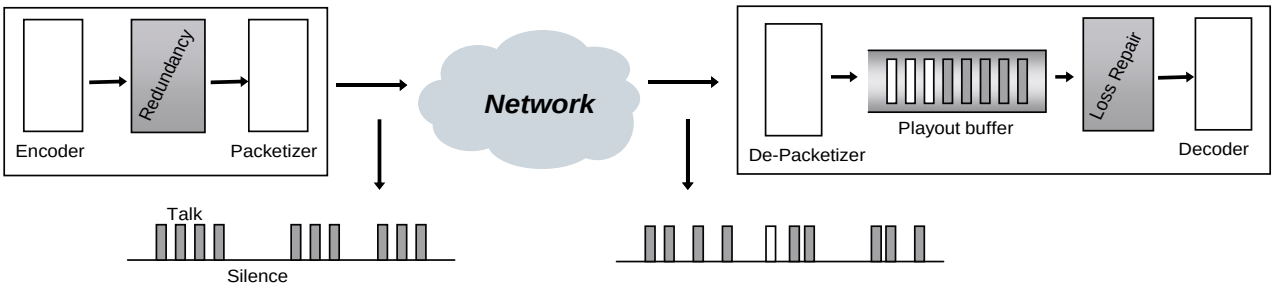


Figure 2.1: A VoIP system: The Big Picture

reduction in the data rate can be achieved by means of Voice Activity Detection (VAD), in which case no signal is encoded during silence periods. Usually, VAD systems tend to elongate the talkspurts by a period called the hangover time. Speech can be modeled with a good approximation as a Talk/Silence process with Talk and Silence periods that follow exponential distributions with a mean of 1.2 and 1.8 sec respectively, after applying hangover.

The packetizer encapsulates a certain number of speech samples (for G.711) or a certain number of frames (for G.729, G.723) into packets of equal sizes and adds the RTP (12B), UDP (8B) and IP (20B) headers.

As the voice packets are sent over an IP network, they are subject to variable delays and network drops.

An important component at the receiving end is the playout buffer whose purpose is to absorb variations in delay and provide a smooth playout. This is achieved by holding arriving packets until a later playout time in order to ensure that there are enough packets buffered to be played out continuously. Further information on playout adjustment schemes is given in Section 2.4.

The playout buffer delivers a continuous stream of packets to the decoder which reconstructs the speech signal. Decoders often implement Packet Loss Concealment (PLC) that produces a replacement for a lost packet, similar to the original one, by filling in silence or noise, by interpolating or even by regenerating the packet from the surrounding ones. While error concealment can mask single packet losses, loss bursts can cause noticeable dropouts in the receiving signal. That is why a source usually has recourse to Forward Error Correction to alleviate loss bursts of small number of packets. The principle of FEC is to send redundant information, along with the primary information, in order to allow a receiver to reconstruct (exactly or approximately) the missing audio signal. FEC allows to significantly improve the perceived quality in case of congestion, but it has the main drawback of increasing the end-to-end delay, since the destination has to wait for the redundant packets to be received in order to reconstruct the missing sample.

Further information on error recovery techniques is given in Section 2.2.

Each of the above components along the path of the packetized voice, may introduce delay and loss. The components of the end-to-end delay are the following:

- **Encoding/decoding and packetization delay**. The packetization delay is the time needed to collect all voice samples that end up in one packet. This delay scales linearly with the packetization interval, and its choice is a tradeoff between effective bit rate and delay. In this dissertation, we will consider that packetization delay is constant and, if not otherwise specified, that it is equal to 20ms[1]. Although the voice encoding/decoding process can take a significant amount of time when performed by software based implementations, they are reducible by using fast hardware implementations. Without access to specific implementation information and processing delay measurement, we will consider that this delay is very small and can be neglected. In practice, this delay should be taken into account in the end-to-end delay.

- **Propagation, serialization and queuing delay in the network.** Propagation and Serialization delays are not compressible and can be significant. Queuing delay, on the other hand, can be reduced, to some extent, if some form of Quality of Service is implemented in the Network.

- **Buffering at the receiver in order to remove jitter and wait for the redundant information.** This delay component is a variable part of the delay that can be significant. Since this component can be completely controlled by the VoIP endsystem, it is of primary concern in this dissertation.

Distortion of the original voice signal may be due to:

- **The use of a low bit rate encoder**

- **Packet losses in the Network** and finally

- **Drops at the receiver when packets arrive after their scheduled playout time.**

Another important impairment is echo, the reflection of the participants' signals, perceived as delayed and attenuated versions of their own voices. The larger the end-to-end delay, the more annoying is the echo. Although one might at first think that echo cannot happen in a packetized voice system, reflections may indeed happen (i) at the four-to-two wires hybrid connection between a packet and a circuit switched network and (ii) at the PC end-point when the microphone picks up the remote person's voice from the speaker as well as multiple reflections in the room and bounces them back. Both types of echo can

[1] A packetization interval of 20ms is used by many VoIP systems, despite the low payload efficiency, because it allows to keep the end-to-end delay relatively low.

be controlled by an echo canceler, that should be located as close to the source of echo as possible.

2.2 Error recovery

An audio transport protocol may cope with packet losses by [92]: (1) retransmitting dropped packets, (2) using error-concealment algorithms to correct the losses, or (3) applying Forward Error Correction (FEC) to reconstruct the missing packet.

Retransmission algorithms based on Automatic Repeat reQuest (ARQ) have been successful in protocols like TCP, but they are typically not acceptable for real-time audio applications since they dramatically increase the end-to-end delay. FEC, on the other hand, is an attractive alternative to ARQ since it provides relatively low-delay performance. The principle of FEC is to send redundant information, along with the original information, so that lost data can be recovered, at least partially, from this redundant information. When FEC fails to recover from a loss, applications can resort to *error concealment* algorithms at the receiver to correct the effect of missing packets [62]. Error concealment algorithms[62, 100, 9, 101] use repair mechanisms like insertion, interpolation or regeneration. These techniques work well for relatively small loss rates ($\leq 10\%$) and for small packets (4-40ms of audio) but break down when the loss length approaches the length of a phoneme (5-100ms). More sophisticated concealment techniques, such as Adaptive Packetization and Concealment (AP/C)[100], exploit the network loss characteristics and the property of long-term correlation within a speech signal together, to mitigate the impact of packet losses. This is accomplished by an adaptive choice of the packetization interval of the voice stream at the sender. Besides, modern speech codecs, such as G.729 [9], use concealment algorithms that are defined as part of their specification. These algorithms interpolate the missing codec parameters based on surrounding and previous values. These techniques lead to a good quality of the reconstructed speech if the number of consecutive lost frames is small and if the loss does not occur at an unvoiced/voiced transition [101]. Hence, error-concealment schemes should not be regarded as substitutes for FEC, but rather a complement to the latter.

FEC techniques can be classified as *media independent* and *media specific*. Media independent FEC uses block codes (*e.g.* based on Reed-Solomon [18] or on parity codes [104, 48]) to provide redundant information. Each code takes a codeword of k data packets and generate $n-k$ additional check packets for the transmission of n packets over the network. Such a code, denoted as an (n, k) code, is able recover all losses in the same block if and only if at least k out of n packets are received correctly. These techniques have the advantage of recovering the loss packet in a bit-exact way. On the other hand, they have the disadvantage of introducing additional delays that can be significant (up to

$n-1$ packet intervals).

Media specific (also called *signal processing*) FEC is used by audio conferencing tools such as *rat* [6]. The principle of the *signal processing* FEC is to transmit each segment of audio, encoded with different quality coders, in multiple packets. When a packet is lost, another packet containing the same segment (maybe encoded differently) can be able to cover the loss. This approach [42] has been advocated by Hardman *et al* [62] and Bolot *et al* [20] for use on the Mbone, and extensively simulated by Podolsky *et al*. [93].

The first transmitted copy of the audio segment is called primary encoding and subsequent transmissions secondary encodings. With *signal processing* FEC, redundant audio segments are piggy-backed onto a later packet, which is preferable to the transmission of additional packets, as this decreases the amount of packet overhead and routing decisions. For example, in the case of a single redundant segment, packet n contains, in addition to its encoded samples, a redundant version of packet $n-1$. This redundant information is usually obtained using a lower-bit-rate, lower-quality encoding than the primary information. This simple scheme only recovers from isolated losses but can be modified (as proposed in [21]) to recover from consecutive losses as well by carrying in packet n redundant versions of packets $n-1$ and $n-2$, or of packets $n-1$, $n-2$ and $n-3$ or of packets $n-1$ and $n-3$ etc.

Obviously, the more redundant information is added at the source, the more lost packets can be reconstructed. However, sending more redundant copies implies increasing the bandwidth requirement at the source (and henceforth the packet loss rate) and increasing the end-to-end delay (since the receiver has to wait longer for the redundant information). Moreover, it would make little sense to add much redundant information when the network load is low and the packet losses are rare.

Therefore, a robust FEC scheme should be adaptive and choose the FEC according to the network characteristics (such as packet loss process, available bandwidth,...) at any given time and depending upon its impact on the end-to-end delay. In the following sections we describe the packet loss process of audio packets in the Internet and we review the rate control schemes that have been proposed in the literature.

2.3 Loss Process of Audio Packets

The efficiency of the FEC depends on the characteristics of the loss process of audio packets. Typically, FEC is more efficient when the consecutive number of lost packets is small.

There has been many research efforts in the measurement and modeling of end-to-end Internet characteristics [91, 19, 112, 71, 25]. The main result is that the correlation structure of the loss process of audio packets can be modeled with low order Markov chains. In

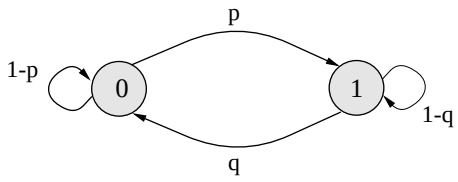


Figure 2.2: The Gilbert model

particular, a two-state Gilbert model was found to be an accurate model in many studies. Moreover, it was found that the distribution of the number of lost packets in a loss period is approximately geometric [20, 112] (loss measurements in Appendix A on page page 135 are in line with these results). These results confirmed that FEC schemes are well suited for interactive audio applications in the Internet. The Gilbert model (depicted in Figure 2.2) is a two-state model in which state 1 represents a packet loss and state 0 represents a packet reaching the destination. The parameters p and q denote respectively the probabilities of passing from state 0 (no loss) to state 1 (loss) and from state 1 to state 0. The stationary probabilities to be in state 1 (π_1) and in state 0 (π_0) are given by:

$$\pi_1 = \frac{p}{p+q} \quad \text{and} \quad \pi_0 = \frac{q}{p+q}$$

In absence of redundant information, the packet loss rate is given by the unconditional probability to be in state 1: $PLR = \pi_1$. The Gilbert model also allows to compute the packet loss rates after reconstruction when FEC is used. The n-stage transition matrix $P^n = [p_{ij}^{(n)}]$, $i, j \in \{0, 1\}$ is given by:

$$P^n = \frac{1}{p+q}\begin{bmatrix} q & p \\ q & p \end{bmatrix} + \frac{(1-p-q)^n}{p+q}\begin{bmatrix} p & -p \\ -q & q \end{bmatrix} \tag{2.1}$$

2.4 Playout Adjustment Algorithms

Jitter is compensated for by means of adaptive playout buffer algorithms at the receiver. Most existing playout adaptation algorithms work by taking some measurements on the delays experienced by packets and updating the playout delay on a talkspurt to talkspurt basis. The main purpose of playout adaptation algorithms is to trade delay for loss. Let D be the playout delay which is defined as the difference between playout time and generation time for all the packets in a talkspurt; clearly, the larger D, the smaller the fraction of late packets.

Classical methods for playout adaptation (in absence of FEC) were proposed in [95, 83]. [95] proposed four adaptation algorithms. Each algorithm computes in some way an estimate of the mean $\hat{d}$ and the standard deviation $\hat{v}$ of the delays experienced by previous packets and adjusts the playout delay at the beginning of each talkspurt to be $D = \hat{d} + 4\hat{v}$.

All four algorithms compute $\hat{v}$ in the same fashion, using an exponentially weighted moving average. The four algorithms differ only in their computation of $\hat{d}$: Algorithms 1 and 2 in [95] compute an exponential moving average of the delays, but with different weighting factors. Algorithm 3 sets $\hat{d}$ to be the minimum delay experienced during the prior talkspurt. Algorithm 4 performs delay spike detection. During a spike, the delay estimate tracks the delays closely, and after a spike, an exponential weighted moving average is used. For a detailed description of this algorithm, see [95]. The adaptive playout adjustment algorithm proposed in [83] tracks the network delay of recently received packets and maintains delay percentile information. This algorithm also performs spike detection. During spike mode, the delay of the first packet in a talkspurt is used as playout delay. During normal mode, the playout delay is the qth percentile among the last w packets received by the receiver.

When FEC is used, existing audio tools compute the playout delay as if FEC were absent (using one of the methods described above), compute the delay needed to make use of FEC and add the two. If FEC is not adaptive, it is clear that this method introduces too much delay when network losses are rare. To cope with this problem, new playout adjustment algorithms were proposed in [99] to integrate more smoothly the FEC packets into the playout buffer. The algorithms proposed in [99] include:
- *Virtual delay* algorithms that are all modifications of algorithms proposed in [95, 83]. The virtual algorithms use any of the existing playout adaptation algorithms which compute the playout delay D as a function of the packet delays but substitute the packet delays by the packet *virtual delays* that are defined as the difference in time between the earlier of the arrival and the recovery times, and the generation time of the packets.
- *A previous optimal* algorithm that uses the optimal delay for the previous talkspurt as playout delay for the next talkspurt. The optimal playout delay can be chosen to meet an arbitrarily chosen criteria. In [99], the optimal playout delay for a talkspurt is defined as the minimum delay that achieves a specified loss rate after reconstruction.
- *An analytical* algorithm that works by attempting to model the impact of network loss and delays on the application playout probability and the end-to-end delay. It then uses this model to find the playout delay D that meets a particular loss rate after reconstruction. For a detailed description of these coupled algorithms, see [99].

2.5 Rate Control

As we mentioned earlier, the addition of FEC repair data to a media stream is an effective means by which that stream may be protected against packet loss. However, the addition of large amounts of repair data when loss is detected will increase network congestion and hence packet loss, leading to a worsening of the problem which the use of FEC was

intended to solve. Therefore, the rate of audio sources should be controlled in order to avoid congestion collapse in the network.

In addition, it has been suggested that audio applications share resources fairly with each other and with current TCP-based applications. One way to ensure this is to implement some form of congestion control that adapts the transmission rate in a way that *fairly* shares congested bandwidth with TCP applications. One definition of *fair* is that of TCP *friendliness* [76] - if a non-TCP connection shares a bottleneck link with TCP connections, traveling over the same network path, the non-TCP connection should receive the same share of bandwidth as a TCP connection.

There has been significant previous research on TCP-Friendly control mechanisms and many control schemes were proposed in the literature. We can distinguish three main classes of control mechanisms:

1. window-based control mechanisms,
2. mechanisms based on additive increase, multiplicative decrease (AIMD) and,
3. equation-based mechanisms.

The control mechanisms closest to TCP are the window-based mechanisms [67, 97]. They maintain a *congestion window* which is used to control the transmission of the packets. Although window-based schemes exhibit a behavior very close to the one of TCP, their main disadvantage is their lack of flexibility. Since these protocols strictly adhere to TCP window dynamics, it would be hard to modify them to take into account timeliness requirements of real-time streams.

Another class of control mechanisms uses additive increase, multiplicative decrease (AIMD) in some form, but do not apply AIMD to a congestion window. The RAP protocol (Rate Adaptation Protocol) [96] employs an AIMD rate control algorithm based on regular acknowledgments sent by the receiver. Another AIMD protocol has been proposed in [105]. This scheme relies on regular RTP/RTCP [102] reports sent between sender and receiver to estimate the loss rates and round-trip times. While AIMD schemes exhibit good response to transient changes in congestion, real-time streams find decreasing the sending rate in multiplicative order in response to a single loss to be unnecessarily severe (as it can noticeably decrease the user-perceived quality [108]). For this reason, equation-based control mechanism seem to be leading candidates for mechanisms to provide relatively smooth congestion control for real-time traffic.

In Equation-based congestion control [76], the sender uses an equation to determine the rate TCP would achieve under the same network conditions and adjusts its sending rate to conform to this allowed sending rate. A key issue, when using these schemes, is to choose a reliable characterization of the TCP throughput. The TCP-Friendly protocols proposed in [108, 23] are based on the TCP response function first reported in [76] and

later formalized in [78]. Since this characterization does not take the *timeouts* into account, it has been reported in [78] that this model is not accurate for loss rates higher than 5%. Another formulation of the TCP response function was derived in [88]. It characterizes the throughput of a TCP connection as a function of the packet size, the round trip time, the TCP re-transmission timeout and the frequency of loss indications per packet sent[2]. Based on this model, Padhye and al. [88] proposed a scheme in which the receiver acknowledges each packet. At fixed time intervals, the sender estimates the packet loss rate experienced during the previous interval and updates the sending rate using this more complex model (see Equation (7.2)). Since this scheme updates the sending rate at fixed time intervals, it is suitable for use with multimedia applications. But it has the disadvantage to have a poor transient response at small time-scales.

Besides, Floyd and al. proposed the TFRC protocol [56]. In TFRC, the sender explicitly adjusts its sending rate as a function of the measured rate of *loss events*, where a loss event consists of one or more packets dropped within a single RTT. Their algorithm for calculating the loss event rate is based on the method of Average Loss Interval. It offers a very good tradeoff between responsiveness to changes in congestion and avoidance of unnecessary abrupt shifts in the sending rate. Unfortunately, TFRC cannot be used, as such, with audio streams because the loss measurement process proposed in [56] supposes that rate controlled flows adjust their sending rate by changing the interval between packets, the packet size being fixed (and in general equal to the MTU). In contrast, audio sources adjust their sending rate by changing the packet size while keeping the interval between packets constant. Modifications to the loss measurement process proposed in Chapter 7, allow sources that send packets at regular intervals (and adjust their packet sizes) to measure the same loss event rate as if they were sending MTU size packets (and adjusting their throughput by varying the interval between packets).

2.6 An Audio Quality Measure: the E-Model

Since the E-model is accepted as a standardized measure of audio quality, offering the possibility to encompass the combined influence of the distortion (caused by low bitrate encodings and the packet loss) and the end-to-end delay, we will use this model to build our utility function as described in Chapter 5. In this section, we give a brief overview of the E-model. A detailed description can be found in [39, 13, 12, 14].

The E-model predicts the subjective quality that is experienced by an average listener combining the impairment caused by transmission parameters (such as loss and delay) into a single rating $R \in [0, 100]$. The rating can then be used to predict subjective user

[2]Further details on Equation Based Congestion Control are given in Section 7.2.2 on page 86, where we detail the different models for long-term TCP throughput.

R-value range	MOS	Speech transmission quality
$100 - 90$	4.50-4.34	best
$90 - 80$	4.34-4.03	high
$80 - 70$	4.03-3.60	medium
$70 - 60$	3.60-3.10	low
$60 - 0$	3.10-1.00	very poor

Table 2.1: Speech transmission quality classes and corresponding rating value ranges

reactions, such as the Mean Opinion Score (MOS). According to ITU-T Recommendation G.107, every rating value corresponds to a speech transmission category, as shown in Table 2.1. A rating below 60 indicates unacceptable quality, while values above 70 correspond to PSTN quality (values above 90 corresponding to very good quality).

The E-model is based on the principle that transmission impairments can be transformed into *Psychological Factors* and that these factors are additive on a psychological scale. The E-model rating R is thus expressed as follows:

$$R = R_0 - I_s - I_d - I_e + A \tag{2.2}$$

where R_0 groups the effects of noise (such as background noise and circuit noise), I_s represents impairment that occur simultaneously with the voice signal (quantization), I_d is the impairment associated with the mouth-to-ear delay, and I_e is the impairment associated with signal distortion (caused by low bit rate codecs and packet losses). The advantage factor A is the deterioration that callers are willing to tolerate because of the 'access advantage' that certain systems have over traditional wire-bound telephony, e.g. the advantage factor for mobile telephony is assumed to be 10. Since no agreement has been reached for the case of VoIP services, we drop the advantage factor in this dissertation.

In this thesis, we focus on the impact of the one-way mouth-to-ear delay (via I_d) and the distortion (via I_e) on the quality of a packetized voice call. Other factors, such as background noise and or quantization effects also impair the quality (via R_0 and I_s) of a packetized voice call, but since they do not depend on the underlying transport network, we use the set of default values that are recommended in [13] for these parameters.

As a result, the rating R can be reformulated as follows:

$$R = 94.2 - I_d - I_e \tag{2.3}$$

From Equation (2.3), it follows that two calls with same rating can give totally different subjective impressions. One might produce a very clear, undistorted speech (i.e. $I_e = 0$) but suffer from a relatively large delay (i.e. $I_d = 10$). Another call might slightly

distort the speech (i.e. $I_e = 10$) while its delay is not noticeable (i.e. $I_d = 0$). However, the E-model predicts that a judging panel will award the same MOS to both calls, although for different reasons. Belowl, we provide details about the influence of these two parameters.

2.6.1 Delay Impairment I_d

The delay impairment I_d models the quality degradation due to the one way mouth-to-ear delay d[3]. The delay impairment I_d is the sum of three contributing impairments:

$$I_d = I_{dte}(d, TEL) + I_{dle}(d, LEL) + I_{dd}(d) \tag{2.4}$$

The terms I_{dte} and I_{dle} capture the impairments due to talker and listener echo respectively. TEL and LEL are the echo losses (in dB) at the points of reflexion and depend on the echo cancellation applied. In this dissertation, we assume that the echo losses at both ends are equal. In the sequel, we denote the echo loss by EL, with $EL = TEL = LEL$. This amounts to assuming that the same echo controllers are used at both ends. For packetized voice calls [39], $EL = \infty$ corresponds to a perfect echo control and $EL = 51$ corresponds to a simple yet efficient echo controller. The third delay-relative impairment, I_{dd} is the loss of interactivity. [13, 12] give analytical formulas to compute I_d as a function of the echo loss and the mouth-to-ear delay.

2.6.2 Equipment Impairment I_e

The impairments introduced by distortion are brought together in I_e. Currently, no analytical expression allows to compute I_e as a function of parameters such as the encoding rate or the packet loss rate. Estimates for I_e must be obtained through subjective measurements. A few values for I_e are given in Appendix A of [14] for several codecs and several packet loss conditions and a recent study [70] provides additional results about the impact of losses on the perceived audio quality. We build our utility function based on the values given in [14], as explained in Section 5.2.

2.7 Non-Elevated Services

As the Internet has established itself as global communication infrastructure conveying different types of traffic ranging from file transfer to interactive audio applications, pro-

[3] ITU-T Recommendation G.107 [13] gives a fully analytical expression for I_d in terms of various delay measures (such as mouth-to-ear delay, delay from the receive side to the point where signal coupling occurs and delay in the four wire loop) and other parameters describing various circuit switched and packet switch inter-working scenarios. In pure VoIP scenarios, the various delay measures collapse into a single one, the mouth-to-ear delay.

viding some kind of service differentiation has been an important research problem for the past few years. This section reviews different proposals for differentiated services and introduces what we call the *non-elevated* services.

In order to enhance the single class best effort service, the Differentiated Services (DiffServ) [43] architecture was proposed as a more scalable solution, compared to previous approaches such as the Integrated Services (IntServ) architecture [27]. While the IntServ approach aimed at providing end-to-end guaranteed service on a per-flow basis, the DiffServ approach proposes a coarser notion of quality of service (QoS), focusing on aggregate flows in the core routers and intending to differentiate between service classes rather than provide absolute per-flow QoS measures. Among Diffserv proposals is the Expedited Forwarding (EF) [44], which aims to provide low queuing delay and extremely low loss guarantees. Several other models for scalable differentiated services such as the SCORE architecture [106] and the core-stateless architecture (called Corelite) [86] have been proposed. They both proposed IntServ type end-to-end absolute performance measures without per-flow state in the network core. A common feature of all these proposals is that they associate priority with low delay and hence higher throughout in case of congestion. Another approach proposed by Kilkki [75] offers applications the choice between several priority levels. This priority level is set for the entire flow as a function of how the flow deviates from its contractual rate; if a flow exceeds its rate, its priority level is low. In this framework, it is typically suitable for a flow to be adaptive: if it conforms to its rate, it will be able to obtain a low delay, but it will not get throughput priority. Although these methods are attractive, they all need some form of admission control, which increase the complexity of their deployment.

A fundamentally different approach was followed by some authors who proposed service differentiation without admission control. Crowcroft [38] first proposed a '1-bit' scheme as a low cost scheme to provide delay differentiation. In this proposal, analyzed by May et al. [79], packets with the service bit set to 1 receive serving priority while constrained to a smaller buffer. Later, Dovrolis et al. [41] described a proportional differentiated model where the quality between classes of traffic is proportional and is thus independent of the load within each class. Moret and Fdida [84] also described a two-class proportional differentiation model. All these proposals couple low delay with improved throughput, and are some form of priority. They should therefore be associated to some form of pricing discrimination.

In contrast, yet a simpler alternative was proposed by Hurley et al. under the name of Alternative Best Effort (ABE) [66] or by Firoiu et al. under the name of Best Effort Differentiated Services (BEDS) [50]. Both proposals offer the tradeoff between packet delay and packet loss probability (or throughput). In the sequel, we will refer to this kind of service as *non-elevated* service. In these proposals, a packet that is marked as 'low delay' will experience a low queuing delay but will be likely to receive more packet

loss. An attractive feature of these tradeoff services is that applications opting for the low delay class do not impact other applications that value higher throughput. Therefore, these services do not need to police how much traffic use the low delay class and they retain the operational simplicity of a single class best effort network. This would make them good candidates for the deployment of a low cost and low complexity differentiated service. The only question that remains is the following: is there a real benefit, for an interactive application such as Internet telephony, to trade delay for losses, since it may be forced to compensate the additional loss by more FEC and thus more end-to-end delay?

Chapter 3

Importance of Delay

Audio quality problems are not surprising because the current Internet provides users with single class Best-Effort service that does not promise anything in terms of performance guarantees. In this chapter, we start by discussing the importance of delays in interactive communications (Section 3.1). Then, we examine the problems brought by today's Best-Effort service (Section 3.2) and finally, we review the possible approaches that can be taken to solve these problems (Section 3.3).

3.1 Importance of Delays and Losses

The perceived quality of a telephone call largely depends on two factors: i) *distortion*, the difference between the received signal and the original one; and ii) *mouth-to-ear delay*, the time between the moment the speaker makes an utterance and the moment the listener hears it.

Packet loss directly contributes to the intelligibility of a word or a sentence. It causes voice clipping and skips. Some codec algorithms can correct for some lost voice packets but typically, only a single packet can be lost during a short period for codec correction algorithms to be effective (as explained in detail in Section 2.2).

Delay can have two effects on speech quality. First, it increases the subjective effect of any echo impairment. Second, even when echo is perfectly controlled, delay above 150 ms can interfere with the dynamics of a conversation and degree of interactions. ITU-T Recommendations G.114 [11] and G.131 [10] specify the following tolerable mouth-to-ear delay for connections with adequately controlled echo:

- a mouth-to-ear delay of up to 150 ms is acceptable for most user applications;
- a mouth-to-ear delay between 150 ms and 400 ms is potentially intolerable[1];

[1] For example, they state that international connections with satellite hops that have transmission times below 400 ms are still considered acceptable.

- a mouth-to-ear delay above 400 ms is unacceptable.

ITU-T Recommendations G.114 also states that highly interactive tasks may experience degradation even for values on the order of 100 ms. In an extensive study, Kitawaki et al. [74] describe the effect of end-to-end delay on speech quality in telecommunications, with human factors such as conversational mode taken into account. They measured the effect of delay using six different tasks involving more or less interruptions in the dialog. The delay detectability threshold was defined as the delay detected by 50% of a task's subject. As the interactivity required by the task decreased, the delay detectability threshold increased from 45 to 300 ms of one-way delay. In delay perception, the assumption was verified that a talker expects a particular response time from his partner, and he notices delays that are outside this expectation time window. In addition, delay perception is greatly influenced by temporal characteristics of conversation speech. They conclude that the effect of delay on a conversation differs appreciably, depending on the content and occasion but that long round-trip transmission delay in the range of 300-400 ms gives to subscribers considerable difficulties in communications.

In summary, even though the relative impact of delay on perceived quality depends on the degree of interactivity required for particular conversations, an average user perceives delay as a hindrance to interactivity from a threshold around 150 ms.

3.2 Does today's Best-Effort service hurt audio quality?

From the previous section, it appears clear that controlling both the mouth-to-ear delay and the distortion is key to offering high quality packet-based voice calls. However, the currently available Best-Effort Internet service was never engineered to handle voice traffic and it provides variable loss rates and delays. Various studies attempted to characterize end-to-end dynamics of the Internet [91, 19, 61, 45, 25] and large scale end-to-end measurement projects such as Surveyor [72], NIMI [3] or the Pinger [5] project measure end-to-end delay, loss and routing over a diverse set of measurement probes throughout the Internet. Some other projects, such as the Sprint IP Monitoring [7] project study performance of backbone networks through non-intrusive measurements. Results from these various studies exhibit a substantial variability, depending a lot on the paths under consideration, with typical packet loss rates from 0 (e.g. an average 0.1% to 0.3% is observed on some backbone networks) to 20% (during peak hour over some international links), and one way delays from a few ms to a few hundreds of ms (with jitter as high as 150 ms observed on some international paths [61, 45]).

3.3 Strategies to alleviate delays and losses

The main challenge in supporting interactive audio over wide area network is to provide synchronized playout of an undistorted audio signal within an acceptable delay (around 150 ms) in face of losses and stochastic end-to-end delays.

We see three different approaches to this problem: i) an end-to-end approach, ii) a network-oriented approach and iii) a 'mixed' approach.

3.3.1 End-to-End Approach

The end-to-end approach (or control approach) consists in taking measures at the source and destination to adapt to network conditions and to wisely control the application's parameters so as to make the most of the available channel. Different mechanisms exist to minimize the effect of loss and delay variations on the quality of the audio delivered to the destination. Compensation for loss end-to-end can be achieved efficiently using error recovery techniques such as packet-level FEC but waiting for the redundant information results in a delay penalty. Adaptive playout buffer algorithms can be used for jitter compensation but this again is accomplished at the expense of end-to-end delay. A large part of the delay budget of an audio application is spent in queues in the networks and in playout buffer at the receiver, while waiting for delayed packets and redundant information.

Alleviating losses and jitter is therefore a solvable problem, but reducing end-to-end delay remains a real challenge.

3.3.2 Network-Oriented Approach

The network-oriented approach tackles the heart of the problem and aims at reducing losses and jitter by improving network performance. This can be done either by resorting to over provisioning or by augmenting the best-effort service through introduction of Quality of Service (QoS) in IP networks. Supporters of over provisioning advocate that if no link in an IP network is ever more than 30% occupied, even in peak traffic conditions, then the packets should flow through without any queuing delays and elaborate protocols to give priority to one class of packets are not necessary. Even if this approach shows very good performances in the network core, in some cases, access links may be expensive and broadband access difficult to obtain, so that QoS might be desirable on access links, even if the core network is lightly loaded. Moreover, over provisioning is not as straightforward as it may appear since adding capacity can sometimes lead to a delay increase in congested networks (Braess paradox [34]). On the other hand, massive deployment of IP quality of service, based on Intserv [27] or Diffserv [43] for example, could involve large implementation efforts that some service providers are not yet ready to make.

The new generation Internet service providing users with a high quality of service end-to-end is therefore still under development and one can anticipate that many IP networks will continue to provide a Best-Effort service for quite some time.

3.3.3 Mixed Approach

A third approach, that lies somewhere in between the end-to-end and the network-oriented approaches, is the introduction of non-elevated services at congested points of the network. These services allow sources to reduce their queuing delay, but at the expense of an increased loss rate in periods of congestion. Non-elevated services constitute an attractive alternative to other types of QoS based on some form of reservation because they retain the operational simplicity of single class Best-Effort networks.

However, since these services offer a tradeoff differentiation between loss and delay (instead of giving a clear priority to delay-sensitive traffic), it is up to the source to evaluate which service best fit its needs, and hence, to take decisions end-to-end. Since additional losses may require the use of more FEC and hence incur an increase in mouth-to-ear delay, it is not clear whether the low delay class really provides a benefit to interactive audio applications.

3.4 Conclusion and Approach of the Thesis

While PSTN was optimized to provide very low delay, conquering delays in IP networks is still a challenge.

In this dissertation, we take an end-to-end approach to tackle the problem of alleviating losses and delays, while trying to manage the delay spent in the playout buffer. In addition, our results find interesting applications when used in conjunction with non-elevated services. Our strategy thus lies between the first and the third approaches introduced in this chapter.

Chapter 4

A First Delay-aware Error Control

4.1 Introduction

This chapter presents a first adaptive error control scheme for real time audio which is *delay aware*, i.e., which incorporates the impact on the end-to-end delay in the choice of FEC. The complete *delay-aware* solution, jointly adapting the playout delay and the redundancy is the subject of Chapter 5.

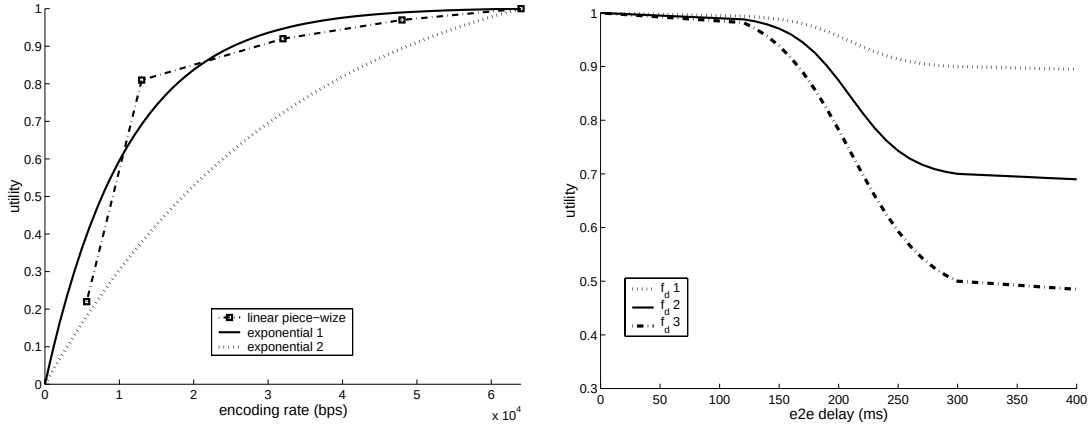
This chapter is organized as follows:

- Section 4.2 describes our utility function approach. We consider the perceived audio quality as a function of the audio reconstructed rate at the destination *and* of the mouth-to-ear delay.
- Section 4.3 presents the delay aware error and rate control. Our control problem is formulated as an optimization problem. This first formulation of the optimization problem is based on the assumptions that (1) losses can be modeled with a Gilbert process, (2) the playout delay is equal to the delay that would be used if FEC were absent plus an additional delay to be able to use FEC (as in *rat* and *freephone* [6, 2]) and (3) the destination plays the best copy received at the destination before the playout time.
- Results of simulations implemented in ns2 [4] are given in Section 4.4 and show that the delay aware scheme increases the utility of a source by avoiding that it wastes delay on the FEC when it is not necessary.

Details about the optimization problem and its solution are in Appendix C.

4.2 A utility function approach which accounts for delay

As mentioned above, the purpose of this chapter is to design an error control algorithm which is *delay aware*. To this end, we take a utility function approach.



(a) (b)

Figure 4.1: Utility as a function of (a) the rate only (for delay = 0), (b) the mouth-to-ear delay only (for encoding rate = 64Kbit/s)

For users employing our audio application, we define the utility not only in terms of sound quality at the destination but also in terms of *interactivity*. In other words, we consider the utility (quality) as being a function of the encoding rate of audio received at the destination *and of the mouth-to-ear delay*. This utility function $f : \Re^+ \times \Re^+ \rightarrow [0, 1]$ was obtained as follows. We characterized separately:

- **The utility as a function of the encoding rate**: $f_r : \Re^+ \rightarrow [0, 1]$. There exists objective and subjective methods to assess the quality of an audio source as a function of the encoding rate. Objective methods use specific signal metrics to assess the quality, such signal-to-noise ratio (SNR) [36]. These metrics, while easy to obtain, are only approximations for two main reasons. First, because they are sensitive to the characteristics of the signal, and hence to the words being pronounced. Second, because they often fail to correlate with perception properties of the human hearing system [109]. Despite these limitations, a SNR model can still give insight into the audio quality. Moreover, operational SNR curves (on a linear scale from 0 to 1) can be modeled using negative exponentials, the quality increasing rapidly at low rates, and more moderately at higher rates. Therefore, we considered two utility functions of the form: $f_r(x) = c_1 + c_2\, e^{-\alpha x}$, where c_1 and c_2 are constants, selected so that $f_r(64Kbits/s) = 1$ and $f_r(0) = 0$. Figure 4.1(a) depicts two of these exponentials, corresponding respectively to $\alpha = 0.0001$ and $\alpha = 0.00003$. Quality assessment models based on subjective measurements provide more accurate results but are more difficult to obtain. A widely used model is MOS (Mean Opinion Scores) where the perceived quality is usually rated on a 1 to 5 scale. Utility functions for adaptive flows can be obtained using score tables (scaled down between 0 and 1), and interpolating between values to obtain piece-wise linear utility functions. We

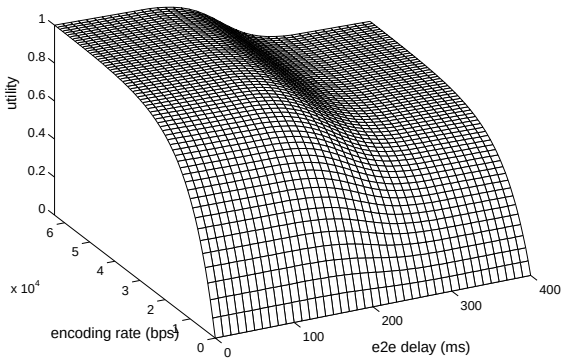


Figure 4.2: Utility as a function of the rate and the mouth-to-ear delay

present such a function in Figure 4.1(a). Besides, exponentially-decay functions were introduced in [29] to describe utility curves for adaptive application. These curves account for the fact that the quality increases slowly at very low rates (below the minimum rate required by the application), then rapidly at intermediate rates and again slowly at higher rates. In this chapter, we restricted ourselves to consider strictly exponential and piecewise linear curves.

- **The utility as a function of the mouth-to-ear delay**: $f_d : \Re^+ \rightarrow [0, 1]$. Even though an objective and unique function can not be set to describe the quality as a function of delay, various studies concluded that, for natural hearing, the mouth-to-ear delay should be approximately 150 ms [11, 10, 37]. While a lower delay can not really be appreciated by an average listener, delays above this threshold will be noticed by the users and will become a hindrance to interactivity. Moreover, it is also recognized that telephony users find one-way delay greater than about 300 ms more like half duplex connection than a conversation. These considerations lead us to consider utility curves like those represented on Figure 4.1(b). These utility functions present the following behavior: for delays below the critical threshold of 150 ms, the quality decreases very slowly as the users do not benefit from getting a lower mouth-to-ear delay. Above this threshold, the quality drops steeply as any increase of the mouth-to-ear delay hurts the interactivity. Then, above 300 ms, since the connection is considered as a half duplex conversation anyway, any further increase of the delay only slightly affects the quality (which is already low).

  Even though it agrees with common intuition, the nature of the exact behavior of this function remains out of the scope of this study. Our goal is not to validate a particular utility function but rather to show that the use of this kind of function allows to incorporate the impact on the mouth-to-ear delay in the choice of FEC.

To this end, we considered a set of utility functions of the form:

$$f_d(x) = \begin{cases} 1 - \gamma_1\, x & \text{if } x \leq 150 \\ b_1 \tanh(\beta(x - b_2)) + b_3 & \text{if } 150 < x < 300 \\ 1 - \delta - \gamma_2\, x & \text{if } x \geq 300 \end{cases}$$

where x is the mouth-to-ear delay (in ms), γ_1, β and γ_2 are parameters representing the steepness of the decrease in each of the 3 regions and δ determines the difference between the full and the half-duplex quality. b_1, b_2 and b_3 are constants selected to ensure the continuity of f_d. The utility functions depicted on Figure 4.1(b) were obtained by tuning these parameters. A user will opt for either one or the other depending on the importance she attaches to delay.

Then, the global utility is defined as the product of f_d and f_r:

$$f(x, y) = f_r(x)\, f_d(y)$$

where x and y represent respectively the reconstructed rate at the destination and the mouth-to-ear delay. Such a function is depicted on Figure 4.2.

In the simulation we have performed, we have used various values for the parameters of the utility functions.

Note that the packet loss rate after reconstruction also impacts the quality. Many codecs employ various concealment techniques which are able to mask such losses as long as $PLR_{FEC} \leq PLR_{max}$ where the threshold PLR_{max} depends on the codec. In this first delay-aware error control, the influence PLR_{FEC} is not incorporated in the utility function. We rather express the following condition: $PLR_{FEC} \leq PLR_{max}$ as a constraint. In the complete solution presented in Chapter 5, PLR_{FEC} is incorporated in the utility function.

4.3 Our Joint rate/error/delay Control

Once we have determined the influence of delay on the quality (utility) of the call, our goal is to **find the best redundant information that will maximize the perceived quality at the receiver**. This is the purpose of this section.

Consider a source with the flexibility to encode its samples at a rate $x \in [0, X_{max}]$ (we suppose X_{max} to be 64 Kbits/s). The quality of the voice call is characterized by a function $f : \Re^+ \times \Re^+ \rightarrow [0, 1]$ of (1) the reconstructed rate at the destination and (2) the mouth-to-ear delay.

The source transmits voice packets to a destination over an unreliable network characterized by:

- **a packet loss process:** Y_i which we suppose to be a Gilbert process where $Y_i \in \{0, 1\}$. If the ith packet is received at the destination, then $Y_i = 0$, otherwise, $Y_i = 1$. The parameters p and q of the Gilbert model are estimated on-line at the receiver using the maximum likelihood estimator.

- **a delay distribution.** We do not make any assumption about the distribution of delays in the network. Instead, we consider that the end-to-end delay d_{e2e}, not including FEC, is known by source. d_{e2e} actually represents the sum of the packetization delay and the playout delay (which is estimated at the destination as described in [83]). We further suppose that the playout delay is controlled in such a way that late losses in the playout buffer can be neglected. We will show in Chapter 5 how to take late losses into account.

The parameters p, q and d_{e2e} (which are all estimated on-line at the receiver) are sent back to the source via the application specific part (APP) of the RTCP receiver reports.

Let R_{max} be the rate available for the audio flow. R_{max} is the result of a TCP-Friendly rate control scheme (such as the one discussed in Chapter 7).

Then, consider that we use the *media specific* FEC scheme described in Section 2.2 to recover from packet losses. Let $K - 1$ denote the maximum number of redundant pieces of information sent along with the primary information. Thus, packet n carries information about at most (i.e. a subset of) packets $n - 1, \ldots, n - K + 1$. Therefore the total number of copies (encoded at different rates, including 0) of a given packet sent by the source is equal to K. The optimal value of K is a priori unknown and is supposed to be in $[1, K_{max}]$. In practice, the larger K, the longer the destination has to wait to receive the redundant information, and thus, the longer the end-to-end delay. Let τ_K represent the delay introduced by the use of FEC. τ_K is the delay between sending the first and the last copy of a given packet and can thus be written as follows: $\tau_K = (K - 1)\, T$, where T is the time interval between two consecutive audio packets (T is also packetization delay).

Further, define the random set Δ by $\Delta = \{i | Y_i = 0, i = 1, \ldots, K\}$, namely the set of copies of a given packet that are received at the destination.

Then, our problem can be stated as follows: Given that we can send at most K_{max} copies of each voice packet, find the optimal number of copies to send, and the optimal encoding rate for each copy, so as to maximize the quality of the voice call subject to the rate constraint. Mathematically, it gives the following optimization problem (which we call P1):

$$\underset{\substack{1 \le K \le K_{max} \\ x_i \ (i=1,\ldots,K)}}{\text{maximize}} \quad \sum_{\Delta \subseteq \{1,\ldots,K\}} P(\Delta) \max_{i \in \Delta} f(x_i, d_{e2e} + \tau_K)$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{K} x_i + R_{overhead} \leq R_{max} \\ x_i \geq 0, i = 1, \ldots, K \\ PLR_{FEC} \leq PLR_{max} \end{cases}$$

where x_i is the encoding rate of the copy placed in ith position in the stream ($i = 1$ corresponds to the primary information); $P(\Delta)$ is the probability to receive the set Δ; $R_{overhead}$ is the bandwidth overhead of the IP/UDP/RTP headers, PLR_{FEC} is the packet loss rate after reconstruction and PLR_{max} is the maximum acceptable value for PLR_{FEC}. The choice of a value for PLR_{max} mainly depends on the efficiency of the error resilience scheme used at the receiver. Typical values of PLR_{max} range between 5 and 10% (if losses are isolated) [68].

The objective function above represents the average quality measured at the destination. This model assumes that different copies of a given packet can not be combined to produce a better quality copy of the original packet. We rather assume that the receiver will send to its audio driver the *best* copy (i.e. leading to the highest quality) it has received of a given packet. The formulation of the objective function could be different if we used layered or multiple description coding [110] schemes for audio.

The problem above appears to be, in general, difficult to solve but a careful analysis of the objective function allowed us to derive solutions for K_{max} up to 5. The methodology remains the same for greater values of K_{max} but the main burden is that the number of terms in the sum grows exponentially with K. For further details about the resolution of the optimization problem, one can refer to Appendix C. In the following, we just describe the general characteristics of the results:

- $\boldsymbol{x_1 \geq x_i}, \forall i = 1, ..., K$: the primary information should always be encoded using the best quality encoding among those used to encode the different copies.
- **if** $\boldsymbol{p + q < 1}$, $\boldsymbol{x_1 \geq x_K \geq x_i}, \forall i = 1, \ldots, K$: it pays to use good quality coders to encode the end packets. Furthermore, for a given number of copies to send, the larger K, the better the audio quality at the destination, but also the larger the end-to-end delay. In this case, our algorithm allows to find the good tradeoff between quality of the reconstructed signal and delay.
- **if** $\boldsymbol{p + q > 1}$, $\boldsymbol{x_1 \geq x_2 \geq \ldots \geq x_K}, \forall i = 1, \ldots, K$: the redundant copies should closely follow the primary packet and the quality of the encodings should decrease as the copies are moved away from the primary packet.

4.4 Simulation Examples

We implemented and tested our *delay aware* error control scheme in the ns2 simulator [4]. This section presents a summary of our results.

This section investigates the behavior of our scheme under a wide range of loss conditions. We consider a simple scenario where n audio (TCP-Friendly) flows and $3n$ SACK TCP flows share a single bottleneck link. The packet loss rate experienced by the connections is varied artificially by changing the number of connections sharing the link. Figures 4.3(a) and (b) represent respectively the packet loss rate experienced by audio flows and the corresponding TCP-Friendly rate constraint as a function of the number of connections sharing the link. The bottleneck bandwidth is 5 Mbits/s and the one-way propagation delay (without queuing) is the same for all connections and is fixed at 80 ms. The graphs show the mean values averaged over the last 300 s of simulation and over all connections. Each point on this graph represents the results averaged over 5 simulation runs and the corresponding 99% confidence intervals. For our experiments, the parameter PLR_{max} was set to 5%.

As explained in Section 4.2, various values can be used for the parameters of the utility function $f = f_r\, f_d$. Figure 4.3(c) shows the mean utility obtained by the sources for three different parameter settings of f_r, for a fixed $f_d = f_{d2}$ (see Figure 4.1(b)). The results obtained with the piecewise linear f_r (MOS) and with the exponential of parameter $\alpha = 0.0001$ (exponential 1 on the figure) were extremely close to each other and the results obtained with the exponential of parameter $\alpha = 0.00003$ differed slightly in the way they spread the rate among the different copies. In all cases, the mouth-to-ear delay (including FEC) was the same. In the rest of the chapter, all the results shown were obtained using the exponential of parameter $\alpha = 0.0001$ for f_r. We now consider three different utility functions, which are defined as follows: Utility i designate $f_r\, f_{d_i}$, for $i \in \{1, 2, 3\}$ where the functions f_{d_i} are the ones depicted in Figure 4.1(b). Utility 1 corresponds to a source that does not attach much importance to the delay. Utility 3 characterizes a source for which delay is an important issue and Utility 2 represents a tradeoff between delay and audio distortion issues.

Figures 4.3 (d), (e) and (f) compare the mean utility obtained when using the delay aware FEC to the one obtained when using the FEC scheme proposed in [23] (which we will refer to as *classical FEC*) for the three utility functions of interest. For clarity, we do not show confidence intervals but we just mention that they were small ($< 2\%$). On these figures, each curve corresponds to a particular parameter setting (i.e. value of K) of the classical FEC scheme. As one can see, there is no optimal setting of the classical FEC which maximizes the utility in all loss conditions. The **delay aware scheme** (bold curve on the figures), in return, **always chooses the optimal amount of FEC (i.e. yielding the maximal utility) depending on network conditions**. Figures 4.3 (g), (h) and (i)

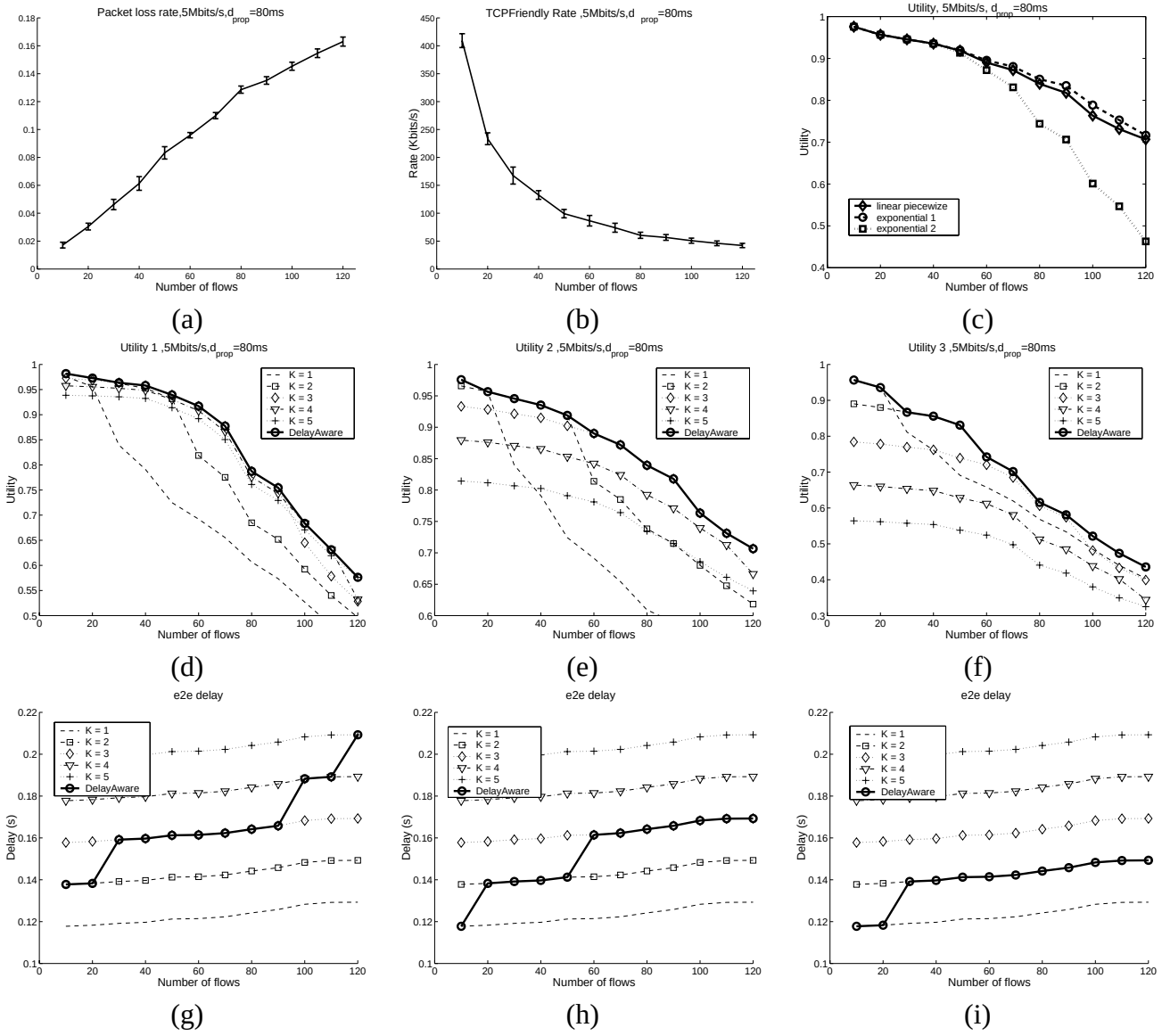


Figure 4.3: Packet loss rate (a). TCP-Friendly rate (b). Utility for different f_r (c). Utility of delay aware FEC vs classical FEC for (d) $f_d = f_{d1}$, (e) $f_d = f_{d2}$, (f) $f_d = f_{d3}$. e2e delay (including FEC) of delay aware FEC vs classical FEC for (g) $f_d = f_{d1}$, (h) $f_d = f_{d2}$, (i) $f_d = f_{d3}$.

show the corresponding mouth-to-ear delays (including packetization and FEC-induced delay) obtained with each scheme. One can observe that, in the delay aware scheme, the end-to-end delay is adapted, depending on loss conditions. The delay is kept small when the losses are moderate and is increased when the losses become more significant. Moreover, when comparing Figures 4.3 (g), (h) and (j), one can see that, depending on the importance the user attaches to delay, the amount of redundancy used is increased more or less rapidly as the loss rate increases. The same simulations were performed with smaller values of the propagation delay (see Figure 4.4 where the propagation delay is 30 ms) and showed that, in the case where network delay is very small (i.e., 30 to 50 ms),

the delay aware scheme leads to performances similar to the classical FEC with parameter $K = 3$ or $K = 4$, which could have been expected since, in this case, the delay induced by the FEC has no consequence on the perceived quality (because we stay below the critical threshold of 150 ms). In light of these results, we can conclude that the delay-aware scheme increases the utility by avoiding a source wasting delay using FEC when it is not necessary (and when it could even hurt the perceived quality).

