Université de Nice - Sophia Antipolis – UFR Sciences
École Doctorale STIC

THÈSE

Présentée pour obtenir le titre de :
Docteur en Sciences de l'Université de Nice - Sophia Antipolis

Spécialité : INFORMATIQUE

par

Víctor Manuel RAMOS RAMOS

Équipe d'accueil : Planète–INRIA (Doctorant invité)

Transmission robuste et fiable du multimédia sur Internet

Soutenue publiquement à l'ESSI le 7 décembre 2004 devant le jury composé de :

Directeurs :	Eitan	ALTMAN	INRIA
	Chadi	BARAKAT	INRIA
Rapporteurs :	Daniel	KOFMAN	ENST
	Uri	YECHIALI	Université de Tel-Aviv
Examinateurs :	Chadi	BARAKAT	INRIA
	Bruno	TUFFIN	INRIA/IRISA

Transmission robuste et fiable du multimédia sur Internet

Víctor Manuel Ramos Ramos

Titre de la thèse en anglais :

Robust and reliable multimedia transmission over the Internet

A mis padres y a Carolina ...

Remerciements

Les mots ne suffiraient pas pour exprimer toute ma reconnaissance envers Eitan Altman, mon directeur de thèse. Ses conseils et ses encouragements ont permis à ce travail d'aboutir. Faire ma thèse sous sa direction a été pour moi un grand honneur. Je le remercie pour sa disponibilité et pour sa patience. Merci d'avoir répondu à toutes mes questions, même les plus simples. La liberté qu'il m'a accordée et les responsabilités qu'il m'a confiées m'ont permis d'atteindre une maturité professionnelle que je n'aurais pas imaginée auparavant.

Je tiens à remercier vivement Chadi Barakat, mon co-directeur de thèse. Travailler avec lui a été un grand plaisir. Je le remercie également pour les responsabilités qu'il m'a confiées, pour m'avoir considéré comme son égal. Cela m'a permis de mener mon travail à bien. Je le remercie pour son grand soutien tout au long de cette thèse même aux moments les plus difficiles.

Un grand merci à Denis Caromel pour avoir accepté de présider mon jury de thèse. Je remercie aussi Uri Yechiali et Daniel Kofman pour m'avoir fait l'honneur d'écrire un rapport sur ma thèse. Je remercie également Bruno Tuffin, chercheur du projet Armor, pour sa participation à mon jury de thèse. Je le remercie égalément de m'avoir accueilli au sein de son équipe et de la souplesse qu'il m'a accordée. Travailler avec lui a été pour moi une excellente expérience dans ma formation comme chercheur.

Merci à Walid Dabbous, responsable du projet Planète. Je le remercie de m'avoir accueilli au sein de son équipe en tant que doctorant invité et de m'avoir soutenu au moment le plus délicat de ma thèse.

Merci à Tania Jiménez pour sa grande amitié et pour ses encouragements tout au long de cette thèse. Merci de m'avoir accueilli avec Eitan au Cesimo au Vénézuela.

Je remercie vivement tous mes amis et collègues, ici, au Mexique et ailleurs. Leur amitié et leur sympathie m'ont aidé à surmonter beaucoup d'épreuves. Je remercie particulièrement Urtzi Ayesta, Florence Clévenot, Marie-Cécile Lafont, Yezekael Hayel et Joanna Moulierac pour leurs remarques et commentaires,

Mes remerciments vont égalément à mes parents et mes soeurs. Merci pour votre soutien inconditionnel, merci pour m'avoir appris à mener les choses jusqu'au bout.

Merci également à la Universidad Autonónoma Metropolitana et au CONACyT de m'avoir soutenu au long de ma thèse. Je remercie particulièrement Miguel Peña et Miguel Cadena, pour leur soutien inconditionnel et pour leur professionnalisme.

Merci de tout mon coeur à Carolina pour avoir été à mes côtés à tout moment, pour son soutien, pour son amour. Elle a été ma plus grande motivation au long de toutes ces années. Merci pour avoir mis au monde Gaëthan et pour me rendre heureux comme tu le fais.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since its creation, the Internet has known a huge success. This is mainly due to the simplicity of its architecture which is based on the Internet Protocol (IP). IP provides a simple best-effort service, implementing a datagram delivery service without any guarantee on the order of delivery, the integrity of such datagrams, or the arrival of each datagram to its destination [77]. This simplicity allows many different types of networks to interconnect together and to form what we know as the Internet.

In order to circumvent the simplicity and disadvantages of IP, the Transmission Control Protocol (TCP) provides a reliable in-order data delivery to the application layer [78]. The main tasks of the TCP protocol are end-to-end flow control, error control, and congestion control. End-to-end flow control means that a sender may not inject into the network more data than the receiver can hold. The error control task consists in the recovery from any loss in the network, this task is done through retransmission based on sequence numbers, positive acknowledgments, and a retransmission timer. The congestion control task consists in adapting the transmission rate of a source according to the network load [42]. The TCP protocol accomplishes these tasks in an end-to-end fashion without any modification of the network; it is in this way that data transmission in the Internet is assured by TCP in a reliable way.

The concept of an integrated services network has been of interest for already quite a longtime [98]. On one hand, a new telephone backbone would provide several quality of service (QoS) levels for carrying different types of traffic. On the other hand, the existing well deployed worldwide Internet would carry voice and video traffic, as well as data. Using

the Internet to carry multimedia traffic is of great interest since the network is already deployed [46].

The Internet was designed initially to carry data traffic by using TCP on the top of IP. The transmission of real-time traffic is a bigger research challenge since it demands very severe delay and quality of service (QoS) constraints. Using TCP is not adequate for carrying this kind of traffic because it introduces delay variability due to retransmission and reordering. Besides, the TCP's congestion control mechanisms introduce rate variability too. Real-time applications are sensitive to delay and rate; moreover, they can tolerate some losses and do not require full reliability.

Several standards exist for transmitting multimedia in packet-switched networks. The dominant standard for multimedia traffic is H.323, which uses RTP encapsulation for audio over UDP/IP [107]. RTP (Real-time Transport Protocol) provides end-to-end network transport functions suitable for real-time multimedia applications, over multicast or unicast network services [92]. RTP is used together with the real-time transport control protocol (RTCP) to provide the application layer statistics about a session such as packet loss rate, jitter, etc. These statistics are useful for multimedia applications for being "fair" with other flows sharing the same link [18, 19, 52, 64]. In this way, they can adapt their transmission rate in a similar way TCP does by using TCP-friendly like mechanisms.

One of the most attractive and already functional real-time services that is being provided is the transmission of voice over the Internet, which is called Internet Telephony (VoIP: Voice over IP) [39, 102]. Transmitting real-time voice over the Internet has several advantages. One major advantage is price. Placing a voice call over a packet switched network like the Internet is much cheaper than making it through the traditional PSTN[1] telephone network. Additional advantages are the possibility of sharing files and white-boards, having secured calls through encryption, caller identification, etc. There are also several advantages from the carrier perspective from using the Internet: an integration of different types of traffic would be easier than with PSTN networks, cheaper switching, better and robust management platforms.

The basic model of VoIP supposes that users participating in a voice conference have access to multimedia computers connected to the Internet. So, the users can be connected through a classic modem connection, an ADSL line, a LAN, or even a wireless connection. Further, if more than two users participate in a conversation a multicast session is normally established.

The general steps involved during a voice session are illustrated in Figure 1.1. The speech is sampled, digitized, packetized, and transmitted to the other end. Then, the receiver application holds the received packets in a buffer, and thus it plays them out

[1]PSTN stands for Public Switched Telephone Network.

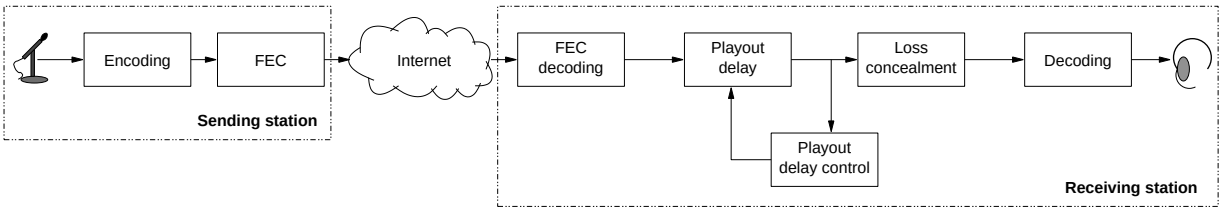


Figure 1.1: General steps in an audio session.

through a sound card. Forward Error Correction (FEC) is normally added to the transmitted packets in order to reduce the loss process at the receiver.

Figure 1.2 depicts the basic scenario of VoIP. Here, two computers are used to establish an audio session and to transmit packets across the network. This basic scenario for VoIP has evolved into two other possible scenarios depicted in Figures 1.3 and 1.4. In Figure 1.3 a computer is connected to the Internet and transmits voice packets to a normal telephone device. The telephone is connected to the central office switch, which connects to the Internet through an Internet telephony gateway. Figure 1.4 shows two standard telephones being used to place a VoIP call. In this case, the user calls the Internet telephony gateway, and based on the caller's ID the user is recognized and logged into the telephony system, then the user types the recipient's number. Next, the gateway starts an H.323 session with the IP address of the recipient's gateway, which knows how to place a call to the receiver's phone. A voice session is next initiated by sending voice in IP packets between both gateways. In this case, the packetization, encoding, decoding, and reassembly processes occur in the gateways.

Two major challenges are imposed by the VoIP architectures just described:

1. The first one is the **implementation** of a VoIP system. Since a VoIP device (a computer or an IP telephone) must be capable of supporting several codecs, the processing power of a VoIP device is much more demanding than for a traditional telephone. Echo cancellation is another problem associated with digital voice transmission. When the VoIP architecture involves at least one telephone, the usual

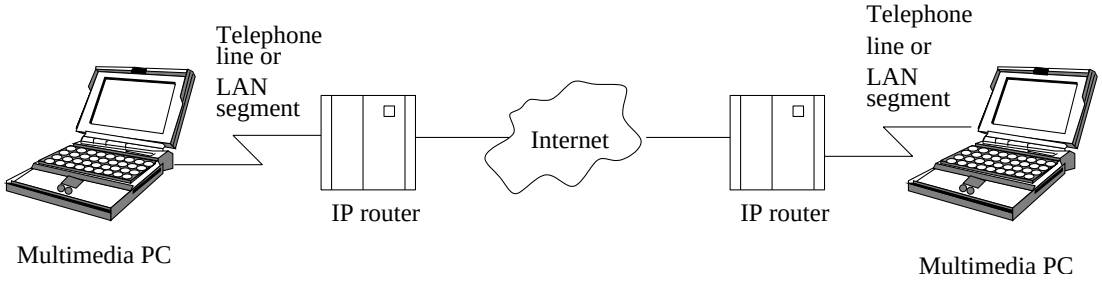


Figure 1.2: Basic model of VoIP.

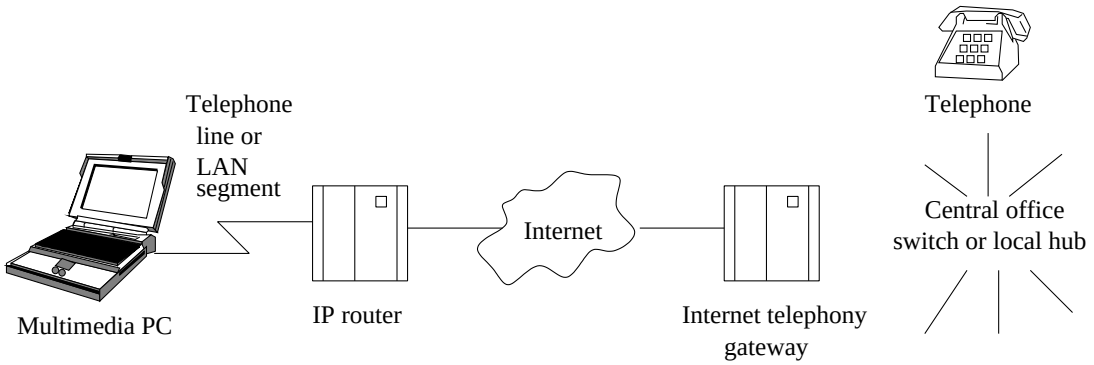


Figure 1.3: PC to phone architecture.

far-end echo is caused by the four-wire to two-wire conversion. In this case, adaptive cancellation is applied to reduce the effects of echo. When there is at least one computer involved in the VoIP architecture, echo can be produced if the user is using just a microphone and speakers; in this case, the user at the other end will hear his/her own voice since the speech is transmitted back by the speakers. In this case, the problem can be solved by using headphones instead of just using speakers [14].

There is another couple of problems related to the implementation of a VoIP architecture. The first one is the support of Dual Tone Multifrequency (DTMF) transmission, and clock synchronization. DTMF can be supported through an input indication (H.245) with H.323, or more recently with SIP (Session Initiation Protocol) [93]. Clock synchronization is a desired characteristic in any VoIP session since it allows to reduce clock drift between sender and receiver [41, 62, 66, 70, 116]. Synchronization can be done with NTP (the Network Time Protocol), or with GSM (Global System for Mobile communications), the latter being the most expensive. Figure 1.5 depicts a testbed used in [62] to synchronize the clocks of two hosts using

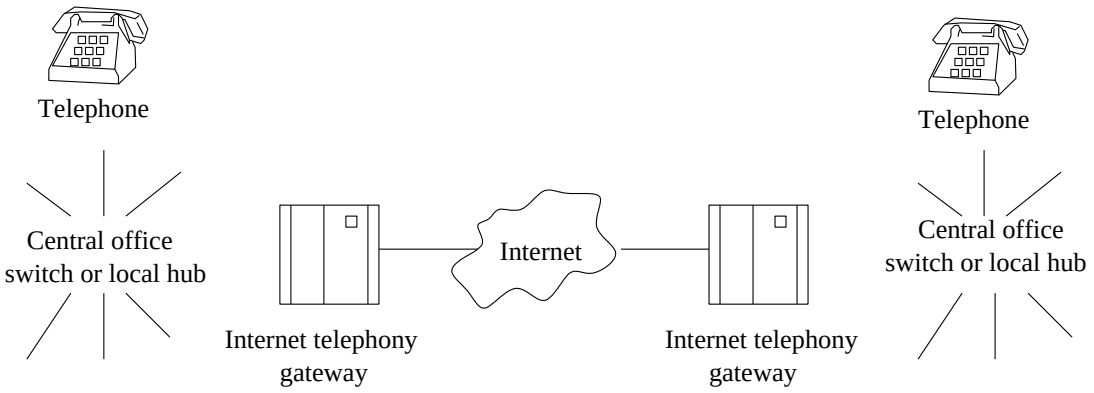


Figure 1.4: Phone to phone architecture.