4.5 Conclusions

In this chapter, we presented an adaptive error control scheme for audio which is *delay aware*, namely, which incorporates the impact on the end-to-end delay in the choice of the FEC. To this end, we took a utility function approach and we defined the quality as being a function of the reconstructed rate at destination *and* of the mouth-to-ear delay. We formalized our control problem as an optimization problem and solved it numerically and theoretically. We showed by simulation that the delay aware scheme does avoid that a source waste delay using FEC when it is not necessary.

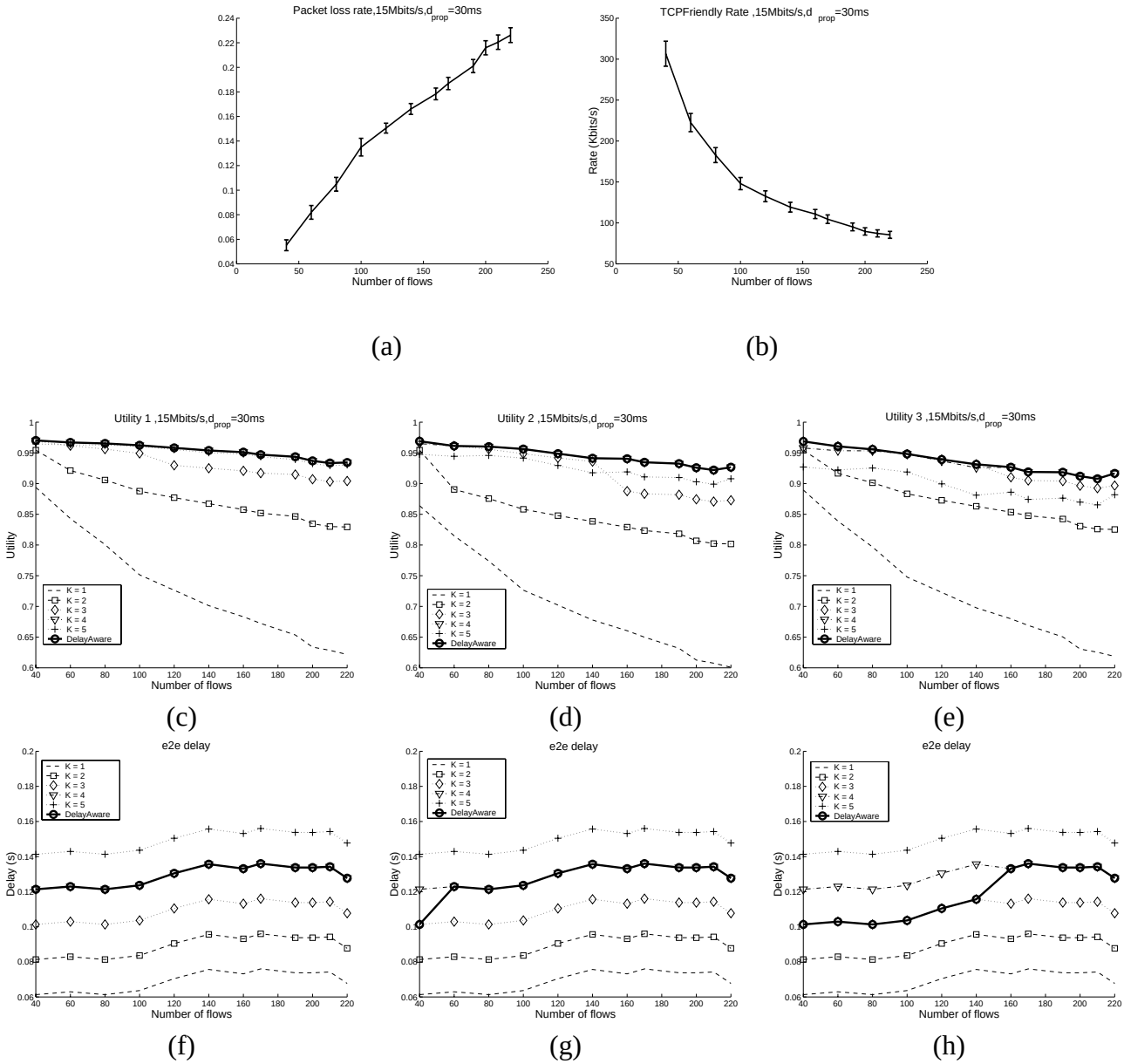


Figure 4.4: Packet loss rate (a). TCP-Friendly rate (b). Utility for different f_r (c). Utility of delay aware FEC vs classical FEC for (d) $f_d = f_{d1}$, (e) $f_d = f_{d2}$, (f) $f_d = f_{d3}$. e2e delay (including FEC) of delay aware FEC vs classical FEC for (g) $f_d = f_{d1}$, (h) $f_d = f_{d2}$, (i) $f_d = f_{d3}$.

Chapter 5

Adaptive Joint Playout Delay and FEC Adjustment for Internet Telephony

5.1 Introduction

The error control scheme presented in Chapter 4 is based on the philosophy that if the source went to the trouble of adding some redundancy (FEC) then the destination should wait for the redundant information to arrive. However, as mentioned in Chapter 1, this philosophy was challenged by Rosenberg *et al* [99], who tackle the problem of the delay introduced by FEC, in the case of non delay aware FEC, and propose a number of playout adaptation algorithms that provide a coupling between FEC and playout buffer adaptation. These coupled playout algorithms improve the delay performance for non delay aware FEC schemes, since they do not require a destination to wait for the FEC packets that are not needed. However, this is less clear in the case of delay aware FEC, since the source sends FEC packets only when necessary. This is one of the questions addressed in this chapter.

In this chapter, we consider the joint problem of FEC and playout adjustment at source and destination of an interactive audio source, while accounting for the impact of delay in the perceived utility. We present two solution methods of increasing complexity that solve the joint problem. The first one ("partial" method, called *N1*) adjusts the playout buffer at destination using the results in [99] (see Section 2.4); the FEC scheme (at the source) is aware of the playout delay computed at the destination and adjusts redundancy as a function of network characteristics and of playout delay, so as to maximize the perceived audio quality. Method N1 supposes that the destination uses a *play first* strategy (to be consistent with the use of a coupled playout algorithm). A second, more elaborate method ("complete" method, called *N2*), jointly chooses both the playout delay and the FEC scheme so as to maximize the perceived audio quality. In the latter method, the playout delay and the FEC scheme are parameters of the optimization problem, while in

the former, the playout delay is adapted separately and the best FEC scheme is chosen given this playout delay. In method N2, we consider the use of both play first and play best strategies at the destination.

We compare our adjustment schemes to two combinations of existing FEC and playout adjustment schemes. The first one (method *O1*) uses the first delay aware FEC adjustment method proposed in Chapter 4 (together with the classical playout adaptation, plus enough delay to wait for all redundant information to be received at destination). The second one (method *O2*) combines the non delay aware FEC scheme proposed in [22] with the coupled playout delay adjustments in [99].

We also address the following questions:

1. Are there significant quality improvements by using the complete joint method N2 ? More generally, how do the different methods compare ?

2. Is it worth using a joint FEC/playout adaptation scheme when delay aware FEC is used or, as mentioned earlier, can we rely on the source to send only those FEC packets that are necessary and wait for all FEC to be received ?

3. With the complete method N2, is it preferable to adopt a *play first* or a *play best* strategy ?

This Chapter is organized as follows:

- Section 5.2 describes our utility functions, which are adapted from the E-model. These utility functions, which are more mature than the one presented in Chapter 4, represent the perceived audio quality as a function of the audio reconstructed rate at the destination, the packet loss rate *and* the end-to-end delay.

- Section 5.3 presents our main results, namely, the derivation of our joint rate-error-playout delay control methods N1 and N2. In Section 5.3.2, we model the channel by a (1) packet loss process that we assume to be a Gilbert loss process and (2) a stationary delay process (not necessarily i.i.d.). We assume that the network delivers packets in sequence. We show that, given this assumption, our method needs only to know the marginal distribution of delays. We further suppose that (3) delay and losses are stochastically independent. Then we write our FEC and playout control problem as an optimization problem (Section 5.3.3), and solve it numerically.

- Results of simulations implemented in ns2 [4] are given in Section 5.4. They show that:

    1. Our complete scheme N2 performs better than the partial scheme N1 and the combinations of existing schemes O1 and O2, in the cases where end-to-end delay is important.

2. Contrary to some intuition, it is worth using a joint playout adjustment algorithm when a delay aware FEC scheme is used. Classical playout delay adjustments as used in *rat* and *freephone* lead to excessive playout delay.
3. When used with the joint adaptation scheme N2, the *play first* and *play best* strategies have similar performance.

5.2 Choice of Utility Functions Which Account for Delay

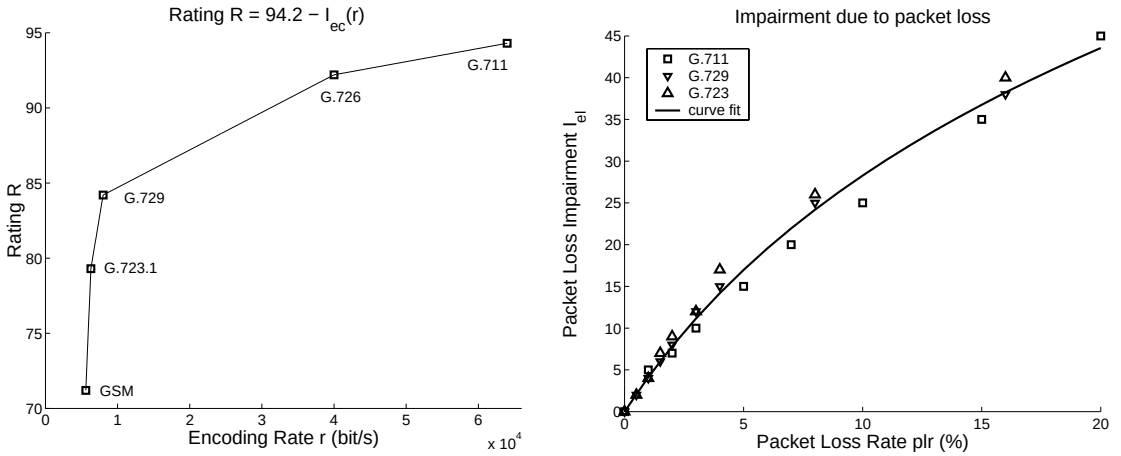
We use the E-model as a basis for defining our utility functions. However, we need to further elaborate on it, for two reasons. First, our optimization framework requires an analytic expression for the rating R as a function of the encoding rate r and the packet loss rate plr, which the E-model does not readily provide. Second, we wish to define different utility functions corresponding to a different modes of interactivity [77] and thus need to consider additional expressions for the delay impairment.

5.2.1 Influence of Distortion

The equipment impairment I_e captures the distortion of the original voice signal due to 1) the use of low bitrate codec and 2) packet losses (that occur in the network and in the playback buffer). In this chapter, we consider these two causes of distortion separately, our goal being to approximate I_e with an expression of the form:

$$I_e(r, plr) = I_{ec}(r) + I_{el}(plr) \tag{5.1}$$

where I_{ec} would represent the impairment due the audio encoding and I_{el} would be the impairment due packet losses. Since we know that the effect of packet loss is to increase the measured distortion, if the distortion increases in the same way for all the codecs as a function of the packet loss rate, then this approximation would be acceptable. Let us first consider the distortion introduced by the encoder (in absence of packet loss). Various values for the intrinsic impairment of a variety of codecs can be found in [14] and [39]. Figure 5.1(a) shows the rating R for voice calls using different bit rate codecs, for zero end-to-end delay and no packet loss. Since these values are obtained though subjective tests, we have at our disposal only a few samples of $I_{ec}(r)$ measured for discrete values of r (let us call them r_i, $i = 1, \ldots, n$) corresponding to the encoding rate of n existing codecs. Knowing the value of our utility function for discrete values of r is sufficient in the case where we wish to choose the best codec among existing ones in order to maximize the utility function of our audio source (see Section 5.3). However, it would be interesting to solve the problem also in the general case where the audio source would be able to encode audio samples with an infinite (large) number of rates ranging from 0 to



(a) (b)

Figure 5.1: Influence of distortion on quality. (a) Rating as a function of the encoding rate (for delay=0 and no loss), (b) Impairment due to loss as a function of the packet loss rate.

64Kbits/s. This could be made possible with some sophisticated coding techniques such as multiple description coding [110]. To obtain a continuous utility function with respect to the encoding rate, we use the values given for existing codecs and simply interpolate between these values to obtain a piece-wise linear utility function (as shown in Figure 5.1(a)).

Now let us consider the impact of packet loss on the measured distortion. In Appendix I of [14], values of I_e are given for several codec types as a function of packet loss rate and packet loss burstiness. In this chapter, we focus on results corresponding to random losses because 1) they are equivalent to the results obtained with bursty losses if the packet loss rate is small ($\leq 5\%$), which will typically be the case of the packet loss rate after reconstruction, and 2) because a precise description of the algorithm used for generating bursty packet losses is missing from [14]. Furthermore, we assume that all codecs implement some form of error concealment, which is a common practice today i.e. it is built-in in G.729A and G.723.1 and can be added on top of G.711.

In Figure 5.1(b), we plotted the values of $I_e - I_{ec}$ as a function of the packet loss rate plr for a variety of codecs. We can see that I_{el} increases with the packet loss rate. Moreover, we can see that the all codec follow more or less the same trajectory. Therefore, we can use a curve fitting technique to represent with a good approximation the impairment due to packet loss as a continuous function of the loss probability, independently of the codec in use. We use a logarithmic curve (see Figure 5.1(b)), as advised in [35] to approximate $I_{el}(plr)$ and obtain:

$$I_{el}(plr) = 34.3 \ln(1 + 12.8\, plr) \tag{5.2}$$

5.2.2 Influence of end-to-end Delay

As explained in [77], there is a dimension that is not captured by the E-model, that of different modes of conversation or interactivity requirements. Different types of conversation require different switching speed and thus have a different sensibilities to delay [74]. For example, a business call might require a higher level of interactivity than a social call. In [77], they conjecture that the fact that the E-model does not account for different modes of conversation may imply that the curves provided capture some kind of averaging of these different modes.

Besides, various studies [11, 10, 37] insist on the fact that, for natural hearing the end-to-end delay should be approximately 150 ms. While the benefit of delays lower than 100 ms can not really be appreciated by an average listener, delays above 150 ms are noticed by the users and become a hindrance to interactivity. In order to account for the great impact of the interactivity threshold (of 150 ms) on a conversation quality, we will also take up the utility function introduced in Section 4.2 on page 31.

We consider three different delay impairment functions, representing different sensibilities to the mouth-to-ear delay. The first delay impairment function $I_{d1}(d)$ considered (see Figure 5.2) is based on the utility function proposed in Section 4.2, and characterizes a user with strong interactivity requirements. The delay impairment from Section 4.2 was scaled to fit in the E-model and is thus expressed as follows:

$$I_{d1}(d) = \begin{cases} \gamma_1\, d & \text{if } d \leq 150 \\ b_1 \tanh(\beta(d - b_2)) + b_3 & \text{if } 150 < d < 300 \\ \delta + \gamma_2\, d & \text{if } d \geq 300 \end{cases}$$

where d is the mouth-to-ear delay expressed in ms, $\gamma_1 = \gamma_2 = 0.01$, $\beta = 0.02$, $\gamma = 50$ and b_1, b_2 and b_3 are constants selected to ensure the continuity of I_{d1}. Figure 5.2 shows the utility function as a function of mouth-to-ear delay (in absence of distortion impairment).

The second and the third delay impairment functions considered are based on the E-model: $I_{d2}(d) = I_d(d, 51)$ represents a user that is annoyed by delay because of echo, but without a clear threshold effect and $I_{d3}(d) = I_d(d, \infty)$ represents a user that attaches a small importance to delay (in an echo-free conversation). Figure 5.2 shows the influence of the mouth-to-ear delay on the utility function for these two delay impairment functions.

5.2.3 Our Utility Functions

In summary, we consider three different utility functions $f_i : \Re^+ \times \Re^+ \times \Re^+ \rightarrow [0, 100]$, $i = 1 \ldots 3$ that correspond to three different interactivity requirements:

$$f_i(r, d, plr) = 94.2 - I_{di}(d) - I_{ec}(r) - I_{el}(plr) \tag{5.3}$$

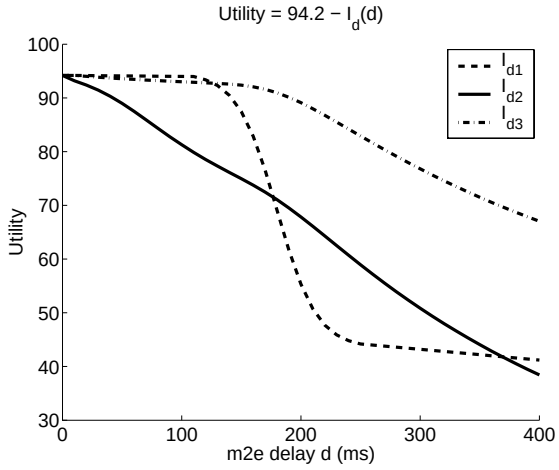


Figure 5.2: Utility as a function of the mouth-to-ear delay for different interactivity levels for $r = 64Kbits/s$ and no loss.

---

where d is the one-way mouth-to-ear delay, r is the codec bit rate and plr is the residual packet loss rate (after FEC is used), $I_{di}(d)$ ($i = 1 \ldots 3$), $I_{ec}(r)$ and $I_{el}(plr)$ are defined above.

5.3 Adaptive Joint Playout Delay and FEC Adjustment

In this section, we present two methods that solve the joint problem of FEC and playout adjustment at source and destination, while accounting for the impact of playout delay in the perceived utility.

5.3.1 Adaptation Mechanisms Overview

The "partial" method N1 adjusts the playout buffer at the receiver using the coupled algorithms proposed by Rosenberg [99]. The receiver periodically reports the value of the playout delay and the parameters characterizing the state of the connection[1] to the sender. Based on these information, the sender adjusts the redundancy so as to maximize the perceived utility. The optimal redundancy scheme is determined by solving the optimization problem introduced in Section 5.3.2.

If the source uses Signal Processing FEC (SP FEC) with method N1, a play first strategy has to be used (since it is the strategy used with coupled playout algorithms). If the source uses media independent FEC like Reed-Solomon FEC (RS FEC), since all the copies are identical, there is only one possible strategy (since the first copy is also the best).

[1]The loss event rate, the parameters of the loss process, the delay distribution and the round-trip time are reported once per round-trip time, but the playout delay needs to be reported only when is updated at the receiver.

The "partial" method accounts for the impact of playout delay on the probability to drop a FEC packet at the receiver due to late arrival. It uses a coupled playout delay adjustment (see Section 2.4) scheme that aims at reducing the delay spent waiting for redundant packets when it is not necessary, but it is not *delay aware* in the sense that it will not try to adaptively concede to a slightly higher loss rate when the mouth-to-ear delay gets close to the interactivity threshold.

The "complete" method N2, on the other hand, jointly chooses both the playout delay and the redundancy at the source so as to maximize the perceived utility. With this method, the receiver periodically reports the parameters characterizing the state of the connection[2] to the sender who determines the optimal redundancy scheme *and* the optimal value of the playout delay so as to maximize the perceived utility. The optimal playout delay is then sent to the receiver and its value is updated as soon as a new talkspurt begins.

With method N2, no constraint is imposed on the decoding strategy. We thus consider both the play first and play best strategy when signal processing FEC is used.

In summary, we study five possible combinations of FEC and playout adjustment method, FEC scheme and decoding strategy: (N1,SP FEC,play first), (N2,SP FEC, play first), (N2,SP FEC,play best), (N1, RS FEC) and (N2, RS FEC).

5.3.2 General Method

Consider the same voice source as in Section 4.3 with the flexibility to encode its samples at a rate x such that $x \in [0, X_{max}]$ (in the general case) or $x \in R$ with $R = \{r_i, i = 1, \ldots, n\}$ (if the source has a limited set of coders at her disposal).

The quality of the voice call is characterized by a utility function $f : \Re^+ \times \Re^+ \times \Re^+ \rightarrow [0, 100]$ as described in Equation (5.3). Our utility function has now three parameters: (1) the reconstructed rate at the destination and (2) the end-to-end delay and (3) the packet loss rate. It should be pointed out that the packet loss rate considered here is the residual packet loss rate after reconstruction of the lost packets using FEC.

The source transmits voice packets to a destination over an unreliable network about which we made some assumptions. For the control scheme, we modeled the network by:

- **a packet loss process:** Y_i which we suppose to be a Gilbert process where $Y_i \in \{0, 1\}$ (see Section 2.3). If the ith packet reaches the destination, then $Y_i = 0$, otherwise, $Y_i = 1$. The parameters p and q of the Gilbert model are estimated on-line at the receiver using the maximum likelihood estimator.

[2] Note that we do not *not* report the value of the playout delay since the latter has to be computed by the source.

- **a stationary delay process: D_i.** We suppose that the network delays of voice packets are identically distributed and follow a given distribution $F_D(d) = P(D_i \leq d)$ [3]. Moreover, we assume that the network delivers packets *in sequence*. The latter assumption can be expressed as follows:

$$Pr(D_{n+1} \leq D_n - T) = 0 \tag{5.4}$$

  which, by induction, is equivalent to:

$$Pr(D_{n+i} \leq D_n - i\,T) = 0 \tag{5.5}$$

  where D_n is the network delay of the nth packet and T is the time interval between two consecutive voice packets. We show that this hypothesis allows us to consider only the marginal distribution of delays and not the joint distributions.

- **independence loss-delay**: we assume that packet losses and network delays are mutually independent.

It is clear that this model is quite simple but we have to find a tradeoff between complexity and tractability since our control scheme has to be implemented in audio sources.

The no-reordering assumption allows to write:

Property 1 For a given packet, if the nth copy arrives after playout time, then all the following copies also arrive after playout time

$$Pr(D_{n+i} \leq D - iT | D_n > D) = 0 \quad , i \geq 0 \tag{5.6}$$

Property 2 For a given packet, if the nth copy arrives before playout time, then all the preceding copies also arrive before playout time (provided they are not lost in the network)

$$Pr(D_{n-i} > D + iT | D_n \leq D) = 0 \quad , i \geq 0 \tag{5.7}$$

Property 3 The joint probability $Pr(\{D_n \leq D\} \cap \{D_{n+1} > D - T\})$ is computed as follows:

[3] The parameters p, q and and the parameters characterizing the delay distribution (which are all estimated on-line at the receiver) are sent back to the source via the application specific part (APP) of the RTCP receiver reports.

$$
\begin{aligned}
& Pr(\{D_n \leq D\} \cap \{D_{n+1} > D - T\}) \\
&= Pr(D_n \leq D) \times Pr(D_{n+1} > D - T | D_n \leq D) \\
&= Pr(D_n \leq D) \times (1 - Pr(D_{n+1} \leq D - T | D_n \leq D)) \\
&= Pr(D_n \leq D) \times \\
&\quad (1 - \frac{Pr(D_n \leq D | D_{n+1} \leq D - T) Pr(D_{n+1} \leq D - T)}{Pr(D_n \leq D)}) \\
&= Pr(D_n \leq D) \times (1 - \frac{Pr(D_{n+1} \leq D - T)}{Pr(D_n \leq D)}) \\
&= Pr(D_n \leq D) - Pr(D_{n+1} \leq D - T) \qquad (5.8)
\end{aligned}
$$

The "no reordering" assumption allows to obtain all needed probabilities from the sole marginal distribution F_D. All information about the joint distribution is encompassed in the no-reordering assumption.

Let R_{max} be the rate available for the audio flow. R_{max} is the result of our TCP-Friendly rate control scheme (which is discussed in Chapter 7).

Let D be the playout delay for each talkspurt.

Then, our general problem can be stated as follows: Given that we can send at most K_{max} copies of each voice packet, find the optimal combination of coding scheme (optimal number of copies to send and optimal encoding rate for each copy) and playout delay D so as to maximize the quality of the voice call subject to the rate constraint.

This problem can be can be formulated as an optimization problem and the formulation depends on the combination of:

1. the FEC/playout adjustment method: (1) N1 or (2) N2.

2. the error recovery technique in use: (1) Signal Processing FEC or (2) Reed-Solomon error coding.

3. the decoding strategy: (1) play the first copy of a packet received correctly, provided it arrived on time or (2) play the best copy received on time.

5.3.3 Detailed Formulation of the Sub-Problems

Method N1 - SP FEC - play first

In this case, the optimization problem can be expressed as follows (P1):

$$\begin{array}{ll} \text{Given:} & p,\, q,\, R_{max},\, F_D(d) \text{ and } D \\ \text{maximize} & \sum_{i\in\Gamma} P_{play}(i)\, f(x_i, D+T, PLR_{FEC}) \\ \text{over all} & \Gamma,\, (x_i\,,\ i\in\Gamma) \\ \text{subject to} & \begin{cases} \sum_{i\in\Gamma} x_i + R_{overhead} \le R_{max} \\ x_i \ge r_0\,,\ i\in\Gamma \end{cases} \end{array}$$

where x_i is the encoding rate of the copy placed in ith position in the stream; T is the packetization delay[4]; $P_{play}(i)$ is the probability that the ith copy is sent to the audio driver; $R_{overhead}$ is the bandwidth overhead of the IP/UDP/RTP headers and PLR_{FEC} is the packet loss rate after reconstruction. Γ is the set of copies of a given packet that are sent by the source i.e., $\Gamma = \{i|$ copy placed in ith position in the stream was sent$\}$. A value $i = 1$ corresponds to the primary information, thus the minimal $\Gamma = \{1\}$ corresponds to a source that sends no redundant information. Without loss of generality, we assume that the set Γ is an ordered set, i.e. if $\Gamma(k)$ denotes the kth element in Γ, we can write $\Gamma(k) < \Gamma(k+1)$. $P_{play}(i)$ is the probability that the ith copy is the first copy correctly received and that it is on time. Let nc denote the number of elements in Γ, namely nc is the total number of copies of a given packet that are sent by the source.

Define K to be the position of the last copy of a given packet sent by the source: $K = \Gamma(nc)$. [5] The optimal value of K is a priori unknown and is supposed to be in $[1, K_{max}]$. In practice, the larger K, the longer the destination has to wait to receive all the redundant information.

$P_{play}(i)$ is a function of the playout delay D, the parameters of the Gilbert model p and q and of the redundancy scheme Γ and can be computed as follows:

$$\begin{aligned} P_{play}(i) &= Pr(\{\{Y_{n+i-1}=0\}\cap\{D_{n+i-1}\le D-(i-1)T\}\}\bigcap \\ &\quad \{\{\{Y_{n+k-1}=0\}\cap\{D_{n+k-1}>D_{n+i-1}+(i-k)T\}\} \\ &\quad \cup\{Y_{n+k-1}=1\}\forall k\in\Gamma, k<i\}) \\ &= Pr(\{\{Y_i=0\}\cap\{D_i\le D-(i-1)T\}\}\bigcap \\ &\quad \{\{Y_k=1\}\cup\{\{Y_k=0\}\cap\{D_k>D_i+(i-k)T\}\} \end{aligned}$$

[4] Recall that the quality is a function of the mouth-to-ear delay. Since playout delay is defined as the difference between playout time and generation time for all the packets in a talkspurt, the mouth-to-ear delay is therefore the sum of playout delay and packetization delay.

[5] As an example, if a scheme $(n, n-2)$ is used, $\Gamma = \{1, 3\}$, $nc = 2$ and $K = 3$.

$$
\begin{aligned}
& \forall k \in \Gamma, k < i\}) \\
& \text{(since the delay and packet loss processes are stationary)} \\
= \; & Pr(\{\{Y_i = 0\} \cap \{D_i \leq D - (i-1)T\}\} \bigcap \\
& \{\{Y_k = 1\} \forall k \in \Gamma, k < i\}) \\
& \text{(using the no-reordering hypothesis (eq.(5.5)))} \\
= \; & Pr(\{\{Y_i = 0\} \cap \{D_i \leq D - (i-1)T\}\}) \times \\
& Pr(\{\{Y_k = 1\} \forall k \in \Gamma, k < i\} | \{\{Y_i = 0\} \cap \\
& \{D_i \leq D - (i-1)T\}\}) \\
= \; & Pr(\{Y_i = 0\}) \times Pr(D_i \leq D - (i-1)T) \times \\
& Pr(\{\{Y_k = 1\} \forall k \in \Gamma, k < i\} | \{Y_i = 0\}) \\
& \text{(by the loss-delay independence)} \\
= \; & Pr(D_i \leq D - (i-1)T) \times \underbrace{Pr(\{\{Y_k = 1\} \forall k \in \Gamma, k < i\} \cap \{Y_i = 0\})}_{a_i} \\
= \; & F_D(D - (i-1)T) \qquad \times \qquad a_i
\end{aligned}
\tag{5.9}
$$

with

$$
a_i \;=\; \begin{cases} \pi_0 & \text{if } i = 1 \\ \pi_1 \, p_{10}^{(i-1)} & \text{if } i > 1 \text{ and } I_d(i) = 2 \\ \pi_1 \prod_{j=2}^{I_d(i)-1} p_{11}^{(\Gamma(j)-\Gamma(j-1))} \, p_{10}^{(i-\Gamma(I_d(i)-1))} & \text{if } i > 1 \text{ and } I_d(i) > 2 \end{cases}
$$

where $I_d(i)$ is the index of the ith copy in the ordered set Γ (e.g. if $\Gamma = \{1, 3\}$, $I_d(3) = 2$), and the probabilities $\{p_{ij}^{(n)}\}$, $i, j \in \{0, 1\}$ are computed using Equation (2.1). As expected, we can see that $P_{play}(i)$ is an increasing function of D.

The residual packet loss rate after reconstruction PLR_{FEC} is defined as the probability that none of the different copies can be played at the receiver and is given by:

$$
PLR_{FEC} = 1 - \sum_{i \in \Gamma} P_{play}(i) \tag{5.10}
$$

The objective function above represents the average quality measured at the destination. In the discrete case, the formulation remains the same but the second constraint is replaced by $x_i \in R, i \in \Gamma$.

Method N2 - SP FEC - play first

In this case, the objective function is the same as in (P1) but the input parameters have changed. The problem (P2) is thus expressed as follows:

$$\begin{array}{ll} \text{Given:} & p, q, R_{max}, F_D(d) \\ \text{maximize} & \sum_{i\in\Gamma} P_{play}(i)\, f(x_i, D+T, PLR_{FEC}) \\ \text{over all} & \Gamma, (x_i\,,\ i\in\Gamma), D \\ \text{subject to} & \begin{cases} \sum_{i\in\Gamma} x_i + R_{overhead} \le R_{max} \\ x_i \ge r_0\,,\ i\in\Gamma \end{cases} \end{array}$$

Method N2 - SP FEC - play best

The problem (P3) is expressed as follows:

$$\begin{array}{ll} \text{Given:} & p, q, R_{max}, F_D(d) \\ \text{maximize} & \sum_{\Delta\subseteq\Gamma\,,\,\Delta\neq\emptyset} P(\Delta)\, \max_{i\in\Delta} f(x_i, D+T, PLR_{FEC}) \\ \text{over all} & \Gamma, (x_i\,,\ i\in\Gamma), D \\ \text{subject to} & \begin{cases} \sum_{i\in\Gamma} x_i + R_{overhead} \le R_{max} \\ x_i \ge r_0, i\in\Gamma \end{cases} \end{array}$$

where x_i is the encoding rate of the copy placed in ith position in the stream, $R_{overhead}$ is the bandwidth overhead of the IP/UDP/RTP headers and PLR_{FEC} is the packet loss rate after reconstruction.

The random variable $\Delta = \{i|\{Y_i = 0\} \cap \{D_i \le D - (i-1)T\}\,,\ i\in\Gamma\}$ represents the set of copies of a given packet that are received at the destination before the playout time of this packet. Without loss of generality, we assume that the set Δ is an ordered set, i.e. if $\Delta(k)$ denotes the kth element in Δ, we can write $\Delta(k) < \Delta(k+1)$.

$P(\Delta)$ is the probability to receive *exactly* the set Δ before playout time. Formally, $P(\Delta)$ is expressed as follows:

$$\begin{aligned} P(\Delta) \;=\; & Pr(\{\{Y_i = 0\} \cap \{D_i \le D-(i-1)T\}, \forall i\in\Delta\} \bigcap \\ & \{\{Y_k = 1\} \cup \{\{Y_k = 0\} \cap \{D_k > D-(k-1)T\}\}, \\ & \forall k \in \Gamma\backslash\Delta\}) \end{aligned} \tag{5.11}$$

Now, let us define the events $\{A_i,\ i = 0\ldots K\}$ as follows:

$$\begin{cases} A_0 &= \{d_1 > D\} \\ A_i &= \{\{d_i \leq D-(i-1)T\} \cap \{d_{i+1} > D-iT\}\} \quad \text{for } i = 1 \ldots K-1 \\ A_K &= \{d_K \leq D-(K-1)T\} \end{cases}$$

From Properties 1 and 2, we can deduce that $P(\bigcup_{i=0}^{K} A_i) = 1$ and $A_i \cap A_j = \emptyset$ for $i \neq j$. Hence, from the Total Probability Theorem, $P(\Delta)$ can be computed as follows:

$$\begin{aligned} P(\Delta) &= \sum_{j=0}^{K} P(\Delta|A_j)P(A_j) \\ &= \sum_{j=K_\Delta}^{K} P(\Delta|A_j)P(A_j) \\ &\quad \text{(since } P(\Delta|A_j) = 0, \forall j < K_\Delta\text{)} \\ &= \sum_{j=K_\Delta}^{K-1} P(\Delta|A_j)(F_D(D-(j-1)T) - F_D(D-jT)) \\ &\quad + P(\Delta|A_K)F_D(D-(K-1)T) \\ &\quad \text{(from Property 3:} \\ &\qquad \begin{cases} P(A_i) = F_D(D-(i-1)T) - F_D(D-iT) \quad \text{for } i = 1 \ldots K-1 \\ P(A_K) = F_D(D-(K-1)T) \end{cases} \\ &\quad \text{)} \end{aligned}$$

where K_Δ is the position of the last copy sent by the source and received on time, i.e. $K_\Delta = \Delta(n_\Delta)$ where n_Δ is the number of elements in Δ and $P(\Delta|A_j)$ is given by:

$$\begin{aligned} P(\Delta|A_j) &= Pr(\{\{Y_i = 0\}, \forall i \in \Delta\} \bigcap \{\{Y_k = 1\}, \forall k \in \Gamma \backslash \Delta,\, k \leq j\}) \\ &= \begin{cases} \pi_{b_1} & \text{if } j = 1 \\ \pi_{b_1} \prod_{k=1}^{nc_j - 1} p_{b_k b_{k+1}}^{(\Gamma(k+1)-\Gamma(k))} & \text{if } j > 1 \end{cases} \end{aligned}$$

where nc_j is the number of elements in Γ that are inferior or equal to j, b_k is a binary value defined as $b_k = 1 - I(\Gamma(k) \in \Delta)$ where I is the indicator function and the probabilities $\{p_{ij}^{(n)}\}$, $i, j \in \{0, 1\}$ are computed using Equation (2.1).

PLR_{FEC} is the probability that none of the copies arrives on time, and is given by:

$$PLR_{FEC} = 1 - \sum_{\substack{\Delta \in \Gamma \\ \Delta \neq \emptyset}} P(\Delta) \tag{5.12}$$

Method N1 - RS FEC

The problem (P4) is expressed as follows:

$$\begin{array}{ll} \text{Given:} & p, q, R_{max}, F_D(d), D \\ \text{maximize} & f(x, D+T, PLR_{FEC}) \\ \text{over all} & (n,k), x \\ \text{subject to} & \begin{cases} x\dfrac{n}{k} + R_{overhead} \leq R_{max} \\ x \geq r_0 \\ k \leq n \end{cases} \end{array}$$

where x is the encoding rate of each copy, $R_{overhead}$ is the bandwidth overhead of the IP/UDP/RTP headers and PLR_{FEC} is the packet loss rate after reconstruction. PLR_{FEC} is a function of the parameters of the loss process p and q, of the parameters of the FEC scheme (n,k) and is a decreasing function of the playout delay D. Detailed computation of PLR_{FEC} as a function of these parameters is given in Appendix D. The final result is as follows:

$$PLR_{FEC} = \frac{1}{k}\sum_{i=1}^{k}\left\{\frac{q}{p+q}(1-F_D(D)) + \frac{p}{p+q}(1-P_{REC}(i))\right\}$$

where

$$\begin{aligned} P_{REC}(i) &= \sum_{g=k+1}^{n-1} P_{PAR}(k,g,i)\left\{F_D(D-(g-i)T) - F_D(D-(g+1-i)T)\right\} \\ &\quad + F_D(D-(n-i)T)P_{PAR}(k,n,i) \end{aligned}$$

$$P_{PAR}(k,g,i) = \sum_{l=1}^{g-k}\sum_{m=0}^{\min(l-1,i-1)} R(m+1,i)R(l-m,g-i+1)$$

with

$$R(m,n) = \begin{cases} P_R(n) & \text{for } m=1 \text{ and } n \geq 1 \\ \displaystyle\sum_{i=1}^{n-m+1} p_r(i)R(m-1,n-i) & \text{for } 2 \leq m \leq n \end{cases}$$

$$P_R(i) = \begin{cases} 1 & \text{if } i=1 \\ q(1-p)^{i-2} & \text{otherwise} \end{cases}$$

and

$$p_R(i) = \begin{cases} 1-q & \text{if } i = 1 \\ q(1-p)^{i-2}p & \text{otherwise} \end{cases}$$

Method N2 - RS FEC

The problem (P5) is the same as (P4) except that D is no longer an input but a parameter. (P5) is thus expressed as follows:

$$\begin{array}{ll} \text{Given:} & p, q, Rmax, F_D(d) \\ \text{maximize} & f(x, D+T, PLR_{FEC}) \\ \text{over all} & (n,k), x, D \\ \text{subject to} & \begin{cases} x\dfrac{n}{k} + R_{overhead} \leq R_{max} \\ x \geq r_0 \\ k \leq n \end{cases} \end{array}$$

5.3.4 Resolution

We implemented numerical methods to solve the different optimization problems formulated above. We used a combination of the algorithm proposed in [94] (which is well suited to maximize a sum of weighted utility functions like ours and gives exact solutions) and of an exhaustive search on discrete parameters (like Γ,n,k etc.). We have a running implementation of our audio source/destination in ns2. The method is optimal if the channel complies with the model described above.

5.4 Simulation Results

We investigated the behavior of the different FEC/playout adjustments schemes under a wide range of loss and delay conditions. The results presented here constitute a small sample of our simulation results but are representative of the behaviors we could observe in our simulations.

We consider a simple scenario where n audio sources share a bottleneck link with $3n$ SACK TCP and an ON/OFF source (CBR 500 Kbits/s when ON) with ON and OFF periods exponentially distributed, with average ON and OFF times of 3s. Packet loss rate is varied artificially by changing the number of TCP and audio connections sharing the link. Figure 5.3 (a) shows the TCP-Friendly rate constraint R_{max} as a function of the number of connections sharing the link. Figures 5.3 (b) and (c) represent respectively the parameters p and q characterizing the losses experienced by audio flows as a function of the number of connections sharing the link. The bottleneck bandwidth is 5 Mbits/s and the one-way propagation delay (without queuing) is 70 ms and the packetization delay

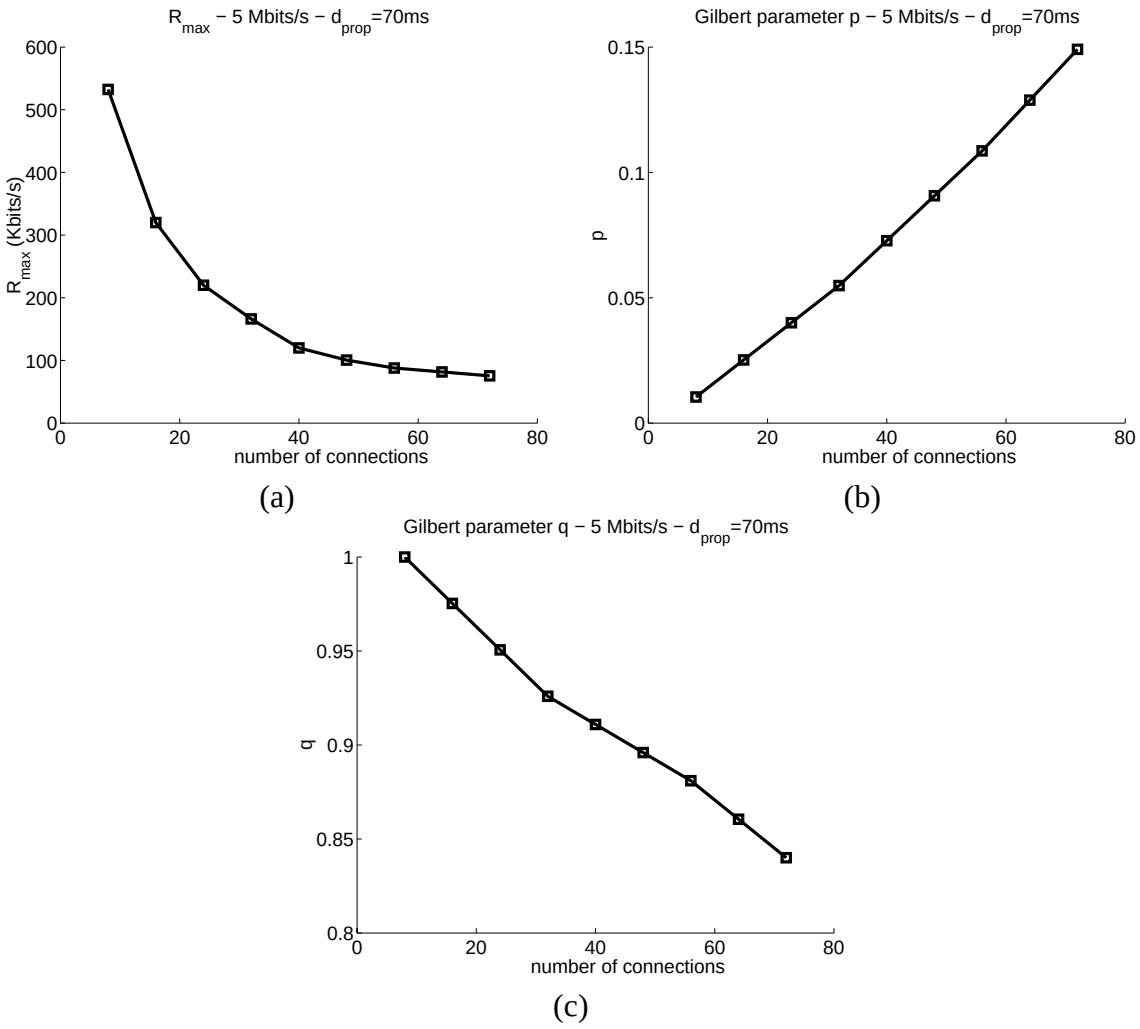


Figure 5.3: (a) TCP-Friendly rate constraint. (b) Gilbert parameter p. (c) Gilbert Parameter q as a function of the number of connections sharing the link.

---

is 20 ms. The graphs show the mean values averaged over the last 300 s of simulation and over all audio connections. Figures 5.4 to 5.7 shows the performance, in terms of (a) mouth-to-ear delay (playout delay + 20 ms of packetization delay), (b) utility measured at the receiver, (c) residual packet loss rate and (d) rate of the reconstructed audio stream at the receiver, for the different methods N1, N2 (Play First), O1 and O2 as a function of the number of connections sharing the link.

Figures 5.4, 5.5 and 5.6 correspond respectively to the results obtained with utility functions f_1 (high interactivity requirements), f_2 (delay is a concern but without threshold effect) and f_3 (delay is marginally important) with a Signal Processing FEC coding scheme. Figure 5.7 was obtained with utility function f_1 and a Reed-Solomon error coding. Algorithm 1 in [95] was used to adjust the playout delay with method O1 and its virtual version was used with N1 and O2. We tested other playout adjustment algorithms proposed [99] and show the results obtained in Appendix E.1. The conclusions presented

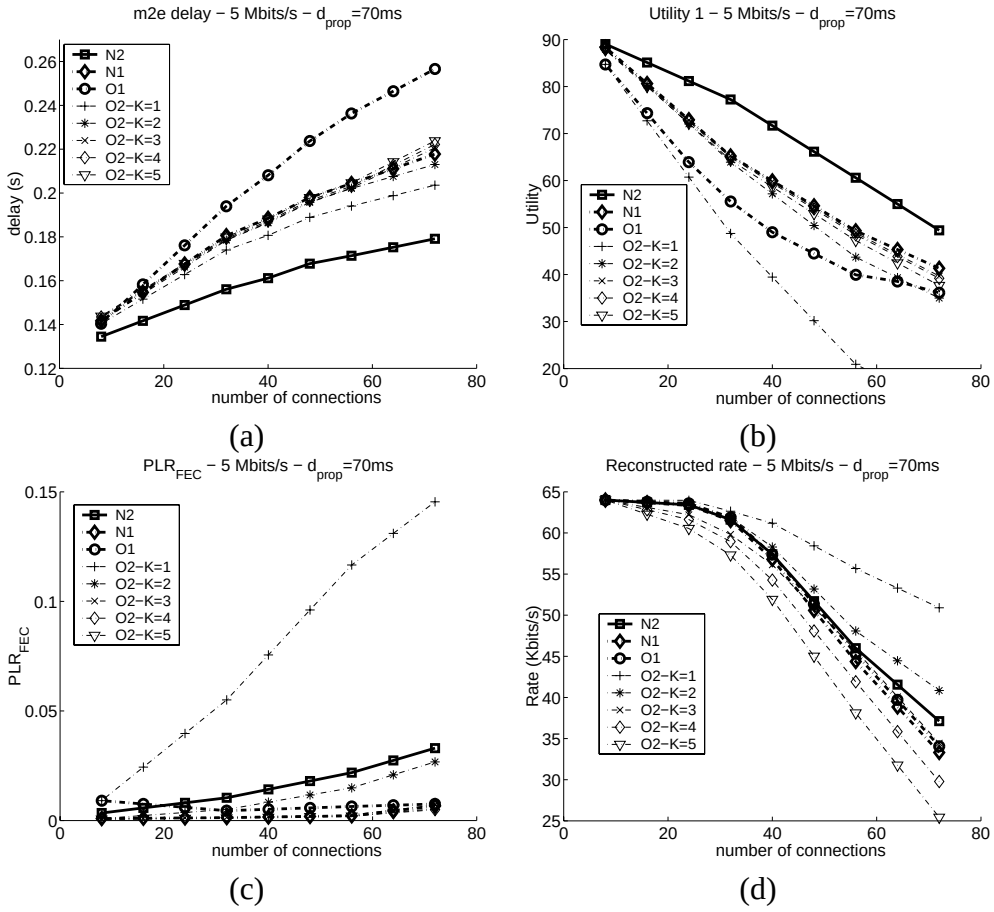


Figure 5.4: Performances of the different methods N1, N2 (Play First), O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utlity f_1.

---

in the sequel remain the same for other existing playout adjustment algorithms.

Figures 5.4 to 5.7 show that:

- Method N2 always gets the highest utility compared to other methods. The gain in utility is significant when utility f_1 is used (Fig. 5.4(b)) with a mouth-to-ear delay 20 to 30 ms smaller than with other methods (Fig. 5.4 (a)); the gain in utility is smaller with utilities f_2 (Fig. 5.5 (b)) and f_3 (Fig. 5.6 (b)) but still visible, with a mouth-to-ear delay 10 to 20 ms smaller than with other methods. This shows that N2 succeeds in trading delay for losses while keeping the residual packet loss rate acceptable. If delay is important (Fig. 5.4 (a)), N2 manages to keep the playout delay much lower than other methods (20 to 30 ms lower) at the price of a slightly higher residual packet loss rate (see Figure 5.4 (c)). On the other hand, if delay issues are less crucial for user, it accepts to increase its playout in order to reduce the residual packet loss rate (Fig. 5.6 (a) and (c)). This shows the benefits of jointly choosing the FEC scheme and the playout delay instead of updating the playout

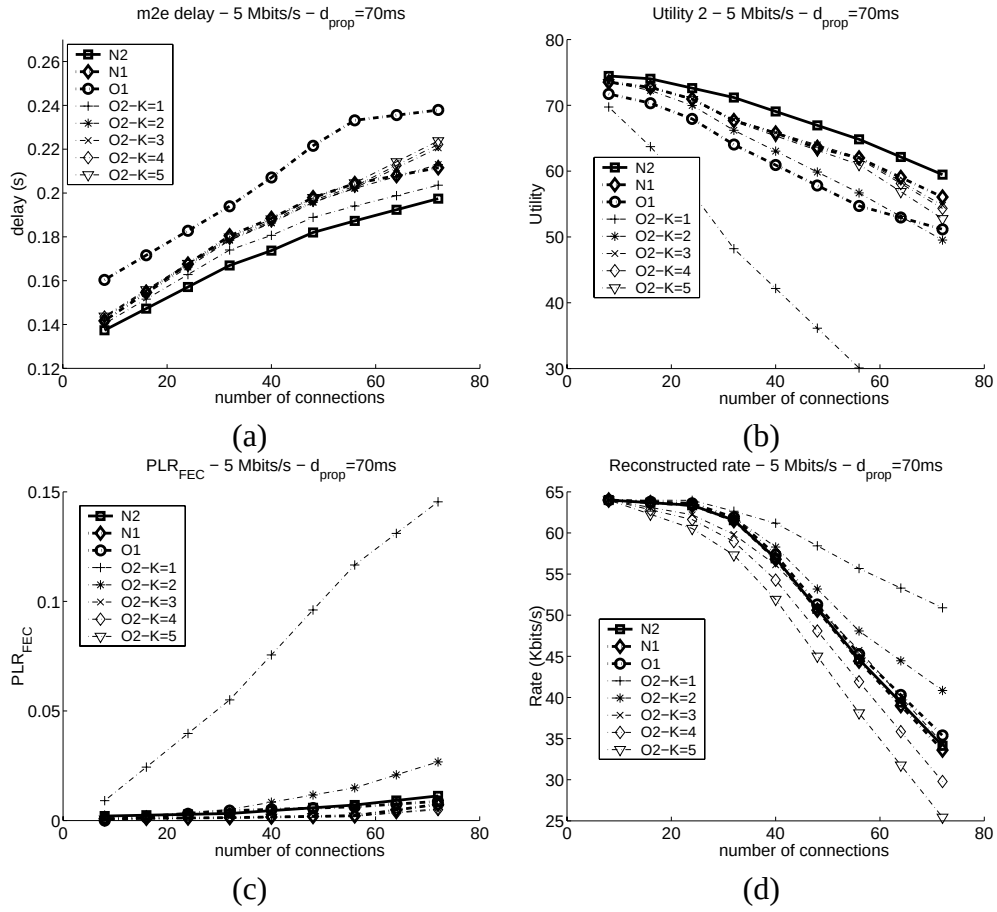


Figure 5.5: Performances of the different methods N1, N2 (Play First), O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utility f_2.

---

delay at the receiver and adjusting the FEC scheme at the sender, based on the current value of the playout delay.

- Method N1 gives results equivalent to the best combination of existing methods. When method O2 is used with Signal Processing FEC, the optimal parameter setting is $K = 3$ (K being the total number of copies sent) for a wide range of network conditions; with Reed-Solomon (n, k), a paradoxical result is that the setting $k = 1$ gives very good result. With this setting, Reed-Solomon is equivalent to a Signal Processing FEC with all the copies encoded with the same quality.

The same simulations were performed with smaller values of propagation delay and results corresponding to a propagation delay of 25 ms are presented in Appendix E.2. In the case where the propagation delay is very small, the method N2 still outperforms other methods, but the results obtained with method N2 are very close to the one obtained with

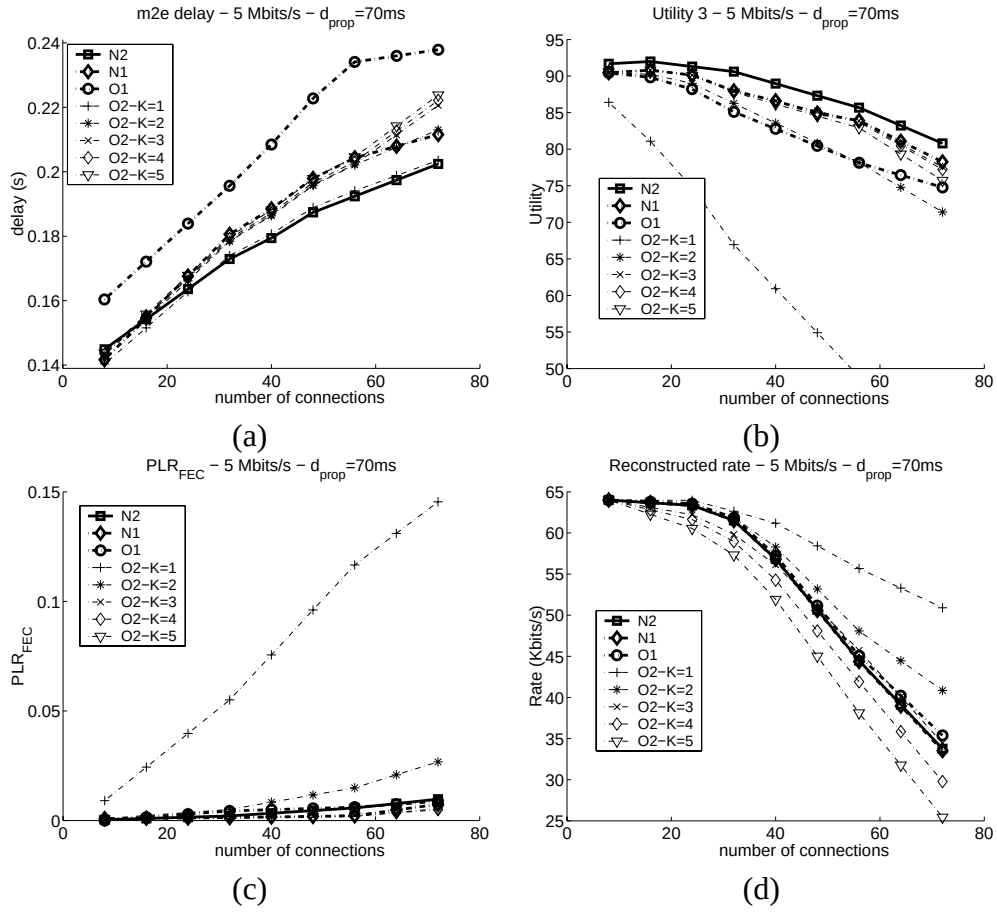


Figure 5.6: Performances of the different methods N1, N2 (Play First), O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utility f_3.