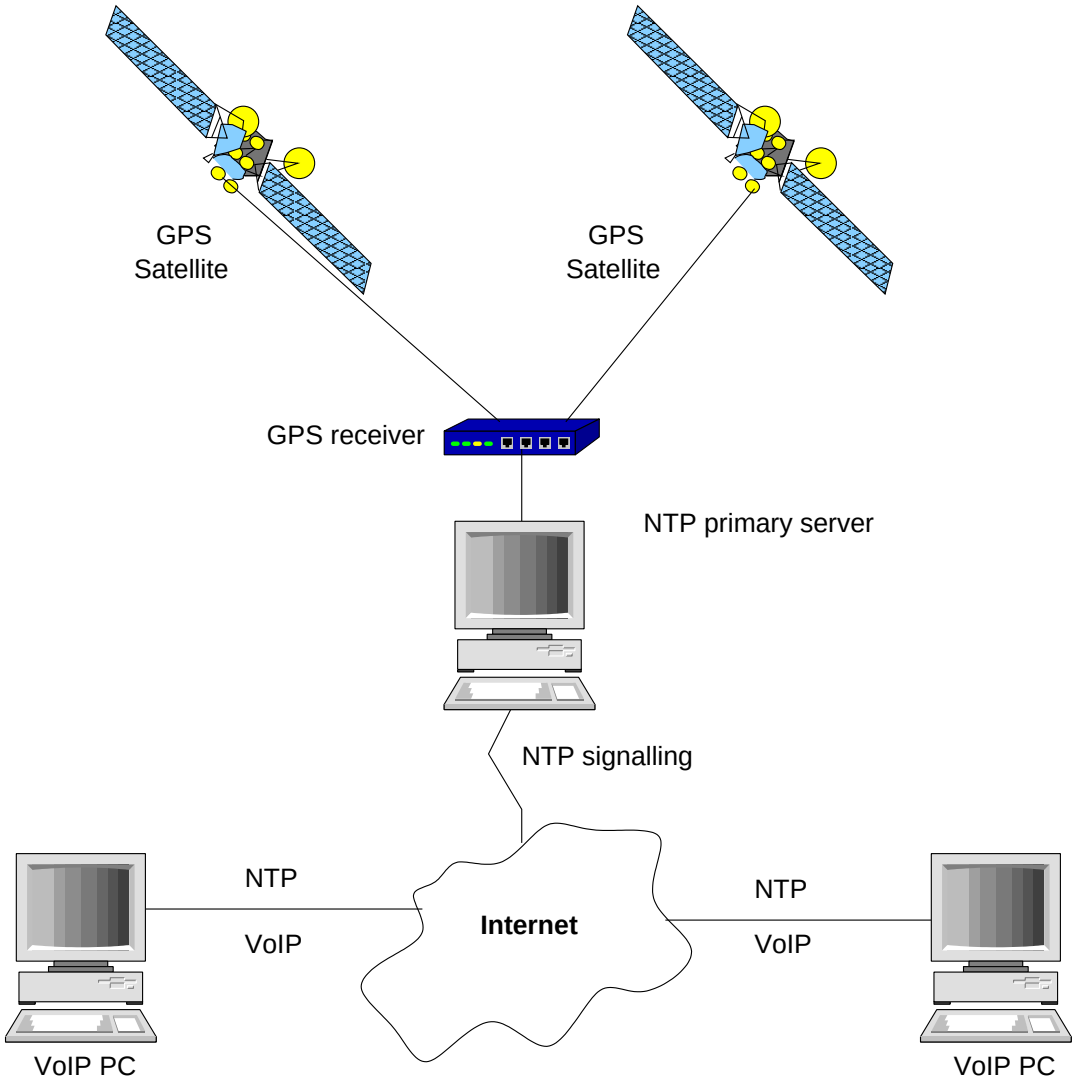


Figure 1.5: NTP synchronization.

NTP. There is a time server using GPS, which provides the time source for the NTP server (middle).

Clock synchronization is beneficial for the performance of playout delay control algorithms, which will be studied in detail in Chapter 4.

2. The second challenge, and the most difficult to solve is the provision of **quality of service**. Transmitting packetized voice over a best-effort network like the Internet poses several problems for providing an acceptable QoS. The main factors affecting the speech quality during a VoIP session are:

   Network bandwidth. A user participating in a VoIP call must have a minimum

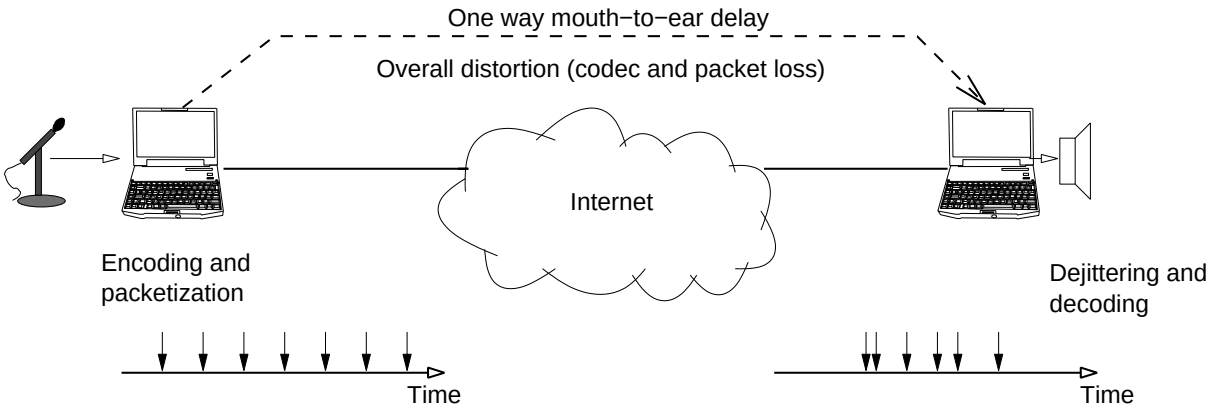


Figure 1.6: Delay components in a VoIP session.

bandwidth available in order to establish and keep a call with an acceptable quality. A user connecting to the Internet with a 33kbps modem would be more limited than a user connecting through an ADSL, or a LAN connection.

Codecs. Since network bandwidth is variable, it is desirable that VoIP applications be fair with other applications sharing the same bandwidth. So, a non TCP application such as VoIP should be "TCP friendly" and adapt its transmission rate according to the available network bandwidth. This is normally done by switching between coders of different rates during a VoIP call according to the state of the network. Thus, when network bandwidth is scarce an application can switch to a lower rate codec, switching back later to a higher rate codec when the network state changes [19, 61, 68].

Delay. Several kinds of delay play a role in IP telephony, some delays are fixed and some others are not. First, there is a constant component of delay composed of quantization, packetization, transmission, and decoding delay. Delays in this set are fixed and depend on the codec used and on the packet size, they can be further reduced if they are implemented in hardware as in the case of IP phone devices. The second class of delay is variable, it is composed of the end-to-end delay plus the playout delay at the receiver [82]. The end-to-end delay varies according to the network state. When the network is congested, packets will take longer to arrive at the receiver. This introduces variability on the end-to-end delay, also known as jitter. Figure 1.6 shows roughly the different types of delay incurred in a VoIP session.

Applications must resort to playout delay algorithms in order to eliminate the introduced jitter [1, 28, 32, 54, 65, 83, 97, 103]. The playout delay is the delay that packets spent in the receiver's buffer. The main task of a playout delay

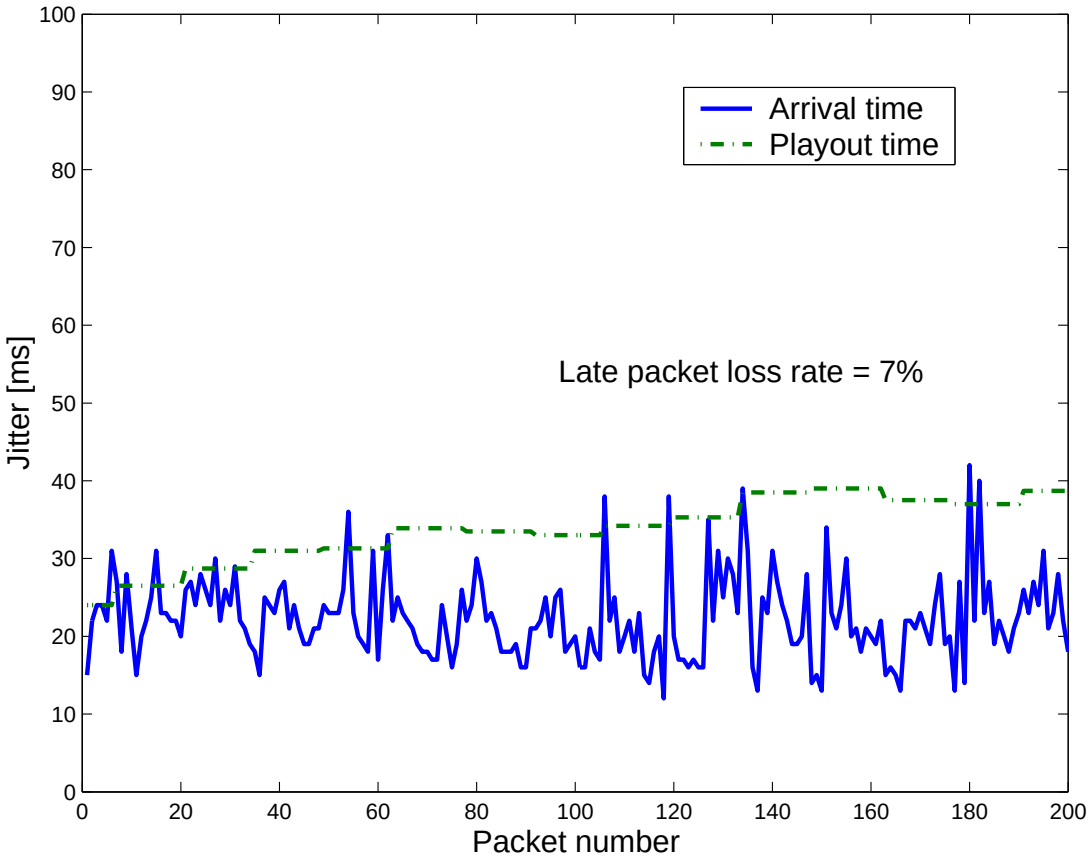


Figure 1.7: Basic operation of a playout delay control algorithm.

algorithm is to trade delay for loss, while at the same time smoothing jitter. Figure 1.7 shows the performance of playout algorithms in [82]. The y-axis shows the packet's interarrival time, and the x-axis shows the packet sequence number during an audio session. Packets with a jitter value over the playout time line are considered lost and are not played out. This increases the overall loss rate but improves the session's interactivity. In Chapter 4 we study in detail playout delay algorithms, and we propose a new approach allowing to control the packet loss percentage seen by an application.

Packet loss. In order to reduce the effects of packet loss, a VoIP system has to resort to sender-based as well as to receiver-based mechanisms. The most common example of sender-based approaches are Forward Error Correction (FEC) techniques [14, 39, 72]. FEC mechanisms work generally by piggybacking redundant versions of the original packets on later packets, so if a packet is lost it can be recovered by waiting for its redundancy to arrive. Another class of FEC mechanisms uses parity coding [31]. Packets are divided into groups of a

given size, and an exclusive-or (XOR) operation is computed over each group. The resulting packet is transmitted together with the next group piggybacked on the first packet. This scheme produces a packet with its size doubled in each group and allows to recover from a single loss. Both FEC schemes introduce additional delay but they allow to reduce the overall loss rate. Additionally, receiver-based mechanisms are often used in order to further reduce the loss rate perceived by an application. Examples of receiver-based mechanisms are insertion, interpolation, and regeneration [38, 72, 88–90]. In Chapter 2 we study through a queuing analysis the performance of a simple FEC mechanism that has been implemented in popular audio tools.

The three phenomena just described are unavoidable in the current Internet and must be compensated by the techniques described above. Moreover, loss and delay are variable which makes even more difficult to provide an acceptable quality of service.

In the last years, different research works have been done on every aspect of real-time multimedia applications. We next identify some well defined research areas in this domain:

Loss and end-to-end delay. Packet loss and delay are often correlated. If packet i is lost it is likely that packet $i+1$ be lost too. Bursty network losses affect the quality of real-time multimedia applications, like VoIP, Internet radio, video conferencing, etc. The dependency between delay and loss has been largely studied, and mechanisms to reduce their effects have been proposed and evaluated [11, 15, 16, 37–39, 52, 63]. In [15], the authors assume that the impact of Internet traffic on a stream of packets sent at regular intervals can be approximated by a batch Bernoulli stream. They propose a FEC mechanism for recovering from losses, and they build the first audio tool able to adapt its transmission rate based on the network state: the Freephone audio tool [34]. More recently, Jiang and Schulzrinne show in [47] that for real-time multimedia sessions with bursty losses the final loss pattern after playout delay and FEC is still bursty. The FEC mechanism proposed in [15] has been implemented in audio tools like Freephone and Rat. The performance of this FEC scheme has been largely studied by modeling and simulation.

Playout delay algorithms. The playout algorithms proposed by Ramjee et al. in [82] are the pioneering reference on this research area. They propose four algorithms for adapting the playout delay at the beginning of a talkspurt in an audio session. They also identify delay spikes and propose a spike detection algorithm for adjusting playout delay. The lack of a parameter for controlling the loss rate in a session inspired a series of works aiming to improve the performance of Ramjee's algorithms [49, 65, 73].

The general idea is to keep track of the packet delay distribution, and to adapt the playout delay on an histogram-like fashion based on the desired loss rate per session. Another class of playout algorithms have been proposed in [55, 57–59, 113]. This class of playout algorithms is very dynamic since they adapt the playout delay in a packet-base fashion. Packets can be played out almost as they arrive by using a technique called time scale modification [99, 108, 114].

Protocols. Several protocols exist for real-time multimedia over the Internet. Protocols for transmission, gateway location [85], signaling [84], QoS statistics reports [92], and synchronization are currently used and deployed. Protocols for gateway location and protocols for signaling are very active directions in this research area. If an IP host wants to call a PSTN user, it must know the IP address of an appropriate gateway. This is made by a *gateway location protocol*. If the gateway's choice is not good, the call's quality can be poor. So, an efficient gateway location protocol aims to find the optimum gateway for placing a call.

In VoIP, since interoperability with PSTN networks is desired, a robust signalling protocol allowing interconnection between both kinds of networks is needed. H.323 and SIP are the most used protocols. H.323 has been used for a long time, but SIP is now being preferred by implementors because it is less complex, easier to learn, it has a lower connection setup latency, and provides a higher interoperability than H.323.

Measuring QoS. As we said before, the main factors affecting the quality of real-time multimedia applications are packet loss, delay and jitter. Understanding the behavior and the dependency between these phenomena is crucial for designing a system that provides a good quality of service.

Now, how can we measure the quality of a real-time multimedia application during a session? There exist in general two ways of doing this. The first one is by reducing the overall loss rate and jitter seen by an application, and keeping these parameters within tolerable levels; thus one can measure, for example, the quality of service by the final average loss percentage or the average playout delay in a given session. We say that QoS is measured *objectively* in this case, since a measurable quantity (the loss rate) is reported by the application to indicate the quality seen during a given session [36, 65]. Another way of measuring quality is by thinking on the final user. In this case a multimedia session is first recorded. There are two signals to be compared: the original transmitted signal, and the received signal corrupted by loss, delay and jitter. The original signal is known as the reference signal. Then, a group of users is given both signals and they must compare them in a *subjective* way after

hearing both. A given session is rated in the range [1 . . . 5]. A user gives a score of 5 when he/she does not perceive any difference between the corrupted signal and the reference signal. So, 5 is given for the best quality, 4 is given for high quality, and so on, with 1 being for unacceptable quality. This subjective way of measuring quality is known as MOS (Mean Opinion Score) [23, 60, 86, 95, 115].

In VoIP, much efforts have been devoted for standardizing ways of measuring quality. For efficiently measuring quality in the subjective way we just described, MOS has to be computed over a large number of user scores, which is not an easy task. So, the general idea in VoIP is to *predict* the subjective quality of a telephone call based on its transmission parameters [48]. Two standards have been proposed by the ITU, the E-Model described in the ITU-T Recommendation G.107 [105], and PESQ (Perceptual Evaluation of Voice Quality) described in the ITU-T Recommendation P.862 [106]. Both standards differ mainly in the assumptions they make for predicting subjective quality. For example, the E-Model does not take into account the dynamics of a transmission but relies on static transmission parameters. PESQ considers playout adaptation but does not include absolute delay into its ranking. PESQ assesses the quality of a voice call entirely by the speech quality. However, one has tendency to compare a VoIP call with a normal PSTN call. The E-model takes this into account, and it is often preferred by researchers since it includes several factors that PESQ does not. For example, the E-model assumes that impairment sources which are not correlated can be added on a psychological scale.