---

method N1. This is mainly due to the fact that when delay is very small, method N2 allows the delay to increase more rapidly, in order to reduce the packet loss rate after reconstruction (that has a greater impact on the quality than the delay, when the latter is very small) and thus, the delay obtained with N2 comes closer to the delay obtained with N1[6].

Figure 5.8 compares the performance of the play first and play best strategies when used with N2 and Signal Processing FEC. In all our simulations, we observed that the two strategies lead to similar results. Consequently, we recommend the use of the play first strategy, which is more simple.

Figure 5.9 (a) shows the average number of copies of a given packet that was sent by

[6]It is important to notice that, in the simulations presented in Appendix E.2, since the propagation delay is smaller, the losses experienced by the audio sources (see Fig. E.2) are higher than the ones presented in Fig. 5.3 (in order to keep a fair share of the bottleneck bandwidth - according to the TCP-Friendly principle).

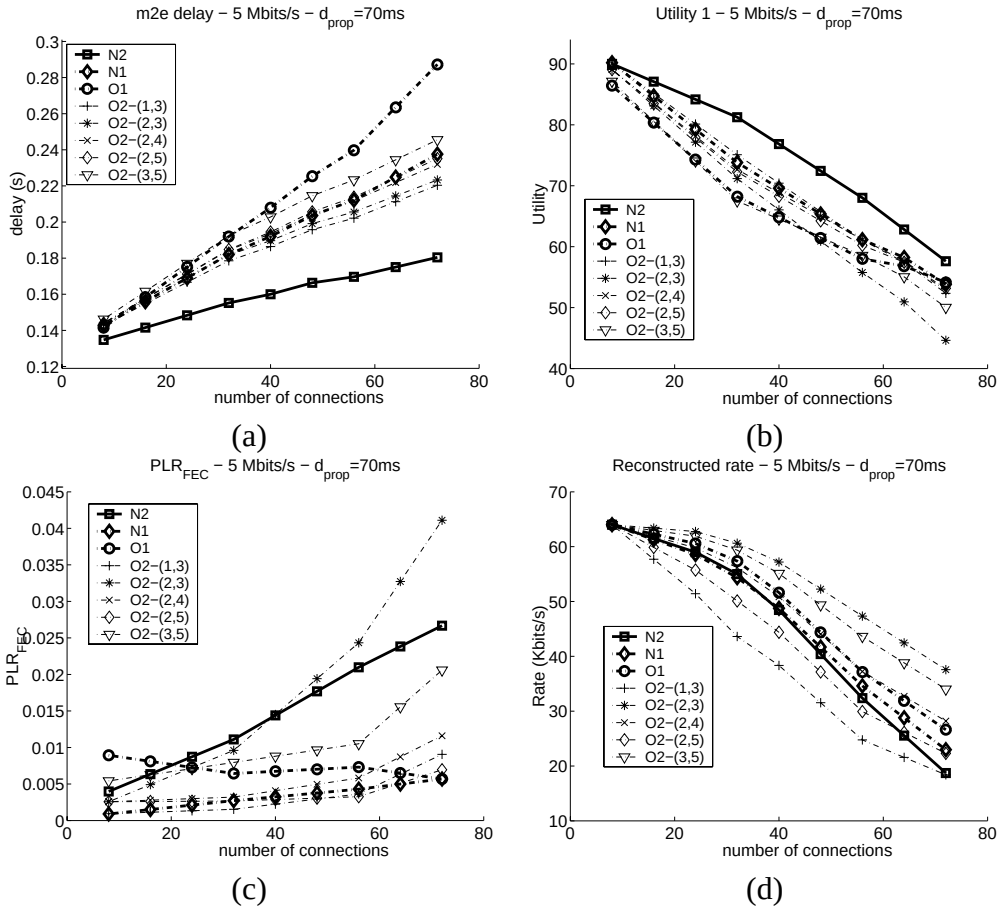


Figure 5.7: Performances of the different methods N1, N2, O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and reconstructed rate of audio for Reed-Solomon FEC with utility f_1.

---

the source when using the methods N2 and O1 with utility f_2. Figure 5.9 (b) shows the probability that the last copy of given packet is discarded at destination because it arrived too late. Figure 5.9 (b) shows that the optimal solution does not always wait for all the FEC packets to arrive (in 40 to 50% of the cases, method N2 does not wait for the last FEC packet to be received). The playout adjustment scheme used by O1 leads thus to excessive playout delay (as can be seen in Figures 5.4 to 5.6 (a)).

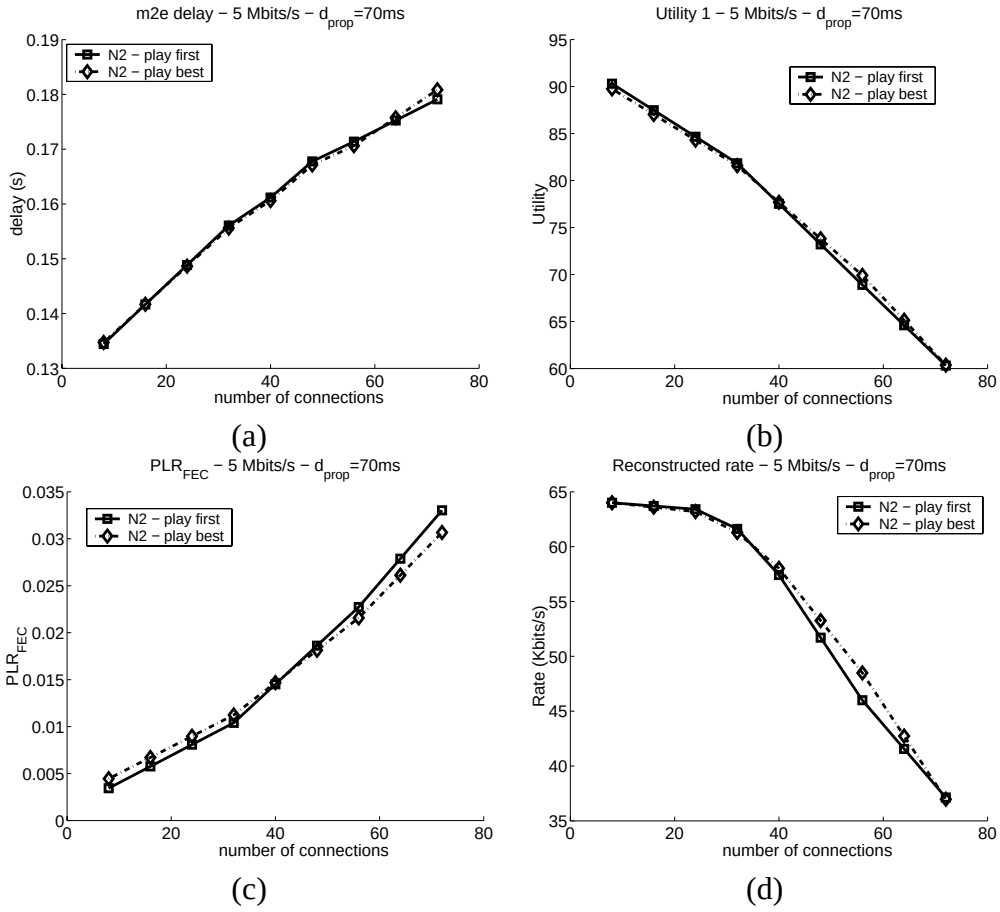


Figure 5.8: Comparison of play first and play best strategies: both strategies achieve equivalent performances.

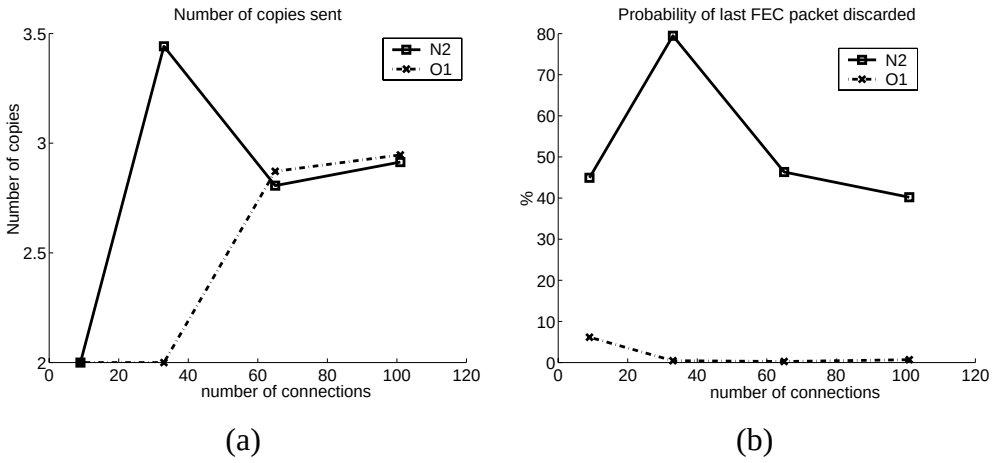


Figure 5.9: The optimal solution (N2) does not always wait for all FEC packets to arrive.

5.5 Conclusions

We designed a method for joint control of delay-aware FEC and playout for interactive audio applications over the Internet. We have shown that, in cases where delay matters (i.e. around a threshold effect), there is real benefit in using the joint method. We have also shown that the improvement brought by delay aware FEC cannot be obtained if the delay aware FEC control is simply piggybacked onto existing adaptive playout control methods; in contrast, it should be incorporated in a complete joint optimization of both FEC and playout.

Chapter 6

Audio on Non Elevated Services

6.1 Introduction

One of our motivations to develop a delay aware FEC and playout adjustment scheme is the emergence of a number of proposals for Internet differentiated services which propose a tradeoff differentiation between delay and losses[1] (or throughput) [50, 65]. In the case of audio sources, this tradeoff is not straightforward. Indeed, in today's Internet, where explicit congestion notification (ECN) [52] is not yet widespread, a source that chooses a low delay class will be likely to experience more losses and hence may be forced to compensate this additional loss by more FEC, and thus more end-to-end delay. This raises the following question: **does an audio source benefit from using a non-elevated differentiated service?**

In this chapter, we provide the following contributions: 1) we propose an adaptive service choosing method to optimize the overall quality of interactive audio conversations over *non-elevated* services and 2) we answer the question about the interest of *non-elevated* services for Voice over IP.

To evaluate the benefit for audio sources to use a *non-elevated* differentiated service, we perform simulations (in the $ns2$ network simulator) where audio sources with the delay-aware FEC and playout adjustment scheme[2] make use of the Alternative Best Effort (ABE) service [66]. We show in Section 6.2 that the use of such a service can lead to quality improvement but that the choice of service depends on 1) network conditions and 2) the importance that users attach to delay. In the light of these results, an adaptive algorithm that allows the source to choose in real time the service leading to the highest quality is proposed in Section 6.3. Finally, we evaluate in Section 6.4 the benefit brought by a *non-elevated* service compared to a single class best effort service.

[1] We refer to this kind of differentiated services as *non-elevated services*.

[2] Method N2 from Chapter 5

6.2 Throughput versus delay: a difficult tradeoff

Among the proposals that emerged for non-elevated services, we chose to look into the proposal for ABE service [66] because it was the only one whose $ns2$ implementation was at our disposal. With ABE, every best-effort packet is marked either *green* or *blue*. Green packets are guaranteed a low bounded delay at every router, but, in return, are more likely to be dropped during periods of congestion than blue ones. As a consequence, they will typically receive less throughput than blue ones (according to the TCP-Friendly rate control). In this framework, the question above can be reformulated as follows: **is it worth being green?**

In the sequel, we make use of our delay aware FEC and playout adjustment scheme to provide an answer to this question.

6.2.1 Simulation description

We consider a simple simulation topology consisting of the well-known dumbbell topology. For each simulation, n blue flows and n green flows compete along a bottleneck link of speed 5Mbits/s. The access links to the bottleneck routers are all provisioned with 10 Mbits/s. The one-way propagation delay (without queueing) d_{prop} is the same for all connections, the maximum time green packets spend in the system is d_g and the maximum waiting time in the virtual queue is d_v (it is equivalent to the maximum waiting time for blue packets).

The adaptive audio applications represent 25% of the connections of each type (green and blue), the other 75% are Sack TCP connections. Network load conditions are varied artificially by varying the number of connections sharing the link (which is equal to $2n$). The graphs show the mean values averaged over the last 300 seconds of simulation and over all connections. Each point on the graphs represents the results averaged over 5 simulation runs and the corresponding 99% confidence intervals.

6.2.2 Small propagation delay

Let us first consider the case where propagation delay is relatively small and equal to 30 ms (with a packetization delay of 20 ms, the minimum delay for green is thus 50 ms). d_g is 40 ms and d_v is 100 ms[3].

Figure 6.1 shows network characteristics for both classes of traffic as a function of the total number of connections sharing the link. Figures 6.1(a) and 6.1(b) represent re-

[3]Note that setting the parameters for the ABE service is not easy because the relative queue sizes (for green and blue) directly influence the tradeoff delay versus losses. We used a maximum queueing delay of 100 ms because we wanted to study a case where queueing delay can be important. Indeed, the introduction of the ABE service would not make sense in a network where queueing delays are small anyway. We depict such a case in Appendix F where $d_v = 60\,ms$ and $d_g = 25\,ms$.

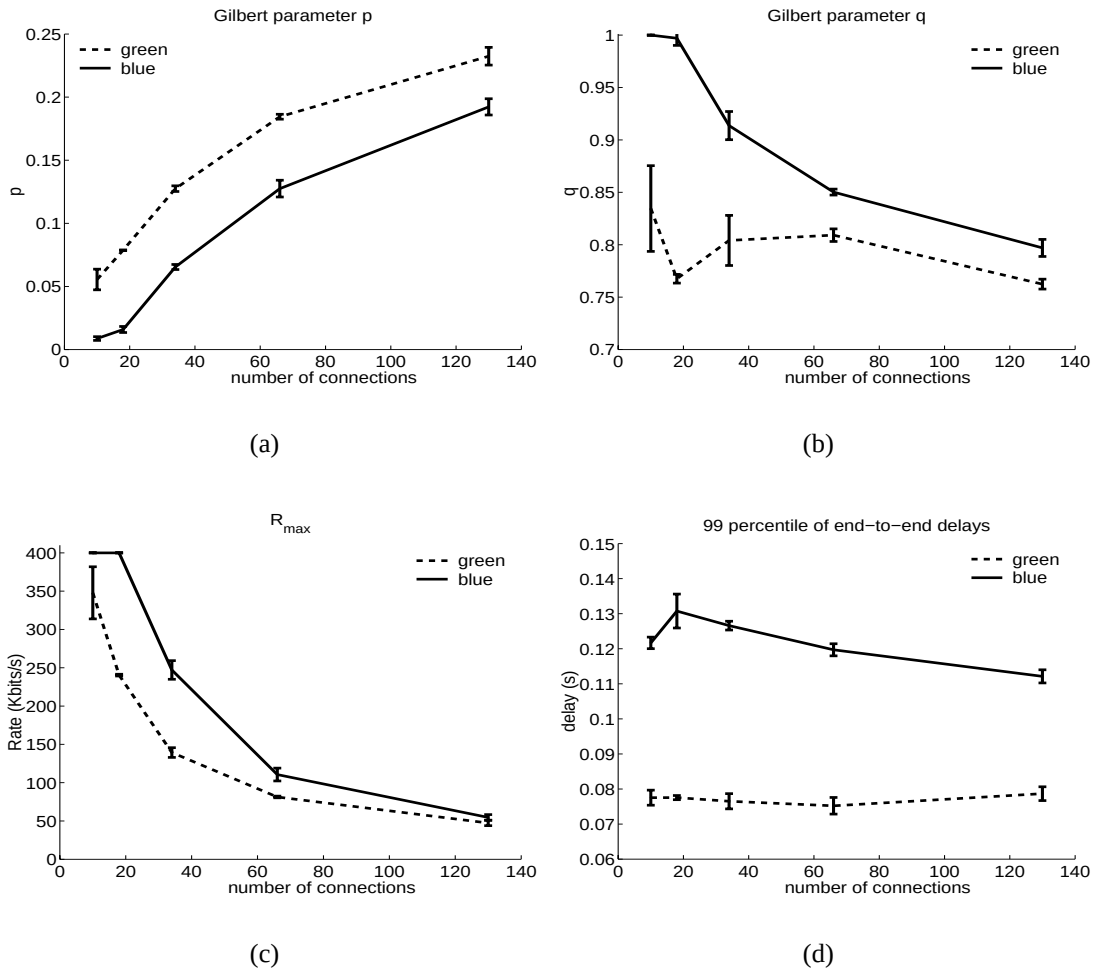


Figure 6.1: Network characteristics for *green* and *blue* traffic. $d_{prop} = 30\,ms$, $d_g = 40\,ms$.

spectively the parameters p and q characterizing the losses experienced by audio flows, Figure 6.1(c) gives the TCP-Friendly rate constraint R_{max} and Figure 6.1(d) depicts the 99% percentile of the end-to-end delay (including network and packetization delay) experienced by green and blue packets. The tradeoff between delay and loss (or equivalently throughput) appears clearly on these four figures.

Figure 6.2 shows the performance of delay-aware audio flows over the *blue* and *green* services. These graphs were obtained with the utility function f_1[4]. Figures 6.2(a), 6.2(c) and 6.2(d) respectively depict the mouth-to-ear delay, the residual packet loss rate and the rate of the reconstructed audio stream at the receiver. The resulting average utility, measured at the receiver, is given in Figure 6.2(b). Even if, in terms of utility, the difference between the blue and green services is minor, one can observe that the sources using the green service receive a mouth-to-ear delay that is 15 ms smaller than the sources using the blue service while the packet loss rate after reconstruction is nearly the same over both

[4] f_1, f_2 and f_3 are defined in Section 5.2 on page 41.

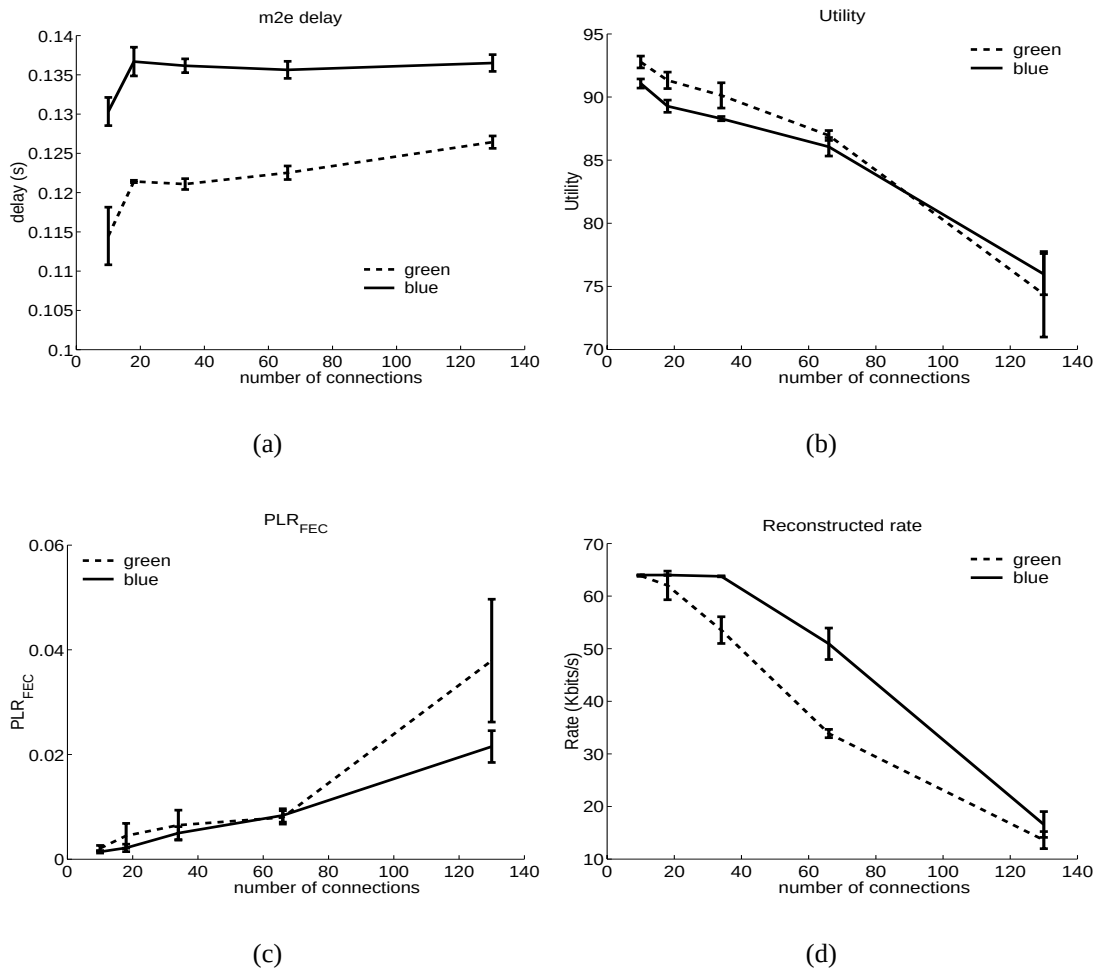


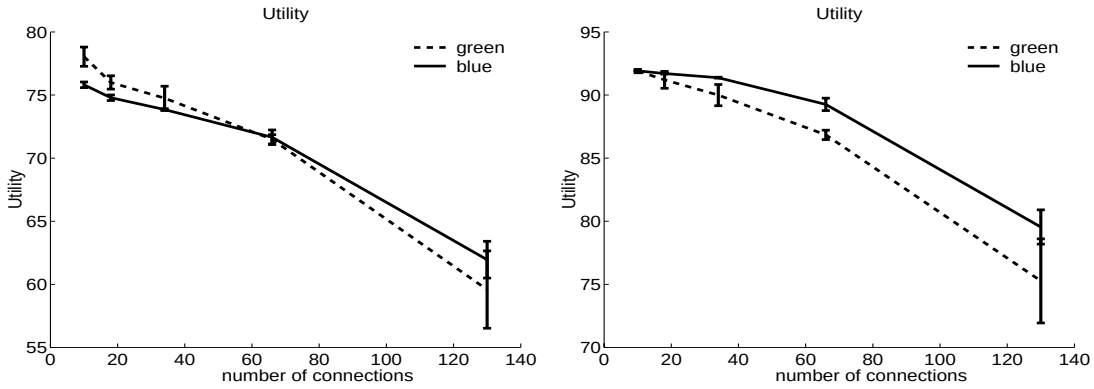
Figure 6.2: Performance of delay-aware audio over *blue* and *green* services with utility f_1. $d_{prop} = 30\,ms$, $d_g = 40\,ms$.

services. Similar results were obtained with utility functions f_2 and f_3 as can be seen on Figure 6.3 where the average utility obtained with both of these functions is depicted.

From these graphs, one can see that, when the delays are small, the difference between the utilities of blue and green flows is minor but there is a small benefit to use the low delay service when 1) network load is low and 2) delay is important to the user (for utility functions f_1 and f_2). In particular, under these network conditions, the low delay class allows to slightly reduce the mouth-to-ear delay without any penalty in terms of packet loss rate after reconstruction compared to the default (blue) class.

6.2.3 Large propagation delay

Consider now larger flows with a propagation delay of 80 ms. d_g and d_v remain the same and are respectively equal to 40 ms and 100 ms. Figure 6.4 shows network characteristics measured by audio flows over both classes of traffic as a function of the total number



(a) (b)

Figure 6.3: Performance of delay-aware audio over *blue* and *green* services : average utility obtained with utility function (a) f_2 and (b) f_3. $d_{prop} = 30\,ms$, $d_g = 40\,ms$.

---

of connections sharing the link. Figure 6.5 shows the performance of delay-aware audio flows over the *blue* and *green* services with the utility function f_1. In this case, one can notice that the difference between the utilities received by green and blue flows is much larger and that the green service brings a significant benefit to users that are sensitive to delay (represented by utility function f_1). In particular, Figures 6.5(a) and 6.5(c) show that the low delay class allows to maintain a mouth-to-ear delay 30 to 40 ms lower than the blue class, at the price of a slightly higher packet loss rate after reconstruction (around 1% to 1.5% higher). In this case, the low delay class provides a real benefit because it allows to keep the mouth-to-ear delay close to 150 ms, while it would not have been possible with a default (blue) service.

Other utility values obtained in the same conditions with functions f_2 and f_3 are depicted in Figure 6.6. Figure 6.6(a) shows that a user who attaches some importance to the end-to-end delay (represented by function f_2) will benefit from being green, up to a certain level of network congestion where the available rate and packet losses become an impediment and the source could choose to switch to blue in order to increase its throughput and reduce its loss rate. On the other hand, a user with a pronounced threshold effect (Figure 6.5) will probably always choose the green service when the delay is close to the threshold. Finally, a user who does not care about delay (see Figure 6.6(b)), will probably choose the blue service in all cases in order to get the smallest residual packet loss rate as possible.

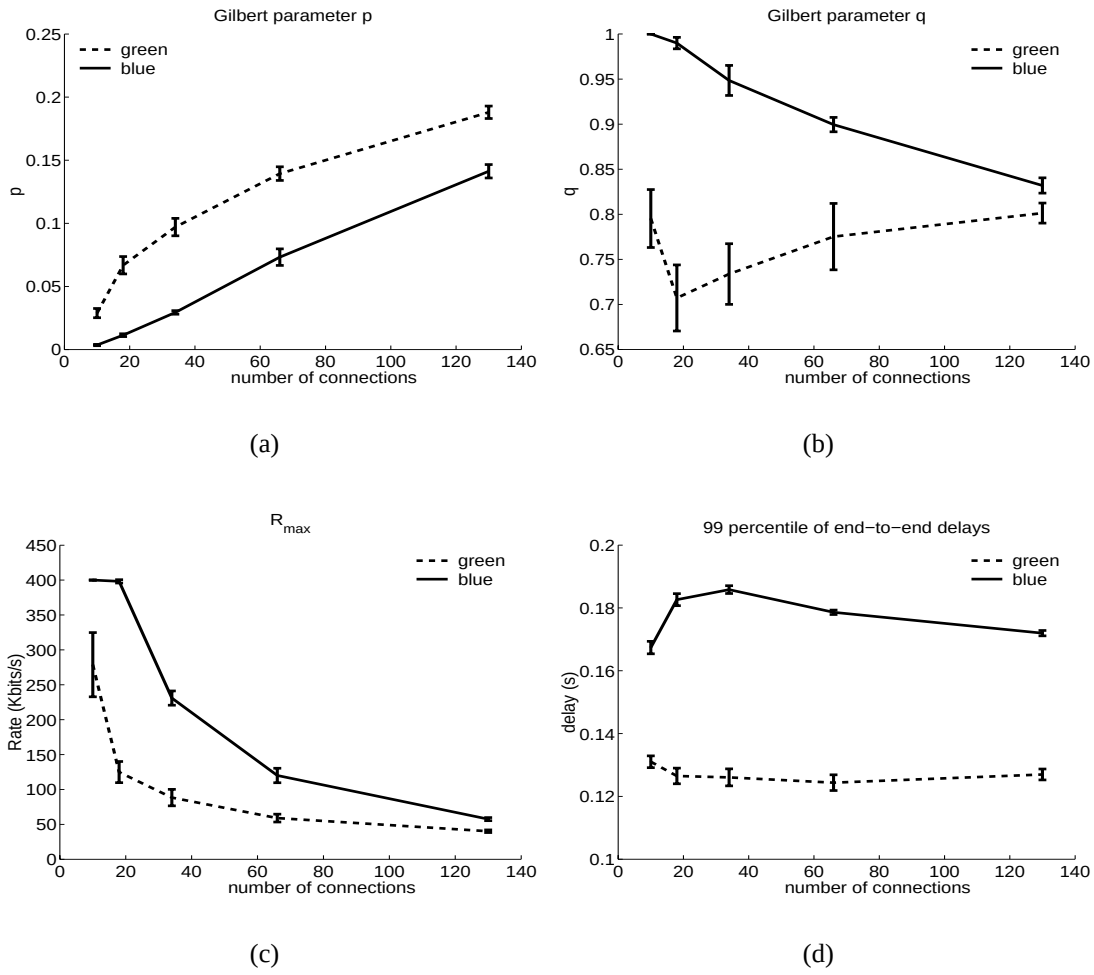


Figure 6.4: Network characteristics for *green* and *blue* traffic. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

6.2.4 Summary

From these results, we conclude that (time sensitive) audio applications benefit, when using ABE, from being green up to a certain network load that depends on (1) the importance that users attach to delay and (2) network topology (e.g., propagation delay , queueing delay, etc.). It is up to the source to determine when it is better to switch from one color to another. This result tends to indicate that there is a need for a color choosing algorithm for audio sources using ABE. In Section 6.3, we propose an adaptive "Color Choosing" algorithm that allows the source to pick the optimal color depending on network load conditions.

6.3 ABE with adaptive Color Choosing algorithm

As we saw in the previous section, an audio source will benefit from using the low delay class under certain network load conditions. In return, in some cases, it will be preferable

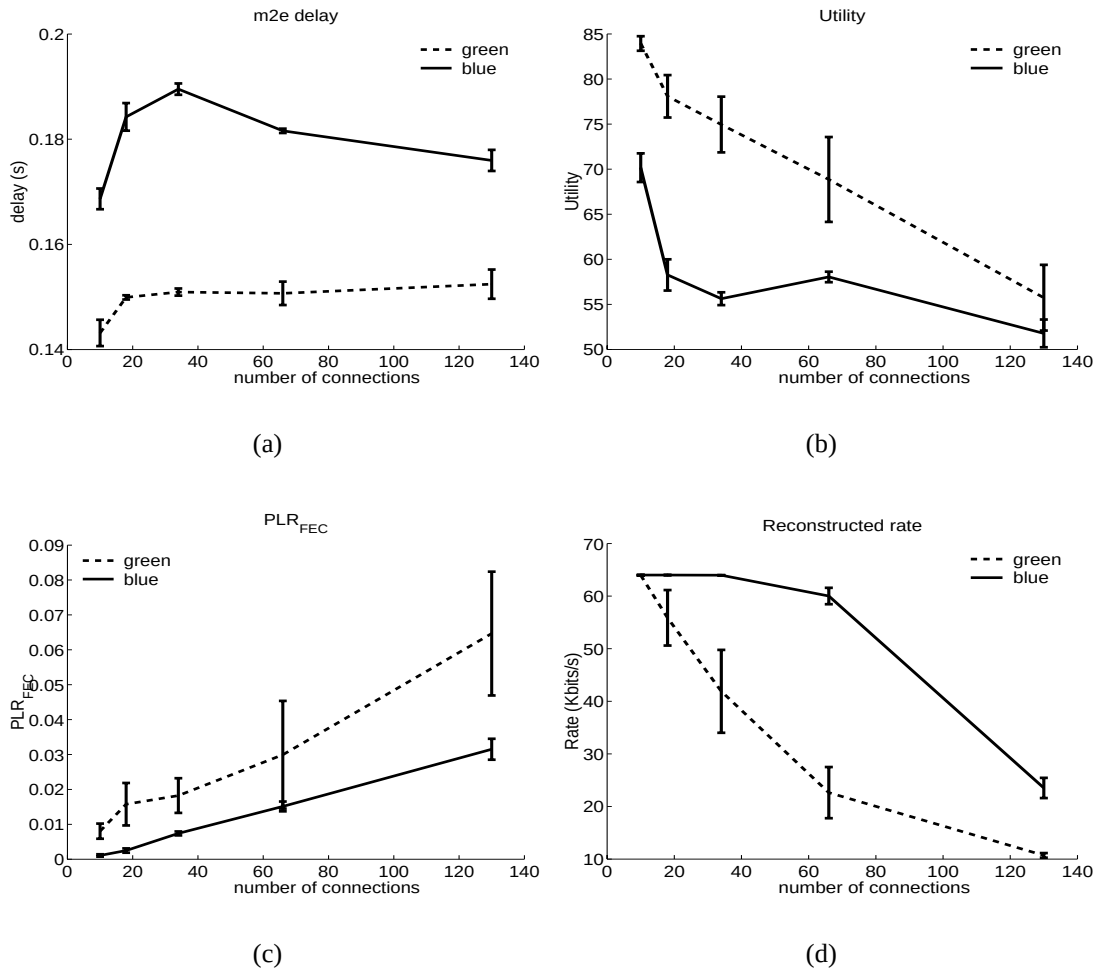


Figure 6.5: Performance of delay-aware audio over *blue* and *green* services with utility f_1. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

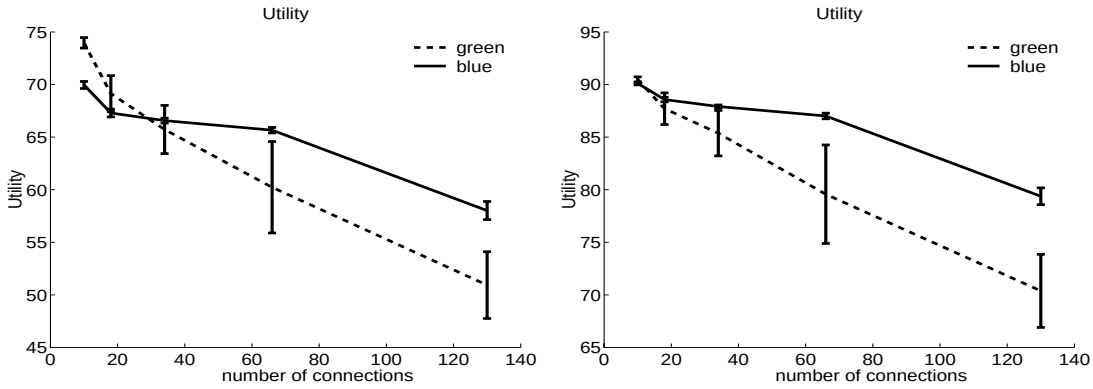
---

that the source use the low loss class of service. This indicates that, in order to make an optimal use of the non-elevated service, the source will need to adapt its choice of service to the network conditions. Therefore, we start, in this section, by proposing an adaptive color (service) choosing algorithm that will allow the audio source to choose the best-suited service depending on network load conditions.

Finally, we will address the following question: if we combine the adaptive error control scheme to the adaptive service choosing algorithm (which is supposed to lead to an optimal use of the non-elevated service) does the non-elevated bring a benefit compared to the single class best effort service?

6.3.1 An adaptive color choosing algorithm

The purpose of the Adaptive Color Choosing algorithm (ACC) is to determine in real time the optimal color of the packets sent by the source. Pseudocode for the ACC algorithm is



(a) (b)

Figure 6.6: Performance of delay-aware audio over *blue* and *green* services : average utility obtained with utility function (a) f_2 and (b) f_3. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

given in Table 6.1 and a detailed description of the algorithm is provided in the sequel.

Let *cur_color* be the current color of the connection. At the start of a connection, *cur_color* can be picked randomly. In order to determine whether the current color is the one leading to the highest utility (or quality), the source needs to estimate the utility it would get if it were using the other color. To this end, the source periodically sends a few packets, that we call "probe packets", using the other color (i.e. the other class of service). Let *probe_color* be the color of the probe packets. The probe packets are sent in small bursts of N consecutive packets and represent a small fraction of the traffic sent by the source.In order to avoid playout disruptions (since probe packets can experience a significantly different delay in the network), the probe packets are sent during silence periods.

At the receiving end, we maintain a "history" of the packets received for both colors in parallel. For each color, a history consists of a record of the last nb packets: in particular, we store the status (i.e. received or lost) and the one-way delay of the packets. Packet losses are detected using sequence number gaps. Both colors use different sequence numbering. In order to avoid undesirable oscillations of the playout delay, the latter is adjusted based only on the packet delays of the current color: *cur_color*.

When it is time to generate a report (i.e. every round-trip time on average), the receiver estimates the values of p, q and ler for both colors based on their respective packet history. p and q are estimated using unbiased estimators as in [112], and ler is computed using the average loss interval method introduced in Chapter 7.

Then, the receiver sends a report including the values of p, q, ler and the parameters [5]

[5]If the delays follow a given distribution, it may be sufficient to send only the parameters characterizing

of the delay distribution for each color back to the sender.

Upon receipt of the nth report, the sender determines whether it is worth changing the current color as follows:

- Based on the parameters included in the report, it computes the new utility associated to each color as explained in Chapter 5 (Section 5.3). The sender maintains an EWMA of this utility in order to filter out small oscillation that could lead to instability of the algorithm.
- Let *ut*[*cur_color*] and *ut*[*probe_color*] be the utilities associated to the current and the probe color respectively. Based on these values, the sender has to determine whether it is worth changing the current color. This is the purpose of the "Color Switching Test".

  Defining the worthiness of a change is a key issue because it directly influences the stability of the algorithm. There is a clear tradeoff between responsiveness to changes in utility and the avoidance of oscillations or unnecessary color switches. Therefore, we designed the "Colors Switching Test", while trying to meet the following requirements:

  1. do not react to sudden brief changes in utility.
  2. do not change color if the benefit is negligible. This requirement is meant to avoid oscillations between two colors when they are more or less equivalent.
  3. avoid changing color too frequently. Since each color change involves a readjustment of the playout buffer (that is always adjusted on the *cur_color* packets), frequent color switches could be noticed by the user and lead to a decrease in perceived quality.

  Based on these requirements, the "Color Switching Test" was defined as follows. The source will decide that it is worth switching to the other service (i.e. switching *cur_color* and *probe_color*) if the following condition:

$$ut[cur\_color] + THRESH < ut[probe\_color] - c(t) \tag{6.1}$$

this distribution. If, on the other hand, the delay distribution cannot be characterized by a known function, a histogram of the delays should be sent to the source. This is what we implemented in $ns2$ but in a real implementation, where the size of the RTCP reports can be an issue, the utility could be computed at the receiver. In that case, the maximum sending rate should either be computed at the receiver and sent back to the sender, or be computed by the source and sent to the receiver. These considerations do not change anything to the results presented in this chapter, but will be of concern when designing the real implementation of this algorithm.

is fulfilled *WORTH_CST* consecutive times. The constant *WORTH_CST*≥ 1 was introduced to meet Requirement 1. The smaller *WORTH_CST*, the more reactive is the algorithm to brief changes in utility. In expression (6.1), *THRESH* represents the minimum utility gain required to trigger a color change (Requirement 2). Finally, the parameter $c(t)$, which was introduced to meet Requirement 3, can be interpreted as the cost of a change. It is increased by a quantity A each time there is a color change and it decreases exponentially with time in absence of change. The parameters A and μ in Table 6.1 ("Color Switching Test") are chosen so that the cost $c(t)$ is prohibitive if the time elapsed since the last color change is smaller than a reasonable interval.

Sender Side Algorithm

Send voice packet
$p =$ ith packet to be sent
if (silence period & less than N probes were
sent since beginning of this silence period)
send p with color $=$ *probe_color*
else
send p with color $=$ *cur_color*

Receive report
report *rr* received
based on parameters included in *rr*,
compute the utility associated to each color:
ut[*cur_color*] and *ut*[*probe_color*]

if "Color switching Test" succeeds
switch *probe_color* and *cur_color*

"Color Switching Test"

$t_{last} =$ time last color switching occurred
$c(t) =$ cost associated to frequent color switching
$t = now$
update $c(t)$: $c(t) = e^{-\mu(t-t_{last})}\ c(t_{last})$

if *ut*[*cur_color*] + *THRESH* < *ut*[*probe_color*] $- c(t)$
if *worth_counter* $=$ *WORTH_CST*
$t_{last} = now$
$c(t_{last}) = c(t) + A$
worth_counter $= 0$
return "Color Switching Test" succeeds
else
worth_counter $=$ *worth_counter* $+ 1$
return "Color Switching Test" fails
else
worth_counter $= 0$
return "Color Switching Test" fails

Receiver Side Algorithm

Receive voice packets
packet p received
this_color $=$ color of p

add packet to history[*this_color*]
if *this_color* $=$ *cur_color*
update playout buffer

Send report
if ⟨time to send a report⟩
update estimates of p, q and ler for
for each color based on their
respective history
send report including:
p, q, ler, $F_D(d)$ of both color

Table 6.1: Pseudocode of the Color Choosing Algorithm. *now* is the current time, variables are initialized as follows: *worth_counter* $= 0$, $t_{last} =$ starting time of the connection and $c(t_{last}) = A$.

6.3.2 Simulations

The Adaptive Color Choosing algorithm (ACC) was implemented in our delay-aware audio sources in the $ns2$ simulator. In this section, we show some simulation results that illustrate the behavior of ACC. Network topology and simulation settings remain the same as in Section 6.2. The following parameters were used for the ACC algorithm: $N = 10$, *THRESH*$= 3$, *WORTH_CST*$= 10$, $A = 10$, $\mu = 1$. The history size was set to $nb = 1000$ for the current color and to $nb = 500$ for the probe color[6].

ACC Parameter Estimation

One of the key factors of the ACC algorithm is the estimation of channel parameters for the probe's color. In order to illustrate the accuracy of the parameters measured by the probes, we started by simulating blue and green audio sources with ACC where we imposed the sources remain with the same current color during the whole simulation. The audio sources were therefore only estimating channel parameters of their own service class and probing the other service class but were not allowed to switch from one color to the other. Figure 6.7 compares network parameters measured by green and blue audio sources to parameters estimated by the probes of the other color. The thin lines on these graphs represent the parameters estimated by the probes. As one can see, the Gilbert parameter p, the maximum sending rate R_{max} and the 99 percentile of end-to-end delays are estimated with a fairly good precision. For the Gilbert parameter q, on the other hand, it is very difficult to obtain an accurate estimation. This comes from the fact that q is sampled only when there is a packet loss (since q is probability to receive the next packet, given that we just lost one), which reduces considerably the number of samples available for q and increases the oscillations in the estimation of this parameter.

Figure 6.8 compares the actual utility received by the green and blue audio flows to the utility estimated by the probes. The different graphs were obtained with utility functions f_1, f_2 and f_3. Figure 6.8 shows that the utility estimated by the probes of a given color follows very closely the actual utility received by sources using this color.

ACC Dynamics

To illustrate the dynamics of the ACC algorithm, we consider a scenario where the number of connections sharing the bottleneck varies during the simulation. The audio sources

[6]Here again, there is clear tradeoff between responsiveness to changes in the network and avoidance of oscillations. We tried other values for the history size, but a size of 1000 packets for the current color's history showed to be a good compromise between stability and responsiveness in our simulations. The history size for probe colors was chosen to be smaller than for the current color since the probe packets represent only a small fraction of the packets sent by the source.

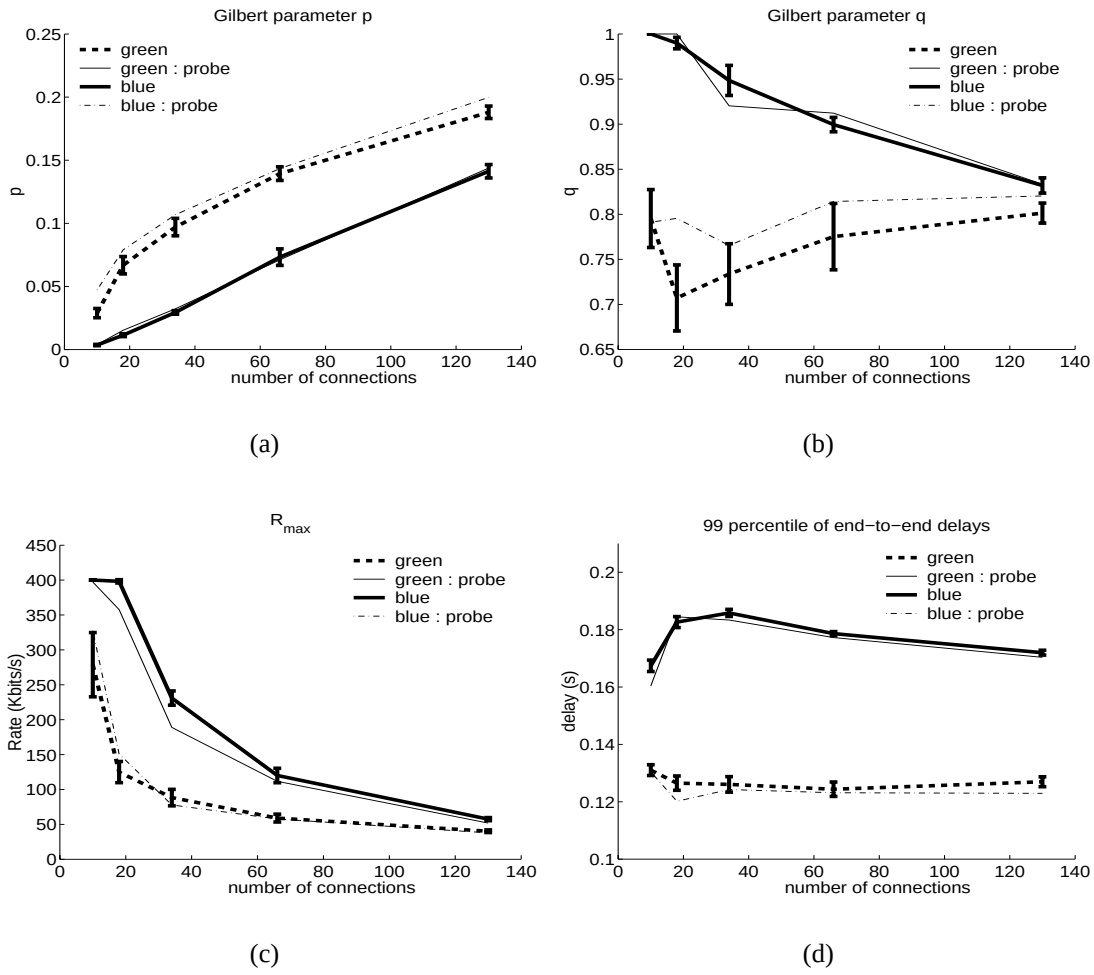


Figure 6.7: Network characteristics estimated by the probes vs actual values for *green* and *blue* traffic. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

---

with ACC are now allowed to switch from one service (colors) to the other[7]. At the start of the simulation, two delay-aware audio flows (one starting with current color green and the other one with blue) implementing ACC compete for bandwidth with two Sack TCP connections[8] (one of each color). To add a supplementary random factor and avoid synchronization effects, we also inject some background traffic consisting of ON/OFF sources with ON and OFF periods exponentially distributed. The background traffic was using on average 25% of the total bottleneck bandwidth. At time $t = 200s$, we increased the number of connections sharing the link and introduced 30 additional Sack TCP connections of each type (blue and green) in the system.

Figure 6.9 illustrates the dynamics of the two delay-aware audio sources. Figures 6.9(a), 6.9(b), 6.9(c) and 6.9(d) represent respectively Gilbert parameters p and q, utility and

[7]We allow connections to switch colors only after 30s of activity in order to avoid that sources oscillate from one service to the other while parameters' estimates have not stabilized yet.

[8]The two Sack TCP connections start at $t = 0s$ and the audio sources start a bit later at $t = 20s$.

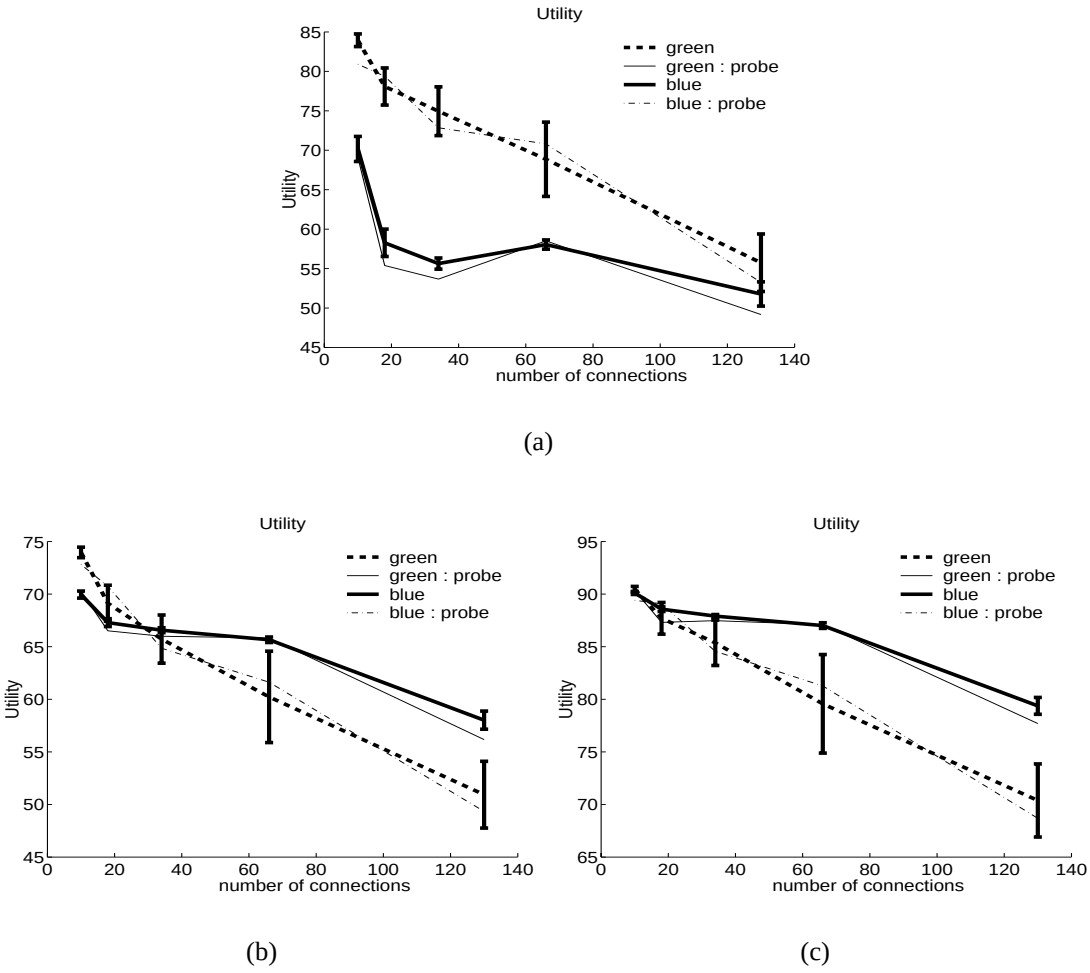


Figure 6.8: Actual vs estimated utility for *green* and *blue* traffic with (a) utility f_1, (b) utility f_2 and (c) utility f_3. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

---

playout delay as a function of time. For each audio source, we show the evolution of these parameters for current and probe color.

In this simulation, utility function f_2 is used. The optimal service is the green one when network load is low (up to $t = 200s$) and the blue one when network is congested (from $t = 200s$). From Figure 6.9(c), we see that the connection that starts blue switches to green at time $t = 50s$ (namely exactly 30s of activity) while the green connection keeps on using this service. When network load increases, the current color's utility decreases rapidly compared to the probe color's one and both connections switch from green to blue (the *start green* and *start blue* connections respectively change color at $t = 205s$ and $t = 206s$).

From Figure 6.9, one can see that the sources successfully estimate, in real-time, the optimal service to be used and they are now able to react within a few seconds to changes in network conditions. An even smaller reaction time could be obtained by decreasing

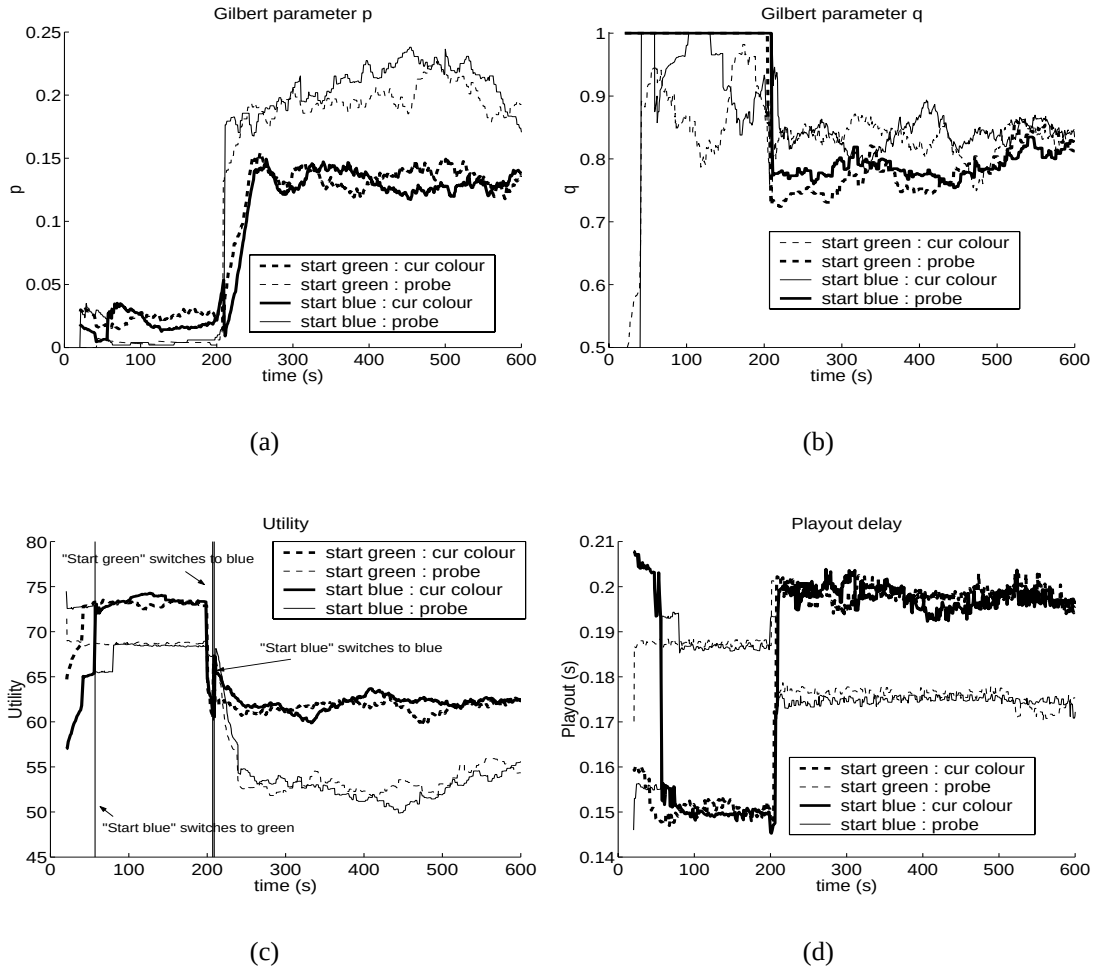


Figure 6.9: Dynamics of delay-aware audio with ACC over ABE. Utility function is f_2. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

---

the history size, but this would be at the price of increased oscillation in the parameter estimation.

Performance of Delay aware audio with ACC

In this section, we investigate the performance of delay-aware audio sources with ACC under the same conditions as in Section 6.2. Figure 6.10 shows the performance of delay-aware audio flows over ABE when d_g is 40 ms, d_v is 100 ms and utility function f_1 is used. On these graphs, "*green* with ACC" refers to connections that start with current color green and implement ACC. All curves correspond to the current color of connections. In order to compare these results to the ones obtained without ACC, we also displayed, with thin lines, the performance of audio sources that do not implement ACC.

Figures 6.10(a), 6.10(c) and 6.10(d) respectively depict the end-to-end delay, the residual packet loss rate and the rate of the reconstructed audio stream at the receiver. The re-

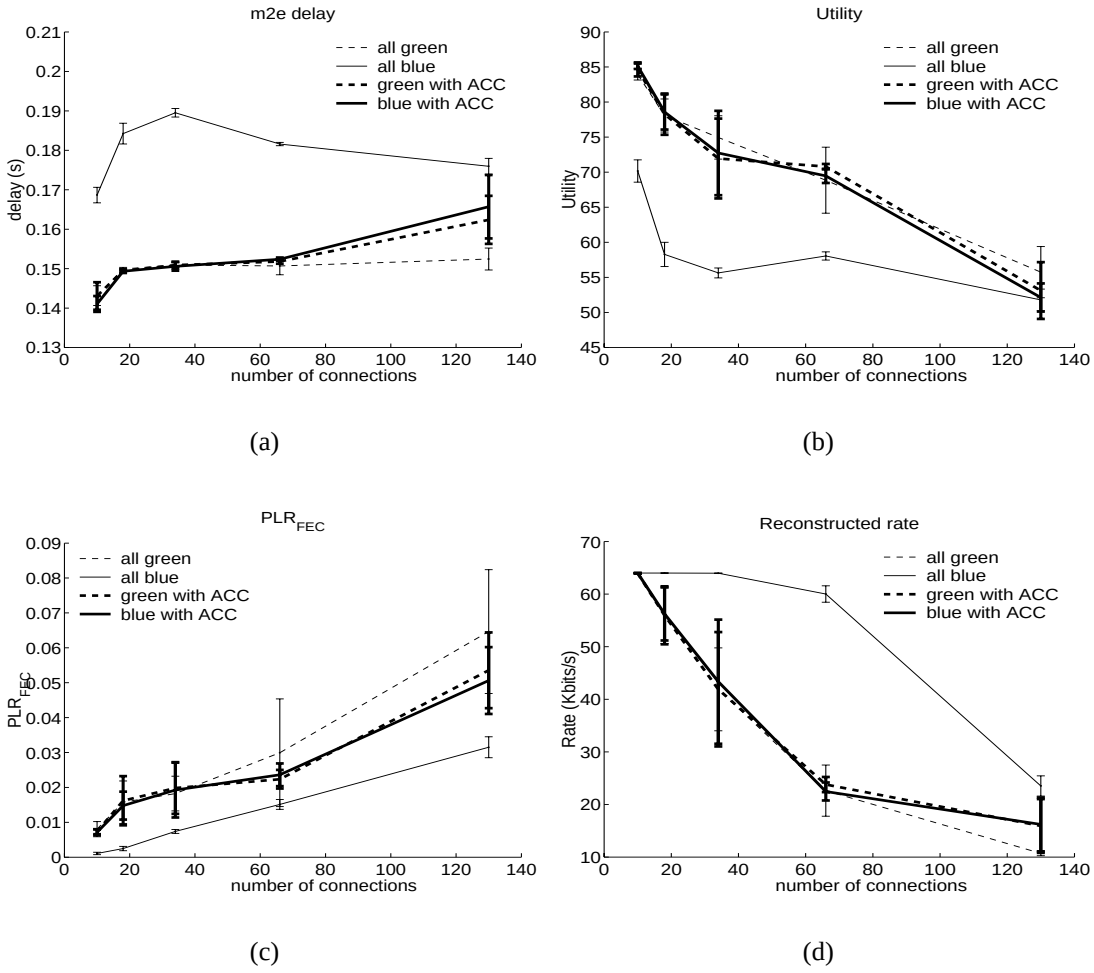
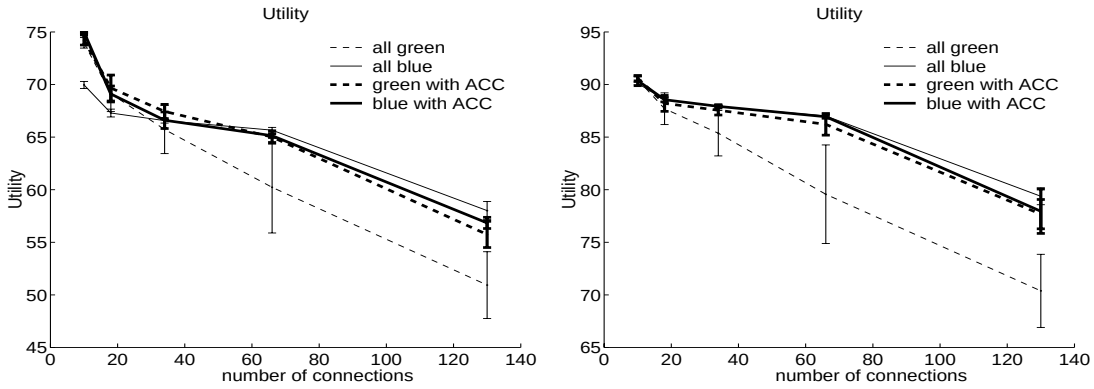


Figure 6.10: Performance of delay-aware audio with ACC over ABE with utility f_1. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

---

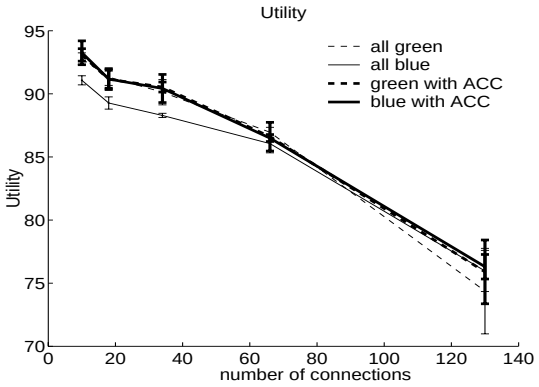
sulting average utility, measured at the receiver, is given in Figure 6.10(b). Similar results were obtained with utility functions f_2 and f_3 as can be seen on Figure 6.11 where the average utility obtained with both of these functions is depicted. Results corresponding to the case of small propagation delays are shown in Figure 6.12.

Figures 6.10(b), 6.11 and 6.12 show that **the ACC algorithm allows a source to adaptively choose the service that maximizes its utility and therefore to make an optimal use of the non-elevated service.**

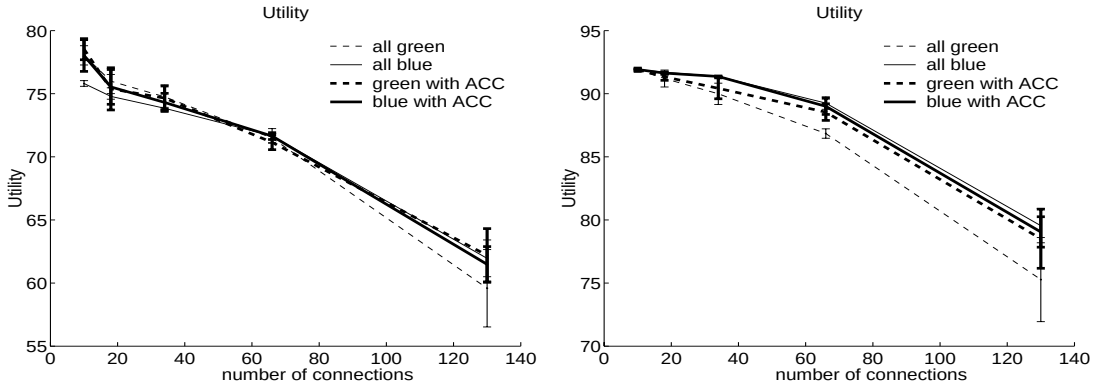


(a) (b)

Figure 6.11: Performance of delay-aware audio with ACC over ABE : average utility obtained with utility function (a) f_2 and (b) f_3. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.



(a)



(b) (c)

Figure 6.12: Performance of delay-aware audio with ACC over ABE : average utility obtained with utility function (a) f_1, (b) f_2 and (c) f_3. $d_{prop} = 30\,ms$, $d_g = 40\,ms$.

6.4 Is Non-Elevated service better than Flat for VoIP?

Now that audio sources are able to make an optimal use of a non-elevated service, the ultimate question that we tackle in this chapter is the following: does a non-elevated service bring a benefit compared to the single class Best-Effort[9] service?

To this end, we take up the simulation scenario introduced in Section 6.2 and we replace the ABE queue by a RED queue with the same parameters as the ABE virtual queue. Namely, the maximum waiting time in the RED queue is d_v. The minimum threshold ($minth$) is set to 40% of the maximum queue size and the maximum threshold is $2 * minth$.

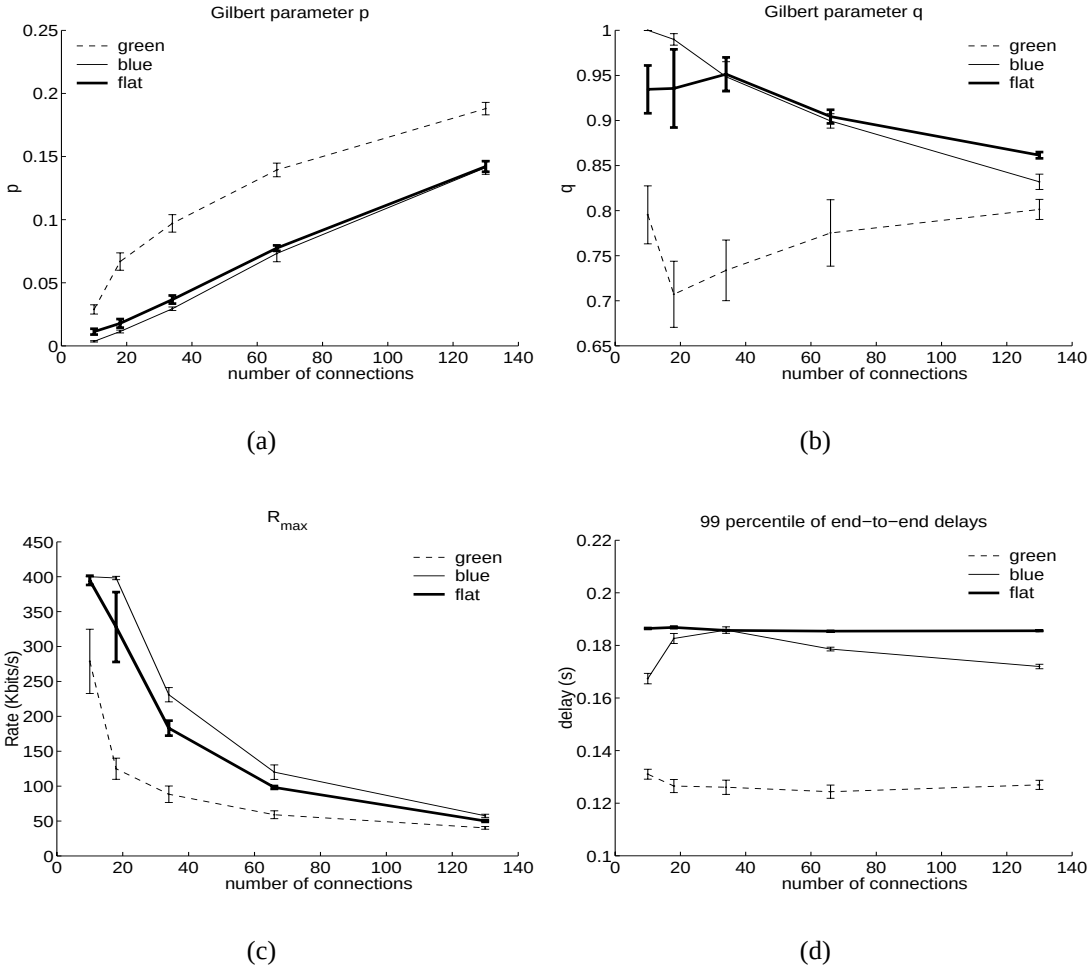


Figure 6.13: Network characteristics for ABE and Flat best-effort services. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

All other things being equal (number and type of connections), Figure 6.13 compares network characteristics of a Flat Best-Effort service to ABE blue and green services[10]

[9] In this work, we refer to a network providing only single class Best-Effort service as a *Flat* network.
[10] ABE results are the same as in Figure 6.4.

when d_{prop} is 80 ms. One can notice that the 99 percentile of end-to-end delays (see Figure 6.13(d)) is sometimes a bit lower for the blue ABE service than for the Flat Best-Effort service. This comes from the fact that, when green packets are discarded because they can not meet their deadline, they make room for blue packets that see their queueing delay reduced.

Figure 6.14 compares the performance of delay-aware audio with ACC over ABE to delay-aware audio over Flat Best-Effort when utility function f_2 is used. For clarity, we also display the performance of delay-aware audio over plain green and blue services (without ACC). Figure 6.15 shows results obtained with utility functions f_1 and f_3 under the same conditions. We also compared ABE to Flat in the case of small propagation delay (d_{prop} is 30 ms). Figure 6.16 depicts channel characteristics in this case and Figure 6.17 shows results obtained with the different utility functions. The reason why even the blue ABE service performs so well compared to the Flat service is again due to the fact that many green packets are dropped and hence make room for blue ones that see their end-to-end delay decrease.

From Figures 6.14(b), 6.15 and 6.17, it appears clearly that, in the case where queueing delay is important, the introduction of a non-elevated service allows to improve the perceived quality of audio applications.

6.5 Conclusions

In this chapter, we evaluated the benefit, for an audio source, to use a non-elevated service. We showed by simulation that the use of such a service can lead to quality improvements, but that the choice of service depends on network conditions (network load and network topology) and on the importance that users attach to delay. This observation lead us to propose an Adaptive Color Choosing (ACC) algorithm that allows audio sources to choose in real-time the service providing the highest audio quality. We showed by simulation that sources using this algorithm successfully estimate the best service to be used and are able to switch rapidly from one service to the other when it is appropriate. The ACC thus allows an audio source to make an optimal use of non-elevated services. Finally, using the optimal error/delay and service (color) control scheme, we showed that interactive audio sources would really benefit from using a non-elevated service coupled with an ACC algorithm, compared to a single class best-effort service, in cases where queueing delay is important. In particular, the introduction of a non-elevated service used in conjunction with an ACC algorithm allows to reduce the mouth-to-ear delay without any significant increase of the packet loss rate ater reconstruction. This confirms that non-elevated services are good candidates for the deployment of low complexity differentiated service, possibly at the edge of the network.

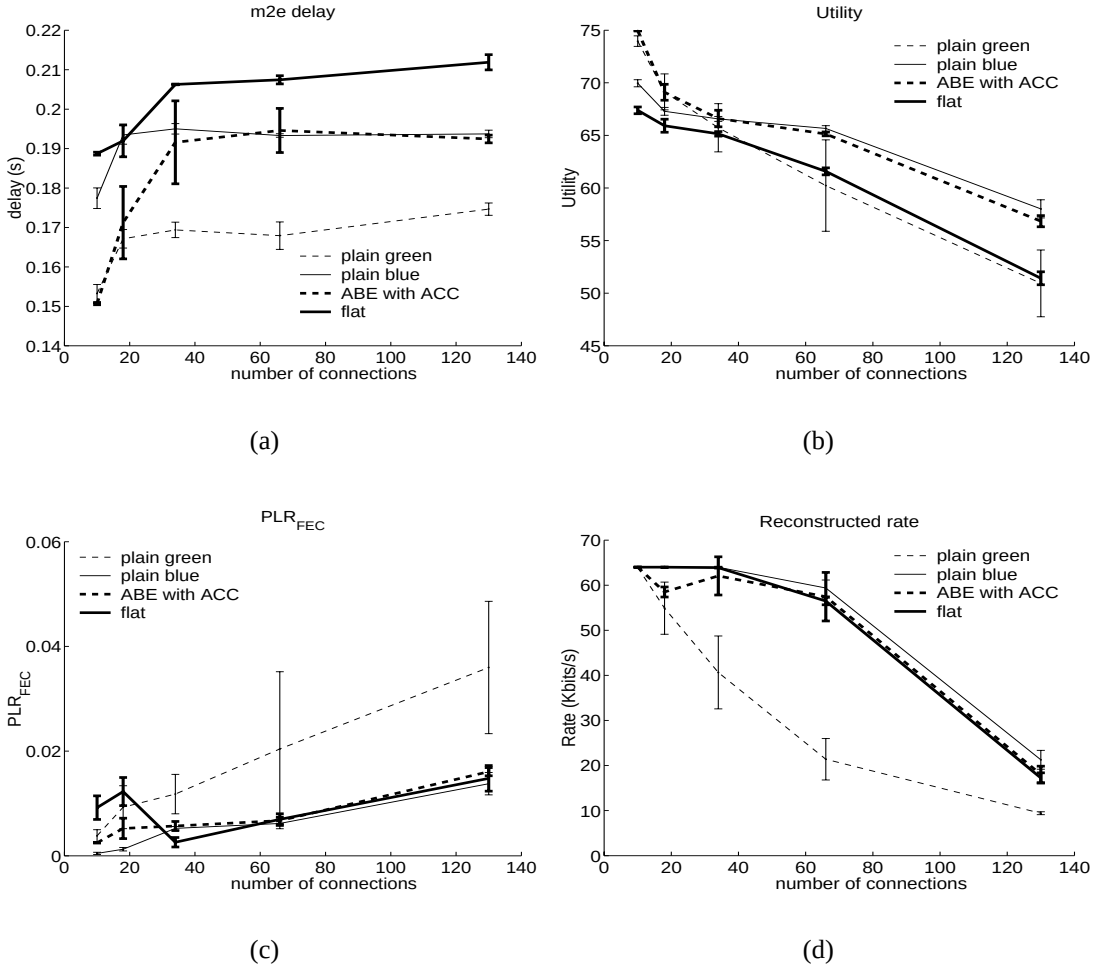


Figure 6.14: Performance of delay-aware audio over ABE vs Flat with utility f_2. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

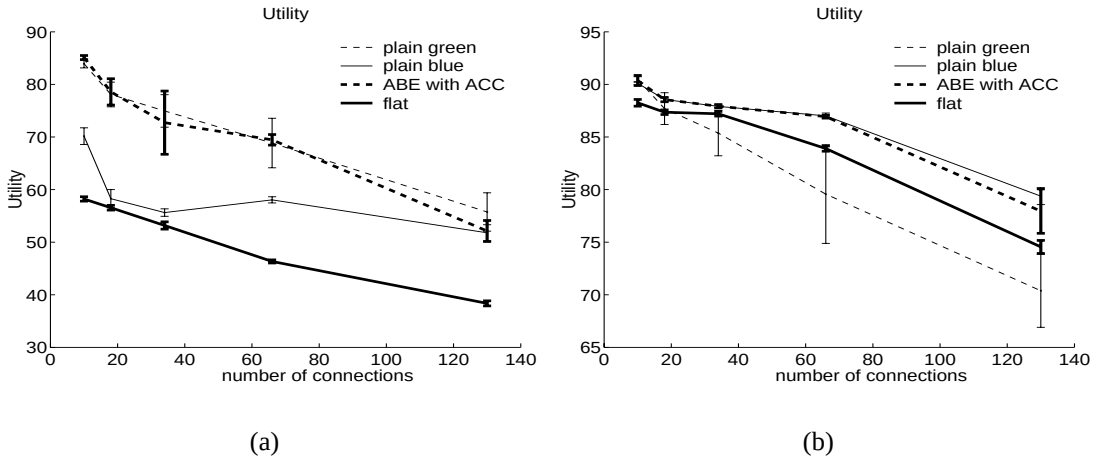


Figure 6.15: Performance of delay-aware audio over ABE vs Flat : average utility obtained with utility functions (a) f_1 and (b) f_3. $d_{prop} = 80\,ms$, $d_g = 40\,ms$.

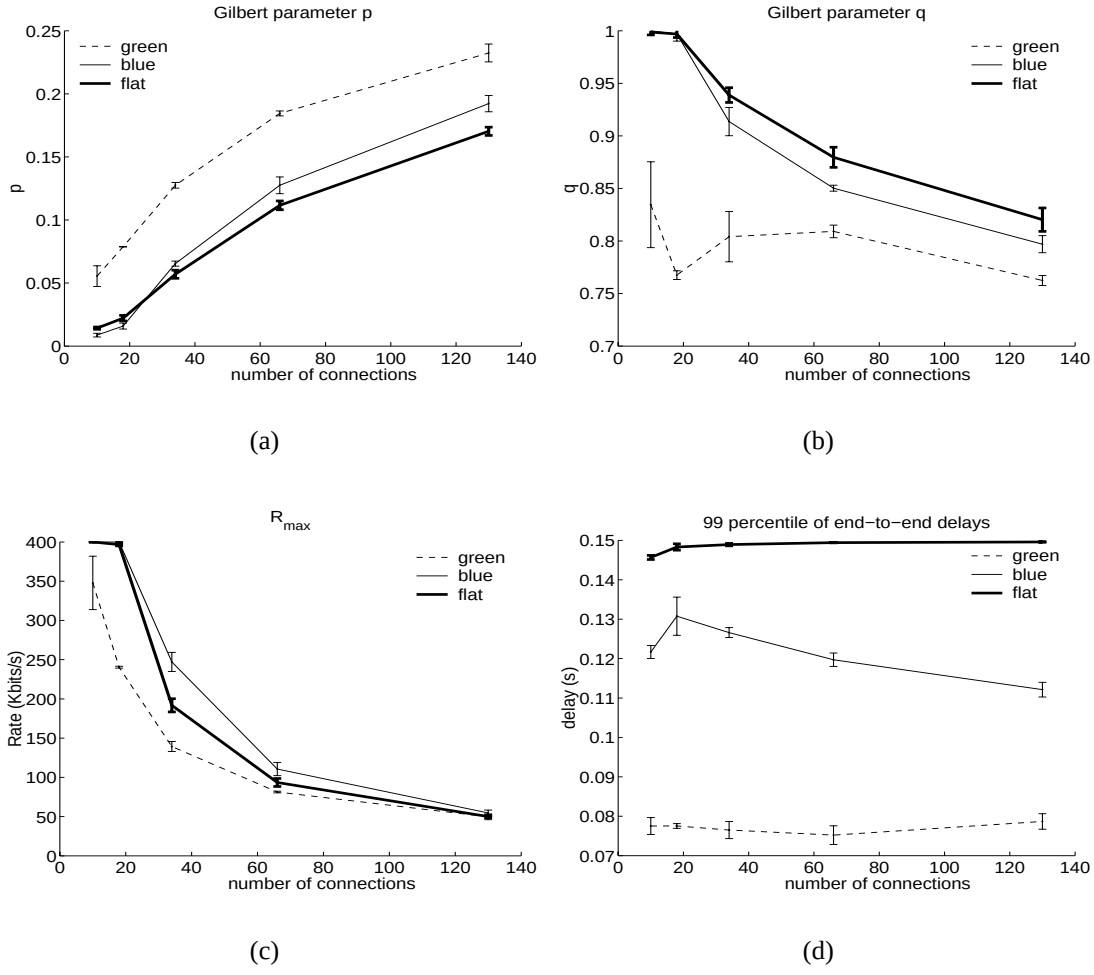


Figure 6.16: Network characteristics for ABE and Flat best-effort services. $d_{prop} = 30\,ms$, $d_g = 40\,ms$.

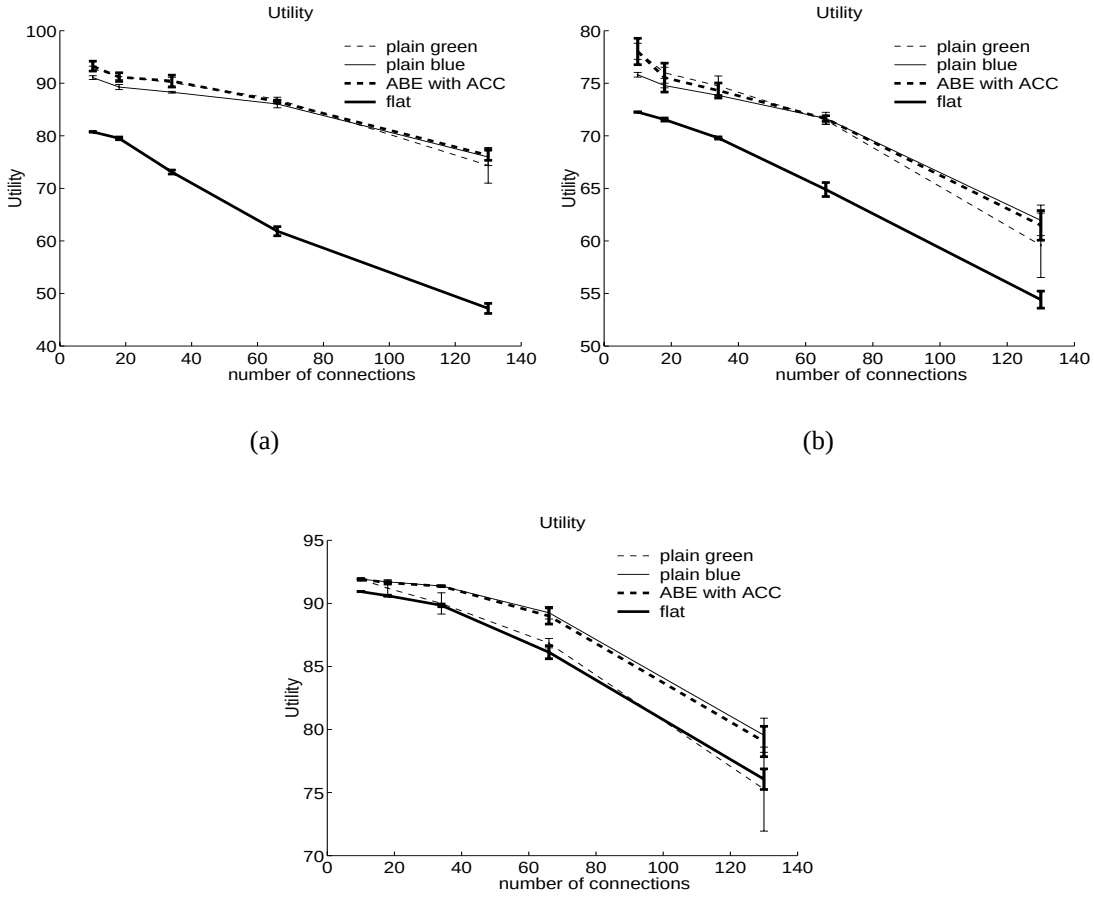


Figure 6.17: Performance of delay-aware audio over ABE vs Flat : average utility obtained with utility functions (a) f_1 and (b) f_2 and (c) f_3. $d_{prop} = 30\,ms$, $d_g = 40\,ms$.

Chapter 7

Congestion Control for Variable Packet Sizes

7.1 Introduction

Current congestion control mechanisms are based on packets (i.e., they aim to reduce or increase the number of packets sent per time interval to adjust to the current level of congestion of the network). This is the correct behavior in a packet rate limited environment. Furthermore, if all flows use the same packet size, for example the maximum transmission unit (MTU), then bandwidth will be shared fairly among them also in the case of a bandwidth limited bottleneck. In an environment where the bottleneck is bandwidth limited but flows may use packets of different sizes, however, resources will not be shared fairly among flows but the resource usage will depend on the packet size. A packet size different from the MTU may be necessary for applications such as VoIP, where a high packet rate needs to be maintained in order to provide audio transmission with a sufficiently low delay.

In this chapter, we study the impact of variations in the packet size on equation based congestion control. Ultimately, it is desirable to design a congestion control mechanism that ensures a fair sharing of resources independently of the packet size of a flow. To this end, it is necessary:

- to determine a rate that is fair to flows competing for the same limited resource, but having different packet sizes, and
- to calculate this rate, which requires to compensate for the bias in the measured loss event rate introduced by the different number of packets per time interval over which the loss event rate is measured.

Three alternative approaches to ensure a fair sharing of resources are possible. If the network is modified so that the response from the network depends on the resources

used and not on the packet rate, much of the performance penalty for flows sending small packets would be removed. In theory, it is possible to modify RED to this end but here we mainly focus on end-to-end mechanisms. We only briefly address possible modifications to the RED gateways in the discussion section. If the network remains unchanged, we can either modify the sender and leave the receiver-side loss measurement mechanism as it is, or we can leave the sender unchanged but modify the loss measurement mechanism. As we will see later in this chapter, a modified loss measurement is more robust under various network conditions and better able to produce a TCP-friendly rate. Only under very stable network conditions do sender-based modifications produce similar results.

We base our analysis on the unicast TFRC protocol [56] and the multicast TFMCC protocol [111]. TFRC and TFMCC are very similar in the way a fair sending rate is calculated and hence the considerations in this chapter apply to both protocols.

We give a quantitative analysis of the bias introduced into the loss measurement process when sampling the bottleneck at different rates. Consequently, three alternative modifications to the loss measurement mechanism of equation based congestion control protocols are presented that remove this bias. These mechanisms are evaluated in detail through mathematical analysis and network simulation. When the bottleneck is bandwidth limited, the presented mechanisms aim to achieve a sending rate comparable to that of a TCP flow with path MTU discovery [82] under similar network conditions. Hence, for a fair sharing of resources, it is no longer necessary that competing flows have the same packet size.

This chapter is organized as follows:

- Section 7.2 discusses the influence packet size on TCP and introduces important definitions related to equation based congestion control.
- In Section 7.3, we present different queueing strategies and discuss their impact on the packet drop rate of a flow. Our analysis of the loss measurement mechanisms is based on these queueing schemes.
- In Section 7.4, we describe the alternative approaches that are possible to ensure a fair sharing of resources.
- In Section 7.5, we propose modifications to equation based congestion control to better cope with different packet sizes.
- We present extensive simulations results for these mechanisms under various network conditions in Section 7.6.
- We discuss how to modify RED so that resources are shared fairly in Section 7.7 and finally give some concluding remarks in Section 7.8.

7.2 Transport Protocols

In this section we give a brief overview of the TCP protocol and equation based congestion control. In particular, we discuss their behavior when used with different packet sizes. Note that throughout this chapter, whenever we refer to packet size we mean the size of the whole packet including IP and transport protocol header. Accordingly, the payload of the packet that can be used by the application is smaller. The smaller the packet size, the more unfavorable the ratio of payload to packet header.

7.2.1 TCP with Small Packets

TCP uses an *additive increase multiplicative decrease mechanism* (AIMD) to detect additional available bandwidth and to react to congestion, where congestion is indicated by packet loss. Upon reception of an ACK, the TCP sender increases the congestion window by $s^2/cwnd$, where $cwnd$ is the size of the congestion window in bytes and s is the segment size in bytes [16]. The resulting increase in window size is approximately one segment per round-trip time without congestion indications. In TCP, there are two mechanism to detect congestion:

- If a data packet is not acknowledged by the receiver within a certain time span (the retransmission timeout value), the sender assumes severe congestion and the congestion window is reduced to one segment.
- Upon packet arrival the TCP receiver acknowledges the sequence number of the last data packet that arrived in order. Therefore, packet loss or packet reordering results in duplicate acknowledgements when new packets arrive. Four acknowledgements for the same sequence number, called a triple duplicate ACK (TDACK), are a strong indication that one or more packets were lost. As a consequence, the sender reduces the congestion window to half of its previous size.

In terms of packets, TCP connections with different segment sizes react in the same way to congestion indications. They increase their window size by one packet per round-trip time in the absence of congestion and halve their window size in response to packet losses. As shown in Figure 7.1, they have approximately the same number of packets in their congestion window under comparable network conditions (i.e., same round-trip time and same packet loss rate). Consequently, TCP throughput scales roughly linearly with the segment size, assuming a drop rate that is independent of the packet size and ignoring the size of the protocol headers [17].

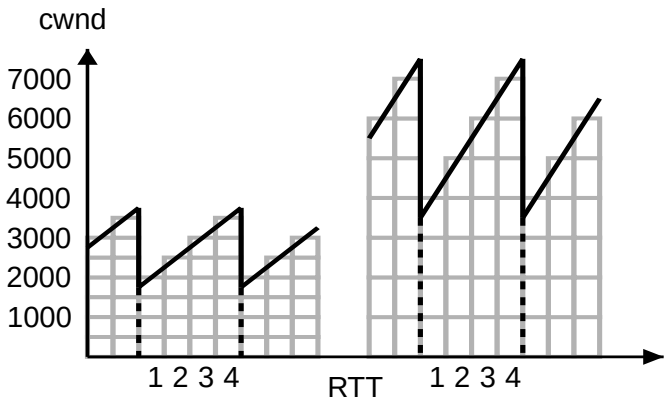


Figure 7.1: Evolution of the TCP congestion window over time

7.2.2 Equation Based Congestion Control

For TCP-friendly equation based congestion control, the sending rate of a flow is adjusted to the rate a TCP flow would achieve under the same network conditions. This is done not through the use of a congestion window as in TCP, but by directly setting the sending rate to a rate that conforms to TCP. The rate is commonly determined using a model for long-term TCP throughput. A simple form of such a model was presented in [78, 58].

$$R_{TCP} = \frac{1.5\sqrt{1/3}\,s}{t_{RTT}\sqrt{p}} \tag{7.1}$$

The expected throughput R_{TCP} of a TCP flow is calculated as a function of the loss rate p, the round-trip time t_{RTT}, and the packet size s. A receiver measures its loss event rate and its RTT to the sender. It then uses Equation (7.1) to calculate R_{TCP}, feeds this rate back to the sender, who adjusts his/her sending rate accordingly. We will denote equation-based congestion mechanisms based on this model by SQRT.

A more complex model that much better captures TCP throughput at higher packet drop rates was presented in [89].

$$R_{TCP} = \frac{s}{t_{RTT}\left(\sqrt{\frac{2p}{3}} + \left(12\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)\right)} \tag{7.2}$$

The class of equation-based congestion control mechanisms based on the more complex model will be denoted by PFTK. As such, TFRC and TFMCC fall into this category.

PFTK mechanisms require the measurement of a loss event rate p rather than a packet loss rate. A *loss event* can consist of several packets lost within the same round-trip time. Since TCP should ideally halve the congestion window only once in response to several losses per round-trip time[1], the PFTK loss measurement process explicitly ignores losses

[1]TCP Tahoe, NewReno and Sack generally halve the congestion window once in response to several

within the same RTT and aggregates them to a single loss event. Consequently, a loss event represents the point in time where the TCP congestion window would be reduced in response to congestion. The initial packet loss that results in a loss event is followed by a period of time where all packet losses will be ignored by the loss measurement process. In this dissertation, we call this period the *Loss Insensitive Period* (LIP). On average, if a rate controlled flow sends N packets per round-trip time, the LIP period consists of $N-1$ packets; since the initial packet loss is taken into account, it is not part of the LIP. A *loss interval* is defined as the number of packets between loss events and the loss event rate is then computed as the inverse of the average loss interval.

The relationship of the TCP sending rate and the packet size discussed in the previous section is also evident from the models for TCP throughput, Equation (7.1) and Equation (7.2). It therefore also holds for equation-based congestion control. When a congestion controlled flow uses a packet size smaller than that of a TCP flow but wishes to avoid the linear decrease in throughput, it needs to increase the packet rate. The impact of such an increase on the dynamics of the congestion control mechanism is analyzed in detail in Section 7.4.

7.3 Queueing

As mentioned in Section 7.1, the resource usage of a flow should determine the throughput that this flow achieves. Yet, the congestion signals flows receive depend to a large degree on the queueing scheme that is utilized at the bottleneck, and not only on the resource usage. A congestion control scheme that aims to achieve a fair sharing of resources among flows with different packet sizes therefore needs to take queue management into account.

The predominant queueing scheme in the Internet is drop-tail. Packets are enqueued at the tail of the router queue, as long as buffer space is available. If the maximum queue length is reached, arriving packets are dropped, until packets from the queue are transmitted and buffer space is freed. Depending on the specific buffer management used, the size of the queue can be

- a certain number of packets (if a packet occupies one bin in the queue independent of its actual size),
- a certain number of bytes (if packets are queued in bins of exactly the packet size), or
- a combination of both (if there is a fixed number of bin sizes and packets are placed in the smallest free bin that can hold the packet or if packets are spread over a

losses per window while Reno reduces the congestion window twice in response to multiple losses in a window of data.

number of small bins of fixed size).

Despite its wide deployment, drop-tail queueing has significant drawbacks. Under some circumstances, queue space can be distributed very unequally among competing flows, which allows some connections to receive a much higher share of bandwidth than others. This phenomenon is called "lock-out". Furthermore, drop-tail queues signal congestion only when the queue is full and therefore operate at (close to) full occupancy for significant periods of time, incurring a large end-to-end delay and increasing the probability of loss bursts. These undesirable properties of drop-tail queues are discussed in more detail in [26], and the authors propose the use of active queue management as an alternative.

Through active queue management, routers are better able to control the queue by proactively dropping packets *before* the buffer overflows, the most promiment example being Random Early Detection (RED) [57]. RED maintains an exponentially weighted moving average of the queue size and drops packets probabilistically, based on the average queue size being between certain thresholds. Below a minimum threshold, no packets are dropped, between the minimum threshold and a maximum threshold the drop probability varies between 0 and a configurable maximum drop probability, and above the maximum threshold all packets are dropped.[2]

As with drop-tail queues, the queue size can be measured either in bytes or in packets. Furthermore, it is possible to base the drop probability on the size of the packet to be dropped.

While a number of other queue management and scheduling algorithms have been proposed (e.g., [47, 46, 107, 40]), we will limit our analysis to drop-tail queueing and RED queueing, as they are the only queue management schemes currently deployed in the Internet.

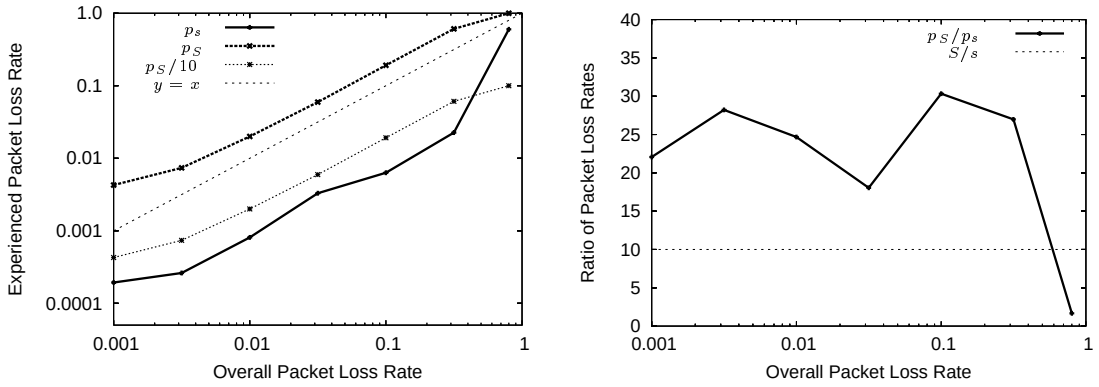
7.3.1 Drop-Tail Packet Drop Probabilities

When only a few flows compete at a bottleneck with a drop-tail queue in packets, the queue occupancy and therefore packet drop probability depend to a large extent on the recent "drop history" and the consequent changes in the sending rates of the flows. However, if the number of flows sharing the bottleneck is high, the packet drop probability is relatively independent from the drop history and flows will experience a full queue with the same probability. The number of packet drops a flow can expect to see only depends on the rate at which it "samples the bottleneck" (i.e., the packet rate) and is independent from the packet size. Also the packet drop pattern depends on the level of statistical multiplexing. With few flows sharing the bottleneck, flows experience bursts of packet

[2] A more "gentle" RED variant which drops packets less aggressively when the average queue size exceeds the maximum threshold is described in [51].

drops when queue is full until the flows react to the congestion (usually about one RTT later). No packets are lost inbetween those intervals of very high packet drop rates. For higher numbers of flows, the bottleneck queue tends to be close to full most of the time and the distinct pattern of packet drops is lost (although also large numbers of flows may synchronize).

When the queue size is measured in bytes rather than packets, small packets have a higher probability of fitting into a nearly full queue. This reduces the number of packet drops for flows with small packets, compared to a queue measured in packets. Under the assumption that each queue occupancy is equally likely, a small packet of size s is S/s times less likely to be dropped than a large packet of size S. However, as discussed before, the bottleneck queue tends to be full rather than empty. If small packets also arrive at a higher rate than large packets (to achieve the same throughput), the queue occupancy distribution is much less favorable for flows with large packets.



(a) Experienced packet loss rates

(b) Ratio of packet loss of large packet CBR flow to small packet CBR flow

Figure 7.2: CBR flows with drop-tail queue in bytes

While a mathematical analysis of the drop probabilities for packets with different sizes in a queue measured in bytes is complex, simulations already provide some insight into the bias introduced by the different packet sizes. For the graph shown in Figure 7.2(a), two CBR flows with the same bitrate and packet sizes of 1000 bytes and 100 bytes, respectively, were run over the same bottleneck. The combined sending rates are slighly higher than the bottleneck bandwidth so as to produce the average packet drop rate indicated on the x-axis. On the y-axis, we plot the packet drop rate experienced by the different flows. In case the queue occupancy were uniformly distributed, we would expect to see a ten times lower drop rate ($p_S/10$) for the flow with small packets than for the flow with large packets (p_S). However, the drop rate for the small flow (p_s) is much lower than that. From Figure 7.2(b) we can see that p_s is in fact too low by a factor of roughly 2.5, unless the

drop rate is close to 100%.

The way the simulation is set up, the queue is close to full all the time, which results in a distribution of queue occupancy that is exceedingly favorable for flows with small packets. When the buffer utilization varies over a larger range of values (as is to be expected when the queue occupancy is driven by TCP's AIMD), we expect the bias to be somewhat less pronounced.

7.3.2 RED Packet Drop Probabilities

As specified in [57], the RED's average queue size avg is calculated from the current queue occupancy q using an exponentially weighted moving average

$$avg \leftarrow (1 - w_q)\, avg + w_q\, q$$

The current packet drop probability p_b varies linearly with the average queue size

$$p_b \leftarrow max_p\, (avg - min_{th})/(max_{th} - min_{th})$$

where max_p is the maximum value for the drop probability and max_{th} and min_{th} are the maximum and minimum queue threshold, respectively. This drop probability is not used directly but transformed in order to provide uniformly distributed packet drops (or more specific a uniform distribution of the number of packets between packet drops).

$$p_a \leftarrow p_b/(1 - count \cdot p_b)$$

Here, $count$ is the number of transmitted packets (i.e., packets that were not dropped) since the last dropped packet. Arriving packets are then dropped with the probability p_a, if the average queue size is between the two thresholds. The drop probability is independent of the size of the incoming packet. A discussion of reasonable values for the RED parameters used above can be found in [53, 55].

The queue thresholds and the queue occupancy can either be measured in packets or in bytes. Furthermore, the packet drop probabilities can be modified accordingly such that large packets are more likely to be dropped than small packets. The author of [54] recommends to use RED in byte mode when bottlenecks are bandwidth limited. For RED in byte mode, the final packet drop probability has to be weighted by the ratio of the size of the current packet s_i to a maximum packet size S

$$p_a \leftarrow \frac{p_b}{(1 - count \cdot p_b)} \frac{s_i}{S}$$

and $count$ needs to be increased by the appropriate fraction of a large packet

$$count \leftarrow count + \frac{s_i}{S}$$

as specified in [33]. In this case, the probability distribution of the number of packets between packet drops is

$$P(L = m) = \begin{cases} 0 & \text{if } \sum_{i=1}^{m} s_i/S > 1/p_b \\ p_b \dfrac{s_m}{S} & \text{if } \sum_{i=1}^{m} s_i/S \leq 1/p_b \end{cases}$$

where s_i is the size of the i^{th} incoming packet after a drop. While this is the distribution seen at the router itself, a single flow will usually experience a different distribution of the number of packets between drops. Only if there is a single flow on the bottleneck link will it experience the same uniform distribution. For a sufficiently high level of statistical multiplexing, the drop probability for a packet of a flow is independent from the packet loss the flow experienced earlier. Therefore, the distribution will be increasingly closer to a geometric distribution as the level of statistical multiplexing increases.

As discussed in [33], RED in byte mode decreases the difference in throughput between flows with large packets and flows with small packets, but with TCP or TCP-friendly congestion control, a significant incentive to use large packets remains. To achieve the same throughput, a decrease in packet size by a factor of η ($\eta \geq 1$) would have to be compensated by a decrease in the loss rate by a factor of η^2 when using SQRT congestion control. The same holds for PFTK with moderate loss rates, while for high loss rates less than a factor of η^2 is sufficient since throughput decreases overproprtionally in the regime of very high loss rates.

7.4 Design Space for Congestion Control with Variable Packet Sizes

According to the original model for long-term TCP throughput as given in Equation (7.2), throughput is directly proportional to the packet size; a flow sending packets of size half the maximum transmission unit (MTU) will achieve half the sending rate of a flow sending MTU sized packets. This linear decrease in throughput is justified if the limited resource at the bottleneck is the number of packets sent over time. If however the limited resource is bandwidth, flows sending smaller packets will suffer unnecessarily as small packets are less costly in terms of resource usage than large packets.

To compensate for the bias in favor of large packets, three alternative design choices come to mind, as shown in Figure 7.3.

- If TCP or TCP-friendly congestion control mechanisms are used with the current queueing schemes in the Internet, the packet drop rate experienced by a flow with small packets is too large for it to achieve the same thoughput as a flow sending large packets. Consequently, a modified queue management scheme can be devised that preferentially drops large packets. Specifically, such a scheme can be implemented by modifying the byte mode of RED accordingly.

- If receivers report to the sender their measured loss event rate, it is possible to have the sender adjust this measurement in order to achieve a fair sending rate.

- Similarly, the receiver could adjust this measurement. However, if the receiver is already being modified, it makes more sense to redesign the whole loss measurement process, taking into account the impact of the packet size on the packet drops experienced with different queueing schemes.

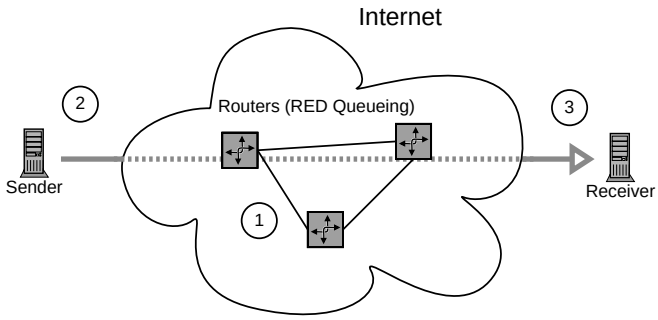


Figure 7.3: Design choices for the modifications

In the following sections, we will discuss the potential advantages and disadvantages of all three alternatives.

7.4.1 Network-Based Approach

When the bottleneck is bandwidth limited, a possible solution to reduce the discrimination against flows sending small packets would be to use RED gateways in byte mode. With these gateways, the fraction of packet drops for each connection sharing the bottleneck is roughly proportional to that connection's share of the bandwith. Figure 7.4 shows the normalized throughput achieved by TCP and TFMCC flows with large packets of 1000 bytes against the throughput of a TFMCC flow sending packets of 100 bytes (VP-TFMCC). Instead of a factor of 10 (as with a per packet drop probability), the TFMCC flow with small packets achieves a throughput only a factor of $\sqrt{10}$ worse than the throughput of the large flows. For high packet drop rates[3] above 10%, the throughput of the flow with small

[3] In fact, the correct parameter to investigate here would be loss event rate rather than packet drop rate.

packets even exceeds the throughput of "normal" TFMCC and TCP because of the over-proportional reduction of throughput of the PFTK formula in the high loss rate regime. Consequently, RED in byte mode does not suffice to ensure fairness between flows using different packet sizes.

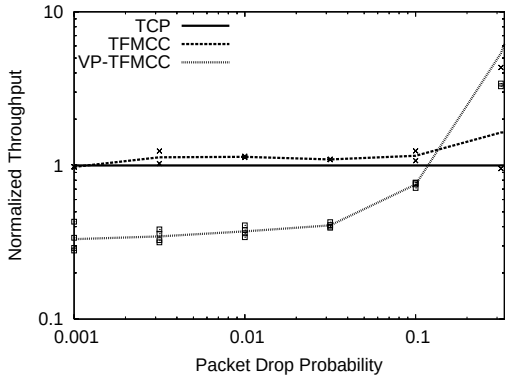


Figure 7.4: TFMCC with small packets and RED in byte mode

Nevertheless, by taking the loss-throughput relationship of the PFTK formula into account, it is possible to modify the packet drop probabilities of RED in byte mode such that flows with different packet sizes achieve the same throughput. We give an overview of how such a modified RED can be built and present a few simple simulation results in Section 7.7. However, changes to the network infrastructure of the Internet are extremely difficult to deploy and therefore the main focus of this chapter will be on an end-to-end approach.

7.4.2 End-to-End Approaches

Let S be the packet size for bulk data transfer, which is the MTU or the minimal MTU that has to be supported by an Internet router. Given a bandwidth limited bottleneck, a first step towards fairness is to use S instead of the actual packet size of the flow in the loss-throughput formula. If the other parameters of the formula (i.e., loss event rate and RTT) remain unchanged, flows will achieve the same throughput and therefore use the same amount of resources at the bottleneck no matter what their packet size is.

Unfortunately, sending packets at a higher rate introduces a bias in favor of flows with small packets in the loss measurement process. The smaller the packet size (i.e., the higher the packet rate) and the higher the packet drop probability, the higher the probability to aggregate several packet drops within the same RTT to a single loss event. In this operating regime, the increase in the number of loss events is no longer proportional to the increase in the number of packets the loss events are sampled over and the measured loss event rate will decrease. A TFMCC flow sending small packets at a high rate will

therefore achieve a higher throughput than a TFMCC flow with large packets or a TCP flow, since the size of the loss intervals is measured in terms of packets.

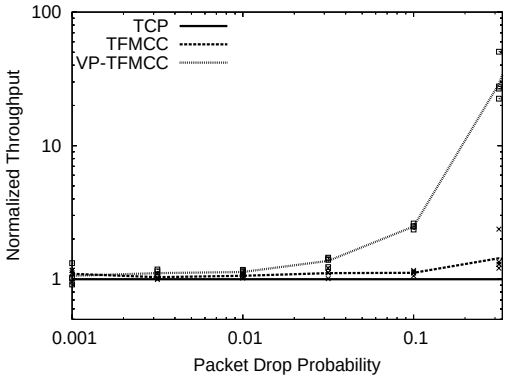


Figure 7.5: TFMCC with large packet size in the formula and per packet drops

Figure 7.5 shows how the throughput for these types of flow varies for different packet drop rates. Here, a fixed RTT of 100 ms[4], a packet size of 1000 bytes for the large TFMCC and the TCP flow and a packet size of 100 bytes for small TFMCC flows (VP-TFMCC) were used. The TCP and the large TFMCC flow achieve approximately the same rate under similar network conditions, with TFMCC being slightly too aggressive in the regime of very high packet drop rates. In contrast, the VP-TFMCC flow is much more aggressive even in the regime of moderate packet drop rates between 1% and 10%, and for drop rates above 50% achieves more than 100 times the TCP throuhgput. If these flows were to compete against each other at the bottleneck, the TFMCC flow sending small packets would lock out the other flows. The higher the packet drop rate, the more pronounced is the bias in favor of small packet sizes.

Consider a rate controlled flow sending N packets of size S per round trip-time. Let θ_n denote the n-th loss event interval measured by this flow. Assume that θ_n is defined as in TFRC [56] and that packets are dropped according to a Bernoulli packet loss process. Let p be the packet drop probability. Then, the random variable θ_n representing the loss interval has the following probability law:

$$P(\theta_n = m) \quad = \quad \begin{cases} 0 & \text{if } m < N \\ (1-p)^i p & \text{if } m = N + i, i \in I\!N \end{cases} \tag{7.3}$$

It should be pointed out that $\theta_n \geq N$ because packet losses within the same round-trip time get aggregated into the same loss event. The expected value of θ_n is given by:

$$E(\theta_n) = N - 1 + \frac{1}{p} \tag{7.4}$$

[4]Note that this effect is independent of the specific RTT value.