Efforts have been made to use the E-model for IP telephony. In [27], Cole and Rosenbluth propose a simplification of the E-model for using it on VoIP systems. They identify the main transport parameters as loss, delay, and jitter and adapt them to the computation of the R-factor. The R-factor on the E-model gives the overall quality on a given audio session. It captures the effects of noise, echo, quantization, delay, special equipments, and certain expectation factors concerning the final user. Similar proposals for simplifying the E-model for its use on packet-switched networks have been proposed in [43, 44, 109–111].

Pricing. The research work on this direction is on pricing schemes for different types of quality of service. For instance, in real-time multimedia applications, a user would not be willing to pay the same price for a constant and high quality session than for a session with variable quality. The best-effort nature of the Internet makes even more interesting this area of research, since in the absence of QoS facilities from the network, pricing is a very big challenge. In [22], Caesar et al. compare several pricing strategies: flat (FL), congestion sensitive (CS), QoS sensitive (QoSS), and a

hybrid scheme that is sensitive to congestion and to QoS (CSQoSS). The study is based on a two class user model: type I users pay any price for the best QoS and type II users request the best QoS at a cost that is less than some maximum price they are willing to pay. In [33], Ganesh and Laevens propose a scheme for which a router writes a social cost on each packet as they pass through. Users switch demand based on this cost. The model maximizes a utility function comprising the value a user gives to bandwidth. A definition of congestion price is proposed as a function of the aggregate data arriving on the link in that slot. The authors show by simulation that transmission rates and prices converge on the long term.

Contributions of this thesis

This thesis is organized in two parts. In the first part we study the performance of interactive real-time applications. In the second part, we analyze the performance of AIMD protocols under variable delay.

First part: Real-time applications

Applications like NeVoT, Vat, Freephone, and Rat have been extensively used by researchers to conduct multicast and unicast voice and video conferences on the Internet. Dependencies between loss and delay have been largely studied over the years, and mechanisms for reducing their effects have been proposed and implemented in real-time tools. One of the goals of the first part of this thesis is to evaluate the performance of these mechanisms on real-time applications. We study in Chapter 2 the performance of a FEC scheme implemented in tools like Freephone and Rat. In these simple schemes, packet i is called the primary copy. A different version of packet i with a larger compression ratio is generated and it is piggybacked on packet $i+\phi$; this is called the redundant copy of packet i. For example, the primary copy can be encoded with PCM μ-law and the redundant copy with a lower bandwidth version code like GSM or LPC. If the primary copy of packet i is lost, then it can still be played out by waiting ϕ packets and decoding its redundancy, given that packet $i+\phi$ is not lost. Since losses on the Internet are often bursty, shifting the redundancy ϕ units apart from the original packet is supposed to reduce the burstiness of the final loss process after FEC decoding. We assume that the quality of the received copy of a packet depends linearly on the amount of redundancy. Then, we show through a queuing analysis that the amount of useful information at the receiver deteriorates for any amount of FEC and for any value of ϕ [5]. This is a very important result, since it means that for this scheme adding FEC is harmful for a VoIP application at the receiver.

Next, in Chapter 3 we generalize our queuing model and we relax the assumptions

we made in the previous chapter on the linear utility function. We consider two aspects that may contribute to a quality improvement: multiplexing with other flows, and the use of non-linear functions relating the quality and the amount of redundancy. The use of non-linear functions in multimedia applications is justified by the fact that the quality of a multimedia session is quite a subjective measure, and it depends on a large number of parameters. Similar utility functions as those we propose have been used in the past relating the quality of a real-time multimedia session and the transmission rate [96]. Besides, we generalize our bottleneck's queuing model to an $M/G/1/K$ queue with an audio flow and an exogenous flow. We find conditions for which adding FEC leads to a quality improvement [4, 6]. Note that although we propose models for a FEC mechanism implemented in audio tools, our models are also valid for any real-time multimedia application using this kind of loss recovery approach.

In Chapter 4 we focus on algorithms for playout delay control. In real-time applications, like VoIP, packets are transmitted in a periodic fashion. This periodicity is lost due to the queuing time introduced in routers, which introduces delay variability, also known as jitter. Playout delay control algorithms are used on audio applications to reduce jitter, they trade loss for delay. Algorithms currently implemented on audio tools lack of a parameter allowing to control the loss percentage after playout of packets. We identify this problem and we propose a set of tunable loss rate algorithms for playout delay based on moving average estimation [81]. We use trace-based simulations to evaluate the performance of our algorithms, and we compare them with algorithms currently implemented in the NeVoT audio tool. Our algorithms overperform those implemented in NeVoT for most of the cases we studied.

Second part: A model for AIMD protocols under variable delay

In the second part of this thesis we propose a model to analyze the performance of AIMD protocols. Further, we model AIMD protocols in presence of variable delay, which is currently becoming an active area of research. Non-interactive multimedia applications often use TCP because their delay constraints are less restrictive than interactive applications. Traditional models for AIMD protocols compute the throughput by modeling the round-trip time as a constant and then replacing it by its average. However, it has been observed that delay variability impacts the throughput of AIMD-like protocols [24]. Delay variability is a common phenomenon in communication networks. Moreover, with the widespread use of wireless networks, studying the impact of delay variability on the performance of AIMD-like protocols is of great importance.

Our contribution on this part of the thesis is the proposal of the first analytical model taking into account delay variability for computing the throughput of AIMD-like

protocols[2]. The model is based on stochastic difference equations, and we solve it for the moments of the window size, as well as for the throughput. We consider two cases for round-trip times: i.i.d. and Markov correlated. We use two versions of the TCP protocol for the validation of our model: the Reno version of TCP, and SACK and Newreno.

We show by analysis and simulation with the NS simulator [67] that an increase in delay variability improves the performance of this class of protocols. Our main result is that, when the packet loss process is constant and when RTT are i.i.d., the throughput of an AIMD mechanism increases with delay variability. This means that current models for AIMD protocols, which only consider the first moment of round-trip delays, underestimate the performance when packet delay is variable [7]. We describe in detail our model in Chapter 5.

Finally, we conclude this thesis in Chapter 6 where we also discuss some perspectives of our future work.

[2]TCP is the best example of this kind of protocols.

Part I

Interactive real-time applications

Chapter 2

FEC perfomance and utility on VoIP

2.1 Introduction

Real-time audio transmission is now widely used over the Internet and has become a very important application. Audio quality is still however an open problem due to the loss of audio packets and the variation of end-to-end delay (jitter). These two factors are a natural result of the simple best effort service provided by the current Internet. Indeed, the Internet provides a simple packet delivery service without any guarantee on bandwidth, delay or drop probability. Audio quality deteriorates (noise, poor interactivity) when packets cross a loaded part of the Internet. In the wait for some QoS facilities from the network like resource reservation, call admission control, etc., the problem of audio quality must be studied and solved on an end-to-end basis. Some mechanisms must be introduced at the sender and/or at the receiver to compensate for packet losses and jitter. Jitter is often solved by some adaptive playout algorithms at the receiver. Adaptive playout mechanisms are treated in detail in [82, 83], and in the next chapter. In this chapter we focus on the problem of recovery from audio packet losses.

Mechanisms for recovering from packet losses can be classified as open loop mechanisms, or closed loop mechanisms [72]. Closed loop end-to-end mechanisms like automatic repeat request (ARQ) are not adequate for real-time interactive applications since they increase considerably the end-to-end delay due to packet retransmission[1]. Open loop mechanisms like FEC (Forward Error Correction) are better adapted to real-time appli-

[1]Note however, that ARQ could perform well as part of a link level protocol for some short links

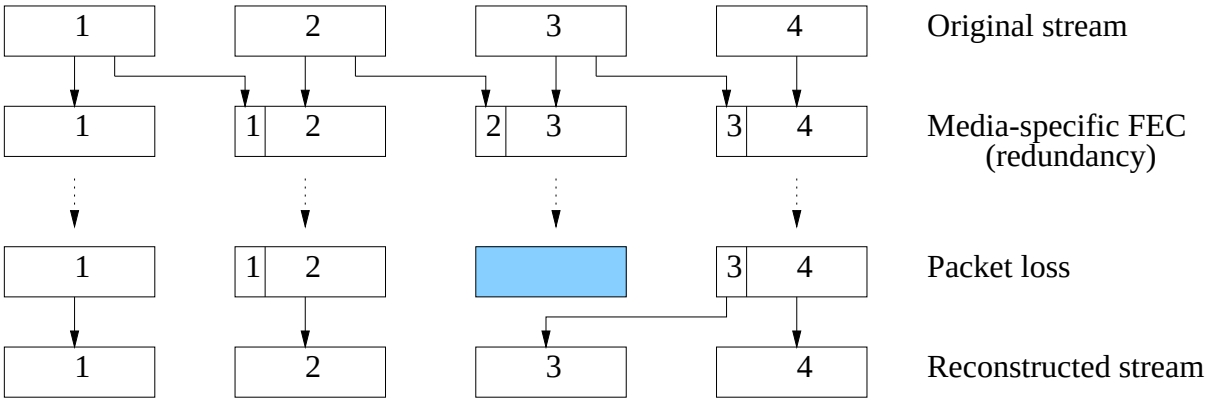


Figure 2.1: The simple FEC mechanism where packet $n+1$ carries redundant information of packet n.

cations given that packet losses are recovered without the need of a retransmission. Some redundant information is transmitted with the basic audio flow. Once a packet is lost, the receiver uses (if possible) the redundant information to reconstruct the lost information. FEC schemes are recommended whenever the end-to-end delay is large so that a retransmission deteriorates the end-to-end quality.

An audio conversation is considered to be *interactive* if the two-way end-to-end delay is less than 250 ms, including media coding and decoding, network transit and playout buffering [71].

FEC has been often used for loss recovery in audio communication tools. It is a sender-based repair mechanism. An efficient FEC scheme is one that is able to repair most of packet losses. Now, when FEC fails to recover from a loss, applications can resort to other receiver-based repair mechanisms like insertion, interpolation, or regeneration, using well known methods [72].

The FEC schemes proposed in the literature are often simple, so that the coding and the decoding of the redundancy can be quickly done without impacting the interactivity. In particular, the redundancy is computed over small blocks of audio packets. Well known audio tools as Rat [79], and Freephone [34], generally work by adding some redundant information of (i.e., a copy of) packet n to the next packet $n+1$, so that if packet n is dropped in the network, it could be recovered and played out in case packet $n+1$ is correctly received. The redundant information carried by a packet is generally obtained by coding the previous packet with a code of lower rate than that of the code used for coding the basic audio flow. For example, a basic audio packet can be coded with PCM and its copy with GSM [94] or LPC [104]. Thus, if the reconstruction succeeds, the lost packet is played out with a copy coded at a lower rate. This has shown to give better quality than playing nothing at the receiver. Figure 2.1 depicts this simple FEC scheme.

In this chapter we address the problem of audio quality under this FEC scheme. In all the chapter, when we talk about FEC in general, it is this particular scheme that we mean. We evaluate analytically the audio quality at the destination as a function of the parameters of the FEC scheme, of the basic audio flow and of the network. The performance of this FEC scheme has been evaluated via simulations [74, 75], and tools like Freephone and Rat have implemented it. In [53], the authors propose to increase the offset between the original packet and its redundancy. They claim that the loss process in the Internet is bursty and thus, increasing the offset could give better performance than having the redundancy placed in the packet following immediately the original one. However, the authors in [53] do not propose any analytical expression permitting to study the impact of this spacing on the audio quality.

In this chapter, we use probabilistic methods and a Ballot theorem [101] to find an explicit expression for the audio quality in the case of a general offset not necessarily equal to one. The audio quality is supposed to be proportional to the volume of data received. We consider a single bottleneck node for the network and we focus on the case when the buffer size in the bottleneck router is only dedicated to the audio flow (or to an aggregate of audio flows implementing the same FEC scheme and sharing the same bottleneck). These assumptions hold when all flows in the network implement FEC, or when a round-robin scheduler with per-flow queuing is used. Under these assumptions, our analysis shows that even for the infinite-offset case ($\phi \to \infty$) which forms an upper bound on the audio quality, adding FEC according to this simple scheme leads always to a deterioration of quality caused by an important increase in network load. Similar negative results have been already obtained using analytical tools for more sophisticated FEC schemes, see [8, 25, 40]. We consider in this chapter both the case in which adding redundant information does not change the amount of useful information in the packet, and thus the total packet size increases with redundancy (along with appropriate scaling of loss probabilities and buffering capabilities), as well as the case in which the total size of the packet does not change, so that adding redundancy results in decreasing the transmitted rate of useful information.

2.2 Analysis of expected goodput

This part is organized as follows. In Subsection 2.2.1, we describe our general model for applications using FEC, and we define a quality function which we will use in the rest of this part. We then present the scenario of fixed amount of useful information per packet, and fixed packet size, in Subsection 2.2.2 and 2.2.3, respectively. In Subsection 2.2.4, we study the simple case when packet n carries redundant information of packet $n-1$

assuming an $M/M/1/K$ queuing model. In Subsection 2.2.5, we solve the problem for the general case when packet n carries redundant information of packet $n-\phi$, with $\phi \geq 1$. Finally, we look in Subsection 2.2.7 at the quality in the case of infinite spacing $\phi \to \infty$. We use this result to infer about multiplexing between several flows in Subsection 2.2.8.

2.2.1 Analysis

In a large network as the Internet, a flow of packets crosses several routers before reaching the other end. Most of the losses from a flow occur in the router having the smallest available bandwidth in the chain of routers, so that one may model the whole chain by one single router called "*the bottleneck*." This assumption has both theoretical and experimental justifications [12, 20]. We shall use the simple $M/M/1/K$ queue to model the network and thus the loss process of audio packets. In other words, we assume that audio packets arrive at the bottleneck according to a Poisson process of intensity λ, and we assume that the time required to process an audio packet at the bottleneck is exponentially distributed with parameter μ. The Poisson assumption on inter-arrival times could be justified by the random delay added to packets by routers located upstream the bottleneck. The service time represents the time between the beginning of the transmission of an audio packet on the bottleneck interface leading to the destination until the beginning of the transmission of the next packet from the same audio flow. Since the two packets may be spaced apart by a random number of packets from other applications, one may use the exponential distribution as a candidate for modeling the service time of audio packets at the bottleneck. The reason for choosing this simplistic model for the network is to be able to obtain simple mathematical formulas that give us some insights on the gain from using FEC.

Let $\rho = \lambda/\mu$ be the intensity of audio traffic. Assume that audio packets share alone the buffer K. This can be the case of a bottleneck crossed only by audio packets, or the case of a bottleneck router implementing a per-flow or a per-class queuing. Thus, for $\rho < 1$, the loss probability of an audio packet in steady state is given by [51]:

$$\pi(\rho) = \frac{1-\rho}{1-\rho^{K+1}}\rho^K, \tag{2.1}$$

and for $\rho = 1$ it is equal to

$$\pi(\rho) = \frac{1}{K+1}.$$

Now, we add redundancy to each packet in a way that if a packet is lost, it can be still "partially" retrieved if the packet containing its redundancy is not lost. The redundancy is located ϕ packets apart from the original packet. It consists in a low quality copy of

the original packet. Let α be the ratio of the volume of the redundant information and the volume of the original packet. α is generally less than one. Along with the possibility to retrieve the lost information in the network, we should consider the negative impact of adding of FEC. Two scenarios should be considered.

- The first, in which the amount of useful information in a packet does not change when adding FEC. In that case, the FEC has a negative impact on the loss probability. Indeed, additional redundant information has an impact on the service times since packets require now more time to be retransmitted at the output of the bottleneck. It may also have an impact on the buffering capacity at the bottleneck since each packet now contains more bits.

- In the second scenario, the packet size does not depend on the amount of added FEC. This means that the amount of useful information in a packet reduces in order to leave space for redundant information of a previous packet.

2.2.2 Constant amount of useful information in a packet

We shall propose the following two possible negative impacts of FEC, in order to study later the tradeoff between the positive and negative impacts:

- **Impact of FEC on service time.** We assume that audio packets including redundancy require a longer service time which is exponentially distributed with parameter $\mu/(1+\alpha)$. This can be the case when our audio flow has an important share of the bottleneck bandwidth. If it is not the case, this assumption can hold when the exogenous traffic at the bottleneck (or at least an important part of it) is formed of audio flows implementing the same FEC scheme. Our assumption also holds when the bottleneck router implements a per-flow scheduling that accounts for the packet size.

- **Impact of FEC on buffering.** The buffering capacity in the bottleneck router will be affected by the addition of FEC in one of two ways: (1) Since packets are now longer by a factor $(1+\alpha)$, we can consider that the amount of buffering is diminished by this quantity, or (2) We can assume that the queue capacity is not function of packet length, but rather of the number of packets. Hence, the queue capacity is not affected by the use of FEC. Let K_α denote the buffer size after the addition of FEC in terms of packets. It is equal to $K/(1+\alpha)$ if the buffer capacity is changed, and it is equal to K otherwise. Thus, the loss probability in the presence of FEC takes the following form:

$$\pi_\rho(\alpha) = \frac{1-\rho(1+\alpha)}{1-\Big(\rho(1+\alpha)\Big)^{K_\alpha}}\Big(\rho(1+\alpha)\Big)^{K_\alpha}. \tag{2.2}$$

Before we define the quality of audio received at the destination, we introduce a random variable Y_n that indicates a successful arrival of a packet at the destination or not. Then,

$$\begin{aligned} Y_n &= 0, \quad \text{if packet } n \text{ is lost, and} \\ Y_n &= 1, \quad \text{if packet } n \text{ is correctly received.} \end{aligned}$$

Let $\phi \geq 1$ be the variable indicating the distance, or the offset, between the original packet and its redundancy. We make the simple assumption that the audio quality is proportional to the amount of information we receive. A quality equal to 1 indicates that we are receiving all the information (the basic audio flow). The quality we get after the reconstruction of an original packet from the redundancy is taken equal to α, where α is the ratio of redundancy volume and original packet volume. We thus define the quality function as the probability that the packet is received correctly,

$$\begin{aligned} Q(\alpha) &= P(Y_n = 1) + \alpha P(Y_n = 0)P(Y_{n+\phi} = 1 | Y_n = 0) \\ &= 1 - \pi_\rho(\alpha)(1 - \alpha P(Y_{n+\phi} = 1 | Y_n = 0)) \,. \end{aligned} \tag{2.3}$$

This equation gives us the audio quality at the destination under an FEC scheme of rate $(1+\alpha)^{-1}$, and of distance ϕ between an original packet and its redundancy. For the case $\alpha = 0$, our definition for the quality coincides with the probability that a packet is correctly received. For the case $\alpha = 1$, it coincides with the probability that the information in an original packet is correctly received, either because it was not lost, or because it was fully retrieved from the redundancy. One may imagine to use another quality function that the one we chose. In particular, one can use a quality function that is not only a function of the amount of data correctly received but also of the coding algorithm used. Different algorithms have been used in [34, 79] for coding the original data and the redundancy. Table 2.1 resumes the notation we will use in the rest of this chapter.

2.2.3 The case of constant packet size

We assume that the total packet size does not depend on the amount of FEC. A packet is constituted of a fraction α of redundant information, and a fraction of $1-\alpha$ of useful information.

[2] As is frequently done, we include in Z_j not only real services but also "potential services": these are services that occur while the system is empty; thus at the end of such a service no packet leaves.

Expression	Definition
$Q(\alpha)$	The audio quality.
ϕ	The offset between the original packet and the packet including its redundancy.
K_α	The size of the queue.
X_j	The random variable which represents the number of packets in the queue just before the arrival of the j–th audio packet.
Z_j	The random variable which represents the number of services between the arrivals of the $j-1$–th and the j–th audio packets.[2]

Table 2.1: Notation used in this chapter.

Since the packet size is constant here, FEC has no impact on the loss probabilities nor on the buffer size (in units of packets). In particular, the probability $\pi_\rho(\alpha)$ does not depend on α, and ρ neither does not change with α. So we do not need to include α and ρ in the notation.

The quality we get after reconstruction of an original packet from the redundancy is taken equal to α. But if we do not lose the original packet, its quality is $1-\alpha$, instead of 1 unit, as before. We thus define the quality function as,

$$\begin{aligned} Q(\alpha) &= (1-\alpha)P(Y_n=1) + \alpha P(Y_n=0)P(Y_{n+\phi}=1|Y_n=0) \\ &= (1-\pi)(1-\alpha) + \alpha\pi P(Y_{n+\phi}=1|Y_n=0))\,. \end{aligned} \tag{2.4}$$

For both scenarios (where useful information or where total information in a packet are constant), we ask the following question: "*How does the audio quality vary as a function of* α*?*" That would permit us to evaluate the benefits from such a recovery mechanism and to find the appropriate amount of redundancy α that must be added to each packet. In the next sections we find the audio quality for different values of ϕ. The only missing parameter is the probability that the redundant information on a packet is correctly received given that the packet itself is lost. This is the function $P(Y_{n+\phi}=1|Y_n=0)$ in (2.3). In the following sections we put ourselves in the stationary regime and we compute this probability.

2.2.4 Spacing by $\phi = 1$

In this section we analyze the case when the redundant information of packet n is carried by packet $n+1$, i.e., $\phi = 1$. This mechanism is implemented in well known audio tools as Freephone [34] and Rat [79]. Let R be the event that the redundancy is correctly received given that the original packet is lost. This represents the case where the next event after the loss of the original packet is a departure and not an arrival.

Fixed amount of useful information per packet. Consider first the scenario in which the useful information in a packet is fixed. Then the probability of event R is given by

$$P(Y_{n+1} = 1 | Y_n = 0) = \frac{1}{\rho(1+\alpha)+1}. \tag{2.5}$$

Substituting (2.5) in (2.3), we obtain

$$Q_{\phi=1}(\alpha) = 1 - \pi_\rho(\alpha)\Big(1 - \frac{\alpha}{\rho(1+\alpha)+1}\Big).$$

To study the impact of FEC on audio quality, we plot $Q_{\phi=1}(\alpha)$ as a function of α for different values of K_α and ρ. In Figure 2.2 we show the results when the buffering capacity at the bottleneck is assumed to change with the amount of FEC ($K_\alpha = K/(1+\alpha)$), and in Figure 2.3 we show the results for the case where the buffering capacity is not changed ($K_\alpha = K$). We see that, for both cases, audio quality deteriorates when α increases (when we add more redundancy), and this deterioration becomes more important when the traffic intensity increases and when the buffer size decreases. The main interpretation of such behavior is that the loss probability of an original packet increases with α faster than the gain in quality we got from retrieving the redundant information. This should not be surprising. Indeed, even in more sophisticated schemes in which a single redundant packet is added to protect a whole block of M packets, it is known that FEC often has an overall negative effect, see [8,25,40]. Yet in such schemes the negative effect of adding redundancy is smaller than in our scheme, since the amount of added information per packet is smaller (i.e., a single packet protects a whole group of M packets). But, we know that for such schemes and in case of light traffic, the overall contribution of FEC is positive [8,40]. This motivates us to analyze more precisely the impact of FEC in our simplistic scheme in case of light traffic.

Define the function $\Delta(\rho) = Q(1) - Q(0)$ and consider the case when the buffering capacity at the bottleneck is not affected by the amount of FEC. This is an optimistic

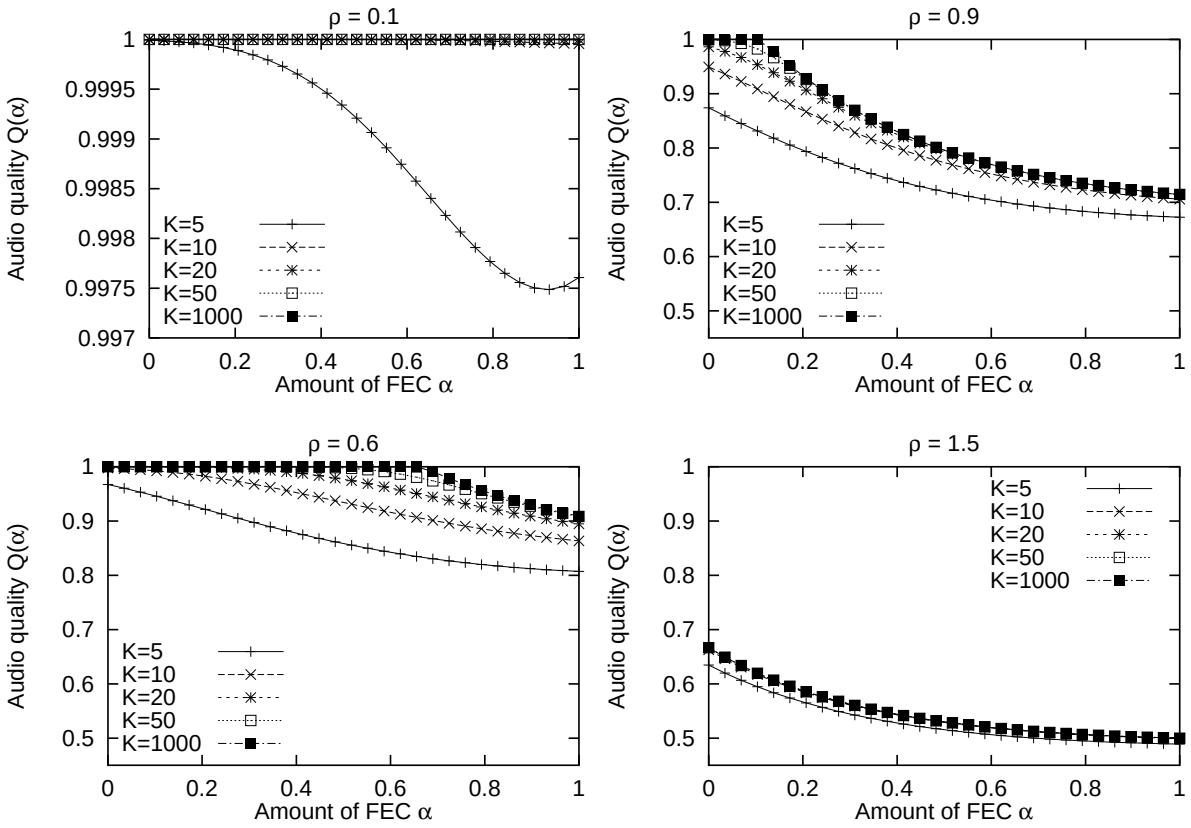


Figure 2.2: $\phi = 1$ and the queue capacity is changed.

scenario where it is very probable to see the gain brought by FEC, of course if this gain exists. We have,

$$\Delta(\rho) = -2(2\rho)^{\frac{K_\alpha}{2}} \Big(\frac{1+\rho}{2\rho+1}\Big)\Big(\frac{1-2\rho}{1-(2\rho)^{\frac{K_\alpha}{2}+1}}\Big) + \frac{1-\rho}{1-\rho^{K_\alpha+1}}\rho^{K_\alpha} \tag{2.6}$$

Finding $\lim_{\rho\to 0}\Delta(\rho)$ would permit us to evaluate the audio quality for a very low traffic intensity. We took $K_\alpha = 2M$ in (2.6) and we expanded $\Delta(\rho)$ in a Taylor series. We found that all the first coefficients of the series $c_0, c_1, ..., c_{M-1}$ are equal to zero, and that the coefficient c_M is negative and equal to $-2(2\rho)^M$. c_i is the coefficient of ρ^i in the Taylor series of $\Delta(\rho)$ and can be computed by

$$c_j = \frac{d}{d\rho^j}\Delta^j(\rho)_{|\rho=0}.$$

Thus, for small ρ, $\Delta(\rho)$ can be written as $-2(2\rho)^M + o(\rho^M)$ and the gain from adding FEC can be seen to be negative. With this simple FEC scheme, we lose in audio quality when adding FEC even for a very low traffic intensity. This loss in quality decreases with

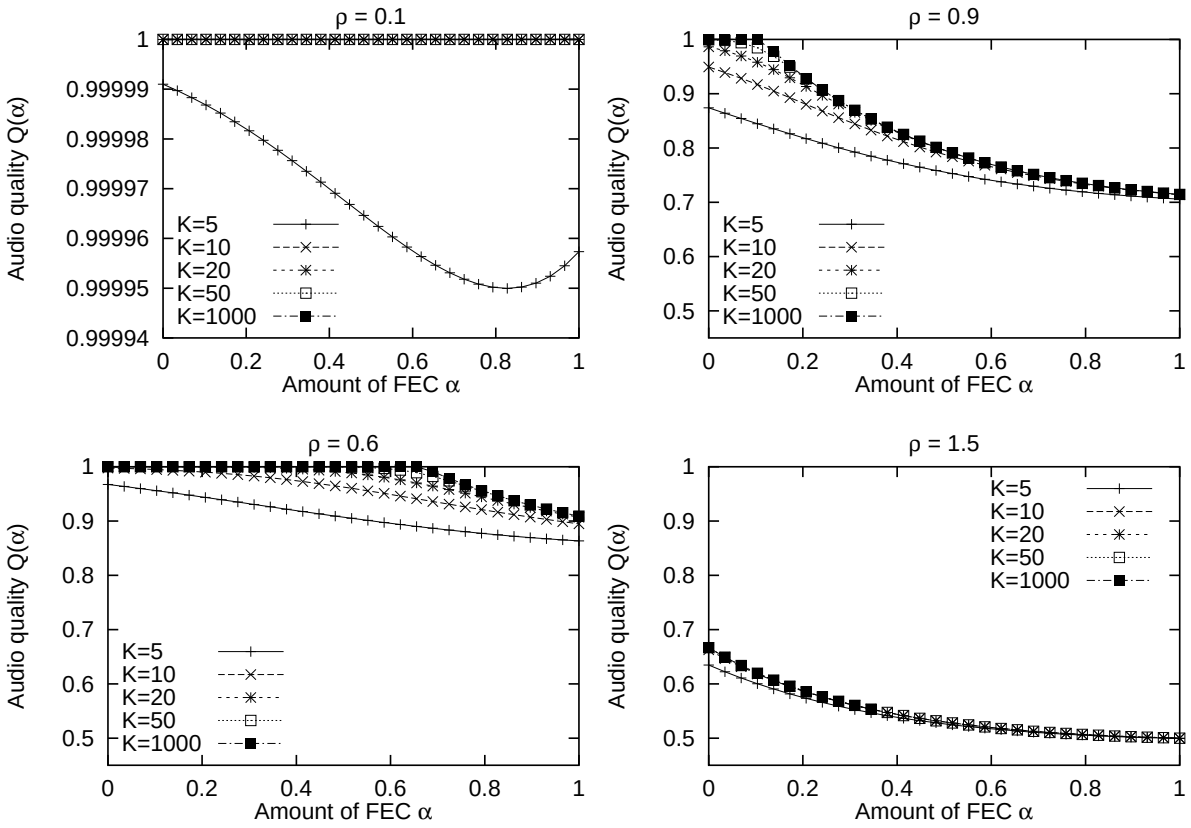


Figure 2.3: $\phi = 1$ and the queue capacity is not changed.

the increase in buffer size.

Fixed packet size. Substituting (2.5) in (2.4), we obtain

$$Q_{\phi=1}(\alpha) = (1-\alpha)(1-\pi) + \frac{\alpha\pi}{\rho(1+\alpha)+1}.$$

In Section 2.2.7 we shall show that we loose in that case, as well as for any general spacing $\phi > 1$. This will be done by considering an optimistic bound obtained by an infinite spacing.