A derivation of this result is given in Appendix G.1. From a practical point of view, $E(\theta_n)$ is composed of the sum of (1) the $N-1$ packets within the LIP and (2) the average number of packets between the end of the LIP and the next packet loss (including the lost packet), given by $1/p$.

Equation (7.4) shows that, if packet dropping rate is not a function of packet size, i.e., if p remains the same, the higher the number of packets per round-trip time, the larger the expected loss interval and hence, the smaller the loss event rate ($= 1/E(\theta_n)$). Consider now a flow sending $N\,\eta$ ($\eta \geq 1$) smaller packets of size $s = S/\eta$ per round-trip time. If the loss measurement process remains unchanged, a flow sending many more smaller packets will overestimate the average loss interval (i.e., $E(\theta_n) = N\eta - 1 + \frac{1}{p}$) compared to a flow sending large packets, leading to an unfair distribution of bandwidth.

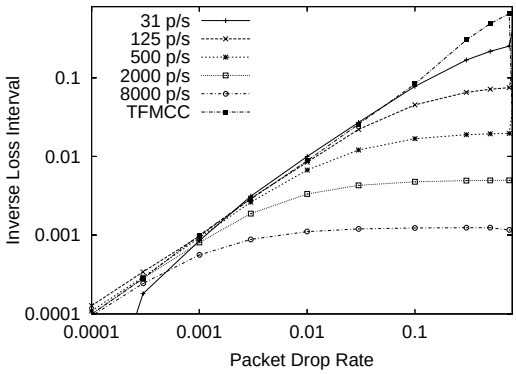


Figure 7.6: Loss Event Rate measured by different CBR flows

The bias introduced in the estimation of the loss event rate is illustrated in Figure 7.6. The figure shows the loss event rate measured by different constant bit rate (CBR) flows and by a TFMCC flow for different packet drop rates, where all flows send packets of 1000 bytes. The TFMCC flow constantly adjusts its sending rate to the measured loss event rate, whereas the CBR flows maintain a constant packet rate. Consequently, when packet drop rates are high, the loss event rate measured by the CBR flows depends almost exclusively on the number of packets per RTT, which is fixed. The relationship of packet loss rate and loss event rate shown in Figure 7.6 coincides perfectly with the results given by Equation (7.4).

Sender-Based Modifications (Unbiasing)

A first naive method to remove this bias in favor of small flows would thus consist in measuring the loss interval as before, computing the bias and then removing this bias from the measured loss interval. A connection sending $N\,\eta$ packets of size s per round-

trip time measures an average loss interval of

$$\begin{aligned} E(\theta_n) &= N\eta - 1 + \frac{1}{p} \\ &= [N - 1 + \frac{1}{p}] + (\eta - 1)N \end{aligned}$$

instead of $E(\theta_n) = N - 1 + \frac{1}{p}$. The simplest way to obtain a correct measure of the loss interval is to substract $(\eta - 1)N$ from the measured interval. Accordingly, if the receiver reports the measured loss event rate to the sender, the sender can adjust the measurement before calculating a TCP-friendly sending rate.[5]

As shown in Section 7.6, this method of unbiasing works well for a simple Bernoulli loss process but its limitations quickly appear under more complex loss conditions. Furthermore, the correction critically depends on the use of correct values for η and N, which is very difficult since N (and possibly also η) varies over time. Directly using the current values can lead to very large variations in the corrected loss interval sizes.[6] However, properly smoothing the values of N and η used for the correction without introducing additional artifacts is equally difficult.

Receiver-Based Modifications

As mentioned in the previous section, modifying the measured loss event rate is very challenging when network loss conditions are not perfectly stable. Indeed, in order to completely remove the bias introduced in the measure of the loss event rate over a certain time span, it is necessary to exactly follow the dynamics of the loss process during this time interval. Since the sender does not have access to this information, it will not be able to accurately estimate this bias under dynamic loss conditions.

A receiver-based approach, on the other hand, allows to closely follow the dynamics of the loss process. It is therefore preferable to modify the loss measurement process itself, instead of trying to modify the outcome of the latter. We will show in the following sections that modifying the loss measurement process is a much more robust approach than a sender-based one.

7.5 Modifications to the Loss Measurement Mechanism

In the following, we will present three methods to counter the effect of increased aggregation of packet losses. All three mechanisms work in packet mode (i.e., when the packet

[5] Since both the sender and the receiver know η and N, the correction can be carried out by either of them but as we will see later, other modifications to the receiver lead to much better results. Therefore, removing the bias only makes sense at the sender, if the receiver cannot be modified.

[6] It is even possible to obtain loss intervals of negative size, since N can change significantly over the course of a loss interval.

drop probability is independent of the packet size) and one of the mechanisms is also suitable for byte mode (with some modifications).

7.5.1 Gateway in Packet Mode

The aim of the modifications to the loss measurement mechanisms presented in this section is to estimate $E(\theta_n) = N - 1 + \frac{1}{p}$, given that the flow actually sends $N\eta$ packets per RTT. For gateways in packet mode, the packet drop probability p is the same for flows sending large packets and flows sending small packets.

Under the assumption of Bernoulli losses, we will show analytically that with these modified loss measurement algorithms, a flow sending $N\,\eta$ ($\eta \geq 1$) smaller packets of size $s = S/\eta$ per round-trip time will estimate the same average loss interval as a flow with packets of size S, no matter the value of η.

Virtual Packets

The main idea of a loss measurement mechanism based on virtual packets is to combine small packets of size s to packets of size S. Whenever a receiver receives S or more bytes (in packets of size s), it records the arrival of a virtual packet. Similarly, a virtual packet is marked as lost, when the amount of bytes lost exceeds S. Figure 7.7(b) gives an overview of how virtual packets are formed.

For this method, it is necessary to modify the definition of loss event and loss interval. The duration of the LIP remains unchanged (i.e., on average it comprises $N\eta - \eta$ packets of size s).

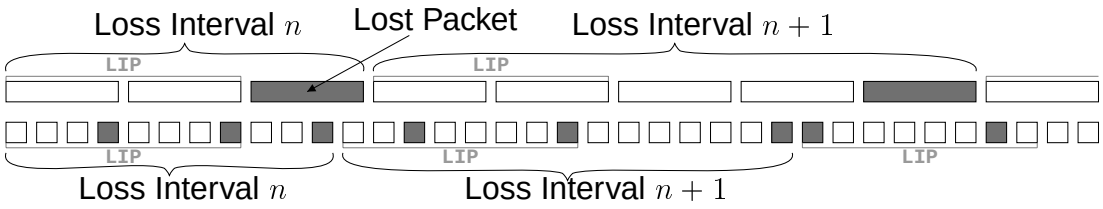
Definition 7.5.1 *A packet loss constitutes a* loss event, *if (a) the LIP following the last loss event ended and (b) at least S bytes were lost since the end of the LIP.*

Definition 7.5.2 *A* loss interval *is measured as the number of virtual packets received between two successive loss events, including the lost packet that ends the loss interval.*

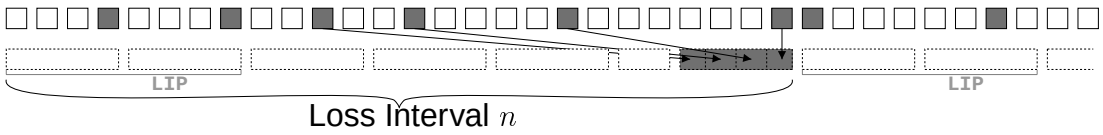
The above mechanism aims at reducing the number of packets that form a loss interval by "normalizing" the number packets through the concept of virtual packets. Note that the size of a loss interval need not be whole-numbered.

From Definition 7.5.1, a flow sending packets of size s experiences a loss event as soon as η packets are lost since the end of the last LIP. From Definition 7.5.2, θ_n is defined over the set $\{N + \frac{i}{\eta}, i \in I\!N_0\}$ and its probability mass function is given by:

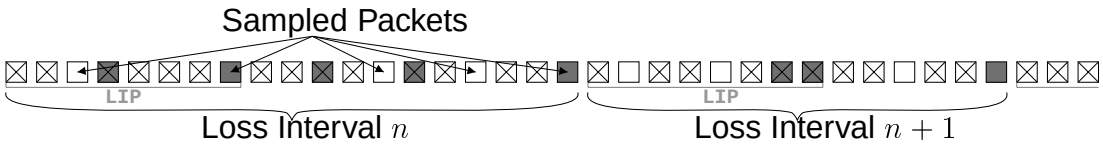
$$P(\theta_n = m) \;=\; \begin{cases} 0 & \text{if } m < N \\ \binom{i+\eta-1}{\eta-1}(1-p)^i p^\eta & \text{if } m = N + \dfrac{i}{\eta}\ ,\ i \in I\!N \end{cases} \tag{7.5}$$



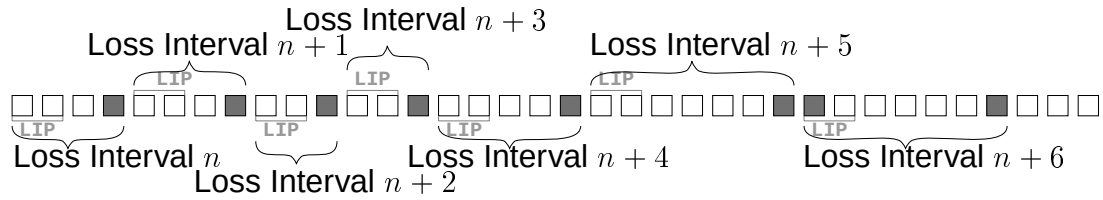
(a) Unmodified Measurement Mechanism



(b) Virtual Packets



(c) Random Sampling



(d) LIP Scaling

Figure 7.7: Schematic illustration of the algorithms

We show in Appendix G.1.1 that the expected loss interval resulting from Equation (7.5) is the same as the one defined in Equation (7.4).

Random Sampling

The same effect of normalization can be achieved in a different way. Instead of aggregating small packets into large packets, the loss interval can be normalized by preventing excess packets (and excess packet losses) to be included in the loss measurement process. Consider a flow that sends packets of size s at the same bitrate and thus at a higher packet rate than a flow with packets of size S would. Upon packet arrival (or the detection of a packet loss), the receiver performs a random experiment that succeeds with the probability $\frac{s}{S}$. Only when the random experiment succeeds is the packet arrival or packet loss taken into account in the loss measurement process. For this mechanism, the same notion of LIP, loss event and loss interval as the one for the original loss measurement process

can be used. An example of random sampling is given in Figure 7.7(c).

In practice, each packet of size $s = S/\eta$ has a probability $p_\eta = 1/\eta$ to be sampled by the loss measurement process. Thus, a loss interval θ_n of size m, ($m \in I\!N$) is measured if:

- j packets are sampled within the $N\,\eta - \eta$ packets following a loss, with $0 \leq j \leq \min(m-1, N\eta - \eta)$
- the following $m - j - 1$ sampled packets are not lost
- the last sampled packet is lost

which translates to:

$$P(\theta_n = m) = \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} {p_\eta}^j (1-p_\eta)^{N\eta-\eta-j} \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} {p_\eta}^{m-j-1} (1-p_\eta)^{k-(m-j-1)} (1-p)^{m-j-1} p_\eta p \tag{7.6}$$

and leads to the expected loss interval given in Equation (7.4), as shown in Appendix G.1.2.

LIP Scaling

A third method to reduce the number of packets contained in a loss interval is to reduce the duration of the LIP. When the loss insensitive period is scaled in proportion to the factor $\frac{s}{S}$, then the LIP of a flow sending small packets should on average contain the same number of packets as the LIP of a flow sending larger packets.

In particular, a flow sending N packets of size S per round-trip time and a flow sending $N\eta$ packets of size s per round-trip time both send $N - 1$ packets per LIP, if the LIP of the small flow is η times smaller than the one of the large flow. As a consequence, both flows will calculate roughly the same loss event rate, given that they experience the same packet drop rate. The LIP scaling mechanism is illustrated in Figure 7.7(d).

A side effect of the LIP scaling method is that it changes the responsiveness of the loss measurement process. Reducing the LIP actually increases the responsiveness of the flow, which can give rise to undesirable oscillations in the measure of the loss interval. To reduce this effect and make sure that the network conditions are measured over the same timescale as a flow sending large packets, we increase the size of the loss history proportionally to the factor $\frac{S}{s}$. This way, the loss history should comprise roughly the same time interval as the one of flows with large packets, while the loss event rate is calculated over many more samples. The impact of these changes of timescale on the dynamics of the algorithm will be analyzed in more detail in the simulation section.

7.5.2 RED in Byte Mode

Consider now a packet loss process where the packet drop probability depends on the packet size as follows:

$$p_S = \eta\, p_s \tag{7.7}$$

where p_S and p_s respectively denote the probabilities for a packet of size S and s ($= S/\eta$) to be dropped.

With RED gateways operating in byte mode the probability to drop a packet of size S is η times larger than the probability to drop a packet of size s. As a consequence, the average interval (in terms of packets) between two packet losses is roughly η times smaller for the flow sending packets of size S than for the flow sending small packets of size s. The average number of bytes between two packet losses is roughly the same for all the flows, independently of the packet size.

For the virtual packet algorithm operating in byte mode, a loss event is declared as soon as a packet of s bytes was lost after the LIP. Thus, we have to relax Definition 7.5.1 and introduce a new definition of a loss event:

Definition 7.5.3 *A packet loss constitutes a* loss event*, if the LIP following the last loss event ended.*

The definition of a loss interval remains the same as the one given in Definition 7.5.2. The random variable θ_n is thus defined as follows:

$$P(\theta_n = m) \;=\; \begin{cases} 0 & \text{if } m < N \\ (1-p_s)^{i-1} p_s & \text{if } m = N - 1 + \dfrac{i}{\eta},\ i \in I\!N \end{cases} \tag{7.8}$$

and the expected loss event interval is given by:

$$E(\theta_n) = N - 1 + \frac{1}{p_S} \tag{7.9}$$

as shown in Appendix G.2.1. It is possible to construct an algorithm for RED in byte mode based on random sampling. This amounts to a random sampling of arrived data packets, whereas lost data packet always need to be taken into account. While the virtual packets and the random sampling mechanism for packet mode are both valid mechanisms of their own with slightly different properties, random sampling in byte mode merely ignores information that is available to the receiver. The number of packets between packet drops is estimated instead of being directly measured as in the virtual packet approach. Generally, we expect random sampling in byte mode to be inferior to virtual packets in byte mode. For this reason we only discuss the approach briefly in Appendix G.2.2.

Adjusting LIP scaling to byte mode results in an even less favorable mechanism. Here, the expected number of lost packets within the LIP does not increase with a decrease in packet size and therefore reducing the duration of the LIP would require introducing artificial packet loss in later intervals. We do not recommend to use LIP scaling in combination with a bottleneck in byte mode.

In addition to a linear dependency of the packet drop probability on the packet size, it is conceivable to derive a packet drop probability from a byte loss probability, where a packet is lost when one or more bytes of the packet are lost. The resulting relationship of packet drop probabilities is then given as:

$$p_S = 1 - (1 - p_s)^{\eta}$$

We propose a loss measurement mechanism based on virtual packets that is designed for such a hypothetical packet marking or packet drop scheme in Appendix G.3.

7.6 Simulations

To investigate the behavior of the different loss measurement mechanisms under more realistic settings than the ones used for the mathematical analysis, we resort to network simulation with the ns-2 network simulator [30]. For the simulation topology, we use the well-known dumbbell topology depicted in Figure 7.8.

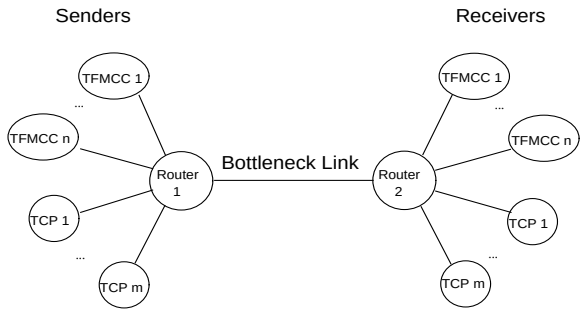


Figure 7.8: Dumbbell topology

The access links to the routers are provisioned with 100 MBit/s, while the bottleneck link between the routers either has a lower capacity, or a loss module is inserted at Router 1 so that the total bandwidth consumed by all flows is well below 100 MBit/s. Unless stated otherwise, we use a propagation delay of 10 ms for the access links and 80 ms for the bottleneck link (in addition to the serialization delay and a possible queueing delay).

When discussing the scenarios, we will denote TFMCC flows which use the same packet size as TCP by TFMCC and TFMCC flows with a different packet size or with a fixed packet rate by VP-TFMCC.

7.6.1 Artificial Channel

Before investigating more complex scenarios where flows interact and compete for resources at a common bottleneck, we will analyze if VP-TFMCC, TFMCC, and TCP flows achieve a similar sending rate when the packet drop rate is independent of the sending rate of the flows. Since the model for TCP (Equation (7.2)) is based on this assumption, equation based congestion control should work best in such an environment.

We use three different loss models for our analysis: Bernoulli loss, Bernoulli loss with a drop rate varying over time, and a Gilbert loss model. The Bernoulli dropper discards each incoming packet with the same packet drop probability. The second loss model provides more variable network conditions where the packet drop probability alternates between high and low. While the average drop probabilities are the same as for the first loss model, the congestion control protocols frequently have to adjust their sending rate, putting more emphasis on the transient behavior of the protocols. In the Gilbert loss model, packet losses are no longer independent but highly correlated. Our time-based model[7] alternates between the no-loss state "0" and the loss state "1" with transition probabilities based on p and q (see Section 2.3 on page 15). The model remains in the current state for a fixed amount of time, τ, after which a random experiment is performed to see if a state change should occur. For the Gilbert model, the average loss rate is $p/(p+q)$ and the average length of a loss burst is $1/q$ time units. To arrive at the same average packet drop probability $\bar{p}$ as in the previous two models, we set $p = a\bar{p}$ and $q = a(1-\bar{p})$. The higher the average packet drop probability the higher the average burst length. The parameter a can be used to modify the burst length while keeping the same average packet drop probability. In the following simulations we use $a = 0.8$ and $\tau = 10$ ms.

With our mechanisms we aim to emulate a TFMCC flow with large packets so the best we can hope for is a throughput similar to the throughput of such a flow. To assess the proposed mechanisms, we therefore normalize the throughput shown in the figures in Section 7.6.1 by dividing by the throughput of a TFMCC flow with large packets. To allow a comparison with TCP, Figure 7.9 depicts the ratio of throughput of plain TFMCC (with large packets) to TCP throughput. This allows as to separate the differences in throughput introduced by using equation based congestion control in general from the ones introduced by using a different packet size.

For the Bernoulli loss model, TFMCC throughput coincides well with TCP throughput and becomes slightly too aggressive in the regime of high loss rates. When the packet drop rate changes over time, TFMCC is a bit more conservative, but in general, equation based

[7] A packet based model would have the disadvantage, that the timescale over which loss bursts occur depends to a large degree on the packet rate. Furthermore, if flows with different packet rates are run concurrently under such conditions, they are no longer independent from each other but would experience very different loss patterns.

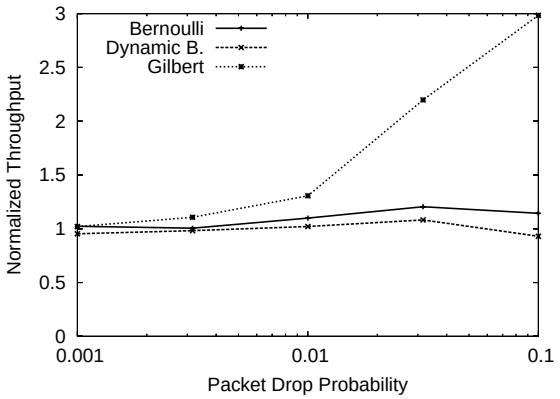


Figure 7.9: Fairness of plain TFMCC with different loss models

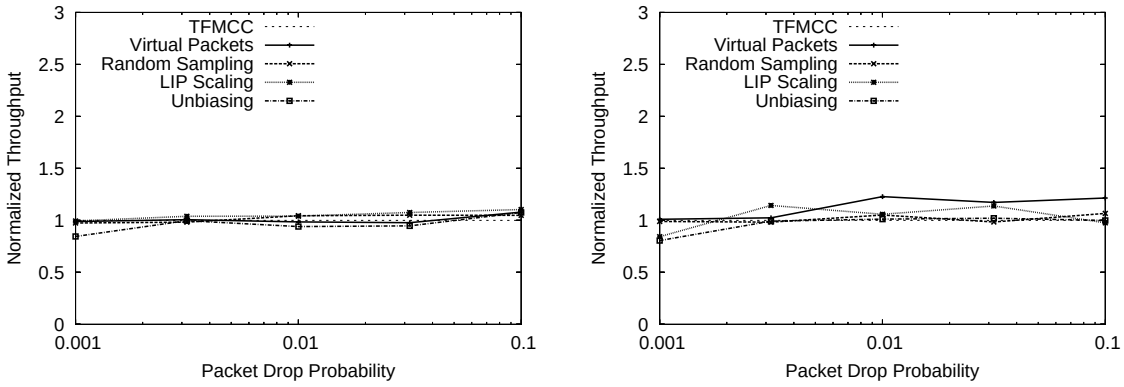
congestion control works very well for both loss models. The results are quite different for the third loss model, where TFMCC is significantly more aggressive than TCP when drop rates are high.

The reason for this discrepancy in TFMCC and TCP throughput results from the correlation of packet drops. Since TCP sends packets back to back, it is likely that a number of them arrive during the loss state of the Gilbert model. For higher loss rates, the congestion window comprises only a few packets and the number of consecutive packet loss is large, leaving too few packets in the pipe to allow fast recovery. Instead of a triple duplicate ACK for one or more segments lost in a congestion window, TCP frequently experiences a timeout; many more than are to be expected with the Bernoulli dropper given the same loss rate. Since the TCP model used for TFMCC is based on different assumptions about the packet loss pattern, TFMCC's sending rate will be too aggressive.

Bernoulli Loss Model

The most basic scenario to investigate is based on a loss module inserted at Router 1, which drops incoming packets with a fixed probability. With a delay for the bottleneck link of 30 ms and 10 ms for the access links, the RTT for all flows is almost constant at 100 ms. Since the achieved sending rates are below the bottleneck bandwidth, no queueing occurs and the queueing strategy has no impact on the simulation. Furthermore, flows running concurrently will not influence each other.

All different mechanisms (i.e., the sender-based unbiasing as well as the three receiver-based loss measurement modifications) result in a throughput very close to the throughput of a TFMCC flow with large packets and thus very close to TCP throughput. When the packet size of the VP-TFMCC flow is set to 100 bytes and the packet rate varies, we obtain a relative throughput of VP-TFMCC as depicted in Figure 7.10(a). Here, the deviation in throughput is nearly always less than 10%. Direct unbiasing is slightly more conservative



(a) VP-TFMCC packet size 100 bytes

(b) VP-TFMCC packet rate 160 packet/second

Figure 7.10: Bernoulli loss

than TFMCC and the other methods are slightly more aggressive, with virtual packets resulting in the throughput most closely resembling TFMCC throughput.

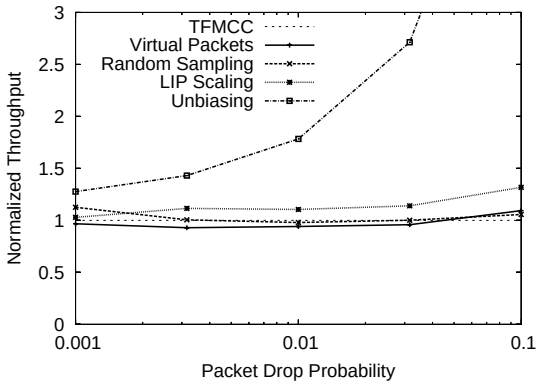
In contrast, when the packet rate is fixed at 160 packets/second and the packet size varies (see Figure 7.10(b)), the virtual packets method has the highest deviation from TFMCC throughput and Unbiasing as well as Random Sampling achieve almost exactly the TFMCC rate. Nevertheless, with less than 20% the difference is relatively small.

As all of the mechanism are based on the assumption of a Bernoulli loss process, these good results are to be expected.

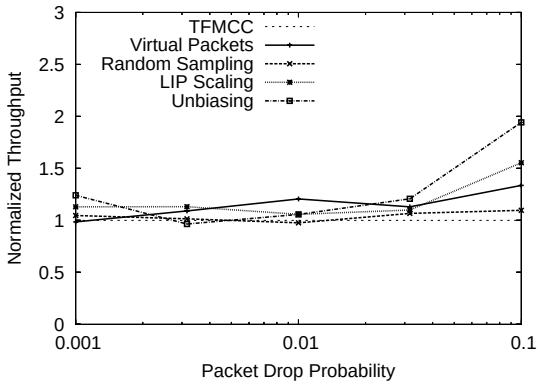
Dynamic Bernoulli Loss Model

To analyze the behavior under more dynamic network conditions we use a Bernoulli packet dropper where the drop rate alternates between 0.5 times the average drop rate and 1.5 times the average drop rate. The packet drop rate changes every 10 seconds (i.e, 24 times over the whole simulation of 250 seconds).

As soon as network conditions become more dynamic the shortcomings of the most simple of the mechanisms, the unbiasing of the loss interval, become obvious. Unbiasing is much more aggressive than TFMCC for both, a fixed packet size and a fixed packet rate, but this effect is much more pronounced for a fixed packet size. For packet drop rates of more than a few percents, the throughput is a multiple of the rate achieved by TCP or TFMCC. The methods of virtual packets and random sampling behave quite well, while LIP scaling is somewhat too aggressive in the case where the packet size is fixed, as shown in Figure 7.11(a). All mechanisms tend to become more aggressive than TFMCC when the packet rate is fixed and the drop probability is high. Under such conditions,



(a) VP-TFMCC packet size 100 bytes



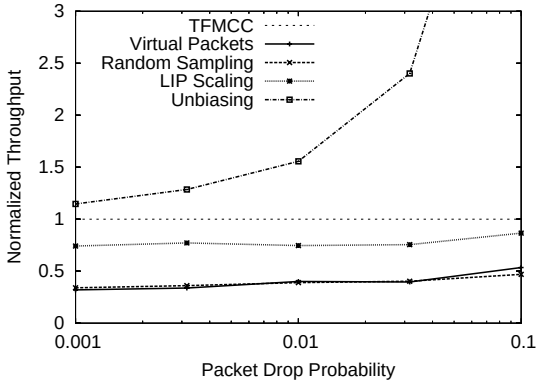
(b) VP-TFMCC packet rate 160 packet/second

Figure 7.11: Dynamic bernoulli loss

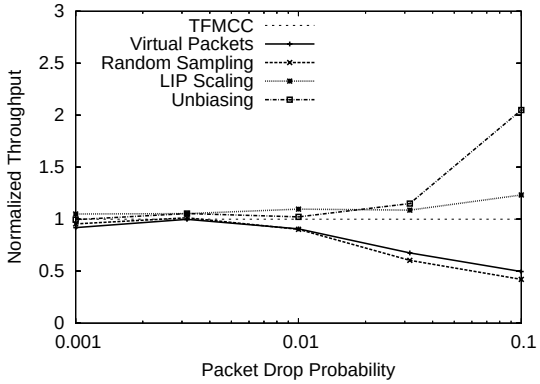
random sampling performs best with only a marginal increase in sending rate compared to plain TFMCC (Figure 7.11(b)).

Gilbert Loss Model

The parameters of the time based Gilbert model specified above ($\tau = 10$ ms and $a = 0.8$) are not intended to closely model network conditions we expect to find in the Internet but are merely used to analyze how the mechanisms perform when the assumption of packet loss independence is not met.



(a) VP-TFMCC packet size 100 bytes



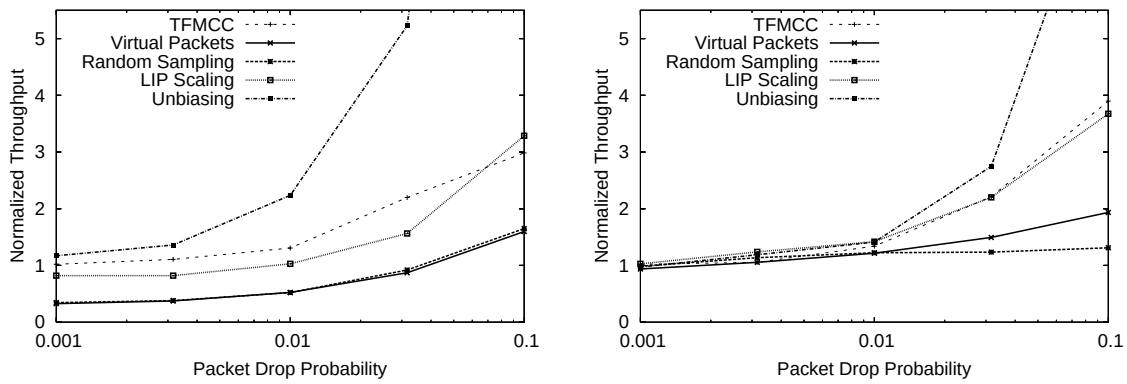
(b) VP-TFMCC packet rate 160 packet/second

Figure 7.12: Gilbert loss model

As far as the throughput of the different mechanisms is concerned, there is a striking

difference to the Bernoulli based experiments. All of the methods are far from fair (for all packet drop rates in case of a fixed packet size and only for higher packet drop rates in case of a fixed packet rate). As in the dynamic Bernoulli experiments, unbiasing is by far the most aggressive scheme, making it unsuitable for all but very simple static network conditions. LIP scaling achieves a somewhat lower sending rate and becomes a bit more aggressive than TFMCC for high loss rates in the fixed packet rate case. Given that TFMCC itself is much more aggressive than TCP under such circumstances (see Figure 7.9), we do not recommend LIP scaling for such network environments.

Virtual packets and random sampling perform alike with a throughput of less than 50% of TFMCC throughput when the interval between packets varies (Figure 7.12(a)) and going from fair to around 50% of TFMCC throughput when the interval between packets is fixed (Figure 7.12(b)). Therefore, these mechanisms achieve a throughput much closer to TCP throughput than that of plain TFMCC, as shown in Figure 7.13.



(a) VP-TFMCC packet size 100 bytes

(b) VP-TFMCC packet rate 160 packet/second

Figure 7.13: Gilbert loss model with throughput relative to TCP

We note that this improvement in fairness is caused by two effects that counterbalance (equation based congestion control in general being too aggressive under such network conditions and the modified loss measurement mechanisms resulting in an overestimation of the loss event rate), not because the mechanisms themselves better model TCP performance. Nevertheless, with these mechanisms we achieve roughly the same performance under normal circumstances and are more conservative than TFMCC under unfavorable network conditions where TFMCC is too aggressive. This is the behavior we would like to see in a modified TFMCC protocol and we therefore recommend the methods virtual packets and random sampling rather than unbiasing and LIP scaling (unless with known favorable network conditions and a high level of statistical multiplexing).

7.6.2 Bandwidth Limited Bottlenecks

After gaining first insight into the performance of the proposed algorithms, in this section we will analyze their performance under more realistic network conditions where flows with small and large packets directly compete at a bandwidth limited bottleneck.

Simulation Setup

In the simulations with bandwidth limited bottlenecks, we use three different parameter settings for the VP-TFMCC flows:

- The packet size is fixed at 100 bytes and the fair bandwidth is 96 KBit/s per flow.
- The packet rate is fixed at 160 packets/s, the maximum packet size is 100 bytes (resulting in a maximum sending rate of 128 KBit/s) and the fair bandwidth is 96 KBit/s per flow.
- The packet rate is fixed at 50 packets/s, the maximum packet size is 200 bytes (resulting in a maximum sending rate of 80 KBit/s) and the fair bandwidth is 64 KBit/s per flow.

The bottleneck capacity is set to the number of flows times their fair bandwidth (i.e., capacity is scaled with the number of flows).

Setting a reasonable queue size for the simulations is not an easy task. Generally, TCP performs better when there is a large amount of buffer space available (so that there is enough space to accommodate TCP's packet bursts), while TFMCC is relatively insensitive to the queue size and therefore outperforms TCP when the queue size is small. Particularly if the queue size is measured in packets, with a potentially large number of small packets in the queue, the queue size available to TCP may vary significantly. When the queue is measured in bytes, a large TCP packet occupies the space of many small VP-TFMCC packets and when TCP sends a burst of packets, it may occupy a large fraction of the queue space. However, due to TFMCC's insensitivity to the available queue size, this has only a relatively small impact.

We chose to set the queue parameters as follows:

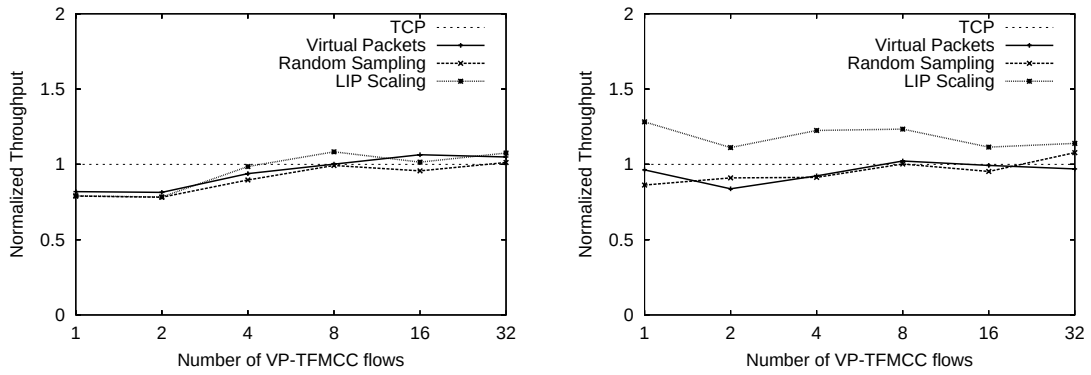
- If the queue is measured in bytes, we set the queue size to twice the bandwidth delay product, assuming a RTT (including buffer delay) of 500 ms.
- If the queue is measured in packets, we set the average packet size to $\frac{2S}{(1.0+S/s)}$ where S is the TCP packet size and s is the size of the VP-TFMCC packets in case of a fixed packet size or the size of a VP-TFMCC packet if the flow were sending at exactly the fair rate in case of a fixed packet rate. The queue size in packets is then set to twice the bandwidth delay product divided by the average packet size.

- For RED queues, we further set the minimum threshold min_{th} to 5% of the queue size, the maximum threshold max_{th} to 50% of the queue size, and the maximum packet drop probability max_p to one drop per 22.5% of the queue size in packets (the average of min_{th} and max_{th}).[8] The $gentle\_$ option of RED is enabled.

All the experiments discussed below were also conducted with half the queue size (i.e., using the equivalent of one bandwidth delay product) with the expected results of a decrease in the level of fairness in favor of TFMCC and VP-TFMCC.

For all the simulations, the same number of VP-TFMCC flows and TCP flows was used (i.e., 1vs1, 2vs2, etc.). The simulation results were averaged over six runs for each parameter setting (together with slight variations in the available bandwidth and the starting times of the flows to provide some degree of randomness).

Packet Mode



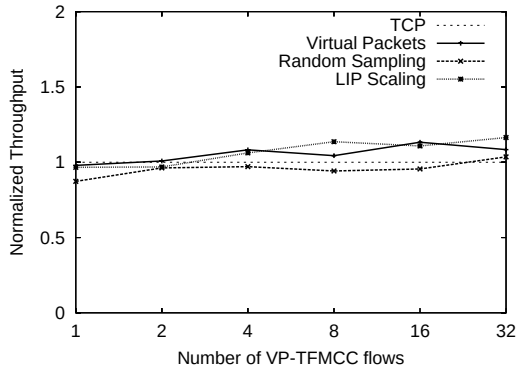
(a) RED queue in packets

(b) DropTail queue in packets

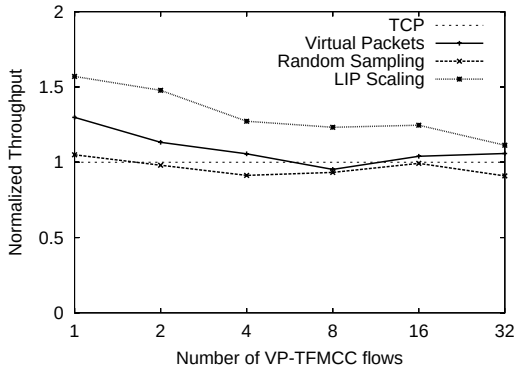
Figure 7.14: VP-TFMCC packet size 100 bytes

In Figures 7.14 to 7.16, we show the throughput of the different VP-TFMCC variants, normalized to TCP throughput, when the decision to drop a packet at the bottleneck is based only on the number of packets and not on the packet size. As is to be expected, the fairness of the VP-TMCC variants improves when RED queuing is used instead of drop-tail queuing. In the simulations, LIP scaling is the most aggressive of the different variants. Particularly when packet loss is correlated as in the drop-tail queue, LIP scaling is significantly too aggressive (as evidenced in the previous simulations with the Gilbert

[8]These are the recommended parameter settings from "More Thoughts on Reference Simulations for Reliable Multicast Congestion Control Schemes", notes from a meeting at Digital Fountain on August 8, 2000 by John Byers, Gavin Horn, Mark Handley, Michael Luby, Will Shaver and Lorenzo Vicisano.

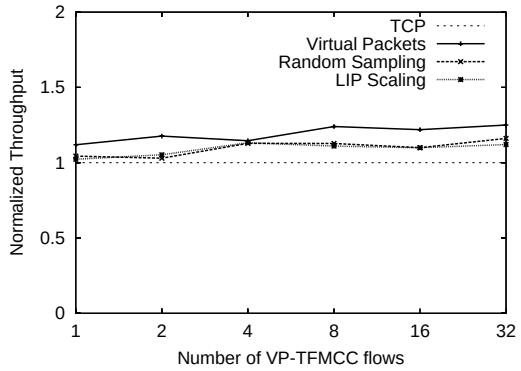


(a) RED queue in packets

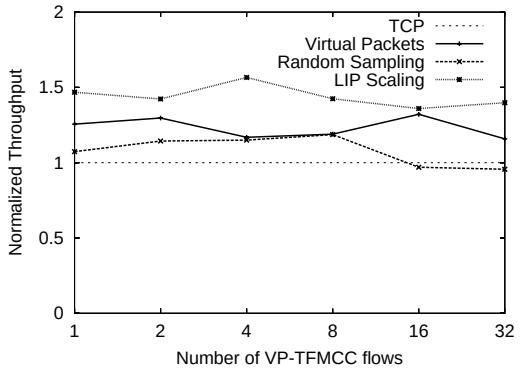


(b) DropTail queue in packets

Figure 7.15: VP-TFMCC packet rate 160 packet/second



(a) RED queue in packets



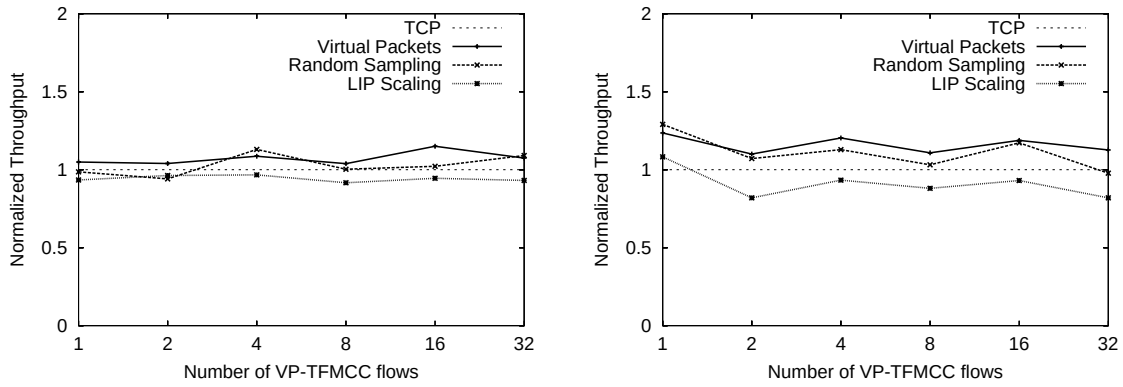
(b) DropTail queue in packets

Figure 7.16: VP-TFMCC packet rate 50 packet/second

loss model). Random sampling and virtual packets perform very similar, with random sampling being somewhat more conservative in most of the simulations.

Surprisingly, in contrast to the other loss measurement variants or plain TFMCC, LIP scaling becomes more aggressive as the buffer size increases. When we compare the simulation results for a bottleneck with drop-tail queue and one bandwidth-delay product worth of buffering to simulations with twice the amount of buffering, random sampling and virtual packets behave like plain TFMCC and become more aggressive with decreasing buffer space. In constrast, LIP scaling is less aggressive (i.e., relatively fair to TCP) with a buffer size of one bandwidth-delay product, but consistently too aggressive for larger buffer sizes (see Figure 7.17). The same effect can be seen with RED queueing,

although on a much smaller scale.



(a) RED queue in packets

(b) DropTail queue in packets

Figure 7.17: VP-TFMCC packet size 100 bytes

The cause for this discrepancy lies in the different timescales over which the loss measurement mechanisms operate. When the buffer size is large, the TCP flows in the simulation tend to synchronize so that the buffer occupancy oscillates and periods of no packet loss alternate with periods of very high packet loss. We can observe two effects that counterbalance:

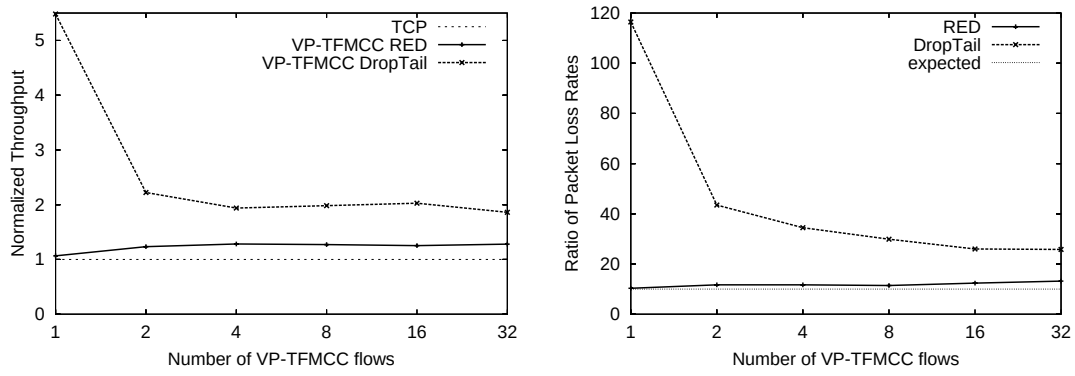
- For random sampling and virtual packets, the number of packets within the loss interval that comprises the non-congested period is comparable to that of TCP. During the same time interval, LIP scaling will experience a loss interval that is η times larger.
- Since TCP backs off within the time frame of one RTT, virtual packets and random sampling will experience (at most) one loss event per congested period. During the same time, a flow with LIP scaling may experience up to η loss events, given a sufficiently high packet drop rate.

Under such circumstances, with LIP scaling the size of the loss intervals is no longer independent of η and although the two effects tend to counterbalance, they will ususally not cancel each other out. This phenomenon can be observed whenever the loss process is time-driven instead of packet-driven.

Byte Mode

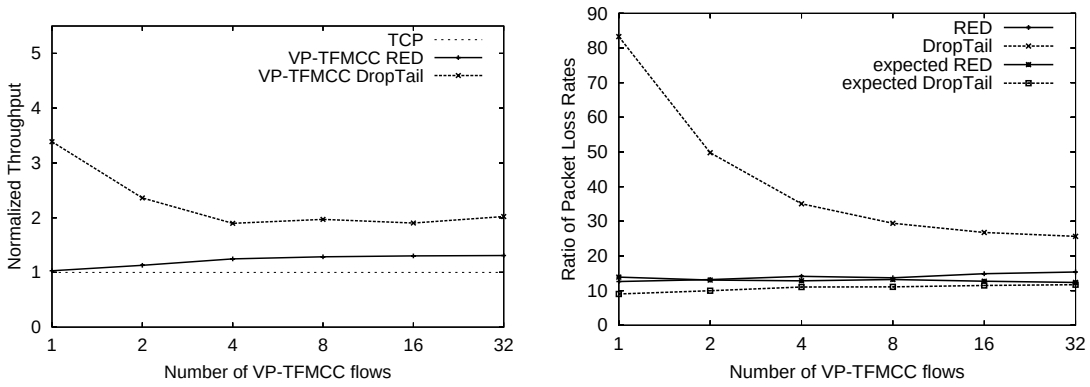
Only the virtual packet method also works in an environment where the packet drop probability is proportional to the packet size. In Figures 7.18 to 7.20, we show how the virtual

packet method performs with RED gateways in byte mode as well as drop-tail gateways with a queue measured in bytes. As for the previous simulations, we depict VP-TFMCC throughput normalized to TCP throughput for both types of gateways. Furthermore, we show the ratio packet drop rates experienced by flows with small and large packets (1) as measured during the simulations and (2) as would be expected if the drop probability were proportional to the packet size.



(a) Normalized Throughput (b) Ratio of Packet Loss Rates (p_S/p_s)

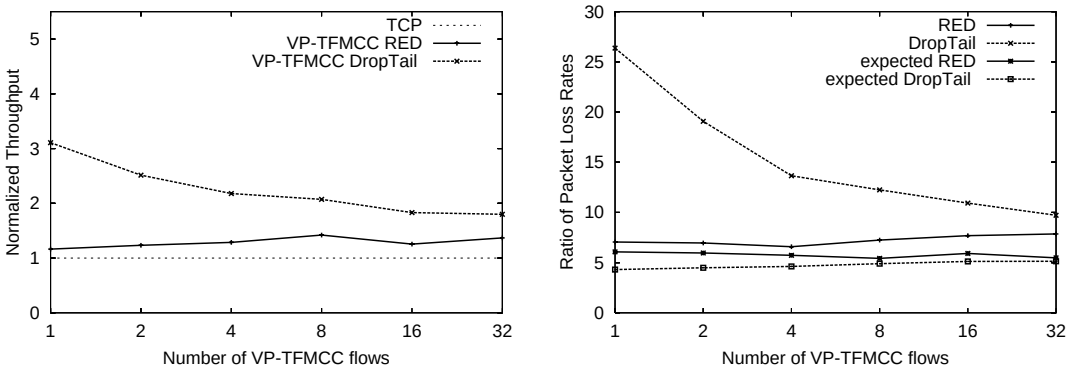
Figure 7.18: Fairness byte mode (VP-TFMCC packet size 100 bytes)



(a) Normalized Throughput (b) Ratio of Packet Loss Rates (p_S/p_s)

Figure 7.19: Fairness byte mode (VP-TFMCC packet rate 160 packet/second)

As in the previous experiments, we observe that fairness towards TCP is significantly higher with RED queues in byte mode than with drop-tail queues, but here the discrepancy is much larger. While with RED queues the packet drop probability is explicitly set proportional to the packet size, for drop-tail queues the ratio of packet drop probabilities for flows with large and small packets depends on the distribution of queue occupancy.



(a) Normalized Throughput

(b) Ratio of Packet Loss Rates (p_S/p_s)

Figure 7.20: Fairness byte mode (VP-TFMCC packet rate 50 packet/second)

Only if the probability for a small packet to fit into the queue is a factor of η larger than the probability for a large packet to fit in, the primary assumption of proportional packet drop rates made in the design of the algorithm is met.

As can be seen from the graphs depicting the ratio of packet drop probabilities, particularly for low levels of statistical multiplexing small packets have an overproportionally higher probability of fitting into the drop-tail queue. Consequently, VP-TFMCC achieves a throughput of roughly twice the TCP throughput (except for very low levels of statistical multiplexing where this ratio is even worse). In contrast, with RED in byte mode a high level of fairness is achieved. While VP-TFMCC is not exactly sending at the same rate as TCP, at around 25% the deviation is comparable to the packet mode case.

With a good model for the dependency of packet drop probability and packet size with drop-tail queues in bytes, the virtual packet method can be adjusted so that VP-TFMCC flows and TCP flows share bandwidth in a fair manner even with such queues. To develop such a model is left for future work.

Both, RED in byte mode and the virtual packets mechanism partly compensate for the bias against flows with small packets. Instead of having part of the removal of the bias in the RED mechanism and the other part in the loss measurement mechanism, RED's byte mode can be altered so that a fair sharing of bandwidth of flows with different packet sizes is achieved without any modifications to the loss measurement process.

7.7 Modifications to the Byte Mode of RED

The advantage of modifying RED instead of the loss measurement mechanisms is that flows need no longer to know what type of bottleneck they are dealing with. If the modifications in the router are implemented in a way that they come into effect when bandwidth

is the limiting factor, but packets are dropped on a per packet basis when the packet rate is the limiting factor, congestion controlled flows will converge to a sending rate that is fair with respect to their resource usage at the bottleneck.

The authors of [33] have done a first step in this direction with the introduction of the SQR dependency of RED packet drop rates. However, this is not sufficient for two reasons: first, because the dependency of TCP's throughput to the loss event rate is not exactly a square-root, but follows a more complex function (PFTK); and second, because there is a discrepancy between the packet drop rate and the loss event rate (as explained in the previous sections).

Based on these observations, it is possible to derive a new dropping strategy for RED in byte mode which ensures that congestion controlled flows with the same round-trip time converge to the same sending rate independently of the packet size.

Let S be defined as the mean packet size (as configured in the RED gateway) and p_S be the packet drop probability computed at the gateway for this mean packet size. Now consider two rate controlled flows sending packets of different size. The first flow sends N_S packets of size S per round-trip time and the second one sends N_s packets of size $s = S/\eta$ per round-trip time.

To ensure fairness between flows sending packets of different sizes, we must drop packets based on their size in a way that all the flows achieve the same throughput (in bytes/sec). The problem can thus be expressed as follows: for any packet size s, find the corresponding drop probability p_s such that:

$$N_s = \eta\, N_S \tag{7.10}$$

Under the Bernoulli loss assumption, results from Section 7.4.2 allow to express Equation (7.2) as a function of the packet loss rate (as opposed to the loss event rate). Consequently, N_S can be expressed as follows:

$$N_S = \frac{1}{\sqrt{\frac{2}{3(N_S-1+\frac{1}{p_S})}} + \left(12\sqrt{\frac{3}{8(N_S-1+\frac{1}{p_S})}}\right)\frac{1}{N_S-1+\frac{1}{p_S}}\left(1+32\left(\frac{1}{N_S-1+\frac{1}{p_S}}\right)^2\right)} \tag{7.11}$$

From Equation (7.11), one can see that N_S is only a function of p_S. We can therefore write $N_S = f_{PFTK}(p_S)$, where the function f_{PFTK} is obtained by solving Equation (7.11) for N_S. Finally, Equation (7.10) can be written as follows:

$$f_{PFTK}(p_s) = \eta\, f_{PFTK}(p_S) \tag{7.12}$$

For any given p_S, p_s can therefore be computed by solving Equation (7.12).

We are aware that to really assess the benefits and limitations of such a modification

to RED, it is necessary to implement the mechanism and run extensive simulations. Yet, it is possible to gain a first insight into the idea in a relatively simple way. For given packet drop rates for a flow sending large packets, we can numerically solve the above equations to obtain the corresponding packet drop rates for a flow sending small packets. If we then use the different drop probabilities for flows of different sizes, we have a first approximation of the behavior of such a modified RED gateway.

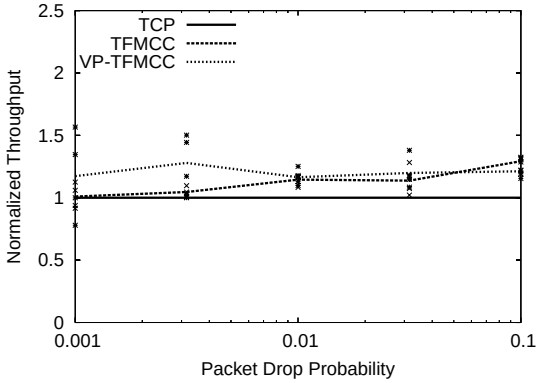


Figure 7.21: Unmodified TFMCC with 100 byte packets

The simulation results for a Bernoulli dropper with packet drop probabilities as defined above are given in Figure 7.21. Both TFMCC flows use the unmodified loss measurement mechanism and still achieve a similar (TCP-friendly) throughput for a wide range of different packet drop rates. With approximately 20%, the deviation from TCP throughput is no worse than that in the simulation results with the modified loss measurement mechanisms. Nevertheless, simulations with a real RED implementation instead of a modified Bernoulli dropper and much more dynamic network conditions will be necessary to really analyze the characteristics of such an approach.

7.8 Conclusions

In this chapter we analyzed the interaction of flows with variable packet sizes with various queuing schemes. We presented three methods to allow flows sending at a packet rate different from that of TCP to estimate a TCP friendly rate. Without these modifications, the loss event rate measured by equation based congestion control mechanisms depends to a large degree on the packet frequency, resulting in too aggressive protocol behavior if flows require a higher packet rate than that of TCP (but reduce their packet size to achieve the same throughput). Two of the proposed mechanisms, random sampling and virtual packets, perform well over a wide range of different network conditions and under unfavorable conditions behave rather less aggressively than TFMCC, which is the desired behavior for modifications to the TFMCC protocol. While random sampling may behave

more stable in an environment where packet drops are highly correlated, virtual packets, also works in case the packet drop probability is proportional to the packet size, as in RED in byte mode. We expect both mechanisms to behave well in real world environments.

Chapter 8

Conclusions

In this dissertation, we studied different problems related to delay issues in the transport of interactive audio over IP networks.