2.2.5 General case: Spacing by $\phi \geq 1$

Now, we consider the more general case when the spacing between the original packet and its redundancy is greater than 1. The idea behind this type of spacing is that losses in real networks tend to appear in bursts, and thus spacing the redundancy from the original packet by more than one improves the probability to retrieve the redundancy in case the original packet is lost. Indeed, a packet loss means that the queue is full and

thus the probability of losing the next packet is higher than the steady state probability of losing a packet. The spacing gives the redundancy of a packet more chance to find a non full buffer at the bottleneck, and thus to be correctly received. We note that the phenomenon of the correlation between losses of packets was already modeled and studied in other papers: [8, 25, 40]. Measurements have also shown that most of the losses are correlated [13, 15, 87].

Here, we are interested in finding the probability that packet $n+\phi$ is lost given that packet n is also lost. This will give us $P(Y_{n+\phi} = 1|Y_n = 0)$ which in turn gives us the expression for the audio quality (as expressed in (2.3)). Since we assume that the system is in its steady state, we can omit the index n and substitute it by zero. We have $Y_0 = 0$ which means $X_0 = K_\alpha$. We are interested in the probability that $X_\phi = K_\alpha$. For the ease of computation, we consider the case $\phi \leq K_\alpha$. We believe that this is quite enough given that a large spacing between the original packet and the redundancy leads to an important jitter and a poor interactivity.

In order to obtain an explicit expression for the probability $P(X_\phi = K_\alpha|X_0 = K_\alpha)$, we first provide an explicit sample-path expression for the event of loss of the packet carrying the redundancy, given that the original packet itself was lost.

Theorem 1. *Let $X_0 = K_\alpha$ and $1 \leq \phi \leq K_\alpha$. then:*

Packet ϕ is not lost if and only if

$$X_\phi < K_\alpha \Leftrightarrow \begin{cases} Z_\phi - 1 \geq 0 \\ \quad or \\ Z_\phi + Z_{\phi-1} - 2 \geq 0 \\ \quad or \\ \quad \vdots \\ \quad or \\ Z_\phi + Z_{\phi-1} + \cdots + Z_1 - \phi \geq 0 \end{cases} \tag{2.7}$$

or equivalently, packet ϕ is lost if and only if

$$X_\phi = K_\alpha \Leftrightarrow \begin{cases} Z_\phi - 1 < 0 \\ \quad and \\ Z_\phi + Z_{\phi-1} - 2 < 0 \\ \quad and \\ \quad \vdots \\ \quad and \\ Z_\phi + Z_{\phi-1} + \cdots + Z_1 - \phi < 0 \end{cases} \tag{2.8}$$

Proof: We can express the number of packets that the $i+1$–th audio packet will find in the queue upon arrival as follows:

$$X_{i+1} = \Big((X_i + 1) \wedge K_\alpha - Z_{i+1}\Big) \vee 0 \quad \forall\, i \geq 0, \tag{2.9}$$

where $\wedge$ and $\vee$ are respectively the minimum and maximum operators. The rest of the proof goes in three steps that are summarized in Lemma 1, Lemma 2 and Corollary 1 below. □

Now, we define

$$\tilde{X}_{i+1} \triangleq (\tilde{X}_i + 1) \wedge K_\alpha - Z_{i+1}. \tag{2.10}$$

This new variable corresponds to the number of packets that would be found in the queue upon the arrival of packet $i+1$ if the queue size could become negative. We next show that it can be used as a lower bound for X_{i+1}.

Lemma 1. *If $\tilde{X}_0 \leq X_0$ then $\tilde{X}_i \leq X_i \quad \forall i \geq 0$.*

Proof: We proceed for the proof by induction. This relation is valid for $i = 0$. Suppose that it is valid for $i \geq 0$. We show that it is valid for $i+1$,

$$\begin{aligned} \tilde{X}_{i+1} &\leq \Big((\tilde{X}_i + 1) \wedge K_\alpha - Z_{i+1}\Big) \vee 0 \\ &\leq \Big((X_i + 1) \wedge K_\alpha - Z_{i+1}\Big) \vee 0 \\ &= X_{i+1}. \end{aligned}$$

□

Lemma 2. *Let $\tilde{X}_0 = K_\alpha$, then*

$$\tilde{X}_i = K_\alpha - \max_{1 \leq l \leq i} \textstyle\sum_{j=l}^{i} (Z_j - 1) - 1 \quad \forall i \geq 0.$$

Proof:

$$
\begin{aligned}
\tilde{X}_0 &= K_\alpha \\
\tilde{X}_1 &= K_\alpha - Z_1 \\
\tilde{X}_2 &= (K_\alpha - Z_1 + 1) \wedge K_\alpha - Z_2 \\
\tilde{X}_3 &= \Big((K_\alpha - Z_1 + 1) \wedge K_\alpha - Z_2 + 1\Big) \wedge K_\alpha - Z_3 \\
&= (K_\alpha - Z_1 - Z_2 + 2) \wedge (K_\alpha - Z_2 + 1) \wedge K_\alpha - Z_3 \\
&= K_\alpha - (Z_1 + Z_2 - 2) \vee (Z_2 - 1) \vee 0 - Z_3 \\
&\vdots \\
\tilde{X}_i &= K_\alpha - \max_{1 \le l < i} \Big\{0, \sum_{j=l}^{i-1} (Z_j - 1)\Big\} - Z_i \\
&= K_\alpha - \max_{1 \le l < i} \Big\{0, \sum_{j=l}^{i-1} (Z_j - 1)\Big\} - Z_i \\
\Rightarrow \tilde{X}_i &= K_\alpha - \max_{1 \le l \le i} \Big\{ \sum_{j=l}^{i} (Z_j - 1)\Big\} - 1.
\end{aligned}
$$

□

Corollary 1. *Expression (2.8) holds if* $X_0 = K_\alpha$ *and* $\phi \le K_\alpha$.

Proof: The right hand side in (2.8) is no other than $\max_{1 \le l \le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1)\Big\}$. Suppose first that $\max_{1 \le l \le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1)\Big\} < 0$. Using Lemma 2 then Lemma 1, we have $\tilde{X}_\phi \ge K_\alpha$ which gives $X_\phi \ge K_\alpha$. Thus, $X_\phi = K_\alpha$.

Now, we need to show that if $X_\phi = K_\alpha$ we get $\max_{1 \le l \le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1)\Big\} < 0$. We define:

$$\phi^* = \min\Big\{ i \quad | \quad \tilde{X}_i < X_i \Big\} \tag{2.11}$$

According to (2.11), we distinguish between the two following cases:

- $\phi^* > \phi$, and
- $\phi^* \le \phi$.

Consider the first case. Using the definition of ϕ^* and Lemma 2, we write:

$$\phi^* > \phi \quad \Rightarrow \quad \tilde{X}_\phi = X_\phi \Rightarrow \tilde{X}_\phi = K_\alpha \,,$$

then

$$\max_{1\le l\le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1) \Big\} = -1 < 0.$$

Now, suppose that $\phi^* \le \phi$, thus $\tilde{X}_{\phi^*} < 0$ and $X_{\phi^*} = 0$. We write,

$$X_\phi \le X_{\phi^*} + (\phi - \phi^*) = (\phi - \phi^*) < \phi \le K_\alpha,$$

if there were no service. Thus, we get in this case $X_\phi < K_\alpha$ which is in contradiction with our assumption that $X_\phi = K_\alpha$. The case $\phi^* \le \phi$ does not appear if ϕ is chosen less or equal to the buffering capacity. Thus, for $X_\phi = K_\alpha$ we have $\max_{1\le l\le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1) \Big\} < 0$. This concludes the proof of Theorem 1. □

According to Ballot's Theorem [101] (see the Appendix in Section 6.2 for details), we have for $k < \phi$:

Lemma 3.

$$P\Big\{ \max_{1\le l\le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1) \Big\} < 0 \,|\, \sum_{l=1}^{\phi} Z_l = k \Big\} = 1 - \frac{k}{\phi} \tag{2.12}$$

Let A be the event that $X_\phi = K_\alpha$ given that $X_0 = K_\alpha$. We sometimes write A^ϕ to stress the dependence on ϕ. We conclude from Theorem 1 that if packet 0 is lost, i.e. if packet 0 finds K_α packets in the system, then

$$A = \left\{ \max_{1\le l\le \phi} \Big\{ \sum_{j=l}^{\phi} (Z_j - 1) \Big\} < 0 \right\}$$

Then, we can represent the probability that packet $n + \phi$ is lost given that packet n is lost as

$$P(Y_{n+\phi} = 0 | Y_n = 0) = P(A)$$

$$= \sum_{k=0}^{\phi-1} P(A | Z_1 + \cdots + Z_\phi = k) P(Z_1 + \cdots + Z_\phi = k) \tag{2.13}$$

Once this probability is computed, the audio quality can be directly derived using (2.3).

Theorem 2. *Define $\rho_\alpha = \rho(1+\alpha)$ and for the scenario in which the useful amount of information in a packet is constant in α; define $\rho_\alpha = \rho$ and $K_\alpha = K$ in the scenario in which the packet size does not depend of α. Consider $1 \leq \phi \leq K_\alpha$. Given that packet n is lost, the probability that packet $n+\phi$ is also lost is given by*

$$P(A) = \sum_{k=0}^{\phi-1} \Big(1 - \frac{k}{\phi}\Big) \Big(\frac{\rho_\alpha}{\rho_\alpha + 1}\Big)^{\phi} \Big(\frac{1}{\rho_\alpha + 1}\Big)^{k} \binom{\phi + k - 1}{\phi - 1} \tag{2.14}$$

where $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ denotes the binomial coefficient. The quality function can be computed by substituting $P(A)$ in (2.3). Note that $P(Y_{n+\phi} = 1 | Y_n = 0) = 1 - P(A)$.

Proof: The second right hand term of (2.13) must be solved by combinatorial reasoning. For that purpose, we define the vector $\vec{Z}$ to be:

$$\vec{Z} = \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_\phi \end{pmatrix}, \tag{2.15}$$

where $\sum_{l=1}^{\phi} Z_l = k$, and we define S be the set of the different sets that $\vec{Z}$ may acquire: $S = \{\vec{Z}\}$. We must sum over all the possible trajectories:

$$\begin{aligned} P(\sum_{l=1}^{\phi} Z_l = k) &= \sum_{S} P(Z_1 = z_1) P(Z_2 = z_2) \cdots P(Z_\phi = z_\phi) \\ &= \sum_{S} \Big(\frac{\lambda}{\lambda + \mu_\alpha}\Big)^{\phi} \Big(\frac{\mu_\alpha}{\lambda + \mu_\alpha}\Big)^{k} \\ &= \Big(\frac{\lambda}{\lambda + \mu_\alpha}\Big)^{\phi} \Big(\frac{\mu_\alpha}{\lambda + \mu_\alpha}\Big)^{k} \binom{\phi + k - 1}{\phi - 1} \end{aligned} \tag{2.16}$$

We define here μ_α as being equal to $\mu/(1+\alpha)$ for the scenario in which the amount of useful information per packet does not depend on α, and $\mu_\alpha = \mu$ in the case of fixed packet size. It's easy to see that the combinatorial part of (2.16) holds. To do that, we can see the problem to be the number of distinguishable arrangements of k indistinguishable objects (the packet audio departures from the bottleneck) in ϕ inter-arrival intervals, just as it's depicted in Figure 2.4.

Using (2.16) we get finally,

$$P(A) = \sum_{k=0}^{\phi-1} \Big(1 - \frac{k}{\phi}\Big) \Big(\frac{\lambda}{\lambda + \mu_\alpha}\Big)^{\phi} \Big(\frac{\mu_\alpha}{\lambda + \mu_\alpha}\Big)^{k} \binom{\phi + k - 1}{\phi - 1}, \tag{2.17}$$

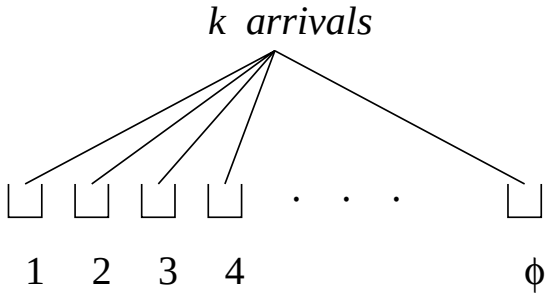


Figure 2.4: Model to solve the combinatorial part.

which yields (2.14) in terms of $\rho_\alpha = \rho(1+\alpha) = \lambda/\mu_\alpha$. The quality function can be obtained by substituting (2.14) in (2.3). The value of $\pi_\alpha(\rho)$ is given in (2.2). □

We trace now plots of the audio quality as given by (2.3) and (2.14) for different values of K_α, ϕ and ρ. Figure 2.5 depicts the behavior of $Q(\alpha)$ when the buffering capacity at the bottleneck is assumed to be divided by a factor $(1+\alpha)$, and Figure 2.6 depicts this behavior when the buffering capacity is not changed.

We notice that, just as in the case of $\phi = 1$, we always lose in quality when we increase the amount of FEC even if we consider a large spacing. But, we also notice that for a given amount of FEC, the quality improves when spacing the redundancy from the original packet. This is the result of an improvement in the probability to retrieve the redundancy given that the original packet is lost. This monotonicity property holds, in fact, for any value of ϕ (not just for $\phi \leq K_\alpha$). We show this theoretically in the next section.

2.2.6 Monotone increase of the quality with the spacing

The steady state probability of loss of a packet n does not depend on ϕ. It thus remains to check the behavior of $P(X_{n+\phi} = K_\alpha | X_n = K_\alpha)$ as a function of ϕ in order to decide on the quality variation (Eq. 2.3). The quality is a decreasing function of this probability. For $\phi \leq K_\alpha$, the latter probability is equal to $P(A^\phi)$, and the monotonicity property can be seen directly from the fact that A^ϕ is a monotone decreasing set (since it requires for more summands to be smaller than zero, as ϕ increases, see Eq. 2.8).

Now, to see that $P(X_{n+\phi} = K_\alpha | X_n = K_\alpha)$ is monotone decreasing for any ϕ, we observe (2.9), which holds for any $i > 0$, and note that X_{i+1} is monotone increasing in X_i. Thus by iteration, we get that X_ϕ is monotone increasing in X_0. Now using this

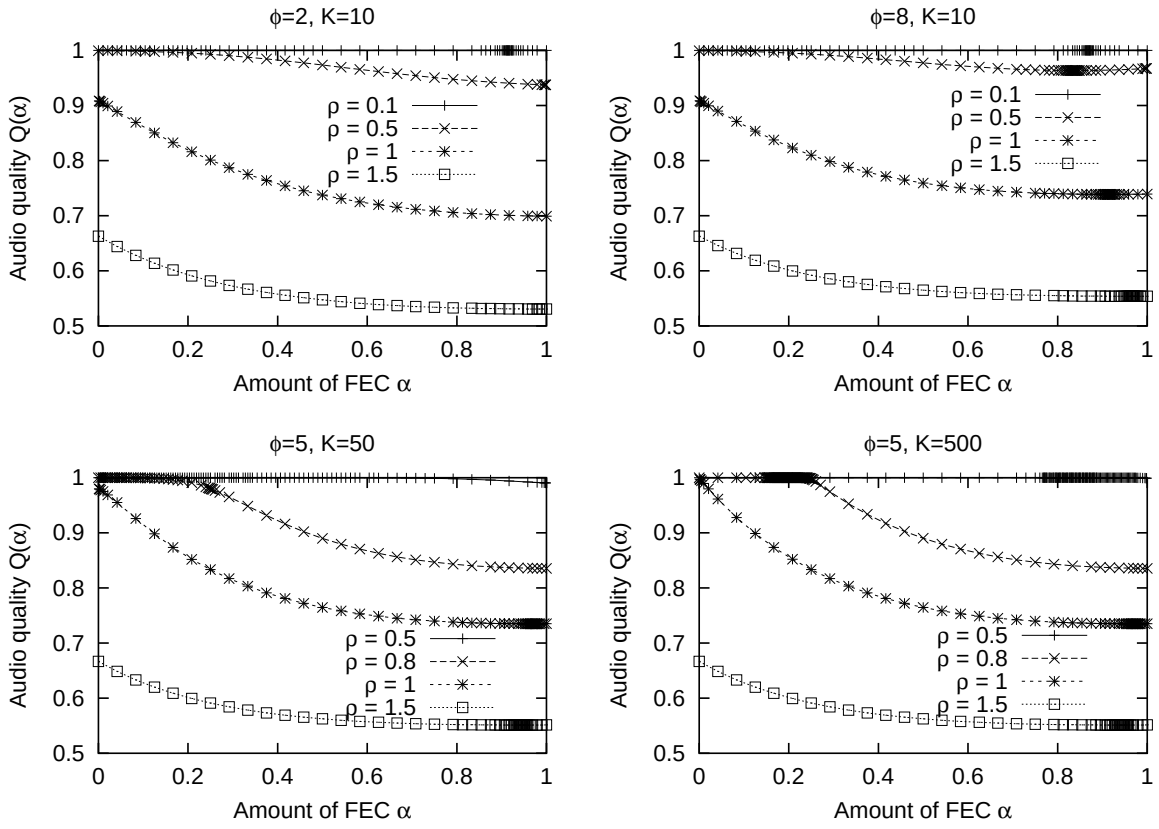


Figure 2.5: Quality behavior in the presence of FEC and spacing $1 < \phi < K_\alpha$ assuming that queue size is changed.

monotonicity, we have

$$\begin{aligned} P(X_{\phi+1} &= K_\alpha | X_0 = K_\alpha) = P(X_\phi = K_\alpha | X_{-1} = K_\alpha) \\ &= \sum_{i=0}^{K_\alpha} P(X_\phi = K_\alpha | X_0 = i, X_{-1} = K_\alpha) \times P(X_0 = i | X_{-1} = K_\alpha) \end{aligned}$$

$$\begin{aligned} &= \sum_{i=0}^{K_\alpha} P(X_\phi = K_\alpha | X_0 = i) P(X_0 = i | X_{-1} = K_\alpha) \\ &\leq \sum_{i=0}^{K_\alpha} P(X_\phi = K_\alpha | X_0 = K_\alpha) P(X_0 = i | X_{-1} = K_\alpha) \\ &= P(X_\phi = K_\alpha | X_0 = K_\alpha). \end{aligned}$$

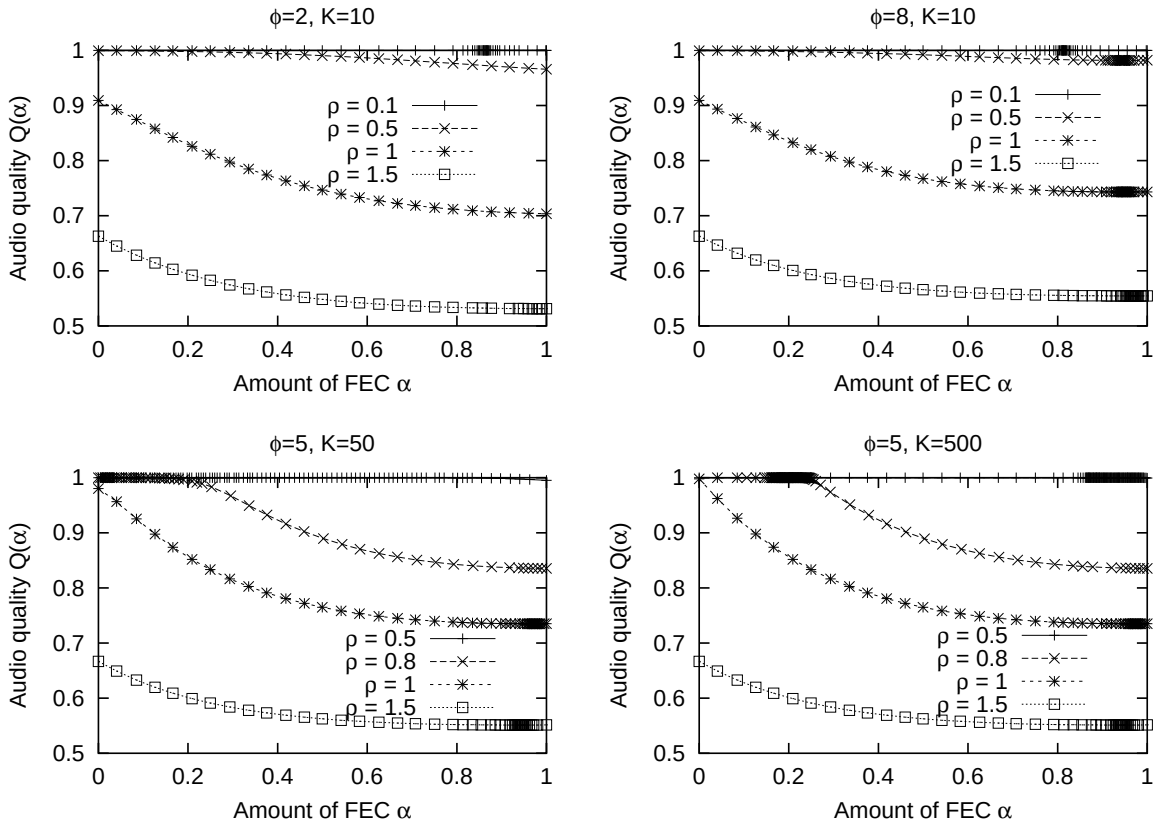


Figure 2.6: Quality behavior in the presence of FEC and spacing $1 < \phi < K_\alpha$ assuming that queue size is not changed.

2.2.7 Limiting case: Spacing $\phi \to \infty$

The case of large ϕ is not of interest in interactive applications, since it means unacceptable delay. However, since we have found that the quality of the audio with FEC improves as the spacing grows, it is natural to study the limit ($\phi \to \infty$) in order to get an upper bound. Indeed, if we see that in this limiting case we do not improve the quality, it means that we lose by adding FEC according to our simple scheme for any finite offset ϕ.

When $\phi \to \infty$, the probability that the redundancy is dropped becomes equal to the steady state drop probability of a packet.

Consider the scenario of fixed amount of useful information per packet. Then, (2.3) can be written as,

$$Q_{\phi\to\infty}(\alpha) = 1 - \pi_\rho(\alpha) + \alpha\pi_\rho(\alpha)(1 - \pi_\rho(\alpha))\,. \tag{2.18}$$

We plot (2.18) in Figure 2.7 as a function of the amount of FEC for different values of K_α and ρ. We see well how, although we are in the most optimistic case, we lose in quality

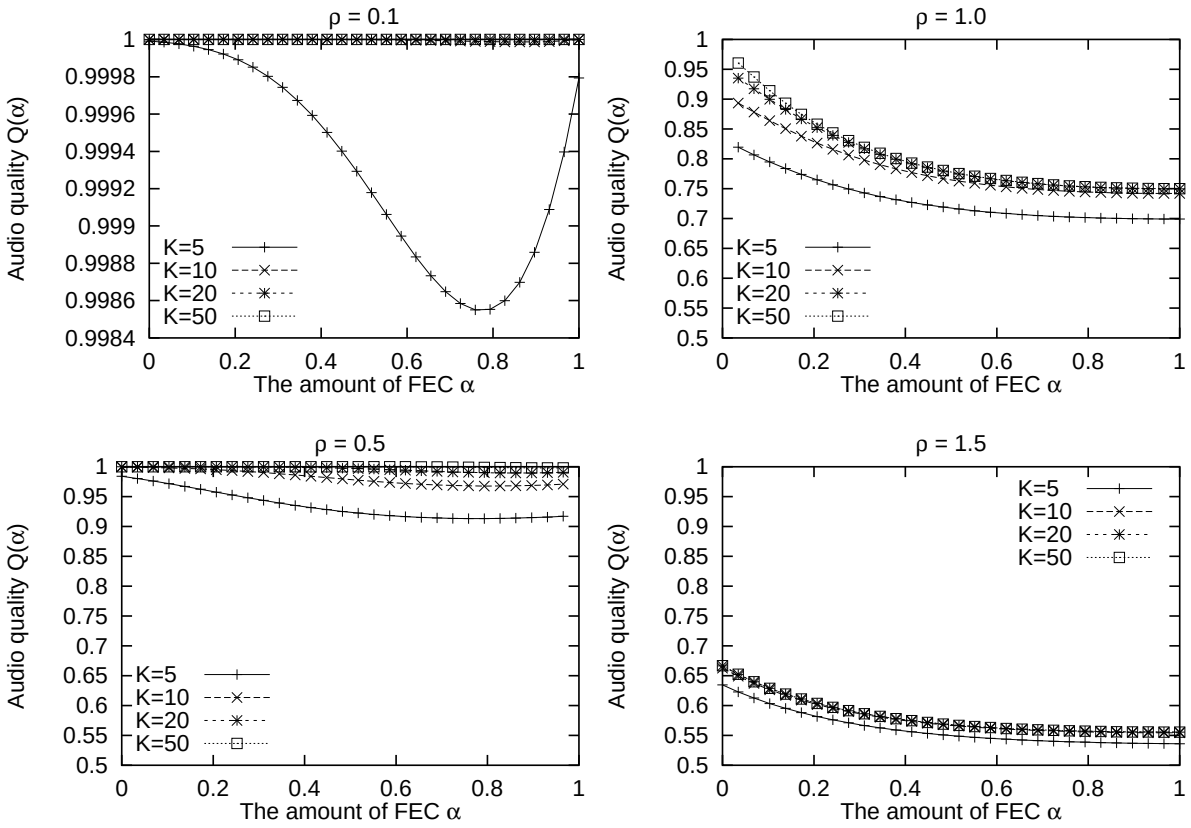


Figure 2.7: Quality behavior in the presence of FEC and spacing $\phi \to \infty$.

when adding FEC. That suggests that this class of FEC mechanisms are not efficient for real time transmission because it never improves the goodput of the connection.

Next we show that this conclusion also holds for the scenario of fixed packet size. In the case of very large spacing, (2.4) can be written as

$$\begin{aligned} Q_{\phi\to\infty}(\alpha) &= (1-\pi)(1-\alpha) + \alpha\pi(1-\pi) \\ &= (1-\pi)(1-\alpha(1-\pi))\,. \end{aligned} \tag{2.19}$$

We see that this is strictly smaller than $1-\alpha$ which is the quality with zero redundancy!

2.2.8 Multiplexing between traffic

We analyze the case when several input flows arrive to the bottleneck, an audio flow and an exogenous flow which represent the superposition of all other flows. We further consider here the case of general independent service time distribution. We use the results of the previous subsection to show that we do not gain in goodput by multiplexing (under both

FEC scenarios) provided that the packet size of all multiplexed streams has the same distribution. This covers both the first scenario (in which packet size increases with FEC) where all streams add FEC, as well as the second scenario in which it does not matter any more how many streams add FEC (since the packet size is not affected by FEC).

We model the multiplexing by the arrival of two independent Poisson input flows, λ_1 and λ_2, representing the audio flow and the exogenous traffic (which represents all other flows) respectively.

In order to compute the goodput of a flow adding FEC, we can use the fact that the spacing between a packet and its redundancy is now random, with a geometric distribution with parameter λ_1/λ_2. We note that we cannot use anymore the exact expressions for loss probabilities derived in Subsection 2.2.5, since the spacing can now be larger than the buffer size. However, we can still use the fact that the quality is increasing with the spacing. Thus, since we saw that we lose by adding FEC for $\phi \to \infty$, we conclude that we lose in goodput by adding FEC when multiplexing flows under the above assumptions. Nevertheless, we shall show in the next section that we may gain in goodput under different assumptions.

2.3 Conclusions

We studied in this chapter the effect on audio quality of a FEC scheme similar to the one used in [34, 79]. This FEC scheme consists in adding a copy of an audio packet to a subsequent one so that the copy can be used when the original packet is lost. First, we considered the case when all flows in the network implement such a FEC scheme. With a simple queuing analysis and assuming linear utility functions, we found an explicit expression for the audio quality (corresponding to the goodput) for a general offset between an audio packet and its copy. This expression showed that adding FEC deteriorates the goodput instead of improving it.

One might wonder whether the conclusions depend on the probabilistic assumptions. However, one can note that the form of the optimistic bound (2.19) does not depend on the exponential assumptions: it would be the same for any service time and interarrival distributions, and even for topologies much more than a single queue. The precise values of π would of course depend on the distribution, and on the form of the network, but the conclusion we draw from the general form of (2.19) will remain the same.

We believe that the main problem with this kind of mechanisms is that the redundant information is constructed at the source using one packet and so the destination has only two choices: either receive the original packet or receive its copy. Better performance could be obtained if we gave the receiver more choices by constructing at the source the

redundancy carried by a packet from a block of audio packets. This will the topic of our future research in this direction.

Chapter 3

Cases where FEC improves audio quality

3.1 Introduction

As we explained in the previous chapter, the negative result we obtained holds in the case when all the flows in the network add FEC, or when the audio flow has its own buffer in network routers. It also holds with the particular linear utility function we considered. In this chapter, we consider other cases where we prove that FEC may improve audio quality.

We address here the questions of how and where this simple FEC scheme, which we recall is implemented in many audio tools as Freephone and Rat, leads to a quality improvement. We consider two aspects that may contribute to this end: multiplexing with other flows and using quality functions which are not proportional to the volume of well received data (goodput). The expected quality is computed by using a *utility function* that indicates the audio quality at the receiver as a function of the transmission rate. In linear utility function we previously used, we supposed that the more the user receives data, the better is the quality and that the increase in quality for a certain amount of redundancy is the same for any value of the transmission rate. In fact, the quality of an audio transmission is quite a subjective measure and depends on a large number of parameters. Yet some simplified non-linear utility functions have been proposed [96] to allow to asses voice quality as a function of the transmission rate (see also [18]).

Our findings in this chapter can be summarized as follows:

– With a linear utility function, adding FEC leads to quality improvement if the (total) rate of the flow(s) adding FEC is small compared to the total rate of the other flows

sharing the same bottleneck and not adding FEC. The addition of FEC in this case does not lead to an important increase in the loss rate which explains this improvement. We start to lose in quality when the (total) rate of flow(s) using FEC increases.

– In the case when all flows add FEC, which is the worst case where adding of FEC has the biggest impact on the load of the network, it is possible to obtain a gain in quality for some particular utility functions. The utility function must increase with the amount of FEC faster than the linear one, and higher increase rates are required for small amounts of FEC.

The results of this chapter serve as guidelines for an efficient use of FEC in audio applications. They will give us an explanation of the gain in audio quality we may perceive in some real scenarios. Queuing models similar to those used in Chapter 2 will be used through this chapter.

The rest of this chapter is organized as follows. In Section 3.2, we investigate the case of a single audio flow sharing the bottleneck with an exogenous traffic not implementing FEC. In Section 3.3, we study the performance of FEC for other non-linear utility functions. We conclude this chapter with some final remarks in Section 3.4.

3.2 Multiplexing and FEC performance

3.2.1 The model

Consider the case of an audio flow implementing FEC and sharing a bottleneck router with some other flows not implementing FEC. We look at the other flows as a single exogenous flow of constant rate and of packet size exponentially distributed. The latter choice can be justified by the mixture of a large number of flows from different sources and of different packet sizes. Let $1/\mu$ denote the average transmission time at the bottleneck of a packet from the exogenous flow. This time is independent of the amount of FEC added to the audio flow. We consider that the original audio packets have a fixed length and we denote by $1/\mu_0$ their average transmission time at the output interface of the bottleneck router.