- In Chapter 4, we presented a first adaptive error control scheme for audio which is *delay aware*, namely, which incorporates the impact on the end-to-end delay in the choice of the FEC. This first delay aware scheme is based on the assumption that the playout delay is equal to the delay that would be used if FEC were absent plus an additional delay to be able to use FEC (as in *rat* and *freephone* [6, 2]). Under this assumption, we showed by simulation that the delay aware scheme does avoid that a source waste delay using FEC when it is not necessary.

- In Chapter 5, we designed a method for joint control of delay-aware FEC and playout for interactive audio applications over the Internet. We took a utility function approach and introduced a set of utility functions that are adapted from the E-Model. These functions encompass the influence of both distortion (introduced by losses and compression) and mouth-to-ear delay on perceived quality. We have shown that, in cases where delay matters (i.e. around a threshold effect), there is real benefit in using the joint method (it performs better than any combination of previously published FEC and playout adjustment scheme). We have also shown that the improvement brought by delay aware FEC cannot be obtained if the delay aware FEC control is simply piggybacked onto existing adaptive playout control methods (as it is done in Chapter 4); in contrast, it should be incorporated in a complete joint optimization of both FEC and playout. We have implemented our method in ns2.

- In Chapter 6, we evaluated the benefit, for an audio source, to use a non-elevated service. We showed by simulation that the use of such a service can lead to quality improvements, but that the choice of service depends on network conditions (network load and network topology) and on the importance that users attach to

delay. This observation lead us to propose an Adaptive Color Choosing (ACC) algorithm that allows audio sources to choose in real-time the service providing to the highest audio quality. We showed by simulation that sources using this algorithm successfully estimate the best service to be used and are able to switch rapidly from one service to the other when it is appropriate. The ACC thus allows an audio source to make an optimal use of Non elevated services. Finally, using the optimal error/delay and service (color) control scheme, we showed that interactive audio sources would really benefit from using a non-elevated service compared to a single class Best Effort service in cases where queuing delay is important. This confirms that non-elevated services are good candidates for the deployment of low complexity differentiated service, possibly at the edge of the network.

- In Chapter 7, we analyzed the interaction of flows with variable packet sizes, such as VoIP flows, with various queuing schemes. We presented three methods to allow flows sending at a packet rate different from that of TCP, to estimate a TCP friendly rate. Without these modifications, the loss event rate measured by equation based congestion control mechanisms depends to a large degree on the packet frequency, resulting in too aggressive protocol behavior if flows require a higher packet rate than that of TCP (but reduce their packet size to achieve the same throughput). Two of the proposed mechanisms, random sampling and virtual packets, perform well over a wide range of different network conditions and under unfavorable conditions behave rather less aggressive than TFRC, which is the desired behavior for modifications to the TFRC protocol. While random sampling may behave more stable in an environment where packet drops are highly correlated, virtual packets, also works in case the packet drop probability depends on the packet size, as in RED in byte mode. We expect both mechanisms to behave well in real world environments.

Future Work

- *More Sophisticated Channel Model* : The Joint FEC and Playout Adjustment scheme proposed in Chapter 5 uses a channel model for both delay and loss that is fairly simplistic. In particular, it can be argued that the hypothesis on loss and delay independence is not valid for a network with drop-tail queues. While we do not expect the use of this simplistic model to have a large impact on the performance of the control scheme, further work will address whether there is any benefit in using more sophisticated models. In particular, a study based on real measurements in the Internet should be conducted to determine the 1) the importance of the error that is introduced when using a simple channel model, and 2) whether another (tractable)

channel model could improve the performance of the algorithm.

- *Implementation in a real audio tool* : The logical continuation of this thesis is to implement the joint FEC and playout adjustment scheme in a real audio software (such as the Robust Audio Tool [6]).

- *Dynamics of ABE* : we showed in Chapter 6 that audio sources benefit from using the ABE service, but that the choice of service depends on network conditions (load and topology) and on the importance that users attach to delay. In our simulations, only 25% of the connections were adaptive and could switch from one service to the other, which means that there was always background traffic using both classes of service. The next question that emerges is the following: how would ABE behave if all sources could switch from one service to the other? How will the system evolve if all sources try to maximize the same (or different) objective functions?

- *Packet Limited versus Rate Limited Bottlenecks* : To be able to deploy the mechanisms proposed in Chapter 7, it is important to know what types of bottlenecks are to be expected in the network. To gain an insight into which of the different mechanisms is appropriate for the Internet, measurements that analyze the relationship of packet drop rates and packet sizes are necessary. The type of bottleneck (and therefore the aforementioned relationship) is likely to differ depending on where in the network the bottleneck is located (i.e., in the backbone or close to the edge). Ultimately, when reasonably accurate information about the characteristics of bottlenecks is available, the performance of the proposed mechanisms should be tested in a real-world environment.

- *Virtual Packets with Drop-tail queues* : The Virtual Packet algorithm for byte mode developed in Chapter 7 works well together with RED but fails if the bottleneck is a drop-tail queue measured in bytes. With a significantly accurate model for the drop rates caused by such a queue, the virtual packet algorithm can be modified to also work in such an environment.

Appendix A

Voice over IP Measurements

A.1 Introduction

In this chapter, we use active and passive traffic measurements to identify the issues involved in the deployment of a voice service over a tier-1 IP backbone network. Our findings indicate that no specific handling of voice packets (i.e. QoS differentiation) is needed in the current backbone but new protocols and mechanisms need to be introduced to provide a better protection against link failures. We discover that link failures may be followed by long periods of routing instability, during which packets can be dropped because forwarded along invalid paths. We also identify the need for a new family of quality of service mechanisms based on fast protection of traffic and high availability of the service rather than performance in terms of delay and loss.

The chapter is structured as follows. Section A.2 briefly presents some related work, while Section A.3 provides detailed information on the measurement approaches followed in this study. Section A.4 describes the model used to assess the subjective quality of voice calls from transport level measurable quantities. In Section A.5 we finally discuss our findings, while Section A.6 presents some concluding remarks.

A.2 Related work

As mentionned in Sections 2.3 and 3.2, past literature on end-to-end Internet measurements has often focused on the study of network loss patterns and delay characteristics [19, 25, 71, 112, 91]. Moreover, all these studies were based on round-trip delay measurements.

While information about delay and losses can give valuable insights about the quality of VoIP, they do not characterize the actual subjective quality experienced by VoIP users. In [35], Cole et al. propose a method for monitoring the quality of VoIP applications based upon a reduction of the E-model [12] to measurable transport level quantities (such

as delay and losses).

Markopoulou et al. [77] use subjective quality measures (also based on the E-model) to assess the ability of Internet backbones to support voice communications. That work uses a collection of GPS synchronized packet traces. Their results indicate that some backbones are able to provide toll quality VoIP, today. In addition, they report that even good paths exhibit occasional long loss periods that could be attributed to routing changes. However, they do not investigate the causes of network failures neither the impact they have on the voice traffic.

A.3 Measurements

In this section we describe the two measurement approaches used in our study, i.e. the passive measurement system deployed in the Sprint IP backbone network and the active measurement system that uses probe packets to study routing protocols stability and link failures.

A.3.1 Passive measurements

The infrastructure developed to monitor the Sprint IP backbone consists of passive monitoring systems that collect packet traces on more than 30 links located in three POPs of the network. Details on the passive monitoring infrastructure can be found in [59].

In this study, we use traces collected from various OC-12 intra-POP links on July 24th, 2001, September 5th, 2001 and November 8th, 2001. A packet trace contains the first 44 bytes of every IP packet that traverses the monitored link. Every packet record is also timestamped using a GPS reference signal to synchronize timing information on different systems [80].

We use the technique described in [90] to compute one-way delays across the Sprint backbone. The basic idea behind that technique is to identify those packets that enter the Sprint backbone in one of the monitored POPs and leave the network in another one. Once such packets are identified computing the delays simply requires to compute the difference between the recorded timestamps.

A.3.2 Active measurements

Passive measurements provide valuable information about network characteristics, but the data collected depend on the traffic generated by other parties, which is completely out of our control. Moreover, given that we do not monitor all the links of the backbone network, we are not able to measure jitter or loss rates through simple passive monitoring (packets may leave the network through not monitored links) [90]. Therefore, our passive

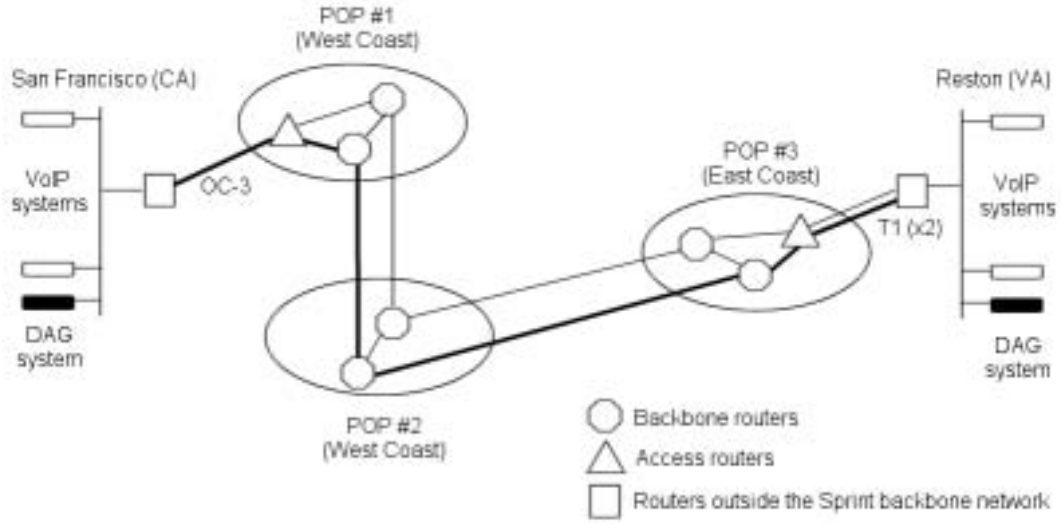


Figure A.1: Topology of the active measurement systems (the thick lines indicate the primary path)

measurements alone cannot provide results on the quality of the voice calls. These are the motivations behind the use of active measurements to complement the passive ones. In an active measurement environment we can perfectly control the amount and the characteristics of the traffic that we inject in the network and thus draw precise conclusions about the impact of the network on the monitored traffic.

Measurement infrastructure

We deployed active measurement systems in two locations of the U.S. (Reston, VA and San Francisco, CA) well connected to the Sprint backbone, i.e. just one router away from the backbone network. Figure A.1 shows the architecture of the testbed and the way the sites are connected through the Sprint network (the thick lines indicate the path followed by our traffic). Note that each access router in a POP is connected to two backbone routers for reliability and, usually, per-destination prefix load balancing is implemented.

The access links to the backbone were chosen to be unloaded in order not to introduce additional delay. At the end of each experiment we verified that no packet losses were induced on the last hops of the paths.

In each site, four systems running FreeBSD generate a traffic made of 200 byte UDP packets at a constant rate of 50 packets per second. We choose this rate so that the probes could be easily used to emulate a voice call compliant to the G.711 standard [8].

An additional system captures and timestamps the probe packets using a DAG3.2e card [32]. The DAG cards provide very accurate timestamping of packets synchronized using a GPS (or CDMA) receiver [80]. The probe packets are recorded and timestamped

right before the access links of the two locations in both directions.

In the experiment we discuss here, probes are sent from Reston (VA) to San Francisco (CA) for a duration of 2.5 days starting at 04.00 UTC on November 27th, 2001. We have run active measurements for several weeks but we have chosen that specific trace because it exhibits an interesting network failure event. In terms of delay, loss and voice call quality we have not measured any significant difference among the many different experiments.

Routing data

We integrate our measurement data with IS-IS routing information collected in POP#2 (see Figure A.1). We use an IS-IS listener [85] to record all routing messages exchanged during the experiment. IS-IS messages permit to correlate loss and delay events to changes in the routing information. In order to illustrate the kind of data that are collected by the listener, we give a brief description of the IS-IS protocol.

IS-IS [87] is a link state routing protocol used for intra-domain routing. With IS-IS, each link in the network is assigned a metric value (weight). Every router[1] broadcasts information about its direct connectivity to other routers. This information is conveyed in messages called Link State PDUs (LSP). Each LSP contains information about the identity and the metric value of the adjacencies of the router that originated the LSP. In general, a router generates and transmits its LSPs periodically, but LSPs are also generated whenever the network topology changes (e.g. when a link or a router goes up or down). Thus, LSPs provide valuable information about the occurrence of events such as loss of connectivity, route changes, etc.

Once a router has received path information from all other routers, it constructs its forwarding database using Dijkstra's Shortest Path First (SPF) algorithm to determine the best route to each destination. This operation is called the decision process. In some transitory conditions (e.g. after rebooting), the decision process can take a considerable amount of time (several minutes) since it requires all the LSPs to be received in order to complete. During that transitory period, a router is responsible to make sure that other routers in the network do not forward packets towards itself. In order to do so, a router will generate and flood its own LSPs with the "Infinite Hippity Cost" bit set[2]. This way, other routers will not consider it as a valid node in the forwarding paths.

[1] IS-IS has been designed within the ISO-OSI standardization effort using the OSI terminology. In this paper, we have instead decided to avoid the use of OSI terms.

[2] This bit is also referred to as the OverLoad (OL) bit.

A.4 Voice call rating

Even though active measurements may provide accurate information on network delay and losses, such statistics are not always appropriate to infer the quality of voice calls. In addition to measurements, we use a methodology to emulate voice calls from our packet traces and assess their quality using the E-model standard [12, 13, 14]. The E-model standard is introduced in Section 2.6 on page 19.

A.4.1 Reduction of the E-model to transport level quantities

In this chapter, we use a simplified analytic expression for the R-factor that was proposed in [35] and that describes the R-factor as a function of observable transport level quantities. In this section, we briefly describe the reduction of equation (2.3) to transport level quantities as proposed in [35] and we introduce the assumptions made about the VoIP connections under study.

Delay impairment I_d

We suppose that the default values proposed in [13] are used for all parameters in the expression of I_d other than the delay itself. In particular, the influence of echo is supposed negligible. The curve obtained describing I_d as a function of the mouth-to-ear delay can then be approximated by a piece-wise linear function [35]:

$$I_d = 0.024d + 0.11(d - 177.3)H(d - 177.3) \tag{A.1}$$

where d is the mouth-to-ear delay[3] and H is the Heavyside function. The Heaviside function is defined as follows:

$$\begin{aligned} H(x) &= 0 \quad \text{if } x < 0 \\ H(x) &= 1 \quad \text{if } x \geq 0 \end{aligned} \tag{A.2}$$

Equipment impairment I_e

In this chapter, we focus on the G.711 coder which does not introduce any distortion due to compression (and hence leads to the smallest equipment impairment value in absence of losses). In addition, we assume that the G.711 coder in use implements a packet loss concealment algorithm. In these conditions, the evolution of the equipment impairment factor I_e as a function of the average packet loss rate can be well approximated by a

[3] As explained in Section 2.1, d is composed of encoding delay and packetization delay, network delay (transmission, propagation and queuing delay) and de-jitter delay introduced by the playout buffer in order to cope with delay variations.

logarithmic function. In particular, if we assume that we are in presence of random losses, the equipment impairment can be expressed as follows [35]:

$$I_e = 30 \ln(1 + 15plr) \tag{A.3}$$

where plr is the total loss probability (i.e., it encompasses the losses in the network and the losses due to the arrival of a packet after its playout time).

In summary, the following expression will be used to compute the R-factor as a function of observable transport quantities:

$$\begin{aligned} R &= 94.2 - 0.11(d - 177.3)H(d - 177.3) - \\ &\quad -0.024d - 30 \ln(1 + 15plr) \end{aligned} \tag{A.4}$$

where d is the mouth-to-ear delay, plr is the total loss probability and H is the Heavyside function defined in equation (A.2).

A.4.2 Call generation and rating

In order to assess the quality of voice calls placed at random times during the measurement period, we emulate the arrival of short business calls. We pick call arrival times according to a Poisson process with a mean inter-arrival time of 60 seconds. We draw the call durations according to an exponential distribution with a mean of 3.5 minutes [73]. The randomly generated calls are then applied to the packet traces for quality assessment.

Since IP telephony applications often use silence suppression to reduce their sending rate, we simulate talkspurt and silence periods within each voice call using for both periods an exponential distribution with an average of 1.5s [69]. Packets belonging to a silence period are simply ignored.

At the receiver end, we assume that a playout buffer is used to absorb the delay variations in the network. The de-jitter delay is defined as the difference between the arrival and the playout time of the first packet of a talkspurt. Within a talkspurt, the playout times of the subsequent packets are scheduled at regular intervals following the playout time of the first one. Packets arriving after their playout time are considered lost. A playout buffer can operate in a fixed or an adaptive mode. In a fixed mode, the de-jitter delay is always constant while in an adaptive mode, it can be adjusted between talkspurts.

In this chapter, we opt for the fixed playout strategy because the measured delays and jitters are very small and a fixed playout strategy would represent a worst case scenario. Thus, we implement a fixed de-jitter delay of 75ms (which is quite high, but still leads to excellent results, as described in Section A.5).

The quality of the calls described above is then computed as follows. For each talkspurt within a call, we compute the number of packet losses in the network and in the playback buffer. From these statistics, we deduce the total packet loss rate e for each talkspurt. In addition, we measure the mouth-to-ear delay d, which is the sum of the packetization delay (20ms, in our case), the network delay of the first packet of the talkspurt and the de-jitter delay.

In order to assess the quality of a call we apply equation (A.4) to each talkspurt and then we define the rating of a call as the average of the ratings of all its talkspurts.

A.5 Results

In this section we discuss our findings derived from the experiments and measurements. We first compare the results obtained via the passive and active measurements and then focus on the impact of link failures on VoIP traffic. We conclude with a discussion of the call rating using the methodology proposed in Section A.4.

A.5.1 Delay measurements

In Figure A.2 we show the one-way delay between two Sprint POPs located on the East and West Coast of the United States. The data shown refers to a trace collected from the passive measurement system on July 24th 2001. However, we have systematically observed similar delay distributions on all the traces collected in the Sprint monitoring project [59]. The delay between the two POPs is around 28.50ms with a maximum delay variation of less than 200μs. Such delay figures show that packets experience almost no queueing delay and that the element that dominates the transmission delay is the propagation over the optical fiber [90].

We performed the same delay measurements on the UDP packets sent every 20ms from Reston (VA) to San Francisco (CA) for a period of 2.5 days. Figure A.3 shows the distribution of the one-way transmission delay. The minimum delay is 30.95ms, the average delay is 31.38ms while the 99.9% of the probes experience a delay below 32.85ms.

As we can see from the figures, the results obtained by the active measurements are consistent with the ones derived from passive measurements. Low delays are a direct result of the over-provisioning design strategies followed by most tier-1 ISPs. Most tier-1 backbones are designed in such a way that link utilization remains below 50% in the absence of link failures. Such strategy is dictated by the need for commercial ISPs to be highly resilient to network failures and to be always capable of handling short-term variations in the traffic demands.

The delay distribution in Figure A.3 shows also another interesting feature: a re-routing event has occurred during the experiment. The distribution shows two spikes that

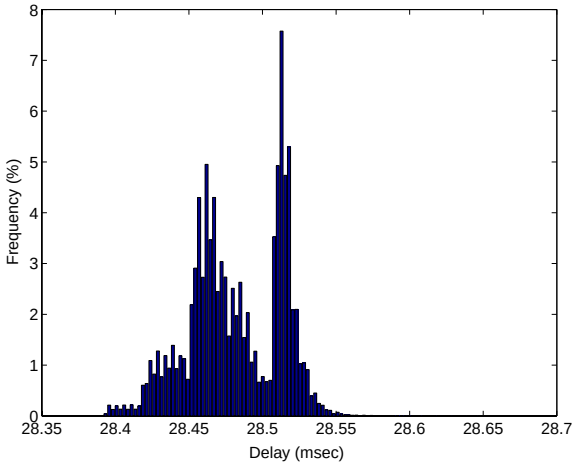


Figure A.2: Passive measurements: distribution of the one-way transmission delay between East and West Coast of the U.S.

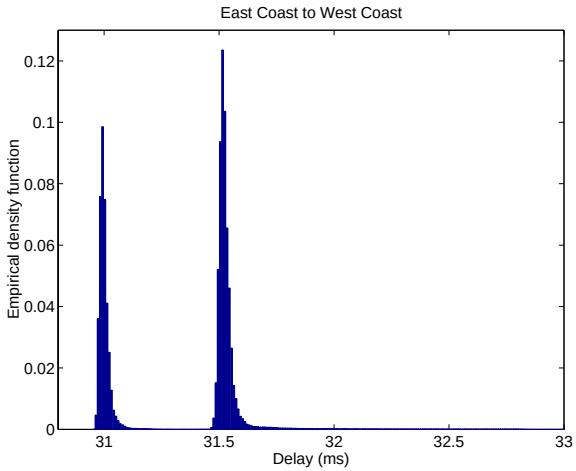


Figure A.3: Active measurements: distribution of the one-way transmission delay from Reston (VA) to San Francisco (CA).

---

do not overlap and for which we can thus identify two minima (30.96ms and 31.46ms), that represent the propagation delays of the two routes[4].

While the difference between the two minima is relatively high (around 500μs), the difference in router hops is just one (derived from the TTL values found in the IP packets). One additional router along the path cannot justify a 500μs delay increase [90]. On the other hand, the Sprint backbone is engineered so that between each pair of POPs there are two IP routhat use too disjoint fiber paths. In our experiment, the 500μs increase in the

[4]The delay distribution derived from passive measurements also shows some spikes. In that case, however, we cannot distinguish between delays due to packet sizes [90] or due to routing, given that we lack the routing information that would let us unambiguously identify the cause of the peaks.

delay is introduced by a 100km difference in the fiber path between the POPs where the re-routing occurred.

A.5.2 Impact of failures on data traffic

In this section we investigate further the re-routing event. To the best of our knowledge there is no experimental study on failures and their impact on traffic on an operational IP backbone network. It can be explained by the difficulties involved in collecting data on the traffic at the time of a failure. Within several weeks of experiments, our VoIP traffic has suffered a single failure. Nevertheless, we believe it is fundamental for researchers and practitioners to study such failure events in order to validate the behaviors and performance of routing protocols, routing equipment and to identify appropriate traffic engineering practices to deal with failures.

The failure perturbed the traffic during a 50 minutes period between 06:30 and 07:20 UTC on November 28th, 2002. During that failure event, the traffic experienced various periods of 100% losses before being re-routed for the rest of the experiment (33 hours).

We now provide an in-depth analysis of the series of events related to the failure and we identify the causes of loss periods. We complement our active measurements with the routing data collected by our IS-IS listener.

Figure A.4 shows the delay that voice probe packets experienced at the time of the failure. Each dot in the plot represents the average delay over a five-second interval. Figure A.5 provides the average packet loss rate over the same five-second intervals.

At time 06:34, a link failure is detected and packets are re-routed along an alternative path that results in a longer delay. It takes about 100ms to complete the re-routing during which all the packets sent are lost. Although the quality of a voice call would certainly be affected by the loss of 100ms worth of traffic, the total impact on the voice traffic is minimal given the short time needed for the re-routing (100ms) and the small jitter induced (about 500μs).

After about a minute, the original route is restored. A series of 100% loss periods follows, each of which lasts several seconds. Figure A.6 shows the one-way delay experienced by all the packets during one of these 100% loss periods (the same behavior can be observed in all the other periods). As we can see from the figure, packets are not buffered by the routers during the short outages (packets do not experience long delays) but they are just dropped because forwarded along an invalid path. Figure A.7 shows the sequence numbers of the packets as received by the end host on the West Coast. Again, no losses nor re-orderings occur during those periods. This is a clear indication that packet drops are not due to congestion events but due to some kind of interface or router failure.

At time 06:48, the traffic experiences 100% losses for a period of about 12 minutes. Surprisingly, during that period no alternative path is identified for the voice traffic. At

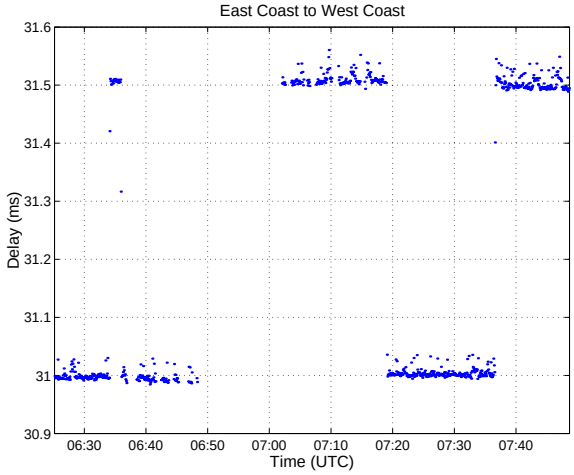


Figure A.4: Average delay during the failure. Each dot corresponds to a five-second interval

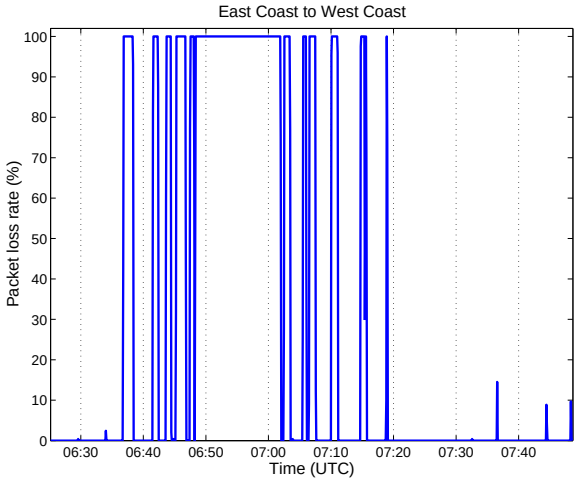


Figure A.5: Average packet loss rate during the failure computed over five-second intervals

---

time 07:02 a secondary path is found but there are still successive 100% loss periods. Finally, at 07:19, the original path is operational again and at time 07:36, an alternative path is chosen and used for the remaining part of the experiment.

The above analysis corresponds to what can be observed from the active measurements. The routing data can provide us more information on the cause of these events. Figure A.8 illustrates the portion of the network topology with the routers involved in the failure. The routers (R_1 to R_5) are located in 2 different POPs. The solid arrows show the primary path used by the traffic. The dashed arrows show the alternative path used after the failure.

Table A.1 summarizes all the messages that we have collected from the IS-IS listener during the experiment. The “Time” column indicates the time at which the LSPs are

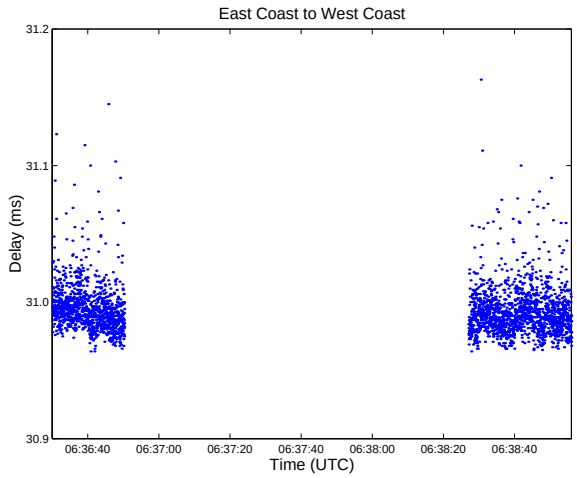


Figure A.6: One-way delay of voice packets during the first 100% loss period

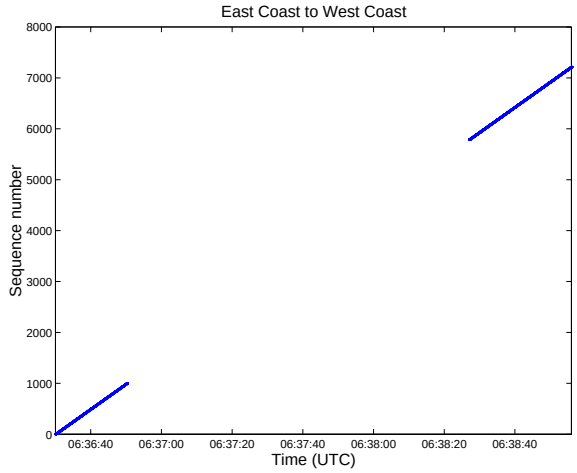


Figure A.7: Sequence numbers of received voice packets during the first 100% loss period

---

received by our listener, the central column ("IS-IS LSPs") describes the LSPs in the format <senders>:<content>, while the third column describes the impact on the traffic of the event reported by IS-IS.

At the time of the first re-routing, routers R_1, R_2 and R_5 report via IS-IS the *loss of adjacency* with R_4. The fact that all the links from R_4 are signaled down is a strong indication that the failure is a router failure as opposed to link failure. As we said earlier, the network reacts to this first event as expected. In about 100ms, R_5 routes the traffic along the alternative path through R_2 (i.e. R_5-R_3-R_2-R_1).

In the period between 06:35 and 06:59, the IS-IS listener receives several (periodic) LSPs from all the five routers reporting that all the links are fully operational. During that time, though, the traffic suffers successive 100% loss periods. For about 13 minutes, R_4 oscillates between a normal operational state (i.e. it forwards the packets without loss or

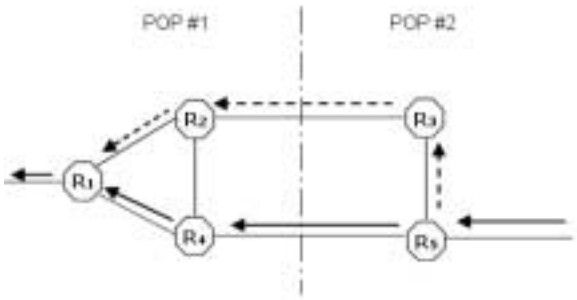


Figure A.8: Routers involved by the failure. The solid arrows indicate the primary path for our traffic. The dashed arrows indicate the alternative path through R_3.

Time	IS-IS LSPs	Impact on traffic
06:34	R_1, R_2, R_5: link to R_4 is down	Re-routed through R_3 in 100ms
06:35	R_1, R_2, R_5: adjacency with R_4 recovered	Re-routed through R_4
from 06:59 to 07:06	R_1: link to R_4 "flaps" 7 times	100% loss periods. Re-routed through R_3
from 07:00 to 07:17	R_2: link to R_4 "flaps" 5 times	100% loss periods. Re-routed through R_3
from 07:04 to 07:17	R_5: link to R_4 "flaps" 4 times	100% loss periods. Re-routed through R_3
07:07	R_1: link to R_4 is down	Re-routed through R_2
07:17	R_1, R_2, R_5: link to R_4 is definitely up	Traffic restored on the original path

Table A.1: Summary of the events occurred during the failure event

additional delay) and a "faulty" state during which all the traffic is dropped. However, such "faulty" state never lasts long enough to give a chance to the other routers to detect the failure.

At time 06:48, R_4 finally reboots. It then starts collecting LSP messages from all the routers in the network in order to build its own routing table. This operation is usually very CPU intensive for a network of the size of the Sprint backbone. It may require minutes to complete as the router has to collect the LSP messages that all the other routers periodically send.

While collecting the routing information, R_4 does not have a routing table and is therefore not capable of handling any packet. As we described in Section A.3, a router is expected to send LSP messages with the "Infinity Hippity Cost" bit set. In our case R_4

does not set that bit. R_5, having no other means to know that R_4 is not ready to route packets, forwards the voice traffic to R_4, where it is dropped.

At time 07:02, R_4 builds its first routing table and the traffic is partially restored but the links R_2-R_4 and R_5-R_4 start flapping resulting again in a succession of 100% loss periods. Note that the traffic is only restored along the alternative path (hence, the longer delays) because the link between R_1 and R_4 is reported to be down. We conjecture that the 100% loss periods are due to R_5 forwarding traffic to R_4 every time the link $R_4 - R_5$ is up, although R_4 does not have a route to R_1.

Most likely the links are not flapping because of an hardware problem but because R_4 is starting receiving the first BGP updates[5] force frequent re-computations of the routing table to add new destination prefixes.

Finally, at time 07:19 all routers report that the links with R_4 are up and the routing remains stable for the rest of the experiment. Traffic is however re-routed again along the alternative path after about 18 minutes even if the original path is operational. This is due to the fact that R_5 modifies its load balancing policy over the two equal cost paths (solid and dashed arrows in Figure A.8). Routers that perform per-destination prefix load balancing (as R_5, in our case) can periodically modify their criteria (i.e., which flow follows which path) in order to avoid that specific traffic patterns defeat the load balancing (e.g., if most of the packets belong to few destination prefixes, one path may result more utilized than the other).

In order to summarize our findings, we divide the failure we observed in two phases:

- The first phase from time 06:34 to 06:59 is characterized by instabilities in the packet forwarding on router R_4: only few LSPs are generated but the traffic experience periods of 100% packet loss. Such "flapping" phase is due to the particular type of failure that involved an entire router and most likely the operating system of the router. The effect on packet forwarding and routing is thus unpredictable and difficult to control protocol-wise.

- The second phase goes from time 06:48 to 07:19 and is instead characterized by a very long outage followed by some routing instabilities and periods of 100% loss. This phase was caused by router R_4 that did not set the "Infinity Hippity Cost" bit. We cannot explain how this problem arised as resetting the hippity bit after the collection of all the BGP updates is a common engineering practice within the Sprint backbone network.

It is important to observe that both the first and the second phase of the failure event are not due to the IS-IS routing protocol. Therefore, we do not expect that the use of a

[5] R_4 can setup the I-BGP sessions, that run over TCP, with its peers only once it has a valid routing table, i.e. it has received all LSP updates.

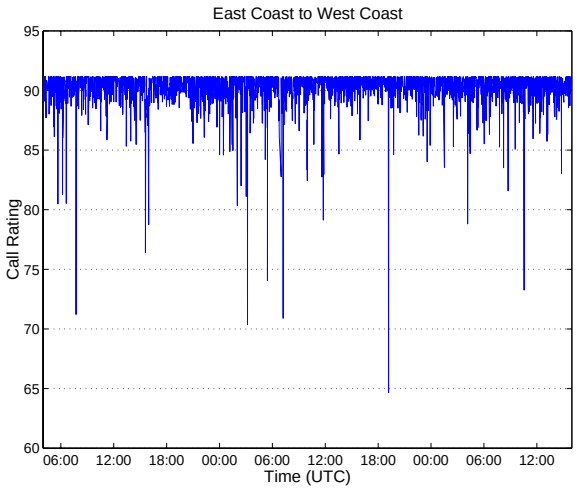


Figure A.9: Voice call ratings (excluding the failure event)

different routing protocol (e.g. "circuit-based" routing mechanisms such as MPLS [98]) would mitigate the impact of failures on traffic.

Instead, it is our opinion that router vendors and ISPs should focus their efforts on the improvement of the reliability of routing equipment, intended both in terms of better hardware architectures and more stable software implementations. Another important direction of improvement is certainly the introduction of automatic validation tools for router configurations. However, such tools would require first to simplify the router configuration procedures. As a side note, introducing circuits or label-switched paths on top of the IP routing will not help in such simplification effort.

A.5.3 Voice quality

This section is devoted to the study of the quality experienced by a VoIP user. Figure A.9 shows the rating of the voice calls during the 2.5 days of the experiment. We did not place any call during the failure event (50 minutes out of the 2.5 days) because the E-model only applies to completed calls and does not capture the events of loss of connectivity.

Figure A.10 shows the distribution of call quality for the 2.5 days of experiment. All these results were derived assuming a fixed playout buffer. One can notice that the quality of calls does not deviate much from its mean value which is fairly high: 90.27. Among the 3,364 calls that were placed, only one experiences a quality rating below 70, the lower threshold for toll-quality. We are currently in the process of investigating what caused the low quality of some calls. Moreover, with 99% of calls experiencing a quality above 84.68, our results confirm that the Sprint IP backbone can support a voice service with PSTN quality standards.

The very good quality of voice traffic is a direct consequence of the low delays, jitter

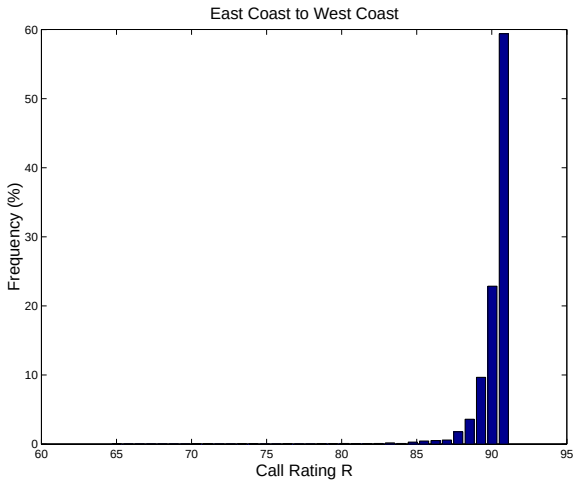


Figure A.10: Distribution of voice call ratings (excluding the failure event)

Loss burst length	Frequency of occurence
1	90.42%
2	8.71%
3	0.71%
4 and above	0.16%

Table A.2: Repartition of loss burst lengths (excluding the failure event)

---

and loss rates that probes experience. Without taking into account the 50 minutes of failure, the average loss rate is 0.19%.

We also studied the probability of having long bursts of losses. The goal was to verify that the assumptions on the distribution of packet losses (in Section A.4 we assumed that the losses were not bursty) and on the performance of packet loss concealment techniques are well suited to our experiment.

For this purpose, we define the *loss burst length* as the number of packets dropped between two packets correctly received by our end hosts. Table A.2 shows the repartition of burst length among the losses observed during the period of experiment. The vast majority (90.42%) of loss events have a burst length 1, while 99.84% of the events have a burst length less than 4. This tends to indicate that the packet loss process is not bursty. Moreover, with a large majority of isolated losses, we can conjecture that packet loss concealment techniques would be efficient in attenuating the impact of packet losses. The results shown in Table A.2 are in line with previous work of Bolot et al. [20] and they suggest that the distribution of burst length is approximately geometric (at least for small loss burst lengths). Future work will include an in-depth study of the packet loss process.

A.6 Conclusion

We have studied the feasibility of VoIP over a backbone network through active and passive measurements. We have run several weeks of experiments and we can derive the following conclusions.

A PSTN quality voice service can be delivered on the Sprint IP backbone network. Delay and loss figures indicate that the quality of the voice calls would be comparable to that of traditional telephone networks.

We have pointed out that voice quality is not the only metric of interest for evaluating the feasibility of a VoIP service. The availability of the service also covers a fundamental role.

The major cause of quality degradation is currently link and router failures, even though failures do not occur very often inside the backbone. We have observed that despite careful IP route protection, link failures can significantly, although infrequently, impact a VoIP service. That impact is not due to the routing protocols (i.e. IS-IS or OSPF), but instead to the reliability of routing equipment. Therefore, as the network size increases in number of nodes and links, more reliable hardware architectures and software implementations are required as well as automatic validation tools for router configurations. Further investigation is needed to identify all the interactions between the various protocols (e.g. IS-IS, I-BGP and E-BGP) and define proper semantics for the validation tools.

The introduction of circuit or label switching networks will not help in mitigating the impact of failures. The failure event we have described in Section A.5 is a clear example of this. As long as the failure is reported in a consistent way by the routers in the network, the IS-IS protocol can efficiently identify alternative routes (the first re-routing event completed in 100ms). The MPLS Fast-ReRoute (FRR) mechanism [1, 49] would provide the same recovery time. On the other hand, a failing and unstable router that sends invalid messages would cause MPLS to fail, in addition to any other routing protocol.

Future work will involve more experiments. Through long-term measurements, we aim to evaluate the likelihood of link and node failures in a tier-1 IP backbone. We also intend to address the problem of VoIP traffic traversing multiple autonomous systems.

Another important area will be the study of metrics to compare the telephone network availability with the Internet availability. On telephone networks, the notion of availability is based on the downtime of individual switches or access lines. The objective of such metric is to measure the impact of network outages on customers. The Federal Communications Commission requires telephone operators to report any outage that affects more that 90,000 lines for at least 30 minutes. Such rule is however difficult to apply to the Internet for a few reasons: i) there is no definition of “line” that can be applied; ii) it is very difficult to count how many customers have been affected by a failure; iii) from a

customer standpoint there is no difference between outages due to the network or due to the servers (e.g. DNS servers, web servers, etc.).

Appendix B

A Note on the Fairness of TCP Vegas

B.1 Introduction

This chapter shows that TCP Vegas' fairness critically requires an accurate estimation of propagation delay. We also show that, in practice, this may be difficult to achieve and we discuss how to choose the parameters α and β that control the window sizes' update.

This chapter is organized as follows:

- we start by summarizing the congestion avoidance scheme of TCP Vegas (Section B.2).
- Section B.3 gives an analysis of the case $\alpha = \beta$ and presents simulation results. Under this setting, we find any over-estimation of the propagation delay of a given connection will increase its rate, an increase that becomes more pronounced with the over-estimation factor.
- Section B.4 is devoted to the case $\alpha < \beta$. Under this setting, we show that the rate of a connection converges to a stable value that depends on the arrival order of all connections. The first connections to be established are favored when the propagation delays are properly estimated. In return, later connections overestimate the propagation delays and hence converge to sending rates higher than what should be. These two effects tend to counterbalance each other but the second tends to dominate.
- Finally, a discussion on our results is addressed and conclusions are drawn in Section B.5.

B.2 TCP Vegas' Congestion Control Algorithm

In this section, we describe the congestion avoidance algorithm of TCP Vegas. As mentioned previously, the bandwidth estimation scheme of TCP Vegas radically differs from the one of TCP Reno. While TCP Reno uses packet losses as congestion feedback, TCP Vegas uses the difference between the expected and actual rates to estimate the congestion state of the network. Because TCP Vegas does not need to engender losses to evaluate the available bandwidth in the network, it utilizes the bandwidth more efficiently than TCP Reno.

The basic idea of TCP Vegas is that the further away the actual throughput gets from the expected throughput, the more congested is the network, which implies that the sending rate should be reduced. The threshold β triggers this decrease. On the other hand, when the actual and expected throughputs are close, the connection is in danger of not utilizing the available bandwidth. The threshold α triggers this increase.

The Congestion Avoidance algorithm of TCP Vegas, first introduced in [28], can be summarized as follows. Once per round trip time,

1. Vegas computes the expected throughput, which is given by:

$$Expected = cwnd/baseRTT$$

   where $cwnd$ is the current window size and $baseRTT$ is the minimum of all measured round trip times.

2. Vegas calculates the current $Actual$ sending rate by using the actual round trip time:

$$Actual = cwnd/RTT$$

   where RTT is the observed round trip time of a packet.

3. Vegas computes the estimated backlog in the buffers by:

$$Diff = (Expected - Actual) * baseRTT$$

4. finally, Vegas updates the window size as follows:

$$cwnd = \begin{cases} cwnd + 1 & if\ Diff < \alpha \\ cwnd & if\ \alpha \leq Diff \leq \beta \\ cwnd - 1 & if\ Diff > \beta \end{cases} \tag{B.1}$$

TCP Vegas controls its window size to keep the measured backlog within the bound-

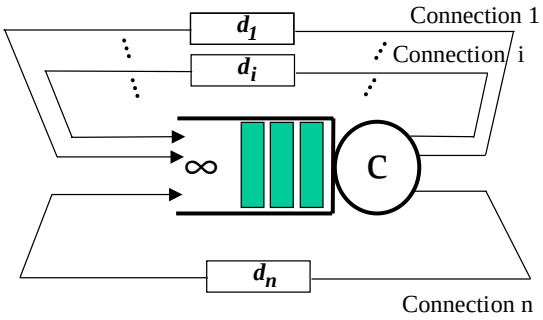


Figure B.1: Network model

aries $[\alpha..\beta]$. The reason behind is that TCP Vegas tries to detect and utilize the extra bandwidth whenever it becomes available without congesting the network. Typical values of α and β are 1 and 3 or 2 and 4 [28, 15] .

B.3 Case 1: $\alpha = \beta$

As we have just seen, TCP Vegas tries to keep a certain amount of packets queued in the buffers. This implies that the value of $baseRTT$ can be greater than the propagation delay (which is the delay when there is no queue). In this section, in which $\alpha = \beta$, we will analyze the influence of an over-estimation of the propagation delay on the fairness of TCP Vegas.

B.3.1 Analysis

Analytical study of the steady state

In this analysis of the fairness of TCP Vegas, we propose a generalization of the equations presented in [81]. We will study the rate distribution, provided by TCP Vegas, at the *steady state* that is when all the window sizes have converged to a stable value.

The network model considered here is illustrated in Figure B.1. It consists of a single bottleneck link, shared by n users. User i ($i = 1, ..., n$) has a propagation delay d_i and uses the window-based flow control of TCP Vegas. The bandwidth of the link is c and the switch adopts a FIFO discipline. The buffer size is assumed to be infinite. This assumption ensures that TCP Vegas does not behave like TCP Reno, which would be the case for small buffer sizes (as shown in [24]).

In the depicted configuration, let us assume that each user i measures a minimum round trip time,

$$baseRTT_i = d_i + x_i \tag{B.2}$$

where x_i (≥ 0) is the propagation delay over-estimation of connection i.

We now consider that the TCP Vegas algorithm has reached a steady state (fixed window sizes). Then, each connection i measures a round trip time $RTT_i = d_i + \Delta$ where Δ is the queuing delay at the switch.

We can deduce from (B.1) that, at the steady state, $Diff = \alpha$. Therefore, we can express the window sizes by:

$$cwnd_i - \frac{baseRTT_i}{RTT_i}\, cwnd_i = \alpha$$

or

$$cwnd_i - \frac{d_i + x_i}{d_i + \Delta}\, cwnd_i = \alpha \tag{B.3}$$

$$cwnd_i = \alpha\, \frac{d_i + \Delta}{\Delta - x_i} \tag{B.4}$$

Let us now derive an expression for the throughput of connection i:

$$rate_i = \frac{cwnd_i}{RTT_i} = \frac{\alpha}{\Delta - x_i} \tag{B.5}$$

This equation clearly shows that any over-estimation x_i of the propagation delay of a given connection results in an increase of its rate which gets greater as the over-estimation factor gets close to Δ .

And finally, if the network capacity c is fully utilized, i.e. $\sum_{i=1}^{n} rate_i = c$, we can deduce the queuing delay at the steady state using:

$$\sum_{i=1}^{n} \frac{\alpha}{\Delta - x_i} = c \tag{B.6}$$

Particular cases

Here we solve Equation (B.6) for two cases of interest:

- **if all connections measure accurately the propagation delay**, $x_i = 0$ for all i. Then, the solution of (B.6) is $\Delta = \frac{n\,\alpha}{c}$ and

  $$rate_i = \frac{c}{n}$$

  This case leads to a fair share of the link bandwidth, which confirms the results in [63, 24].

- **if connection** i **starts when connections** $1, ..., i-1$ **are in equilibrium**, the value of

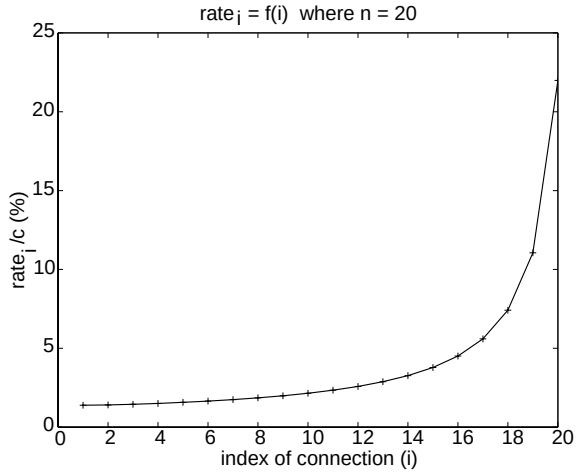


Figure B.2: Repartition of the rates for n = 20

---

x_i ($i = 1, ..., n$) has been determined by Bonald in [24]. He showed that $baseRTT_i$, which is the measure of the round trip time in the steady state reached by connections $1, ..., i-1$ was closely approximated by

$$baseRTT_i = d_i + \frac{\alpha}{c}\,(i-1)\,S_{i-1}$$

where $S_0 = 0$ and for all $i \geq 1$, $S_i = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{i}$.
Therefore the solution of (B.6) is $\Delta = \frac{\alpha}{c}\,n\,S_n$ and

$$rate_i = \frac{c}{n\,S_n - (i-1)\,S_{i-1}}$$

for $i = 1, ..., n$.

Figure B.2 shows the repartition of the rates for $n = 20$. The last connection to be established gets 10 times more bandwidth than the earlier connections. This confirms the critical influence of the propagation delay over-estimation on the fairness.

Quantification of the influence of the propagation delay over-estimation

In order to quantify the influence of the propagation delay over-estimation on the rate distribution we computed two partial derivatives of interest:

- the partial derivative of the rate of a connection with respect to its over-estimation factor is

$$\frac{\frac{\partial rate_i}{\partial x_i}}{rate_i} = \frac{\sum_{k \neq i} \frac{1}{(\Delta - x_k)^2}}{1 + (\Delta - x_i)^2 \sum_{k \neq i} \frac{1}{(\Delta - x_k)^2}} (\Delta - x_i) \qquad \text{(B.7)}$$

This derivative can have significant values when x_i gets close to Δ[1]. This shows that the influence of the over-estimation of the propagation delay of a connection on its rate increases with the over-estimation factor.

- the partial derivative of the rate of a connection with respect to the over-estimation factor of another connection is

$$\frac{\frac{\partial rate_i}{\partial x_k}}{rate_i} = -\frac{1}{(\Delta - x_i) + \sum_{j \neq k} \frac{(\Delta - x_k)^2 (\Delta - x_i)}{(\Delta - x_j)^2}} \tag{B.8}$$

This quantity (which is negative) has significant values only when x_i and x_k are close to Δ and $x_i < x_k$. In practice, this means that the cross influence of the over-estimation of a connection on the rate of another connection is important only for connections with an important over-shooting.

So far, we have considered that all window sizes did stabilize. In fact, when $\alpha = \beta$, the window sizes will oscillate around the steady state values considered in this section. An analytical study of the system dynamics is quite complex. Therefore, we performed simulations to study these oscillations and to check if Hasegawa's hypothesis was true.

B.3.2 Simulations

The results presented in this section were obtained with two different simulators: the Network Simulator (ns) developed at Lawrence Berkeley Laboratory and our own implementation of the congestion avoidance algorithm of TCP Vegas, to cross-check our results.

Simulation setup

The simple network model that was simulated is the one described in Section B.3.1. It consists of a single bottleneck shared by n connections (see Figure B.1). The following parameters were used: the link bandwidth c = 1 Mbps, the propagation delays $d_i = d$ = 0.2 s for all i (all users have the same propagation delay), the number of users n = 10, 20 and 40, and the buffer size is infinite. The successive connections (i = 1,...,n) join the network every 2 seconds, starting from connection with index 1. In addition, we introduced a random part to the propagation delay in order to take into account the influence of very small variations of the queue size (the random part was a zero mean Gaussian with variance equal to the variance of an M/M/1 queue loaded at 90%). Each simulation lasted for 120 seconds.

[1] Without reaching Δ, since the connection's rate has to be finite and smaller than c.

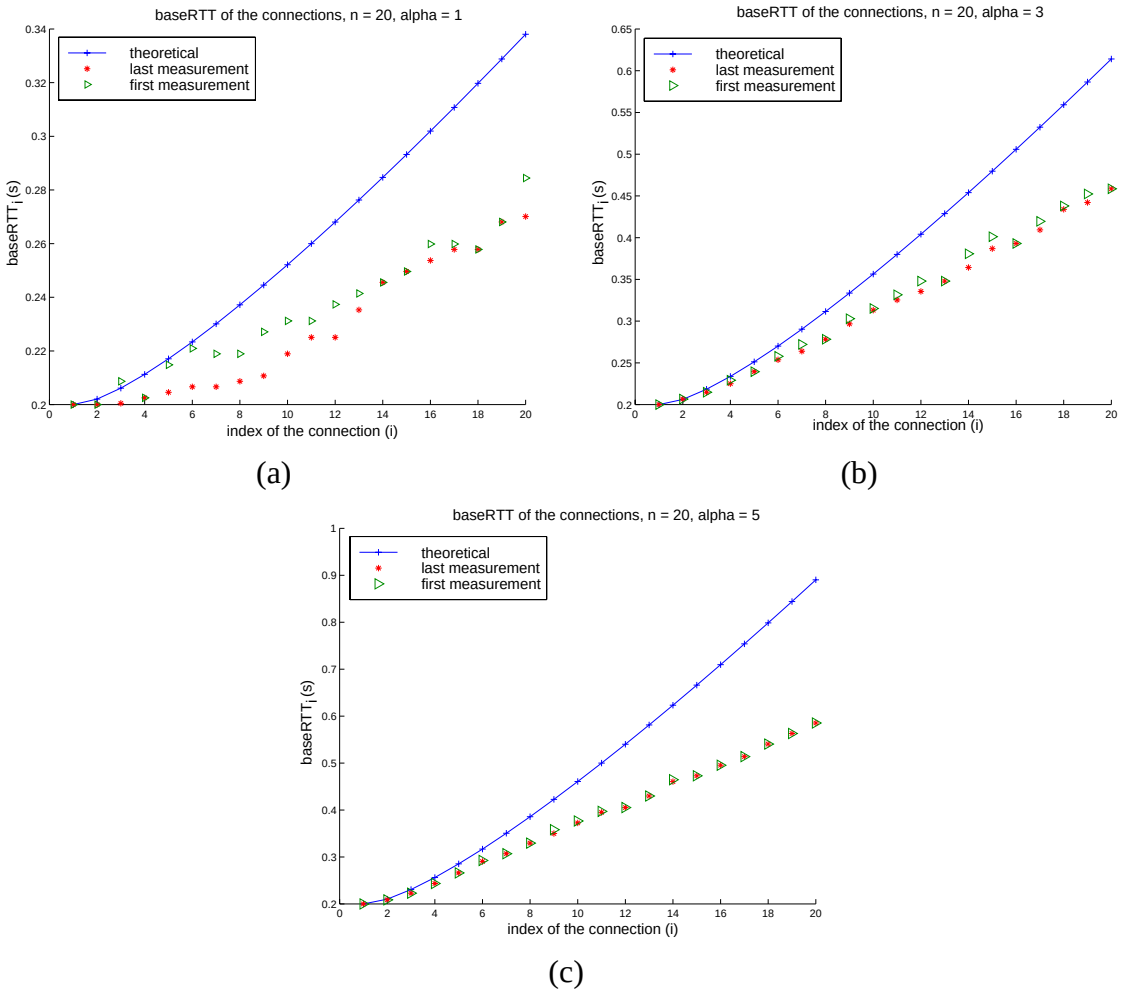


Figure B.3: baseRTT of the connections for (a) $\alpha = 1$, (b) $\alpha = 3$ and (c) $\alpha = 5$.

Results

In this section, we present some results that exhibit the behavior of the TCP Vegas congestion avoidance phase.

In Figure B.3 (a,b,c), we show the values of $baseRTT$ measured by the different connections. Each figure corresponds to a different value of the α parameter. The x-axis and y-axis represent respectively the index of the connection (recall connections join successively the network) and the corresponding $baseRTT$. The solid line represent the theoretical values of $baseRTT$ given by Bonald (not taking the oscillations into account). The triangles and the stars represent respectively the values of $baseRTT$ at the beginning and at the end of the connection. Their simulation values are very close to each other and are far from the value of the propagation delay (especially for the late connections). This means that *the oscillations are not sufficient to allow the connections to measure accurately the propagation delay*. However, the theoretical value of $baseRTT$ is pessimistic compared to the real values.

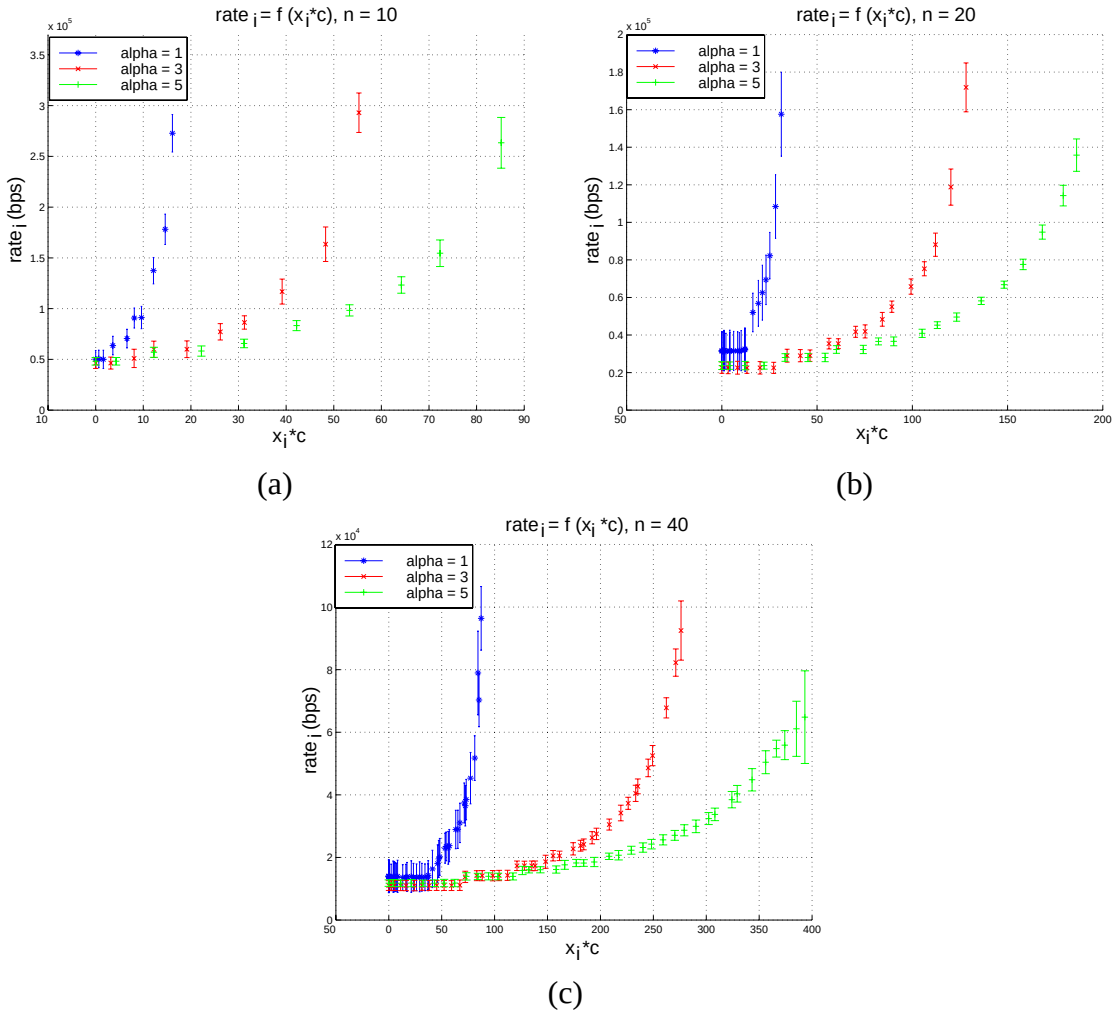


Figure B.4: Repartition of the rates for (a) $n = 10$, (b) $n = 20$ and (c) $n = 40$.

Let us now investigate the influence of the propagation delay over-estimations on the rates distribution. Figure B.4 (a,b,c) shows the rates of the different connections ($rate_i$) as a function of their over-estimation factor (x_i) for different values of α and n. The vertical bars represent the amplitude of the rate oscillations. We can see that any over-estimation of the propagation delay of a connection results in an increase of its rate which gets worse (for a given α) when the over-estimation factor increases. This effect is very critical as the late connections can receive up to 5 times more bandwidth than the earlier connections. This demonstrates the unfairness of TCP Vegas.

Another point of interest is the increase of the amplitude of the rate oscillations with the rate value. This is explained in the following. For all connections, the window size oscillates around its mean value, all oscillation amplitudes being similar. This leads to oscillations of the queuing delay Δ, which influences the rate value following:

$$\frac{\frac{\partial rate_i}{\partial \Delta}}{rate_i} = -\frac{1}{\Delta - x_i} \tag{B.9}$$

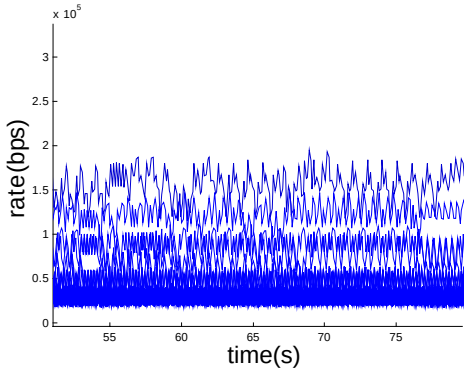


Figure B.5: Evolution of the rates over time for $n = 20$ and $\alpha = 3$.

This derivative increases as x_i approaches Δ, explaining why the rate oscillations increase with the over-estimation factor.

The dynamics of the oscillations are illustrated in Figure B.5 where the evolution of the rates of the connections as a function of the time is plotted (we chose a small simulation window to facilitate the reading of the plot). The oscillations' peaks are not synchronized in time and therefore don't lead to an under-utilization of the link capacity.

B.3.3 Conclusion for $\alpha = \beta$

We have shown that any over-estimation of the propagation delay of a connection results in an increase of its rate which gets worse as the over-estimation factor increases. We also have found that the rate oscillations did not allow to compensate this effect. As a result, the late connections, which have an important over-estimation factor, can get a lot more bandwidth than the earlier connections. Because of this, the enhanced TCP Vegas, when $\alpha = \beta$, does not achieve fairness among the connections. This leads to non deterministic transfer times.

B.4 Case 2: $\alpha < \beta$

We now turn to the case $\alpha < \beta$ in which stabilization of the window sizes, that would oscillate for $\alpha = \beta$ is observed. We now propose to analyze the joint impact of 1) the difference between α and β, and 2) the over-estimation of the propagation delay on the fairness of TCP Vegas. The network model and notations are the same as those depicted in Section B.3.

B.4.1 Analysis of the fairness

At the steady state, in the case $\alpha < \beta$, we can deduce from (B.1) that $\alpha \leq Diff_i \leq \beta$ for all i. Therefore, we can express the window sizes by:

$$\alpha \frac{d_i + \Delta}{\Delta - x_i} \leq cwnd_i \leq \beta \frac{d_i + \Delta}{\Delta - x_i} \tag{B.10}$$

and derive the following expression for the throughput of connection i:

$$\frac{\alpha}{\Delta - x_i} \leq rate_i \leq \frac{\beta}{\Delta - x_i} \tag{B.11}$$

This equation holds the two reasons of unfairness of TCP Vegas. First, if the propagation delays are correctly estimated ($x_i = 0$), the rate of a connection converges to a value that lies between two bounds that depend on the parameters α and β. Therefore some connections could receive β/α times more bandwidth than other connections. Second, late connections will probably receive more bandwidth than earlier ones as the boundary values increase with over-estimation of the propagation delay. The adjective *probably* refers to the fact that the actual convergence value, because of possible overlap between boundaries of successive connections, can not be assessed to a greater value for later connections. Moreover, the convergence values depend on the arrival order of connections, as more than one solution exists to the equation $\sum_{i=1}^{n} rate_i = c$.

Let us now detail the case in which all connections measure accurately the propagation delay. We state that earlier connections will be favored and will receive more bandwidth, as shown in the heuristic argument that follows.

Let us consider the simple case where only two connections are sharing a bottleneck link. Figure B.6 illustrates the convergence region of TCP Vegas for 2 users, but the same geometric picture can be easily extended to a case with more users. In the figure, α_i and β_i lines for connection i denote the sets of window size pairs $\{(cwnd_1, cwnd_2)|Diff_i = \alpha\}$ and $\{(cwnd_1, cwnd_2)|Diff_i = \beta\}$, respectively. The fairness line represents window size pairs of equal throughputs of connections, i.e. $\{(cwnd_1, cwnd_2)|rate_1 = rate_2\}$. Connection 1 increases its window size in regions (1), (4) and (7), and decreases it in regions (3), (6), and (9). Similarly, user 2 increases its window size in regions (7), (8) and (9), and decreases it in regions (1), (2) and (3). The only region where neither user updates its window size is region (5). The arrows in other regions indicate the directions in which the window sizes may get updated. Now, if we suppose that connection 1 starts first and connection 2 joins the network when connection 1 is in steady state, the initial conditions of the system are situated on the x-axis between the lines α_1 and β_1. And, starting from that region, the window sizes will converge to a point in the hachured part of region (5), assuming that the distance between α_i and β_i lines is sufficiently large compared to the

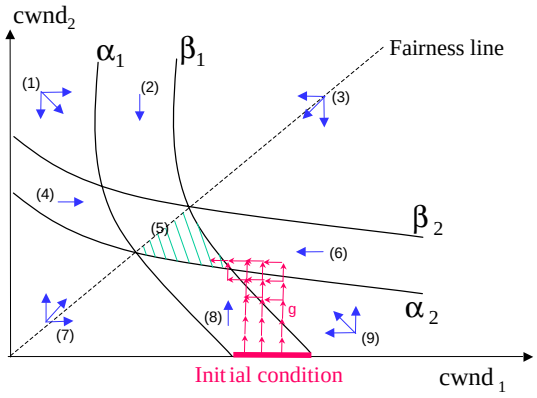


Figure B.6: Convergence region of TCP Vegas

amount ($\delta > 0$) by which users update their window sizes. In the hachured part of region (5), the rate of connection 1 is greater than the one of connection 2 and this explains the bias in favor of early connections. Of course, the greater the difference between α and β, the greater will be the unfairness.

In the next section, we present some simulation results that illustrate our analysis.

B.4.2 Simulations

Using the setup of Section B.3.2, with $\alpha < \beta$, we simulated two scenarios: in the first one, we imposed that all connections have an accurate estimation the propagation delay ($x_i = 0$) while in the second one, more realistic, the propagation delay is estimated by the connections.

Case 1: without over-estimation of the propagation delay

Figure B.7 illustrates the rate distribution between users for different values of the parameters (α, β). The x-axis and y-axis represent respectively the index and the rate of the connections. As expected, we see that earlier connections receive more bandwidth than later ones. Moreover, the unfairness increases with the ratio β/α. We can also notice that the oscillations disappeared.

Case 2: with over-estimation of the propagation delay

Now, we investigate the joint impact of α being unequal to β and the propagation delay over-estimation.

In figure B.8 (a,b,c), we plotted the rates of the connections as a function of their over-estimation factor, for different values of (α, β), and n. We see that the two effects tend to compensate each other and so the overall fairness increases as β furthers off α. However,

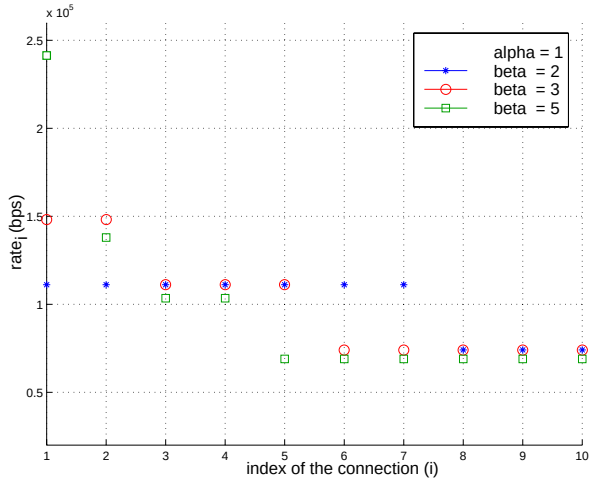


Figure B.7: Rate distribution between the connections for $n = 10$ and without propagation delay over-estimation

---

the effects do not cancel out as the influence of the over-estimation factor dominates. This can be seen in the figure as the rates increase with increasing values of x_i.

B.4.3 Conclusion for $\alpha < \beta$

Under this setting, the rate of a connection converges to a stable value that depends on the arrival order of the connections. When the propagation delays are properly estimated, the earliest established connections are favored and receive more bandwidth. On the other hand, the later connections over-estimate the propagation delays and therefore gain a larger portion of the bandwidth. These two effects tend to counterbalance each other but the second tends to dominate.

B.5 Final Conclusion

In this chapter, we have studied the fairness of TCP Vegas. We have considered the two cases $\alpha = \beta$ and $\alpha < \beta$.

When $\alpha = \beta$, any over-estimation of the propagation delay of a given connection results in an increase of its rate that gets greater as the over-estimation factor increases. The rate oscillations do not allow for compensation of this effect. This results in unfair distribution of bandwidth among the users.

In the case $\alpha < \beta$, we showed that two reasons of unfairness of TCP Vegas are 1) the over-estimation of the propagation delay of a connection and 2) the fact that $\alpha \neq \beta$. The analysis of these two factors evidenced that, although their effects counterbalance, they do not cancel each other out. The over-estimation problem is dominant and causes unfairness.

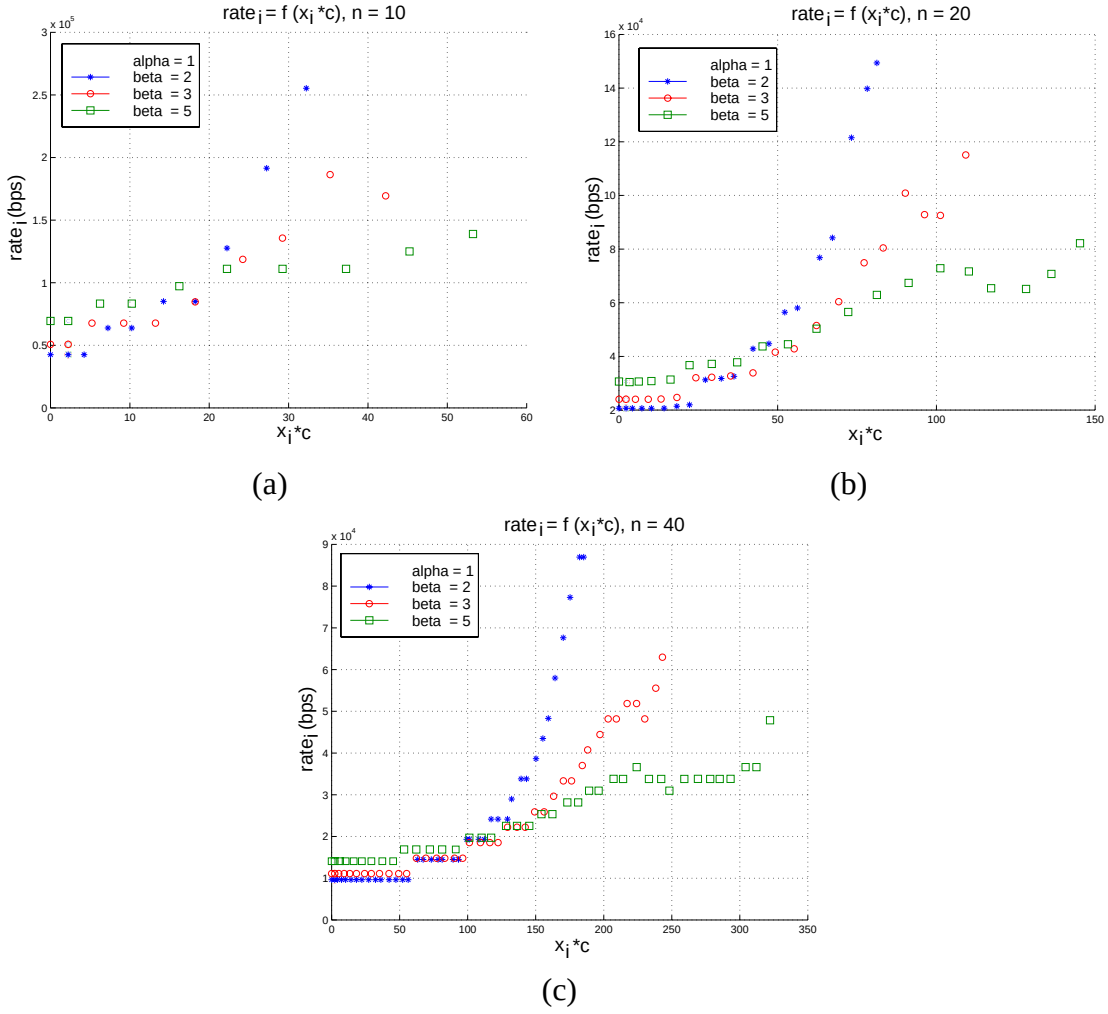


Figure B.8: Repartition of the rates for (a) $n = 10$, (b) $n = 20$ and (c) $n = 40$.

Our final conclusion is that the use of TCP Vegas in the future (instead of Reno) should rely on $\alpha = \beta$ but will require that propagation delays be correctly estimated. There is no obvious way to achieve this.

Appendix C

Delay-Aware Error Control: The optimization problem

In this Appendix, we describe the method used to solve the problem P1 on page 33, which we can rewrite as follows:

$$\begin{array}{ll} \underset{\substack{1\leq K\leq K_{max} \\ \underline{x}^K=(x_1,\ldots,x_K)\in[0,X_{max}]^K}}{\text{maximize}} & F_K(K,\underline{x}^K) \\ \text{subject to} & \left\{ \begin{array}{l} \sum_{i=1}^{K} x_i + R_{overhead} \leq R_{max} \\ PLR_{FEC} \leq PLR_{max} \end{array} \right. \end{array}$$

with $F_K(K,\underline{x}^K) = \sum_{\Delta\subseteq\{1,\ldots,K\}} P(\Delta) \max_{i\in\Delta} f(x_i, d_{e2e}+\tau_K)$. For clarity, we use the notations $F_K(K,\underline{x}^K) = F_K$ and $f(x_i, d_{e2e}+\tau_K) = f_{i,K}$ when the context permits. In the following, we give the details for $K_{max} = 5$. We have:

$$\begin{aligned} F_1 &= \frac{q}{p+q} f_{1,1} \\ F_2 &= \frac{q}{p+q}(1-p) \max_{i\in\{1,2\}} f_{i,2} + \frac{p\,q}{p+q} f_{1,2} + \frac{p\,q}{p+q} f_{2,2} \\ F_3 &= \frac{q}{p+q}(1-p)^2 \max_{i\in\{1,2,3\}} f_{i,3} + \frac{p\,q}{p+q}(1-p) \max_{i\in\{1,2\}} f_{i,3} \\ &\quad + \frac{p\,q^2}{p+q} \max_{i\in\{1,3\}} f_{i,3} + \frac{p\,q}{p+q}(1-p) \max_{i\in\{2,3\}} f_{i,3} \\ &\quad + \frac{p\,q}{p+q}(1-q) f_{1,3} + \frac{p^2\,q}{p+q} f_{2,3} + \frac{p\,q}{p+q}(1-q) f_{3,3} \end{aligned}$$

$$
\begin{aligned}
F_4 \;=\; & \frac{1}{p+q} * \{q(1-p)^3 \max_{i\in\{1,2,3,4\}} f_{i,4} + p\,q(1-p)^2 \max_{i\in\{1,2,3\}} f_{i,4} \\
& + p\,q^2(1-p) \max_{i\in\{1,2,4\}} f_{i,4} + p\,q^2(1-p) \max_{i\in\{1,3,4\}} f_{i,4} \\
& + p\,q(1-p)^2 \max_{i\in\{2,3,4\}} f_{i,4} + p\,q(1-p)(1-q) \max_{i\in\{1,2\}} f_{i,4} \\
& + p^2\,q^2 \max_{i\in\{1,3\}} f_{i,4} + p\,q^2(1-q) \max_{i\in\{1,4\}} f_{i,4} \\
& + p^2\,q(1-p) \max_{i\in\{2,3\}} f_{i,4} + p^2\,q^2 \max_{i\in\{2,4\}} f_{i,4} \\
& + p\,q(1-p)(1-q) \max_{i\in\{3,4\}} f_{i,4} + p\,q(1-q)^2 f_{1,4} \\
& + p^2\,q(1-q) f_{2,4} + p^2\,q(1-q) f_{3,4} + p\,q(1-q)^2 f_{4,4}\} \\
F_5 \;=\; & \frac{1}{p+q} * \{q(1-p)^4 \max_{i\in\{1,2,3,4,5\}} f_{i,5} + p\,q(1-p)^3 \max_{i\in\{1,2,3,4\}} f_{i,5} \\
& + p\,q^2(1-p)^2 \max_{i\in\{1,2,3,5\}} f_{i,5} + p\,q^2(1-p)^2 \max_{i\in\{1,2,4,5\}} f_{i,5} \\
& + p\,q^2(1-p)^2 \max_{i\in\{1,3,4,5\}} f_{i,5} + p\,q(1-p)^3 \max_{i\in\{2,3,4,5\}} f_{i,5} \\
& + p\,q(1-p)^2(1-q) \max_{i\in\{1,2,3\}} f_{i,5} + p^2\,q^2(1-p) \max_{i\in\{1,2,4\}} f_{i,5} \\
& + p\,q^2(1-p)(1-q) \max_{i\in\{1,2,5\}} f_{i,5} + p^2\,q^2(1-p) \max_{i\in\{1,3,4\}} f_{i,5} \\
& + p^2\,q^3 \max_{i\in\{1,3,5\}} f_{i,5} + p\,q^2(1-p)(1-q) \max_{i\in\{1,4,5\}} f_{i,5} \\
& + p^2\,q(1-p)^2 \max_{i\in\{2,3,4\}} f_{i,5} + p^2\,q^2(1-p) \max_{i\in\{2,4,5\}} f_{i,5} \\
& + p^2\,q^2(1-p) \max_{i\in\{2,3,5\}} f_{i,5} + p\,q(1-p)^2(1-q) \max_{i\in\{3,4,5\}} f_{i,5} \\
& + p\,q(1-p)(1-q)^2 \max_{i\in\{1,2\}} f_{i,5} + p^2\,q^2(1-q) \max_{i\in\{1,3\}} f_{i,5} \\
& + p^2\,q^2(1-q) \max_{i\in\{1,4\}} f_{i,5} + p\,q^2(1-q)^2 \max_{i\in\{1,5\}} f_{i,5} \\
& + p^2\,q(1-p)(1-q) \max_{i\in\{2,3\}} f_{i,5} + p^3\,q^2 \max_{i\in\{2,4\}} f_{i,5} \\
& + p^2\,q^2(1-q) \max_{i\in\{2,5\}} f_{i,5} + p^2\,q(1-p)(1-q) \max_{i\in\{3,4\}} f_{i,5} \\
& + p^2\,q^2(1-q) \max_{i\in\{3,5\}} f_{i,5} + p\,q(1-p)(1-q)^2 \max_{i\in\{4,5\}} f_{i,5} \\
& + p\,q(1-q)^3 f_{1,5} + p^2\,q(1-q)^2 f_{2,5} + p^2\,q(1-q)^2 f_{3,5} \\
& + p^2\,q(1-q)^2 f_{4,5} + p\,q(1-q)^3 f_{5,5}\}
\end{aligned}
$$

The original maximization problem can therefore be divided into the sub-problems of finding the constrained maxima of $F_K(K, \underline{x}^K)$, $K = 1, \ldots, K_{max}$, where $\underline{x}^K$ is the variable and K is fixed.

Still, the maximization of F_K is made difficult by the presence of the max functions. To get around this, we partitioned $\Re^K$ into 2^{K-1} sub-spaces σ_i characterized by $\{x_j < x_k < \ldots < x_l\}$ where $\{j, k, \ldots, l\}$ are all the permutations of the set $\{1, \ldots, K\}$.

Considering F_K over each of these sub-spaces allowed to remove the max functions. Moreover, we could identify the subspaces in which the maxima of F_K occur. These sub-spaces depend on the values of p and q. And we could finally rewrite the optimization problem P1 into the problem P2:

$$\begin{array}{c} \text{maximize} \\ 1 \leq K \leq K_{max} \end{array} \quad \max_{\underline{x}^K \in [0, X_{max}]^K} F_K(K, \underline{x}^K)$$

subject to the same constraints as P1, where the functions F_K are defined as follows:

$$\begin{aligned} F_1 &= \frac{q}{p+q} f_{1,1} \\ F_2 &= \frac{q}{p+q} f_{1,2} + \frac{p\,q}{p+q} f_{2,2} \end{aligned}$$

if $p + q < 1$

$$\begin{aligned} F_3 &= \frac{1}{p+q} * \{q\, f_{1,3} + p^2\, q f_{2,3} + p\, q(2-p-q) f_{3,3}\} \\ F_4 &= \frac{1}{p+q} * \{q\, f_{1,4} + p^2\, q(2-p-q) f_{2,4} + p^2\, q(1-q) f_{3,4} \\ &\quad + p\, q(3 - 3p + p^2 + 2pq - 3q + q^2) f_{4,4}\} \\ F_5 &= \frac{1}{p+q} * \{q\, f_{1,5} + p^2\, q(pq + (1-q)^2) f_{2,5} \\ &\quad + p^2\, q(2(1-p)(1-q) + (1-q)^2 + (1-p)^2) f_{3,5} \\ &\quad + p^2\, q(1-q)^2 f_{4,5} + p\, q((1-p)(1-q)^2 + (1-p)^2(1-q) \\ &\quad + (1-q)^3 + 2pq(2-p-q) + (1-p)^3) f_{5,5}\} \end{aligned} \tag{C.1}$$

if $p + q > 1$

$$\begin{aligned} F_3 &= \frac{1}{p+q} * \{q\, f_{1,3} + p\, q f_{2,3} + p\, q(1-q) f_{3,3}\} \\ F_4 &= \frac{1}{p+q} * \{q\, f_{1,4} + p\, q f_{2,4} + p\, q(1-q) f_{3,4} + p\, q(1-q)^2 f_{4,4}\} \\ F_5 &= \frac{1}{p+q} * \{q\, f_{1,5} + p\, q f_{2,5} + p\, q(1-q) f_{3,5} + p\, q(1-q)^2 f_{4,5} \\ &\quad + p\, q(1-q)^3 f_{5,5}\} \end{aligned}$$

The constraint on the packet loss rate after reconstruction can be formulated as a set of constraints on the values of $\underline{x}^K = (x_1, \ldots, x_K)$. Actually, Table 1 shows PLR_{FEC} for a given amount of redundancy. Hence, a maximum value for PLR_{FEC} amounts to imposing a minimum amount of redundancy. If r_0 denotes the minimum rate used to

K	Redundancy	PLR_{FEC}
1	none	$p/(p+q)$
2	-1	$p(1-q)/(p+q)$
3	-2	$(p^2q + p(1-q)^2)/(p+q)$
	-1-2	$(p(1-q)^2)/(p+q)$
4	-3	$(p(3pq - p^2q - 2q^2p + (1-q)^3))/(p+q)$
	-1-3	$(p(1-q)(pq + (1-q)^2))/(p+q)$
	-1-2-3	$p(1-q)^3/(p+q)$
5	-4	$(p^2q((1-p)^2 + 2(1-p)(1-q)$ $+3(1-q)^2) + p^3q^2 + p(1-q)^4)/(p+q)$
	-2-4	$(p^3q^2 + 2p^2q(1-q)^2 + p(1-q)^4)/(p+q)$
	-1-2-4	$(p^2q(1-q)^2 + p(1-q)^4)/(p+q)$
	-1-2-3-4	$(p(1-q)^4)/(p+q)$

Table C.1: Loss rates after reconstruction. Note: in the column Redundancy, -1-2 means that packets n-1 and n-2 were sent in packet n.

---

encode audio samples, $PLR_{FEC} < PLR_{max}$ is equivalent to $x_i \geq r_0$, for all i in the minimal set of copies which yield a packet loss rate smaller than PLR_{max}.

Once the formulation of the original problem is simplified, the maximization of the objective functions F_K,$K = 1, \ldots, K_{max}$ can be carried out using classical methods, the choice of method dependant on the utility function $f(x, y)$. If $f(x, y)$ is differentiable with respect to x and strictly concave, the maximizing values $\underline{x}^K$ can be found by the Lagrangian method. If $f(x, y)$ is a non-linear concave function of x, numerical methods such as SQP (Sequential Quadratic Programming) can be used. In addition, if $f(x, y)$ is a piecewise linear function of x, linear programming methods provide a solution to the maximization problem.

Appendix D

Reed-Solomon FEC: computation of PLR_{FEC}

This appendix gives the detailed computation of the residual packet loss rate after reconstruction when (1) a Reed-Solomon code (n, k) is used, (2) packets are sent over a channel that complies to the assumptions made in Section 5.3.2 and (3) the playout delay is D.

$$PLR_{FEC} = \frac{1}{k} \sum_{i=1}^{k} \overline{P}(i) \tag{D.1}$$

where $\overline{P}(i)$ is the probability that the ith packet in the block is lost for the application i.e. that the ith packet is either (1) dropped in the network and could not be reconstructed or (2) received after its playout time and could not be reconstructed:

$$\begin{aligned}
\overline{P}(i) &= Pr(\{Y_i = 1\} \cap \{\text{packet } i \text{ can not be reconstructed }\}) \\
&\quad + Pr(\{\{Y_i = 0\} \cap \{D_i > D\}\} \bigcap \{\text{packet } i \text{ can not be reconstructed }\}) \\
&= Pr(\{\text{packet } i \text{ can not be reconstructed}\} | \{Y_i = 1\}) \times Pr(\{Y_i = 1\}) \\
&\quad + Pr(\{\text{packet } i \text{ can not be reconstructed}\} | \{\{Y_i = 0\} \cap \{D_i > D\}\}) \\
&\quad \times Pr(\{\{Y_i = 0\} \cap \{D_i > D\}\}) \\
&= (1 - Pr(\{\text{packet } i \text{ can be reconstructed}\} | \{Y_i = 1\})) \times Pr(\{Y_i = 1\}) \\
&\quad + (1 - Pr(\{\text{packet } i \text{ can be reconstructed}\} | \{\{Y_i = 0\} \cap \{D_i > D\}\})) \\
&\quad \times Pr(\{\{Y_i = 0\} \cap \{D_i > D\}\})
\end{aligned} \tag{D.2}$$

With a Reed-Solomon code (n, k), any missing (dropped or received late) packet in a block can be reconstructed if and only if at least k packets among the n packets in this block are received before its playout time:

$$\{\text{packet } i \text{ can be reconstructed}\}$$
$$\Longleftrightarrow$$
$$\{\exists\, k'\ (k' \geq k) \text{ packets in the block } \{1,\ldots,n\} | \{\{Y_j = 0\} \cap \{D_j \leq D + (i-j)T\}\ ,\ j \in \{1,\ldots,n\}\}\}$$

From Property 1, if packet i $(i \leq k)$ arrives after its playout time, then packets j $(i < j \leq n)$ will also arrive after i's playout time:

$$Pr(\{D_j \leq D + (i-j)T\} | D_i > D) = 0 \quad \forall j\ ,\ i < j \leq n$$

This implies that, if packet i $(i \leq k)$ is received after its playout time, it will not be reconstructed, since the following packets will also arrive after its playout time and we need k packets to be received before i's playout time in order to reconstruct it:

$$Pr(\{\text{packet } i \text{ can be reconstructed}\} | \{\{Y_i = 0\} \cap \{D_i > D\}\})) = 0$$

Hence, Equation (D.2) can be re-written:

$$\begin{aligned} \overline{P}(i) &= Pr(\{Y_i = 1\}) \times (1 - Pr(\{\text{packet } i \text{ can be reconstructed}\} | \{Y_i = 1\})) \\ &\quad + Pr(\{\{Y_i = 0\} \cap \{D_i > D\}\}) \\ &= \frac{p}{p+q}(1 - P_{REC}(i)) + \frac{q}{p+q}(1 - F_D(D)) \end{aligned} \tag{D.3}$$

where $P_{REC}(i)$ is the probability to reconstruct the ith packet in the block, given that it was lost in the network. $P_{REC}(i)$ is expressed as follows:

$$\begin{aligned} P_{REC}(i) &= Pr(\{\exists\, k'\ (k' \geq k) \text{ packets in the block } \{1,\ldots,n\} \text{ such that} \\ &\quad \{Y_j = 0\} \cap \{D_j \leq D + (i-j)T\}\ ,\ j \in \{1 \ldots n\}\} | \{Y_i = 1\}) \end{aligned} \tag{D.4}$$

We compute $P_{REC}(i)$ conditionally to the events $\{A_j,\ j = 0,\ldots,n\}$ on the delays of packets in the block:

$$\begin{cases} A_0 &= \{d_1 > D - (1-i)T\} \\ A_j &= \{\{d_j \leq D - (j-i)T\} \cap \{d_{j+1} > D - (j+1-i)T\}\} \quad \text{for } j = 1 \ldots n-1 \\ A_n &= \{d_n \leq D - (n-i)T\} \end{cases}$$

where d_j is the delay experienced by the jth packet in the block. From Properties 1 and 2, $P(\bigcup_{i=0}^{n} A_j) = 1$ and $A_i \cap A_j = \emptyset$ for $i \neq j$. Hence, from the Total Probability Theorem, $P_{REC}(i)$ can be computed as follows:

$$
\begin{aligned}
P_{REC}(i) &= \sum_{g=0}^{n-1} P_{REC}(i|A_g) \times P(A_g) + P_{REC}(i|A_n) \times P(A_n) \\
&= \sum_{g=k+1}^{n-1} P_{REC}(i|A_g) \times P(A_g) + P_{REC}(i|A_n) \times P(A_n) \\
&\quad \text{(since } P_{REC}(i|A_g) = 0 \text{ , } \forall g \leq k\text{)} \\
&= \sum_{g=k+1}^{n-1} P_{REC}(i|A_g)[F_D(D-(g-i)T) - F_D(D-(g+1-i)T)] \\
&\quad + P_{REC}(i|A_n)F_D(D-(n-i)T)
\end{aligned}
$$

$P_{REC}(i|A_g)$, $(g = k+1, \ldots, n)$ is the probability to receive at least k packets in the block, given that we can wait until we have received the gth packet in the block and that we lost the ith packet, namely, it is the probability to receive at least k packets among the g first packets in a block, given that we lost the ith packet. We will denote this probability by $P_{PAR}(k, g, i)$.

$$
\begin{aligned}
P_{REC}(i|A_g)) &= Pr(\{\exists\, k' \ (k' \geq k) \text{ packets in the set } \{1, \ldots, g\} \text{ such that} \\
&\quad \{\{Y_j = 0\} \text{ , } j \in \{1 \ldots g\}\}|\{Y_i = 1\}) \\
&= P_{PAR}(k, g, i)
\end{aligned}
$$

$P_{PAR}(k, g, i)$ is the probability to loose between 1 and $g-k$ packets in $\{1, \ldots, g\}$, $(g \geq k+1)$, given that the ith packet is lost.

Now, let $p_R(i)$ denote the probability to receive exactly $i-1$ packets until the next packet loss, conditionally to the fact the we just lost packet: $p_R(i) = Pr(\{Y_j = 0, \forall j \in \{1, \ldots, i-1\}\} \cap \{Y_i = 1\}|\{Y_0 = 1\})$. Similarly, let $P_R(i)$ denote the probability that at least $i-1$ packets will be received correctly until the next packet loss, conditionally to the fact that we just lost a packet: $P_R(i) = Pr(\{Y_j = 0, \forall j \in \{1, \ldots, i-1\}\}|\{Y_0 = 1\})$. For a Gilbert loss process, these probabilities are expressed as follows:

$$
p_R(i) = \begin{cases} 1-q & \text{if } i = 1 \\ q(1-p)^{(i-2)}p & \text{otherwise} \end{cases}
$$

and

$$
P_R(i) = \begin{cases} 1 & \text{if } i = 1 \\ q(1-p)^{(i-2)} & \text{otherwise} \end{cases}
$$

Using these probabilities, the probability $R(m, n)$ that $m-1$ packets are lost in the

next $n-1$ packets following a packet loss (conditionally to the fact that we just lost a packet) can easily be computed by recurrence [60]:

$$R(m,n) = \begin{cases} P_R(n) & \text{for } m = 1 \text{ and } n \geq 1 \\ \sum_{i=1}^{n-m+1} p_r(i) R(m-1, n-i) & \text{for } 2 \leq m \leq n \end{cases}$$

Similarly, we can define the 'backward' probabilities $\tilde{p}_R(i)$ to be the probability to have received exactly $i-1$ packets since the last packet loss, conditionally to the fact the we just lost packet: $\tilde{p}_R(i) = Pr(\{Y_{-j} = 0, \forall j \in \{1, \ldots, i-1\}\} \cap \{Y_{-i} = 1\} | \{Y_0 = 1\})$; $\tilde{P}_R(i)$ to be the probability that at least $i-1$ packets were received correctly since the last packet loss, conditionally to the fact that we just lost a packet: $\tilde{P}_R(i) = Pr(\{Y_{-j} = 0, \forall j \in \{1, \ldots, i-1\}\} | \{Y_0 = 1\})$ and $\tilde{R}(m,n)$ to be the probability that $m-1$ packets are lost in the last $n-1$ packets preceding a packet loss (conditionally to the fact that we just lost a packet). The reversibility of the Gilbert loss process allows to write: $\tilde{p}_R(i) = p_R(i)$, $\tilde{P}_R(i) = P_R(i)$ and $\tilde{R}(m,n) = R(m,n)$.

The probability $P_{PAR}(k,g,i)$ to loose between 1 and $g-k$ packets in $\{1, \ldots, g\}$, $(g \geq k+1)$, conditionally to the fact that the ith packet is lost is now easy to compute:

$$\begin{aligned} P_{PAR}(k,g,i) &= \sum_{l=1}^{g-k} \sum_{m=0}^{\min(l-1, i-1)} \tilde{R}(m+1, i) R(l-m, g-i+1) \\ &= \sum_{l=1}^{g-k} \sum_{m=0}^{\min(l-1, i-1)} R(m+1, i) R(l-m, g-i+1) \end{aligned} \quad \text{(D.5)}$$

Appendix E

Adaptive Joint Playout Buffer and FEC Adjustment: Further Simulation Results

E.1 Comparison between Different Playout Algorithms

Figures E.1 (a) to (d) compare the performances obtained with method N2 to the ones obtained with method N1 using different playout adjustment algorithms at the receiver and with Signal Processing FEC. Utility f_1 is used and the channel characteristics are the same as shown in Fig. 5.3. We compare the *virtual* [99] versions of Algorithms 1 and 4[1] from [95], to the *virtual* version of the algorithm proposed in [83] (that we call 'Window' in Fig. E.1) and to the *previous optimal* algorithm proposed in [99]. Figure E.1 (a) shows that the delays obtained with the virtual Algorithm 1, the previous optimal and the virtual 'window' algorithm are quite close to each other. Figure E.1 (b) shows that the virtual version of Algorithm 1 gives good results in a wide range of network conditions. Even though the virtual version of Algorithm 4 had a very small playout delay, it got a very low utility compared to other algorithms (see Fig. E.1 (b)). This is mainly due to the poor performances of this algorithm regarding the packet loss rate after reconstruction. Indeed, this algorithm leads to higher loss probabilities than other algorithms because it attempts to track the network delays too closely and it looses a lot of packets whenever the delay estimate is small (our results concerning Algorithm 4 confirm the results obtained in [99]).

[1] We omit results obtained with algorithms 2 and 3, since they do not outperform the other algorithms.

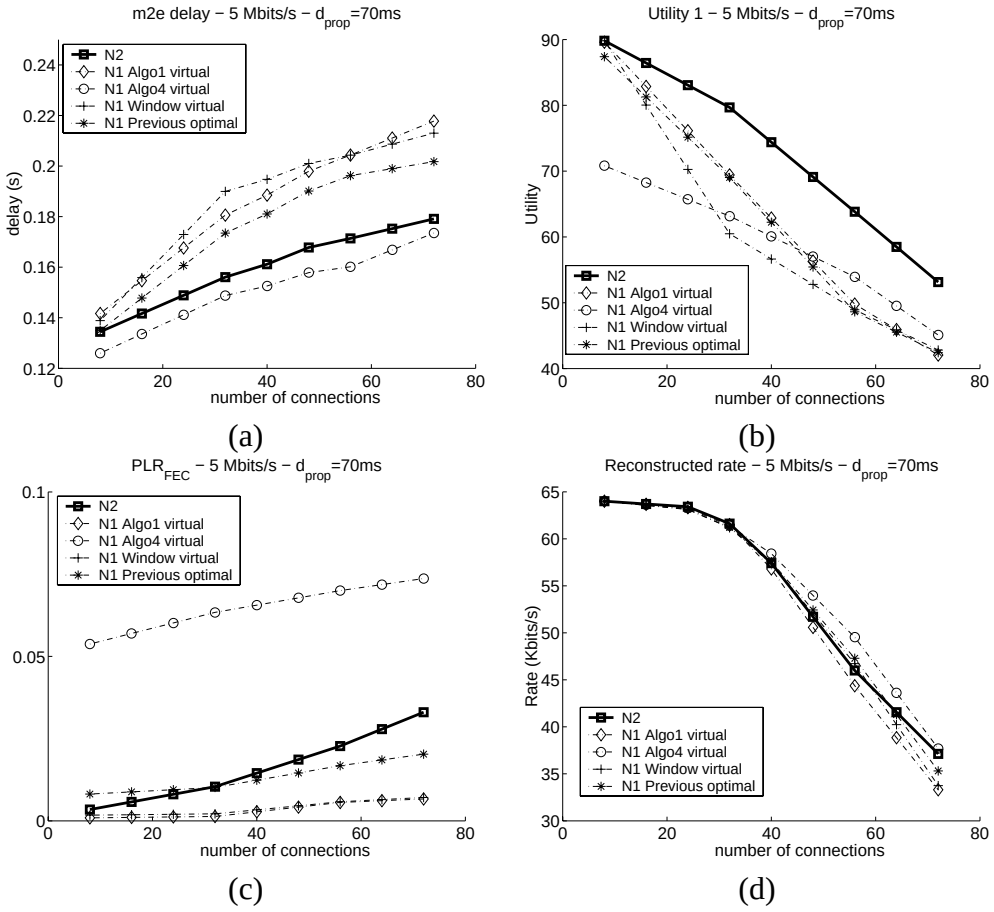


Figure E.1: Performances of the different methods N2 and N1 using different playout adjustment algorithms: (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utility f_1

E.2 Case of Small End-to-End Delays

Figures E.2 (a), (b) and (c) show respectively the TCP-Friendly rate constraint R_{max}, the Gilbert parameter p and the Gilbert parameter q as a function of the number of connections sharing the link for a bottleneck bandwidth of 5Mbits/s and a one-way propagation delay (without queuing) of 25ms.

Figures E.3 to E.6 show the performances, in terms of (a) mouth-to-ear delay, (b) utility measured at the receiver, (c) residual packet loss rate and (d) rate of the reconstructed audio stream at the receiver, for the different methods N1, N2 (Play First), O1 and O2 as a function of the number of connections sharing the link. Figures E.3, E.4 and E.5 correspond respectively to the results obtained with utility functions f_1, f_2 and f_3 with a Signal Processing FEC coding scheme. Figure E.6 was obtained with utility function f_1 and a Reed-Solomon error coding. Algorithm 1 in [95] was used to adjust the playout delay with method O1 and its virtual version was used with N1 and O2. For clarity of the figure, we only show the best curves obtained with method O2. These curves were

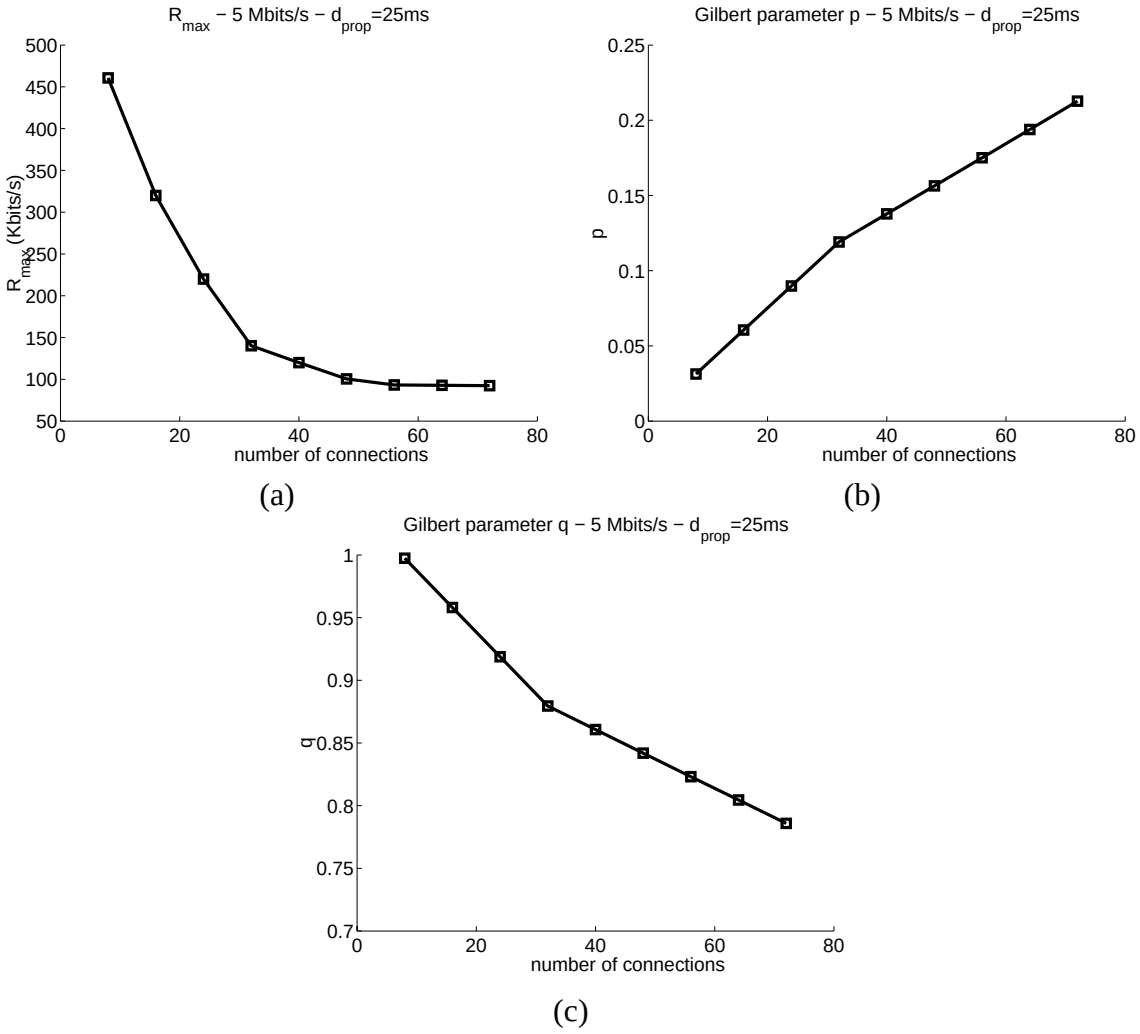


Figure E.2: a) TCP-Friendly rate constraint. (b) Gilbert parameter p. (c) Gilbert Parameter q as a function of the number of connections sharing the link.

---

obtained with $K = 3$ in the case of Signal Processing FEC and with $(k, n) = (1, 3)$ and $(k, n) = (2, 3)$ in the case of Reed-Solomon FEC.

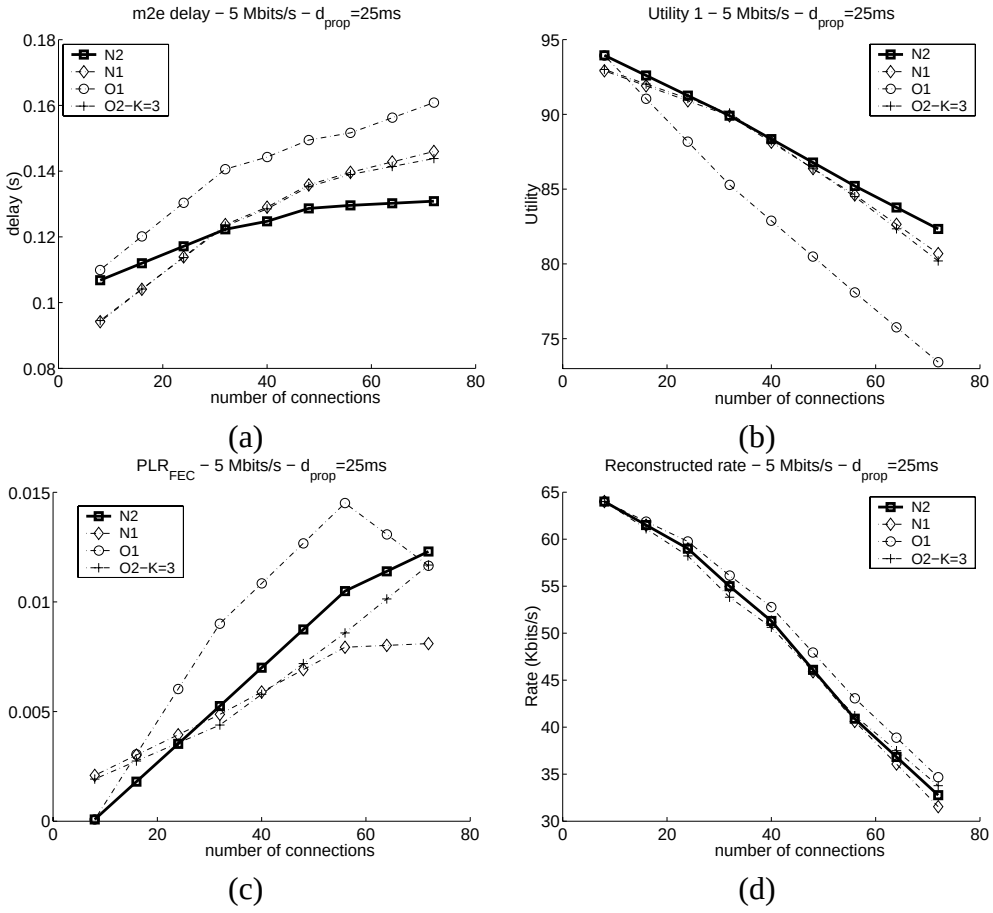


Figure E.3: Performances of the different methods N1, N2 (Play First), O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utility f_1.

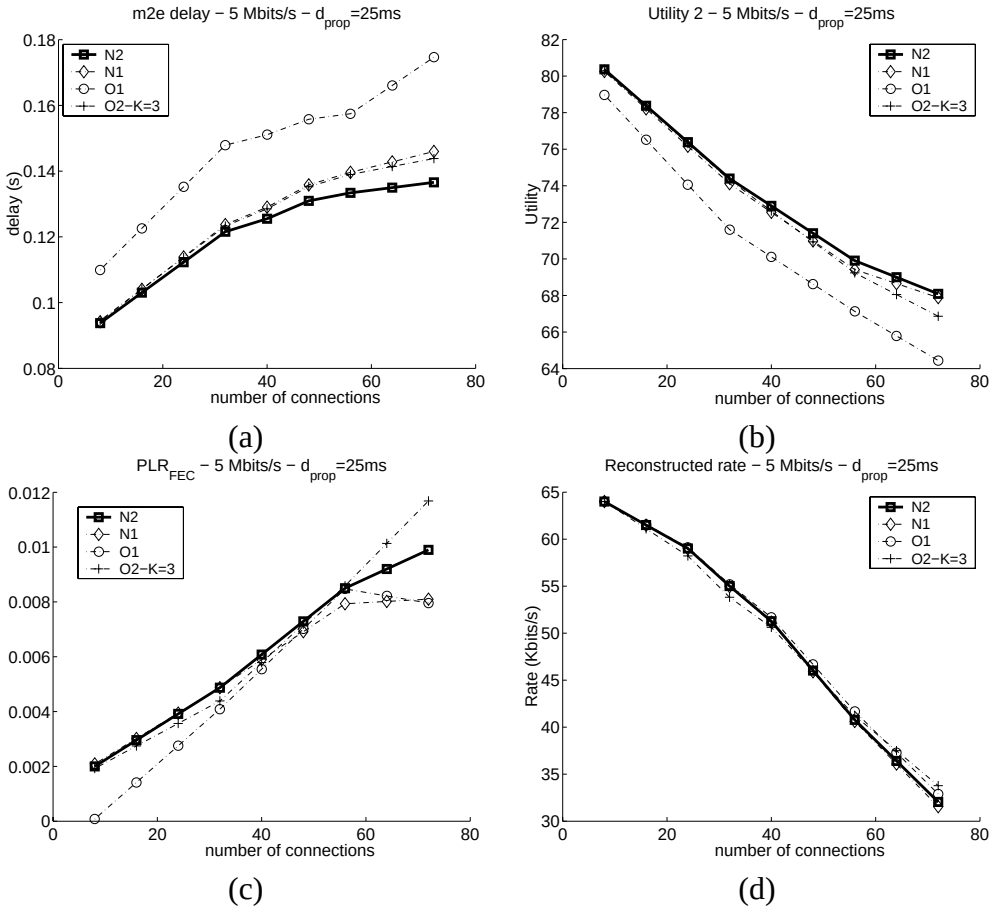


Figure E.4: Performances of the different methods N1, N2 (Play First), O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utility f_2.