Let us suppose that packets (audio + exogenous) arrive at the bottleneck router according to a Poisson process of constant rate λ. Suppose also that audio packets arrive at the bottleneck according to a Poisson process. This latter assumption can be justified by the fact that audio packets cross multiple routers before arriving at the bottleneck, so that their inter-arrival times can be approximated by an exponential distribution. Let $\beta \in [0, 1]$ denote the fraction of arriving packets belonging to the audio flow; this quantity represents the probability that a packet arriving at the bottleneck is of audio type. Suppose finally

that the bottleneck router implements the classical Drop Tail policy and has a buffer of size K packets (packet in service included). Packets from different flows share the K places of the buffer and are served in a FIFO (First-In First-Out) fashion. The system can be then considered as an $M/G/1/K$ queuing system where packets arrive according to a Poisson process and where service times (or transmission times in our settings) are independent and identically distributed. This system can be then solved using some known results from queuing theory [26, 51]. Our main objective is to find an expression for the audio quality at the destination as a function of the different system parameters as well as the amount of FEC added to the original packets by the audio source.

3.2.2 The analysis

Suppose first that the audio flow does not implement FEC. We look at the audio quality at the moments at which packets would arrive at the destination. We take a value equal to 1 as the quality obtained when the audio packet is correctly received, and 0 as the quality when the packet is lost in the network. The *average audio quality* during the conversation is equal to $Q = 1 - \pi$, where π denotes the stationary probability that a packet is dropped in an $M/G/1/K$ system. This probability is equal to:

$$\pi = \frac{1 + (\rho - 1)f}{1 + \rho f},$$

where ρ is the total system load (or the total traffic intensity) given by:

$$\rho = \lambda(\frac{\beta}{\mu_0} + \frac{1 - \beta}{\mu}),$$

and f is the $K-2$ th coefficient of the Taylor series of a complex function $G(s)$ defined as $G(s) = (B^*(\lambda(1 - s)) - s)^{-1}$. $B^*(s)$ is the Laplace Stieltjes transform of the service time distribution [26]. In our case,

$$B^*(s) = \int_0^\infty b(t)e^{-st}dt = \beta e^{-s/\mu_0} + (1 - \beta)\mu/(\mu + s), \quad \text{for } Re(s) \geq 0\,.$$

The coefficient f can be computed by developing the Taylor series of the function $G(s)$ with some mathematical symbolic software. It can also be computed using the theorem of residues as follows:

$$f = \frac{1}{(K-2)!} \left. \frac{d^{K-2}G(s)}{ds^{K-2}} \right|_{s=0} = \frac{1}{2\pi i} \oint_{D_r} G(s) \frac{ds}{s^{K-1}}, \quad (3.1)$$

where D_r is any circle in the complex plane with center 0 and with radius chosen small enough so that the circle does not contain any pole of the function $G(s)$.

Now, adding FEC to the audio flow increases the transmission time of audio packets at the output interface of the bottleneck router. This increases the system's load which changes the stationary probabilities. Let $\alpha \in [0, 1]$ denote the ratio of the volume of FEC at the tail of a packet and the volume of the original packet. The new transmission time of audio packets becomes $(1 + \alpha)/\mu_0$, and the new system load becomes

$$\rho_\alpha = \lambda\big(\frac{\beta(1+\alpha)}{\mu_0} + \frac{(1-\beta)}{\mu}\big). \quad (3.2)$$

In the same way we can compute the new transform of the transmission time, the new coefficient f, and the new drop probability of an audio packet (it is the same for exogenous packets given that the arrival processes of both flows are Poisson). For simplicity, we assume in this part that the size of the bottleneck buffer in terms of packets does not change with the addition of FEC. Henceforth, when we add an index α to a function, we mean the new value of the function after the addition of an amount α of FEC. The quality after the addition of FEC becomes

$$Q_\alpha^\phi = (1 - \pi_\alpha) + U(\alpha)\pi_\alpha(1 - \pi_\alpha^\phi). \quad (3.3)$$

The first term corresponds to the quality obtained when the original audio packet is correctly received. The second term corresponds to the quality obtained when the redundant copy is correctly received and the original packet is lost. $U(\alpha)$ indicates how much quality we get from an amount α of FEC. The quantity π_α^ϕ indicates the probability that the packet carrying the redundancy is dropped given that the original packet is also dropped. ϕ represents the offset (in number of audio packets) between the original packet and the one containing its copy. In this section, and as in the first part, we will only consider the case of a linear utility function $U(\alpha) = \alpha$. We keep the study of the impact of other utility functions until Section 3.3.

The exact computation of Q_α^ϕ requires the computation of π_α^ϕ. This latter function is quite difficult to compute given the multiplexing of packets from both flows at the bottleneck. We must sum over all the possible numbers of non-audio packets inserted

between audio packets. What we can do instead is to find bounds on this probability and thus bounds on the quality. From Section 2.2.6, the probability that a packet is lost given that the n–th previous packet is lost is a decreasing function of n and it converges to π_α when $n \to \infty$. We can write $\pi_\alpha \leq \pi_\alpha^\phi \leq \pi_\alpha^0$, with π_α^0 being the probability that a packet (from any flow) is lost given that the previous packet is also lost. This gives us the following two bounds on the quality: $Q_\alpha^0 \leq Q_\alpha^\phi \leq Q_\alpha$, where

$$Q_\alpha^0 = (1-\pi_\alpha) + \alpha\pi_\alpha(1-\pi_\alpha^0)\,, \tag{3.4}$$

$$Q_\alpha = (1-\pi_\alpha)(1+\alpha\pi_\alpha)\,. \tag{3.5}$$

We use these two bounds to study how the audio quality varies for different amounts of FEC and for different intensities of audio traffic. We are sure that if we gain in Q_α^0 (lose in Q_α), we will gain (lose) in quality for any offset. Our main objective here is to show how the quality varies with FEC for different values of β. The analysis in the first part has shown that we always lose in quality for $\beta = 1$ (i.e., when the audio flow occupies 100% of the bandwidth at the bottleneck). All that we still need to do is to find the expression for the lower bound on the quality which can be found from the expression of π_α^0.

Theorem 3. *π_α^0 is given by $1 + \frac{B_\alpha^*(\lambda)-1}{\rho_\alpha}$, with*

$$B_\alpha^*(\lambda) = \beta e^{-\lambda(1+\alpha)/\mu_0} + (1-\beta)\mu/(\mu+\lambda)\,,$$

and ρ_α given by equation (3.2).

Proof: Consider a general $M/G/1/K$ queuing system. We have to compute the probability that a packet (say 1) is dropped given that the previous packet (say 0) is also dropped. Let $a(t) = \lambda e^{-\lambda t}$ be the distribution of time intervals between arrivals (of packets from both flows), and let $b(t)$ be the distribution of service times. Let $r(t)$ be the distribution of the residual time for the packet in service when packet 0 arrives (there is certainly a packet in service since packet 0 is supposed to be dropped). Using the results in [51], we write $r(t) = \frac{1-B(t)}{\sigma}$. $B(t)$ is the cumulative distribution function of the service time and σ is the average service time. In our case,

$$B(t) = \beta 1\{t \geq (1+\alpha)/\mu_0\} + (1-\beta)(1-e^{-\mu t}),$$

and $\sigma = \rho_\alpha/\lambda$. The probability π_α^0 is no other than

$$\pi_\alpha^0 = \int_0^\infty \frac{1-B(t)}{\sigma}(1-e^{-\lambda t})dt\,.$$

This is the probability that the inter-arrival time between packet 0 and packet 1 is less than the residual time of the packet in service, and we summarize over all the possible values of the residual service time. With a simple computation on this expression and by using the new values of the load intensity and the Laplace Stieltjes Transform of the service time distribution after the addition of FEC, we can prove the theorem.

□

3.2.3 Numerical results

We solve numerically the model for the two bounds on the audio quality (Eq. 3.4 and 3.5). We set K=10 packets and λ=10000 packets/s. Without loss of generality, we set μ_0=μ. We consider four values of ρ: 0.5, 0.8, 1, and 1.5. For every value of ρ, we plot the audio quality as a function of β and α. Recall that β is the fraction of audio packets and α is the amount of FEC. Figure 3.1 shows the results.

We conclude from the above figures that it is possible to obtain a gain with the simple FEC scheme we are studying. This requires that the intensity of the audio flow is small compared to the intensity of the other flows not implementing FEC. The gain diminishes as long as the intensity of the flows implementing FEC increases. It disappears when most of the flows start to implement FEC. This means that a FEC scheme with a simple linear utility function is not a viable mechanism. The gain that we may obtain in some cases is the result of the fact that the exogenous flows are not adding FEC and then they are not so aggressive as audio flows.

3.3 Utility functions and FEC performance

We seek now for a FEC mechanism able to improve the quality in the worst case when all flows in the network implement FEC. Suppose that the audio flow (or an aggregate of audio flows) uses alone the bottleneck resources ($\beta = 1$). The negative results obtained in the first part can be caused by the linear utility function adopted in the analysis. Adding an amount of FEC α increases the drop probability of an audio packet, which reduces the first term in the right-hand side of (3.3) more than it increases the second term. To get a gain, the second term must increase faster than the decrease in the first term. This can be achieved if the utility function increases faster than linearly as a function of α.

Indeed, it has been shown in [96] that multimedia applications have different utility functions than a simple linear one. These functions are typically non-linear. They are convex around zero and concave after a certain rate (between 0 and 1, with 1 being the rate that gives a utility function equal to one). Multimedia applications, and audio applications in particular, have strong delay constraints so that the quality deteriorates sharply when

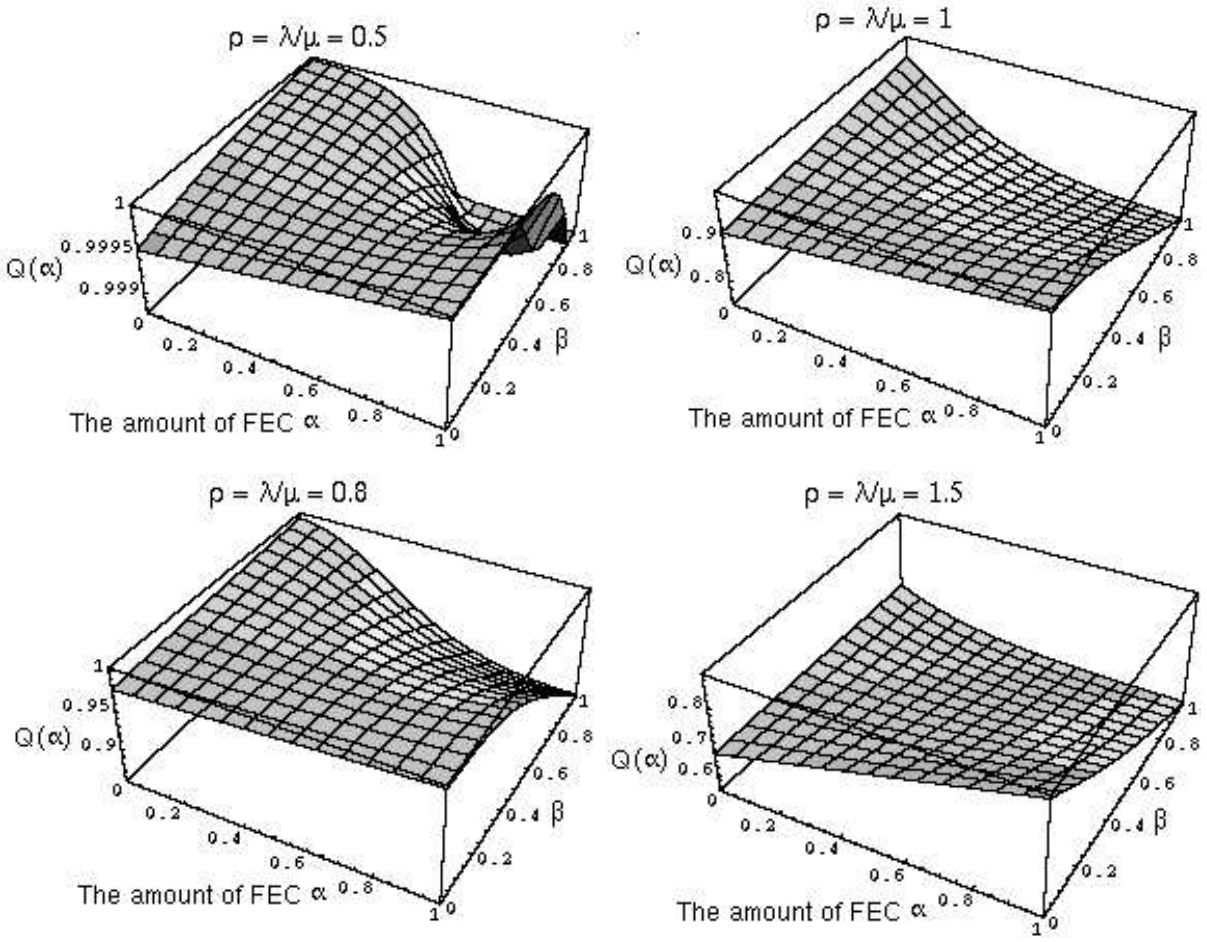


Figure 3.1: Audio quality for an $M/G/1/K$ queue with two flows: the audio flow and the exogenous flow. β represents the probability that an arriving packet belongs to the audio flow. We see clearly how when $\beta \to 0$, Q_α^ϕ starts having an increasing behavior, and this gain becomes more important as ρ increases.

the transmission rate falls below a certain value. This kind of utility functions can be very useful for FEC mechanisms since the reconstruction of a packet from a copy of volume $\alpha < 1$ may give a quality close to that of the original packet.

For the scenario of fixed amount of useful information per packet, we obtain a gain in quality when the redundant information we add to the original packet is small so that it does not contribute to a big increase in loss probability π, and at the same time, if reconstructed in case of the loss of the original packet, it gives a quality close to 1. Such behavior can be also obtained by coding FEC with a lower-rate codec as GSM [94]. Analytically speaking, a utility function leads to an improvement of quality if for $\alpha < 1$, we have

$$Q_\alpha^\phi = (1 - \pi_\alpha) + U(\alpha)\pi_\alpha(1 - \pi_\alpha^\phi) > 1 - \pi_0\,,$$

with π being the stationary drop probability before the addition of FEC.

In the scenario of fixed packet size (which does not depend on the amount of FEC), we gain by adding FEC if

$$Q_\alpha^\phi = U(1-\alpha)(1-\pi_\alpha) + U(\alpha)\pi_\alpha(1-\pi_\alpha^\phi) > 1-\pi$$

(where we assumed $U(0)=0, U(1)=1$).

In the sequel we shall show how FEC can improve the quality for the scenario of fixed amount of useful information per packet. Similar improvement can be shown to occur also in the scenario of fixed packet size.

3.3.1 Some bounds on quality improvement

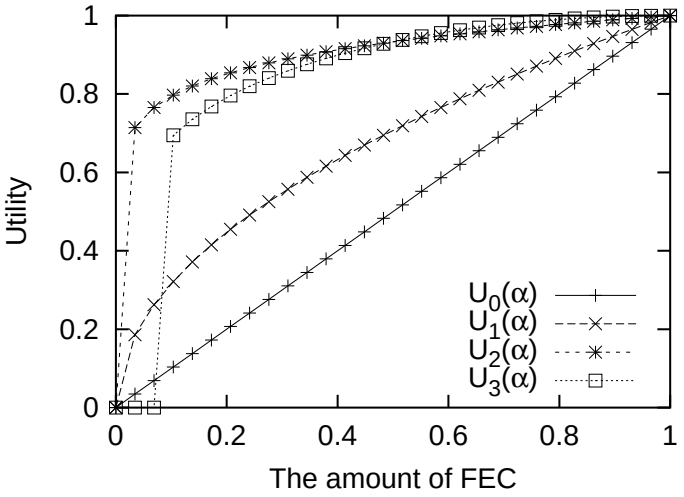
We already showed that we do not gain by multiplexing if the distribution of the sizes of all packets does not depend on the amount of FEC. We thus consider only the scenario in which the useful information is fixed per packet in the flow that adds FEC, and thus the packet size of this flow increases with FEC; we assume that other flows do not use FEC.