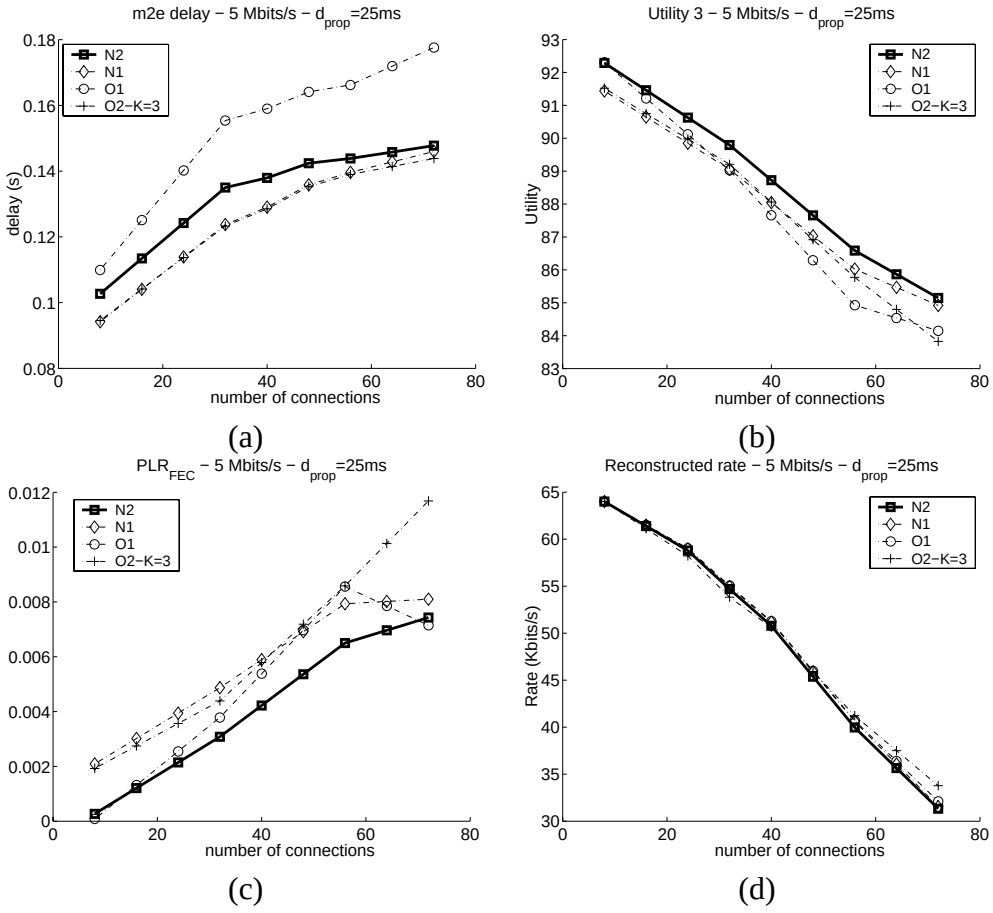


Figure E.5: Performances of the different methods N1, N2 (Play First), O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and (d) reconstructed rate of audio for SP FEC with utility f_3.

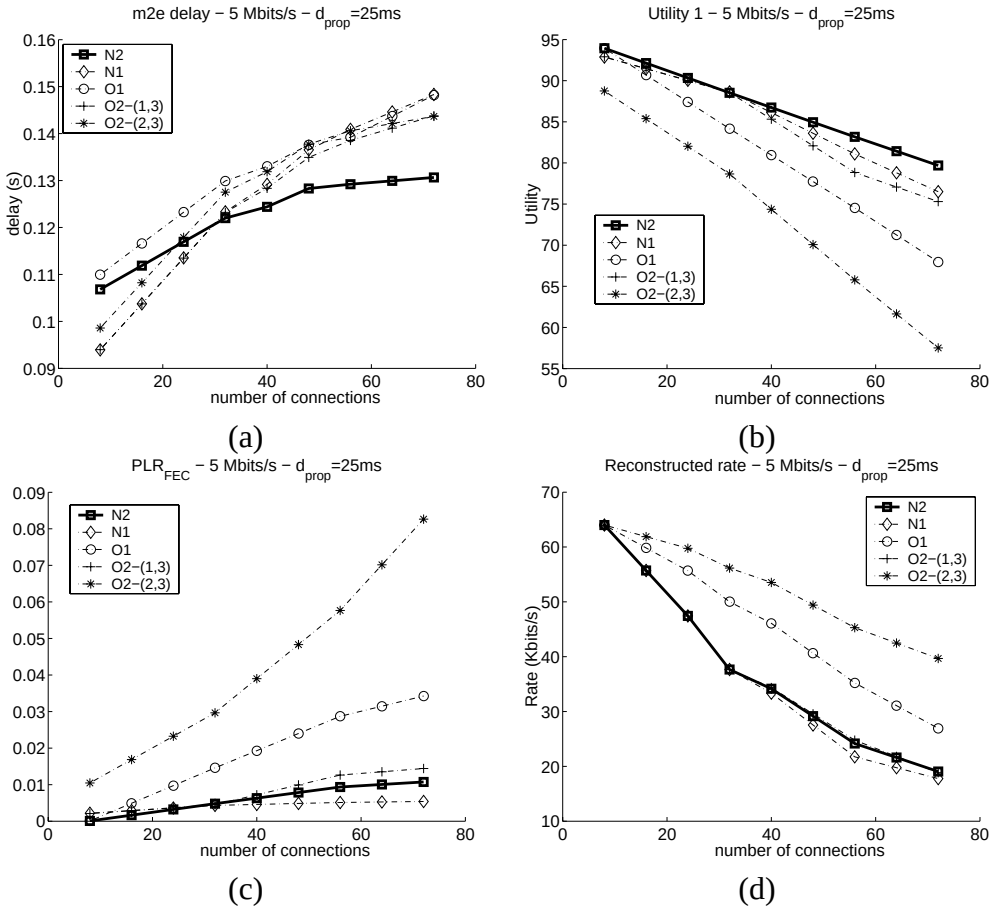


Figure E.6: Performances of the different methods N1, N2, O1 and O2 (with different FEC parameter settings). (a) Mouth-to-Ear delay, (b) utility measured at the destination, (c) residual packet loss rate and reconstructed rate of audio for Reed-Solomon FEC with utility f_1.

.

Appendix F

Audio on Non Elevated Services : Further Simulations

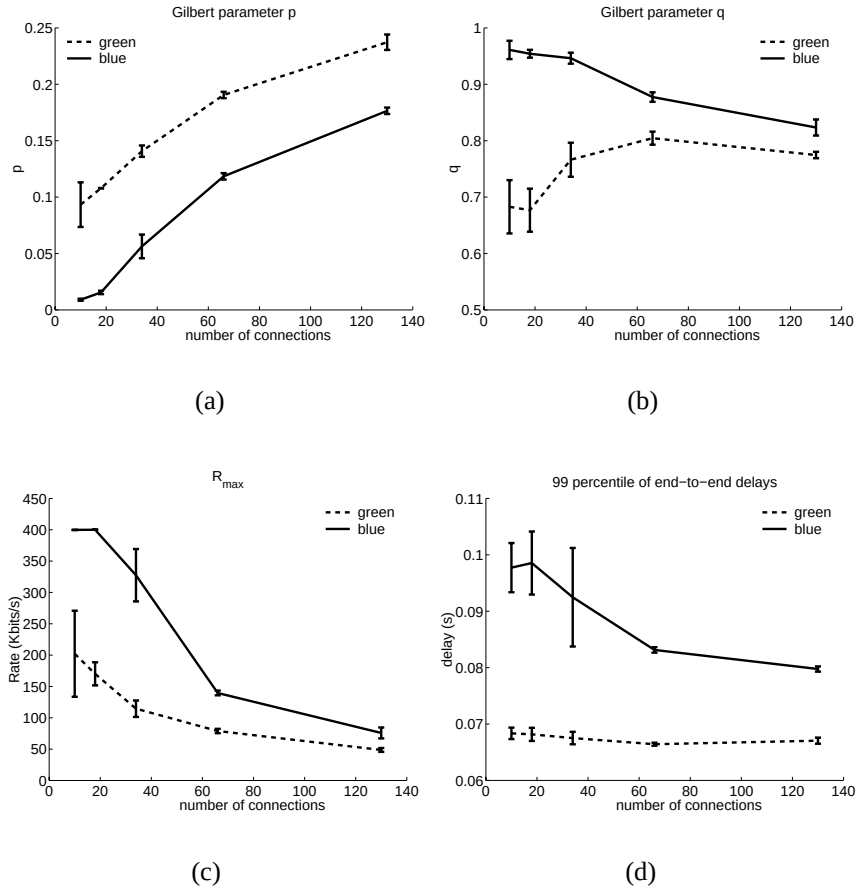


Figure F.1: Network characteristics for *green* and *blue* traffic. $d_{prop} = 30ms$, $d_g = 25ms$, $d_v = 60ms$.

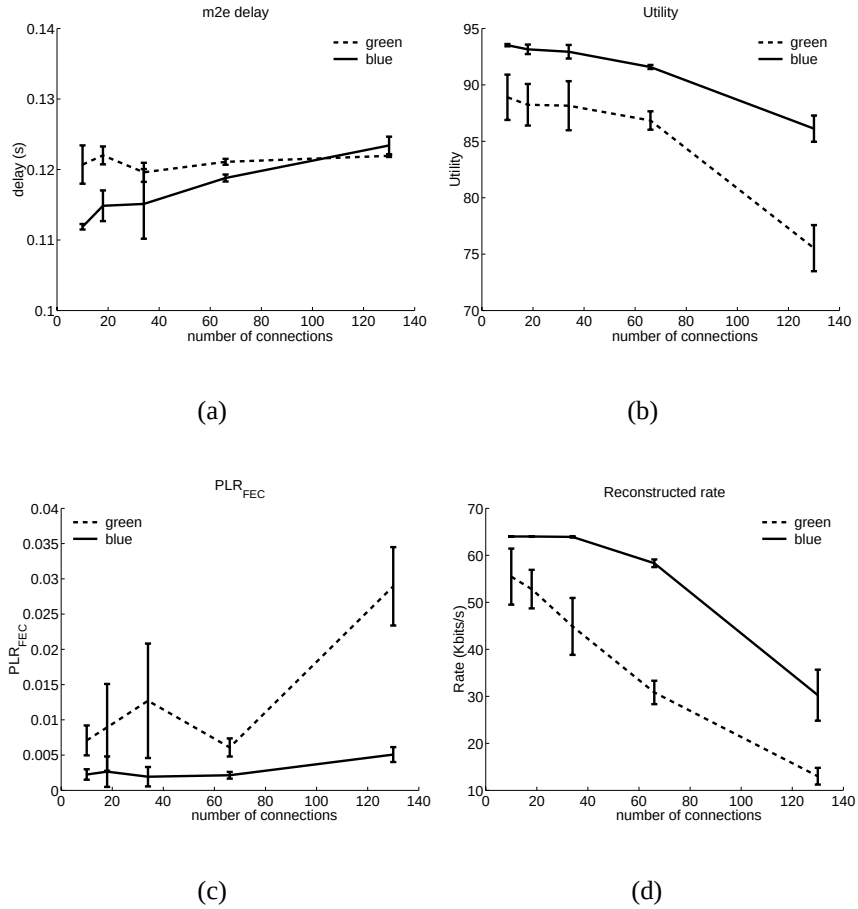


Figure F.2: Performance of delay-aware audio over *blue* and *green* services with utility f_2. $d_{prop} = 30ms$, $d_g = 25ms$, $d_v = 60ms$.

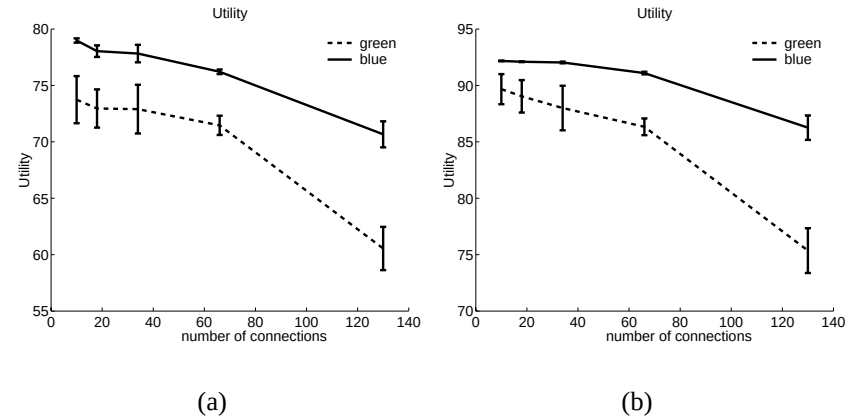


Figure F.3: Performance of delay-aware audio over *blue* and *green* services : average utility obtained with utility function (a) f_2 and (b) f_3. $d_{prop} = 30ms$, $d_g = 25ms$, $d_v = 60ms$.

Appendix G

Congestion Control for Variable Packet Size : Analysis

G.1 Gateway in Packet Mode

From Equation (7.3), the expected loss interval obtained with the unmodified loss measurement mechanism is derived as follows:

$$\begin{aligned} E(\theta_n) &= \sum_{m=0}^{\infty} m\, P(\theta_n = m) \\ &= \sum_{i=0}^{\infty} (N+i)(1-p)^i p \\ &= Np \sum_{i=0}^{\infty} (1-p)^i + p \sum_{i=0}^{\infty} i(1-p)^i \\ &= N + \frac{1-p}{p} \\ &= N - 1 + \frac{1}{p} \end{aligned}$$

G.1.1 Virtual Packets

Based on Equation (7.5), the expected loss interval measured with the virtual packets algorithm is given by:

$$\begin{aligned} E(\theta_n) &= \sum_{i=0}^{\infty} (N + \frac{i}{\eta}) P(\theta_n = N + \frac{i}{\eta}) \\ &= N + \sum_{i=0}^{\infty} \frac{i}{\eta}\, P(\theta_n = N + \frac{i}{\eta}) \end{aligned}$$

$$
\begin{aligned}
& \text{because } \sum_{i=0}^{\infty} P(\theta_n = N + \frac{i}{\eta}) = \frac{1}{p^\eta}\, p^\eta = 1 \\
= \;& N + \sum_{i=0}^{\infty} \frac{i}{\eta} \binom{i+\eta-1}{\eta-1} (1-p)^i p^\eta \\
= \;& N + \frac{p^\eta}{\eta}(1-p) \sum_{i=0}^{\infty} i \binom{i+\eta-1}{\eta-1} (1-p)^{i-1} \\
= \;& N + \frac{p^\eta}{\eta}(1-p)\, \eta \frac{1}{p^{\eta+1}} \\
= \;& N - 1 + \frac{1}{p}
\end{aligned}
$$

G.1.2 Random Sampling

With this method, the nth loss interval, θ_n, is defined as follows (see Section 7.5.1):

$$
\begin{aligned}
P(\theta_n = m) = \;& \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} {p_\eta}^j (1-p_\eta)^{N\eta-\eta-j} \\
& \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} {p_\eta}^{m-j-1} (1-p_\eta)^{k-(m-j-1)} (1-p)^{m-j-1} p_\eta p \\
= \;& \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p)}\right)^j \\
& \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} (1-p_\eta)^{k-(m-j-1)} (1-p)^{m-1} (1-p_\eta)^{N\eta-\eta}\, {p_\eta}^m\, p \\
= \;& (1-p_\eta)^{N\eta-\eta}\, p\, (1-p)^{m-1} \\
& \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p)}\right)^j \left(\frac{1}{p_\eta}\right)^{m-j} {p_\eta}^m \\
= \;& (1-p_\eta)^{N\eta-\eta}\, p\, (1-p)^{m-1} \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j
\end{aligned}
$$

The expected loss interval is then given by:

$$
\begin{aligned}
E(\theta_n) = \;& (1-p_\eta)^{N\eta-\eta}\, p \sum_{m=1}^{\infty} m\, (1-p)^{m-1} \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \\
= \;& (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\, (1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right.
\end{aligned}
$$

$$
\begin{aligned}
&\quad + \sum_{m=N\eta-\eta+1}^{\infty} m\,(1-p)^{m-1} \sum_{j=0}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \Bigg] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right. \\
&\quad \left. + \left(\sum_{m=0}^{\infty} m\,(1-p)^{m-1} - \sum_{m=0}^{N\eta-\eta} m\,(1-p)^{m-1} \right) \left(1 + \frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right. \\
&\quad + \frac{1}{p^2}\left(1 - (N\eta-\eta)(1-p)^{N\eta-\eta+1} + (N\eta-\eta+1)(1-p)^{N\eta-\eta} - 1\right) \\
&\qquad \left. \left(1 + \frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right. \\
&\qquad \left. + \frac{(1-p)^{N\eta-\eta}}{p^2}(1 - (N\eta-\eta)\,p) \left(1 + \frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right] \\
&\qquad + \underbrace{\frac{1+(N\eta-\eta)\,p}{p} \left[(1-p_\eta)(1-p) + p_\eta\right]^{N\eta-\eta}}_{A} \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{j=0}^{N\eta-\eta-1} \underbrace{\sum_{m=j+1}^{N\eta-\eta} m\,(1-p)^{m-1}}_{B} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right] + A
\end{aligned}
$$

where B is given by:

$$
\begin{aligned}
B &= \sum_{m=0}^{N\eta-\eta} m\,(1-p)^{m-1} - \sum_{m=0}^{j} m\,(1-p)^{m-1} \\
&= \frac{(N\eta-\eta)(1-p)^{N\eta-\eta+1} - (N\eta-\eta+1)(1-p)^{N\eta-\eta} + 1}{p^2} \\
&\quad - \frac{j\,(1-p)^{j+1} - (j+1)(1-p)^j + 1}{p^2} \\
&= \frac{(1-p)^{N\eta-\eta}}{p^2}[(N\eta-\eta)(1-p) - (N\eta-\eta+1)] - \frac{(1-p)^j}{p^2}[\,j\,(1-p) - (j+1)] \\
&= -\frac{(1-p)^{N\eta-\eta}}{p^2}[(N\eta-\eta)p + 1)] + \frac{(1-p)^j}{p^2}(jp+1)
\end{aligned}
$$

thus

$$
\begin{aligned}
E(\theta_n) &= A - \frac{[(1-p_\eta)(1-p)]^{N\eta-\eta}}{p}[(N\eta-\eta)p+1)] \sum_{j=0}^{N\eta-\eta-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \\
&\quad +(1-p_\eta)^{N\eta-\eta} \sum_{j=0}^{N\eta-\eta-1} \binom{N\eta-\eta}{j} j \left(\frac{p_\eta}{1-p_\eta}\right)^j \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p} \sum_{j=0}^{N\eta-\eta-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{1-p_\eta}\right)^j \\
&= A - \frac{[(1-p_\eta)(1-p)]^{N\eta-\eta}}{p}[(N\eta-\eta)p+1)] \\
&\qquad \left[\sum_{j=0}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j - \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta}\right] \\
&\quad +(1-p_\eta)^{N\eta-\eta} \left[\sum_{j=0}^{N\eta-\eta} \binom{N\eta-\eta}{j} j \left(\frac{p_\eta}{1-p_\eta}\right)^j - (N\eta-\eta)\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p} \left[\sum_{j=0}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{1-p_\eta}\right)^j - \left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&= A - \frac{[(1-p_\eta)(1-p)]^{N\eta-\eta}}{p}[(N\eta-\eta)p+1)] \\
&\qquad \left[\left(1+\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} - \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta}\right] \\
&\quad +(1-p_\eta)^{N\eta-\eta} \left[\frac{p_\eta}{1-p_\eta}(N\eta-\eta)\left(1+\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta-1} - (N\eta-\eta)\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p} \left[\left(1+\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta} - \left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&= A - \frac{(N\eta-\eta)p+1}{p}\left\{[(1-p_\eta)(1-p)+p_\eta]^{N\eta-\eta} - {p_\eta}^{N\eta-\eta}\right\} \\
&\quad +(1-p_\eta)^{N\eta-\eta} \left[p_\eta(N\eta-\eta)\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta} - (N\eta-\eta)\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p} \left[\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta} - \left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&= \frac{(N\eta-\eta)p+1}{p}{p_\eta}^{N\eta-\eta} + p_\eta(N\eta-\eta) - (N\eta-\eta){p_\eta}^{N\eta-\eta} + \frac{1}{p} - \frac{p_\eta^{N\eta-\eta}}{p} \\
&= p_\eta(N\eta-\eta) + \frac{1}{p}
\end{aligned}
$$

$$= \quad (N-1) + \frac{1}{p}$$

G.2 RED in Byte Mode

G.2.1 Virtual Packets

Using Equation (7.8), we compute the expected loss event interval as follows:

$$\begin{aligned}
E(\theta_n) &= \sum_{m=0}^{\infty} m\, P(\theta_n = m) \\
&= \sum_{i=1}^{\infty} (N-1+\frac{i}{\eta})\, P(\theta_n = N-1+\frac{i}{\eta}) \\
&= N-1+\frac{1}{\eta}\sum_{i=1}^{\infty} i\,(1-p_s)^{i-1}p_s \\
&= N-1+\frac{1}{\eta}\,\frac{1}{{p_s}^2}\,p_s \\
&\quad \text{(From Equation (7.7))} \\
&= N-1+\frac{1}{p_S} \qquad \text{(G.1)}
\end{aligned}$$

G.2.2 Random Sampling

With this method, each packet of size $s = S/\eta$ will be sampled by the loss measurement process with a probability of:

- $p_\eta = 1/\eta$ if it is received,
- 1 if it is lost

Thus, a loss interval θ_n of size m, ($m \in I\!N$) is measured if:

- j ($0 \leq j \leq \min(m-1, N\eta-\eta)$) packets are sampled within the $N\,\eta-\eta$ packets following a loss,
- the $m-j-1$ packets are sampled among the k packets received between the end of the LIP period and the next packet loss,
- the $k+1$st packet following the LIP period is lost packet.

which is translated as follows:

$$
\begin{aligned}
P(\theta_n = m) &= \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} {p_\eta}^j (1-p_\eta)^{N\eta-\eta-j} \\
&\quad \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} {p_\eta}^{m-j-1} (1-p_\eta)^{k-(m-j-1)} (1-p_s)^k p_s \\
&= \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^j \\
&\quad \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} \left((1-p_\eta)(1-p_s)\right)^{k-(m-j-1)} \\
&\quad (1-p_s)^{m-1} (1-p_\eta)^{N\eta-\eta} {p_\eta}^{m-1} p_s \\
&= (1-p_\eta)^{N\eta-\eta} {p_\eta}^{m-1} p_s (1-p_s)^{m-1} \\
&\quad \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^j \left(\frac{1}{1-(1-p_\eta)(1-p_s)}\right)^{m-j} \\
&= \frac{p_s (1-p_\eta)^{N\eta-\eta}}{1-(1-p_\eta)(1-p_s)} \left(\frac{p_\eta(1-p_s)}{1-(1-p_\eta)(1-p_s)}\right)^{m-1} \\
&\quad \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j
\end{aligned}
$$

The expected loss interval is then given by:

$$
\begin{aligned}
E(\theta_n) &= \underbrace{\frac{p_s (1-p_\eta)^{N\eta-\eta}}{1-(1-p_\eta)(1-p_s)}}_{A} \sum_{m=1}^{\infty} m \left(\underbrace{\frac{p_\eta(1-p_s)}{1-(1-p_\eta)(1-p_s)}}_{Z}\right)^{m-1} \\
&\quad \sum_{j=0}^{\min(m-1, N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&= A \sum_{m=1}^{N\eta-\eta} m\, Z^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&\quad + A \sum_{m=N\eta-\eta+1}^{\infty} m\, Z^{m-1} \sum_{j=0}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&= \underbrace{A \sum_{j=0}^{N\eta-\eta-1} \underbrace{\sum_{m=j+1}^{N\eta-\eta} m\, Z^{m-1}}_{B} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j}_{C}
\end{aligned}
$$

$$+ A \underbrace{\sum_{m=N\eta-\eta+1}^{\infty} m\ Z^{m-1}}_{D} \left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} \tag{G.2}$$

where B, C and D are computed as follows:

$$\begin{aligned}
B &= \sum_{m=0}^{N\eta-\eta} m\ Z^{m-1} - \sum_{m=0}^{j} m\ Z^{m-1} \\
&= \frac{(N\eta-\eta)Z^{N\eta-\eta+1} - (N\eta-\eta+1)Z^{N\eta-\eta} + 1}{(1-Z)^2} - \frac{j\ Z^{j+1} - (j+1)Z^j + 1}{(1-Z)^2} \\
&= \frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)Z - (N\eta-\eta+1)\right] - \frac{Z^j}{(1-Z)^2}\left[\ j\ Z - (j+1)\right] \\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right] + \frac{Z^j}{(1-Z)^2}\left[j(1-Z)+1\right]
\end{aligned}$$

$$\begin{aligned}
C &= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right] \sum_{m=j+1}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&\quad + \frac{1}{1-Z} \sum_{m=j+1}^{N\eta-\eta} \binom{N\eta-\eta}{j} j\ \left[Z\left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)\right]^j \\
&\quad + \frac{1}{(1-Z)^2} \sum_{m=j+1}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left[Z\left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)\right]^j \\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right] \\
&\quad \left[\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} - \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^{N\eta-\eta}\right] \\
&\quad + \frac{1}{1-Z}\left[Y(N\eta-\eta)(1+Y)^{N\eta-\eta-1} - (N\eta-\eta)Y^{N\eta-\eta}\right] \\
&\quad + \frac{1}{(1-Z)^2}\left[(1+Y)^{N\eta-\eta} - Y^{N\eta-\eta}\right] \\
&\qquad \text{with} \quad Y = Z\left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right) = \frac{1}{1-p_\eta} - 1 \\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right] \left[\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} - \left(\frac{Y}{Z}\right)^{N\eta-\eta}\right] \\
&\quad - \frac{Y^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1\right] \\
&\quad + \frac{(1+Y)^{N\eta-\eta-1}}{(1-Z)^2}\left[Y(N\eta-\eta)(1-Z)+1+Y\right]
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} \\
&\quad +\frac{(1+Y)^{N\eta-\eta-1}}{(1-Z)^2}\left[Y(N\eta-\eta)(1-Z)+1+Y\right] \\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} \\
&\quad +\frac{1}{(1-Z)^2}\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[Y(N\eta-\eta)(1-Z)(1-p_\eta)+1\right]
\end{aligned}
$$

$$
\begin{aligned}
D &= \sum_{m=0}^{\infty} m\, Z^{m-1} - \sum_{m=0}^{N\eta-\eta} m\, Z^{m-1} \\
&= \frac{1}{(1-Z)^2} - \frac{(N\eta-\eta)Z^{N\eta-\eta+1} - (N\eta-\eta+1)Z^{N\eta-\eta}+1}{(1-Z)^2} \\
&= \frac{Z^{N\eta-\eta}}{(1-Z)^2}[-(N\eta-\eta)Z+(N\eta-\eta+1)] \\
&= \frac{Z^{N\eta-\eta}}{(1-Z)^2}[(N\eta-\eta)(1-Z)+1)]
\end{aligned}
$$

thus

$$
\begin{aligned}
E(\theta_n) &= A\,C + A\,D\,\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} \\
&= -A\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} \\
&\quad +A\frac{1}{(1-Z)^2}\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[Y(N\eta-\eta)(1-Z)(1-p_\eta)+1\right] \\
&\quad +A\frac{Z^{N\eta-\eta}}{(1-Z)^2}[(N\eta-\eta)(1-Z)+1)]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta} \\
&= A\frac{1}{(1-Z)}\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[Y(N\eta-\eta)(1-p_\eta)+\frac{1}{1-Z}\right] \\
&= \frac{p_s\,(1-p_\eta)^{N\eta-\eta}}{1-(1-p_\eta)(1-p_s)}\,\frac{1-(1-p_\eta)(1-p_s)}{p_s} \\
&\quad \left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[\frac{p_\eta}{1-p_\eta}(N\eta-\eta)(1-p_\eta)+\frac{1-(1-p_\eta)(1-p_s)}{p_s}\right] \\
&= N-1+\frac{p_\eta}{p_s}+(1-p_\eta) \\
&= N-1+\frac{1}{\eta p_s}+\frac{\eta-1}{\eta} \\
&= N-1+\frac{1}{p_S}+\frac{\eta-1}{\eta} \qquad \text{(G.3)}
\end{aligned}
$$

While the above loss measurement mechanism introduces a bias of $\frac{\eta-1}{\eta}$, this can easily be compensated for since η is known to the receiver.

G.3 Hypothetical RED (channel with byte errors)

Consider a packet loss process where the packet drop probability is derived from a byte loss probability. In this case, a packet is lost if one or more bytes of the packet are lost. Thus, if p_S and p_s respectively denote the probabilities for a packet of size S and $s(= S/\eta)$ to be dropped, both probabilities are related as follows:

$$p_S = 1 - (1 - p_s)^\eta \tag{G.4}$$

G.3.1 Virtual Packets

When the virtual packet algorithm is used with the hypothetical RED dropping strategy, the definition of loss event remains the same as for RED in byte mode (Definition 7.5.3) but the definition of loss event interval has to be modified as follows.

Definition G.3.1 *A* loss interval *is measured as the number of* entire *virtual packets received between two successive loss events, including the lost packet that ends the loss interval; the lost packet being counted as an* entire *virtual packet*[1].

Eventhough there is no intuitive justification for Definition G.3.1, we show later in this section that definining the loss interval in this way allows to ensure that all flows measure the same loss event rate, no matter their packet size.

Based on Definition G.3.1, the random variable θ_n is thus characterized as follows:

$$P(\theta_n = m) = \begin{cases} 0 & \text{if } m < N \\ \sum_{l=0}^{\eta-1} (1 - p_s)^{\eta i + l} p_s & \text{if } m = N + i,\ i \in I\!N_0 \end{cases} \tag{G.5}$$

[1]Note that counting the lost packet as an *entire* virtual packet and then rounding down (by taking the entire number of virtual packets between loss events) is equivalent to counting the lost packet as a small packet and then rounding up.

And the expected loss event interval is given by:

$$\begin{aligned}
E(\theta_n) &= \sum_{m=0}^{\infty} m\, P(\theta_n = m) \\
&= \sum_{i=0}^{\infty}(N+i)\sum_{l=0}^{\eta-1}(1-p_s)^{\eta i+l}p_s \\
&= \sum_{i=0}^{\infty}(N+i)p_s(1-p_s)^{\eta i}\sum_{l=0}^{\eta-1}(1-p_s)^{l} \\
&= \sum_{i=0}^{\infty}(N+i)p_s(1-p_s)^{\eta i}\frac{1-(1-p_s)^{\eta}}{p_s} \\
&= \sum_{i=0}^{\infty}(N+i)(1-p_S)^{i}p_S \\
&\quad \text{(From Equation (G.4))} \\
&= N-1+\frac{1}{p_S} \qquad \text{(G.6)}
\end{aligned}$$

In practice, θ_n as defined in Equation (G.5) represents the sum of (1) the number of virtual packets[2] that are ignored because they are part of the nth lip period i.e. $(N\,\eta-\eta)/\eta$ and (2) the number of 'entire' virtual packets received between the end of the LIP period and the next packet loss (including the lost packet, which, alone counts for a complete virtual packet). At first, it may seem surprising that the expected value of the loss interval defined in this way (using a truncated part of the interval until the next loss) is exactly equal to the desired expected value given in Equation (G.6). But this can be explained by taking a closer look at the expected number of 'entire' virtual packets received between the end of a LIP period and the next packet loss.

Let υ_n be number of virtual packets received between the end of the nth LIP period and the next packet loss (including the lost packet that counts for an entire virtual packet). $\lfloor \upsilon_n \rfloor$ is the number of 'entire' such packets and thus $\delta_n = \upsilon_n - \lfloor \upsilon_n \rfloor$ represents the truncated part of the loss interval.

We have:

$$P(\upsilon_n = 1 + \frac{m}{\eta}) = (1-p_s)^m p_s \ \ , \ m \in I\!N_0 \qquad \text{(G.7)}$$

[2]Note that the virtual packets are made up of small packets.

$$\begin{aligned}
E(\upsilon_n) &= \sum_{m=0}^{\infty}(1+\frac{m}{\eta})\,P(\upsilon_n = 1+\frac{m}{\eta}) \\
&= 1+\sum_{m=0}^{\infty}\frac{m}{\eta}\,P(\upsilon_n = 1+\frac{m}{\eta}) \\
&= 1+\frac{1}{\eta}\,(1-p_s)\sum_{m=0}^{\infty}\,m\,(1-p_s)^{m-1}p_s \\
&= 1+\frac{1}{\eta}\,(1-p_s)\frac{1}{{p_s}^2}p_s \\
&= 1+\frac{1-p_s}{\eta\,p_s} \\
&= \frac{1}{\eta\,p_s}+\frac{\eta-1}{\eta}
\end{aligned} \tag{G.8}$$

Equation (G.8) shows that the fact that we count the lost packet as an entire virtual packet introduces a bias of $\frac{\eta-1}{\eta}$ in the measurement of $E(\upsilon_n)$.

Now, δ_n is defined as follows:

$$\begin{aligned}
P(\delta_n = \frac{j}{\eta}) &= \sum_{i=1}^{\infty}P(\upsilon_n = \frac{j+\eta\,i}{\eta})\;,\; j=0\ldots\eta-1 \\
&= \sum_{i=1}^{\infty}(1-p_s)^{j+\eta(i-1)}p_s \\
&= \sum_{i=0}^{\infty}(1-p_s)^{j+\eta i}p_s \\
&= p_s(1-p_s)^j\frac{1}{1-(1-p_s)^\eta} \\
&\quad \text{(From Equation (G.4))} \\
&= p_s\,(1-p_s)^j\,\frac{1}{p_S}
\end{aligned} \tag{G.9}$$

The bias introduced by the truncation of the loss interval, δ_n, is thus given by:

$$\begin{aligned}
E(\delta_n) &= \sum_{j=0}^{\eta-1}\frac{j}{\eta}\,P(\delta_n = \frac{j}{\eta}) \\
&= \sum_{j=0}^{\eta-1}\frac{j}{\eta}\,p_s\,(1-p_s)^j\,\frac{1}{p_S} \\
&= \frac{p_s}{\eta p_S}\sum_{j=0}^{\eta-1}j\,(1-p_s)^j \\
&= \frac{p_s}{\eta\,p_S}\,\frac{1}{{p_s}^2}\,\{(\eta-1)(1-p_s)^{\eta+1}-\eta(1-p_s)^\eta+(1-p_s)\}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\eta\, p_S\, p_s}\{(\eta-1)(1-p_s)(1-p_S) - \eta(1-p_S) + (1-p_s)\} \\
&= \frac{1}{\eta\, p_S\, p_s}\{p_S + (\eta-1)\, p_S\, p_s - \eta\, p_s)\} \\
&= \frac{1}{\eta\, p_s} - \frac{1}{p_S} + \frac{\eta-1}{\eta}
\end{aligned}
\tag{G.10}
$$

Equation (G.10) shows that the bias introduced by the the truncation of the loss interval exactly compensates for (1) the bias introduced by the fact that the lost packet is counted as a virtual packet and (2) the fact that the number of virtual packets received between losses is smaller when sending small packets, i.e. $1/(\eta p_s)$ than when sending large packets, i.e. $1/p_S$. So that finally, we can write:

$$
\begin{aligned}
E(\theta_n) &= \frac{N\,\eta - \eta}{\eta} + E(\lfloor \upsilon_n \rfloor) \\
&= N - 1 + E(\upsilon_n) - E(\delta_n) \\
&= N - 1 + \frac{1}{p_S}
\end{aligned}
\tag{G.11}
$$

Bibliography

[1] Advanced topics in MPLS-TE deployment. White paper, Cisco Systems, 2001.

[2] Freephone. http:www-sop.inria.fr/rodeo/index.html.

[3] National Internet Measurement Infrastructure Infrastructure (nimi). http://www.ncne.nlanr.net/nimi/.

[4] Network simulator. Available from http:www-mash.cs.berkeley.edu.ns.

[5] PingER project. http://www-iepm.slac.stanford.edu/pinger/.

[6] Robust audio tool. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/.

[7] Sprint IP Monitoring project. http://ipmon.sprintlabs.com.

[8] Pulse code modulation (PCM) of voice frequencies. ITU-T Recommendation G.711, November 1988.

[9] Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (cs-acelp), March 1996.

[10] Control of talker echo. ITU-T Recommendation G.131, August 1996.

[11] One-way transmission time. ITU-T Recommendation G.114, February 1996.

[12] Speech processing, transmission and quality aspects (STQ); overall transmission plan aspects for telephony in private networks. ETSI Guide 201 050, February 1999.

[13] The e-model, a computational model for use in transmission plannning. ITU-T Recommendation G.107, May 2000.

[14] Provisional planning values for equipment impairment factor Ie. Appendix to ITU-T Recommendation G.113, February 2001.

[15] J. S. Ahn, P. B Dansig, Z. Liu, , and L. Yan. Evaluation of TCP Vegas: Emulation and experiment. In *ACM SIGCOMM Computer Communication Review*, volume 25, pages 185–195, 1995.

[16] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, Internet Engineering Task Force, April 1999. Obsoletes RFC 2001.

[17] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random losses. In *Proceedings of ACM Sigcomm*, Cannes, France, September 1997.

[18] R. Blahut. *Theory and Practice of Error control codes*. Addison-Wesley, 1993.

[19] J.-C. Bolot. End-to-end packet delay and loss behavior in the internet. In *Proceedings of ACM Sigcomm*, pages 189–199, August 1993. San Francisco, CA.

[20] J-C. Bolot and A. Vega Garcia. The case for FEC-based error control for packet audio in the internet. In *to appear in ACM Multimedia Systems*.

[21] J-C. Bolot and A. Vega Garcia. Control mechanisms for packet audio in the internet. In *Proceedings of IEEE Infocom*, March 1996.

[22] J-C. Bolot, S. Fosse Parisis, and D. Towsley. Adaptive FEC-based error control for internet telephony. In *Proceedings of IEEE Infocom*, March 1999.

[23] J-C. Bolot and T. Turletti. Experience with rate control mechanisms for packet video in the internet. *ACM Computer Communication Review*, 28(21), January 1998.

[24] Thomas Bonald. Comparison of TCP Reno and TCP Vegas via fluid approximation. Technical report, INRIA, 1998.

[25] Michael Borella. Measurement and interpretation of internet packet loss. *Journal of Communication and Networks*, 2(2):93–102, June 2000.

[26] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, Internet Engineering Task Force, 1998.

[27] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. RFC 1633, June 1994.

[28] L.S. Brakmo and L.L. Peterson. TCP Vegas: End to end congestion avoidance on a global internet. *IEEE J. of Selected Areas in Communication*, 13:1465–1480, 1995.

[29] L. Breslau and S. Shenker. Best-effort versus reservations: A simple comparative analysis. *Proceedings of ACM Sigcomm*, August 1998. Vancouver, Canada.

[30] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[31] R.W. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195, December 1990.

[32] J. Cleary, S. Donnelly, Ian Graham, A. McGregor, and M. Pearson. Design principles for accurate passive measurements. In *Proceedings of Passive and Active Measurement Workshop*, April 2000.

[33] S. De Cnodder, O. Elloumi, and K. Pauwels. RED behavior with different packet sizes. In *Proc. Fifth IEEE Symposium on Computers and Communications*, Antibes-Juan les Pins, France, July 2000.

[34] J. E. Cohen and F. P. Kelly. A paradox of congestion in a queueing network. *J. Appl. Prob.*, 27:730–734, 1990.

[35] R.G. Cole and J.H. Rosenbluth. Voice over IP performance monitoring. *Computer Communication Review*, 31(2):9–24, April 2001.

[36] T. Cover and J. Thomas. *Elements of Information theory*. John Wiley and Sons, 1991.

[37] R. V. Cox and P. Kroon. Low bit-rate speech coders for multimedia communication. *IEEE Communications Magazine*, pages 34–41, December 1996.

[38] Jon Crowcroft. All you need is just 1 bit. Keynote presentation in IFIP Conf. on Protocols for High Speed Networks, October 1996. http://www.cs.ucl.ac.uk/staff/jon/hipparch/dollarbit/.

[39] D. de Vleeeschauwer, J. Janssen, G. H. Petit, and F. Poppe. Quality bounds for packetized voice transport. *Alcatel Telecom Review*, pages 19–24, January 2000.

[40] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking Research and Experience*, 1(1):3–26, October 1990.

[41] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of ACM Sigcomm*, pages 109–122, September 1999.

[42] C. Perkins et al. RTP payload for redundant audio data. RFC 2198, 1997.

[43] S. Blake et al. An architecture for differentiated services. RFC 2475, December 1998.

[44] V. Jacobson et al. An expedited forwarding PHB. RFC 2598, June 1999.

[45] A. Fei, G. Pei, R. Liu, and L. Zhang. Some measurements on delay and hop-count of the Internet. In *IEEE Globecom'98*, Sydney, Australia, November 1998.

[46] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A new class of active queue management algorithms. In *Proceedings of ACM NOSSDAV*, Port Jefferson, New York, June 2001.

[47] W. Feng, D. Kandlur, D. Saha, and K. Shin. Stochastic fair blue: A queue management algorithm for enforcing fairness. In *Proceedings of IEEE Infocom*, April 2001.

[48] D. R. Figuereido and E. de Souza e Silva. Efficient mechanisms for recovering voice packets in the internet. In *IEEE Globecom'99*, pages 1830–1836, 1999.

[49] Clarence Filsfils. Sub-100ms convergence: MPLS FRR. RIPE41 meeting, January 2002.

[50] V. Firoiu, X. Zhang, and Y. Guo. Best effort differentiated services: Tradeoff service differentiation for elastic applications. In *IEEE International Conference on Telecommunications*, June 2001. http://www-net.cs.umass.edu/ vfiroiu/papers/beds-conf.pdf.

[51] Victor Firoiu and Marty Borden. A study of active queue management for congestion control. In *Proceedings of IEEE Infocom*, Tel-Aviv, Israel, March 2000.

[52] S Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communications Review*, 21(5):8–23, 1994.

[53] S. Floyd. RED: Discussions of setting parameters. http://www.icir.org/floyd/REDparameters.txt, November 1997.

[54] S. Floyd. RED: Discussions of byte and packet modes. http://www.icir.org/floyd/REDaveraging.txt, March 1997 (with additional comments from January 1998 and October 2000).

[55] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED. *under submission*, August 2001.

[56] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM Sigcomm*, pages 43 – 56, Stockholm, Sweden, August 2000.

[57] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[58] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[59] Charles Fraleigh, Sue Moon, Christophe Diot, Brian Lyles, and Fouad Tobagi. Packet-level traffic measurements from a tier-1 IP backbone. Technical Report TR01-ATL-110101, Sprint ATL, November 2001.

[60] Pascal Frossard. FEC performance in multimedia streaming. *IEEE Communications Letters*, 5(3):122–124, March 2001.

[61] O. Hagsand, K. Hanson, and I. Marsh. Measuring Internet Telephony quality: where are we today? In *IEEE Globecom'99*, pages 1843–1842, December 1999.

[62] V. Hardman, A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the internet. In *Proceedings of INET'95*, June 1995.

[63] G. Hasegawa, M. Murata, and H. Miyahara. Fairness and stability of congestion control mechanisms of TCP. In *11th ITC Specialist Seminar*, pages 255–262, October 1998.

[64] G. Hasegawa, M. Murata, and H. Miyahara. Fairness and stability of congestion control mechanisms of TCP. In *IEEE Globecom'99*, pages 1329–1336, 1999.

[65] P. Hurley, J.-Y. Le Boudec, and P. Thiran. The asymmetric best-effort service. In *IEEE Globecom 1999*, Rio de Janeiro, Brazil, December 1999.

[66] 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.

[67] S. Jacobs and A. Eleftheriadis. Providing video services over networks without quality of service guarantees. In *RTMW'96*, October 1996. Sophia Antipolis, France.

[68] N. Jayant. Effects of packet loss on waveform coded speech. In *Proc. 5th Int. Conference on Computer Communications*, pages 275–280, October 1980.

[69] W. Jiang and H. Schulzrinne. Qos measurement of internet real-time multimedia services. Technical Report CUCS-015-99, Columbia University, December 1999.

[70] W. Jiang and H. Schulzrinne. Comparison and optimization of packet loss repair methods on VoIP perceived quality under bursty loss. In *Proceedings of ACM NOSSDAV*, pages 73–81, May 2002.

[71] Wcuyu Jiang and Henning Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proceedings of ACM NOSSDAV*, Jun 2000.

[72] S. Kalidindi and M.J. Zakauskas. Surveyor: an infrastructure for internet performance measurements. In *Proceedings of the Internet Society Conference (INET)*, June 1999.

[73] S. Keshav. *An Engineering Approach to Computer Networking*. Addison-Wesley Publishing Company, 1997.

[74] Nobuhiko Kiatawaki and Kenzo Itoh. Pure delay effect on speech quality in telecommunications. *IEEE Journal on Selected Areas in Communications*, 9(4):586–593, May 1991.

[75] K. Kilkki and J. Ruutu. Simple integrated media access - an internet service based on priorities. In *6th International Conference on Telecommunication Systems*, March 1998.

[76] J. Mahdavi and S. Floyd. TCP-Friendly unicast rate-based flow control. In *Draft posted on end2end mailing list*, Jannuary 1997. http://www.psc.edu/networking/papers/tcp_friendly.html.

[77] Athina Markopoulou, Fouad Tobagi, and Mansour Karam. Assessment of VoIP quality over Internet Backbones. In *Proceedings of IEEE Infocom*, June 2002.

[78] M. Mathis, J. Semke, J. Madhavi, and T. Ott. Macroscopic behavior of TCP congestion avoidance algorithm. *Computer Communication Review*, 27(3), July 1997.

[79] M. May, J. Bolot, C. Diot, and A. Jean-Marie. 1-bit schemes for service discrimination in the internet: Ananlysis and evaluation. Technical report 3238, INRIA, 1997.

[80] Jorg Micheel, Stephen Donnelly, and Ian Graham. Precision timestamping of network packets. In *Proceedings of ACM Sigcomm Internet Measurement Workshop*, November 2001.

[81] J. Mo, R.J. La, V. Anantharam, and J. Walrand. Analysis and comparison of TCP Reno and Vegas. In *IEEE Globecom'99*, 1999.

[82] J. C. Mogul and S. E. Deering. Path MTU discovery. RFC 1191, Internet Engineering Task Force, November 1990.

[83] S. B. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustment: Performance bounds and algorithms. *ACM/Springer Multimedia Systems*, 5:17–28, Januray 1998.

[84] Y. Moret and S. Fdida. A proportional queue control mechanism to provide differentiated services. In *International Symposium on Computer System*, Belek, Turkey, 1998 October.

[85] Richard Mortier. Python routing toolkit, 2002. http://www.sprintlabs.com/Department/IP-Interworking/Routing.

[86] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Relative delay differentiation and delay class adaptation in core-stateless networks. In *Proceedings of IEEE Infocom*, March 2000.

[87] D. Oran. OSI IS-IS intra-domain routing protocol. RFC 1142, February 1990.

[88] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM Sigcomm*, 1998.

[89] Jitendra Padhye, Victor Firoiu, Donald F. Towsley, and James F. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, April 2000.

[90] Dina Papagiannaki, Sue Moon, Chuck Fraleigh, Patrick Thiran, Fouad Tobagi, and Christophe Diot. Analysis of measured single-hop delay from an operational backbone network. In *Proceedings of IEEE Infocom*, June 2002.

[91] V. Paxson. End-to-end internet packet dynamics. In *Proceedings of ACM Sigcomm*, September 1997. Cannes, France.

[92] C. S. Perkins, O. Hodson, and V. Hardman. A survey of packet-loss recovery techniques for streaming audio. September/October 1998.

[93] M. Podolsky, C. Romer, and S. McCanne. Simulation of fec-based error control for packet audio in the internet. In *Proceedings of IEEE Infocom*, April 1998.

[94] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for qos management. In *IEEE Real-Time Systems Symposium*, December 1997.

[95] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of IEEE Infocom*, 1994.

[96] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proceedings of IEEE Infocom*, March 1999. New York.

[97] I. Rhee, V. Ozdemir, and Y. Yi. TEAR: TCP emulation at receivers - flow control for multimedia streaming. Technical report, Department of Computer Science, NCSU, April 2000.

[98] Eric C. Rosen, Arun Viswanathan, and Ross Callon. Multiprotocol Label Switching Architecture. RFC 3031, January 2001.

[99] J. Rosenberg, L. Qiu, and H. Schulzrinne. Integrating packet FEC into adaptive voice playout buffer algorithms on the internet. In *Proceedings of IEEE Infocom*, March 2000.

[100] H. Sanneck. Concealment of lost speech packets using adaptive packetization. In *Proceedings IEEE Multimedia Systems 1998*, Austin, TX, June 1998.

[101] H. Sanneck and N. Le. Speech property-based FEC for Internet Telephony applications. In *Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference (MMCN)*, pages 38–51, San Jose, CA, January 2000. ftp://ftp.fokus.gmd.de/pub/glone/papers/Sann0001:Speech-FEC.ps.gz.

[102] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, 1996.

[103] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for Real-Time Applications. INTERNET-DRAFT - draft-ietf-avt-rtp-new-08.ps, July 2000.

[104] N. Shacham and P. Mc Kenney. Packet recovery in high-speed networks using coding and buffer management. In *Proceedings of IEEE Infocom*, volume 1, pages 124–131, June 1990.

[105] D. Sisalem and H. Schulzrinne. The loss-delay adjustment algorithm: A TCP-Friendly adaptation scheme. In *Proceedings of ACM NOSSDAV*, July 1998. Cambridge,UK.

[106] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM Sigcomm*, September 1999.

[107] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *Proceedings of ACM Sigcomm*, Vancouver, British Columbia, September 1998.

[108] W. Tan and A. Zakhor. Error resilient packet video for the internet. In *ICIP'98*, volume 3, pages 458–462, October 1998.

[109] J. Tribolet, P. Noll, B. McDermott, and R. Crochiere. A study of complexity and quality of speech waveform coders. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, pages 586–590, April 1978. Tulsa.

[110] V. A. Vaishampayan. Design of multiple description scalar quantizers. *IEEE Transactions on Information Theory*, 39(3):821–834, May 1993.

[111] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. In *Proceedings of ACM Sigcomm*, pages 275 – 286, San Diego, CA, August 2001.

[112] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of temporal dependence in packet loss. In *Proceedings of IEEE Infocom*, March 1999. New York, NY.

List of Figures

List of Tables

Curriculum Vitae

Catherine Boutremans was born in Belgium in 1975. She received the degree of Electrical Engineer in Telecommunications from the Faculté Polytechnique de Mons in July 1997. From January to June 1997, she participated in an Erasmus exchange programme and she completed her Master's Thesis in the Signal Processing laboratory (LTS) at the Swiss Federal Institute of Technology (Ecole Polytechnique Fédérale de Lausanne, EPFL). From 1997 to 1998 she attended the doctoral school in Communication Systems (SSC) at EPFL where she was awarded with a 3-year full scholarship of the SSC. In October 1998 she joined the Institute for computer Communications and Applications (ICA) which later became the Laboratory for computer Communications and Applications (LCA). She initially worked on congestion control algorithms, focusing fairness issues. She then started to work on error control for Internet Telephony, addressing the problems related to large delays in interactive communications. Since 2001, she has also been working on a Voice over IP measurement project in conjunction with Sprint Advanced Technology Labs where, in the summer/autumn of 2001, she undertook an internship.

Publications

- C. Boutremans, J. Y. Le Boudec. “Audio on Non Elevated Services”. Submitted for publication.
- J. Widmer, C. Boutremans, J. Y. Le Boudec. “End-to-End Congestion Control for Flows with Variable Packet Size”. Submitted for publication.
- C. Boutremans, J. Y. Le Boudec. “Adaptive joint playout buffer and FEC adjustement for Internet Telephony”. To appear in *Proceedings of IEEE Infocom*, San Francisco, California, April 2003.
- C. Boutremans, G. Iannaccone, C. Diot. “Impact of link failures on VoIP performance”. In *Proceedings of ACM NOSSDAV*, Miami, Florida, May 2002.
- C. Boutremans, J. Y. Le Boudec. “Adaptive Delay Aware Error Control for Internet Telephony ”. In *Proceedings of 2nd IP-Telephony workshop*, Columbia University, New York, April 2001, pp. 81-92.
- A. Meylan, C. Boutremans. “Realisation of an Adaptive Audio Tool”. In *Proceedings of 7th International Workshop*, IDMS 2000, Enschede, The Netherlands, October 2000.
- M. Vojnovic, J. Y. Le Boudec, C. Boutremans. “Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times”. In *Proceedings of IEEE Infocom*, Tel Aviv, Israel, March 2000, pp. 1303-1312.
- C. Boutremans, J. Y. Le Boudec. “A Note on the Fairness of TCP Vegas”. In *Proceedings of International Zurich Seminar on Broadband Communications*, Zurich, Switzerland, February 2000, pp. 163-170.