Again, we use here the bounds on the quality $Q_\alpha^0 \leq Q_\alpha^\phi \leq Q_\alpha$, with

$$\begin{aligned} Q_\alpha^0 &= (1-\pi_\alpha) + U(\alpha)\pi_\alpha(1-\pi_\alpha^0)\,, \\ Q_\alpha &= (1-\pi_\alpha)(1+U(\alpha)\pi_\alpha) \end{aligned}$$

A utility function that improves the lower bound improves the quality for any value of ϕ. A utility function that does not improve the upper bound will not lead to an improvement of quality whatever is the value of ϕ. Using the upper bound, we can find the maximum quality that this simple FEC scheme can give and this is for the best utility function. Indeed, the best utility function is one that jumps directly to one just after 0. This could be subjectively justified by using redundant packets coded at very small rates, as LPC or GSM. A very small amount of FEC ($\alpha \simeq 0$) that does not change the load of the network (i.e., that does not change π), will then lead to the same quality as the original audio packet. The question that one may ask here is: "*why to send large original packets in this case, given that we are able to obtain the same quality with small packets?*" The important processing time required by low-rate codes could be the answer to this question. We are not addressing this issue here, and we will only focus on the computation of an upper bound for the FEC scheme we are studying. Let Q^{max} be the maximum quality that we could obtain, thus $Q^{max} \simeq (1-\pi) + \pi(1-\pi) = 1-\pi^2$.

This Q^{max} has to be compared to the quality $(1-\pi)$ we get in the absence of FEC. Given that Q^{max} is larger than $(1-\pi)$, we conclude that we can always find a utility function and an offset between original packets and redundancies so as to gain in quality.



$$U_0(\alpha) = \alpha$$
$$U_1(\alpha) = \sqrt{\alpha}$$
$$U_2(\alpha) = \alpha^{\frac{1}{10}}$$
$$U_3(\alpha) = u(\alpha - \alpha_0)\left(\frac{1-\cos\pi\alpha}{2}\right)^{\frac{1}{10}}$$

Figure 3.2: Possible utility functions for rate adaptive applications.

Note that we are not considering the impact of the coding and decoding delays on the audio quality. The impact of these delays will be the subject of a future work. We also conclude from our analysis here that the FEC scheme we are studying cannot improve the quality by more than a factor of π. This means that the maximum gain in quality we could obtain is 100% and this gain is an increasing function of the network load. For example, for a network that drops 1% of packets, we cannot improve the quality by more than 1%, and for a network that drops 10% of packets we can get an improvement up to 10%.

Without loss of generality, we consider the family of utility functions that jump from zero to 1 at a value α_0. We denote such functions by $U_{\alpha_0}(\alpha)$. These are the utility functions of the so called hard real-time applications. We also consider the upper bound on the quality (an infinite offset). When increasing the amount of FEC with such applications from 0 to α_0, the quality deteriorates since its equal to $(1 - \pi_\alpha)$. When we cross α_0, the quality jumps from $(1 - \pi_{\alpha_0})$ to $(1 - \pi^2_{\alpha_0})$ and it resumes then its decrease with α. For such applications, the FEC scheme improves the quality if $\pi^2_{\alpha_0} < \pi$ and the maximum gain that we could obtain is a factor of $\frac{(\pi-\pi_{\alpha_0})}{(1-\pi)}$. This maximum gain corresponds to an amount of FEC slightly larger than α_0. It is not clear how the gain varies as a function of network load. But, what we can say here is that the FEC scheme behaves better with functions having a small α_0. After a certain threshold on α_0, the above condition becomes unsatisfied and it becomes impossible to gain in quality.

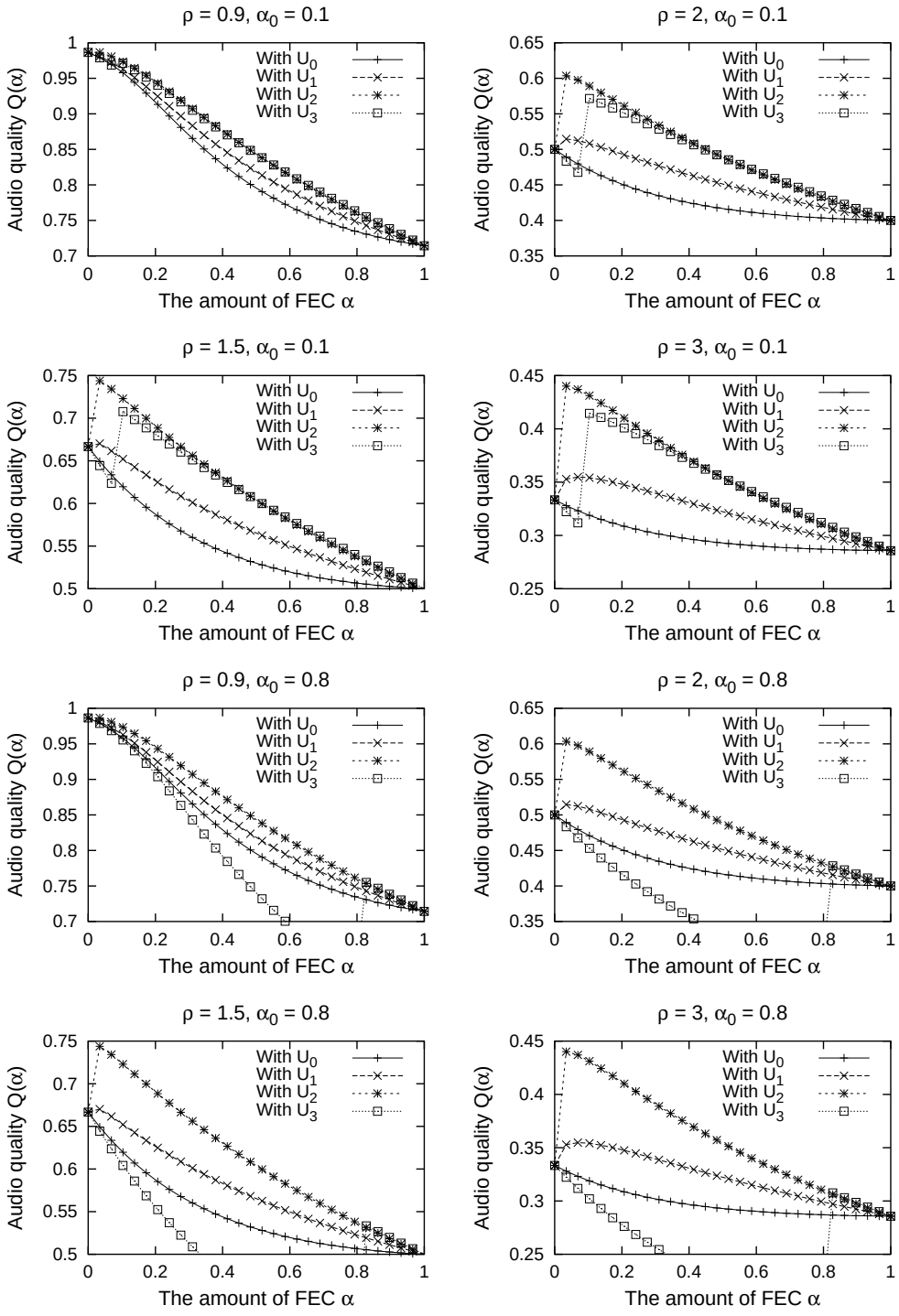


Figure 3.3: Lower bound for audio quality with $K = 20$, $\alpha_0 = 0.1$ (top) and $\alpha_0 = 0.8$ (bottom).

3.3.2 Some numerical results

We give in Figure 3.2 some possible utility functions[1] that could serve to our needs, and that are similar in their form to the utility functions proposed in [96]. In Figure 3.2, $U_3(\alpha)$

[1]The function $u(\alpha)$ is the step unit function. It is equal to 1 if $\alpha > 0$, and is equal to zero otherwise. α_0 represents the initial value giving a significant quality.

is plotted with $\alpha_0 = 0.1$.

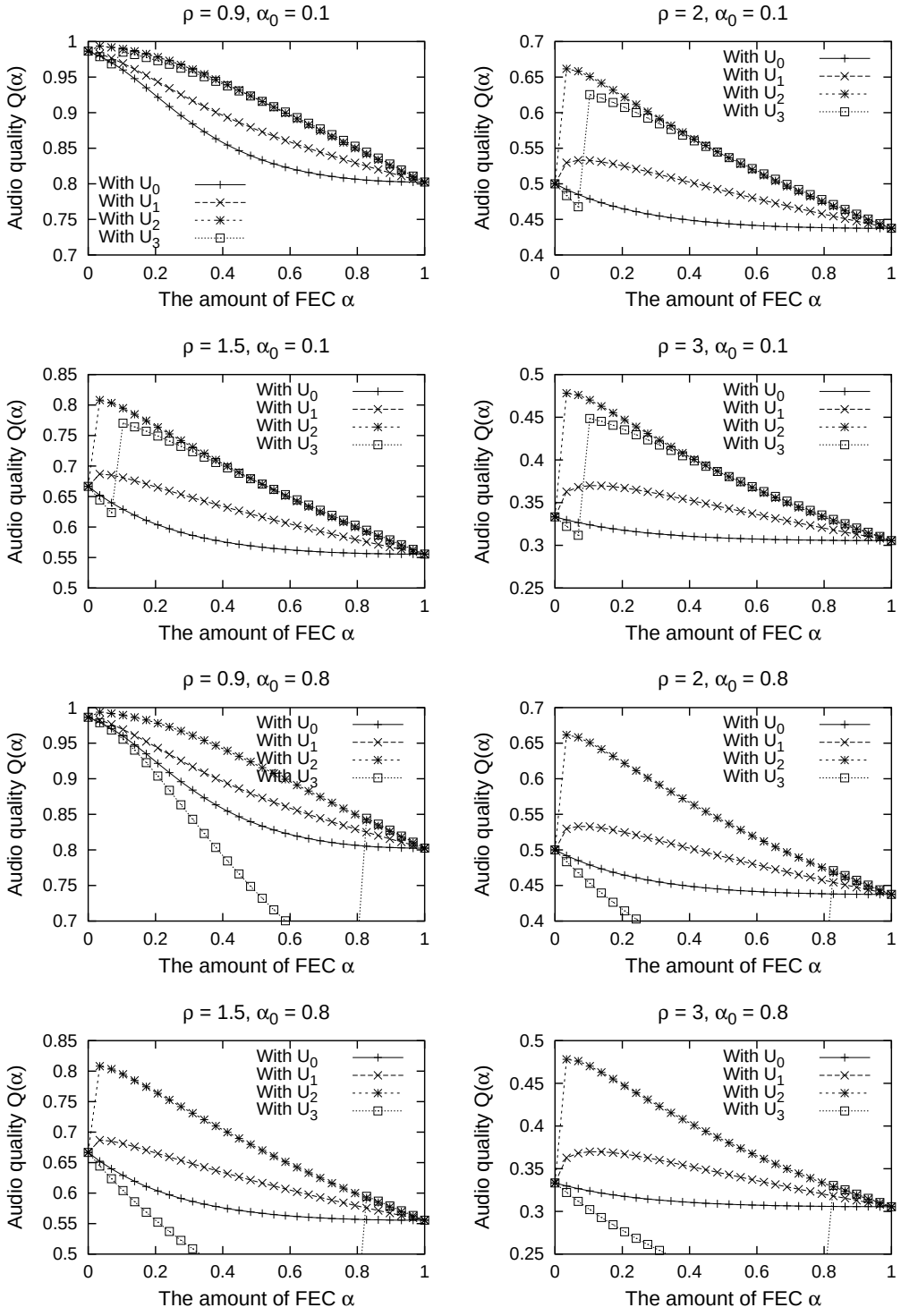


Figure 3.4: Upper bound for audio quality with $K = 20$, $\alpha_0 = 0.1$ (top) and $\alpha_0 = 0.8$ (bottom).

We solve the model numerically for the two bounds on the quality. We compute first the stationary distribution of the model for different values of α and ρ. We set K to 20 and λ to 10000 packets/sec. Then, for the the different utility functions in Figure 3.2, we plot the upper and lower bounds on the quality (Q_α and Q^0_α). Figure 3.3 shows plots for the lower bound and Figure 3.4 shows plots for the upper bound. The top four plots were obtained with $\alpha_0 = 0.1$ and the four bottom plots with $\alpha_0 = 0.8$ in both figures. We see clearly how the jump in the utility function results in a jump in quality and how this jump leads sometimes to better quality than that at α_0 and sometimes not. We also see how the case $U(\alpha) = \alpha$ does not present any improvement in quality.

3.4 Conclusions

In this chapter we extended the model we presented in Chapter 2 by studying cases where the FEC scheme may be helpful. The first case is when the audio flow has a small rate compared to an exogenous traffic that does not implement FEC. The second case is when the utility function of the audio application presents an important jump at small transmission rates. We gave conditions on where the FEC scheme can improve the audio quality. We also gave an upper bound on the gain in audio quality we could obtain.

Although we found some regions where the FEC scheme can behave well, we believe that this scheme is not the appropriate solution for improving the quality of audio applications. In the current Internet, this scheme profits from the fact that most of the other flows do not implement FEC. This will not be the case when a large number of flows start to add FEC to their packets. There is also a problem with the mechanism in case of applications with different utility functions than linear. We found that we get a gain especially when a small amount of redundancy gives the same performance as the big original packet. But it then seems intuitive to reduce the volume of original packets to reduce the drop probability and to gain even more in quality instead of adding FEC that does not improve the performance by more than 100%. There is no need to send long packets if we are able to get good quality with small ones.

Chapter 4

Playout delay control

4.1 Introduction

Delay, jitter, and packet loss in packet-switched wide-area networks are the main factors impacting audio quality of interactive multimedia applications. Today's Internet still operates in a best-effort basis, and thus, the impact of jitter, delay, and packet loss must be alleviated by employing end-to-end control mechanisms.

Audio applications such as NeVoT [91] and Rat [79] generate packets spaced at regular time intervals. The traffic generated by an audio source is divided into periods of activity, called *talkspurts*, and periods of silence. Silence periods are periods where no audio packets are transmitted.

Audio packets encounter variable delay while crossing the Internet, which is mainly due to the variable queuing time in routers. This delay variability modifies the periodic form of the transmitted audio stream. To playout the received stream, an application must reduce or eliminate this delay variability, by buffering the received packets and playing them out after a certain deadline. Packets arriving after their corresponding deadline are considered late and are not played out. If the playout delay is increased, the probability that a packet will arrive before its scheduled playout time also increases. This reduces the number of packets artificially dropped in the playout buffer. However, very long playout delays have a negative impact on the interactivity of an audio session. Obviously, there exists a trade-off between delay and loss due to late packets. For interactive audio, packet delays up to 400 ms and loss rates up to 5% are considered adequate [45].

We focus in this chapter on the tradeoff between loss and delay for playout delay control algorithms. Using measurements of packet end-to-end delay of audio sessions done with NeVoT, we present and validate a Moving Average (MA) algorithm that adjusts the playout delay at the beginning of each talkspurt. To prove the efficiency of our algorithm, we compare it with earlier work done by Ramjee et al. [82]. We present two versions of our algorithm: an offline algorithm and an online one. First, we explain the offline MA algorithm, which serves as a reference for our work. Then, we show how an online hybrid algorithm can be implemented by combining the ideas proposed by Ramjee et al. with the moving average algorithm we propose.

Moving average estimators have been used in different fields of research. For example, in [117] Zhang et al. show that moving average estimation is a good approach to predict end-to-end delay for TCP connections. We use a moving average technique not for predicting end-to-end delay, but rather for predicting an optimal value of the playout delay per-talkspurt in an audio session.

One characteristic that most of the playout delay adaptation algorithms lack is the ability to fix the loss percentage to some a priori value. By changing a measure of variability, the algorithms proposed by Ramjee et al. can achieve different loss percentages. However, there is no explicit relationship between the measure of variability that we can adapt in these algorithms and the average loss percentage. The average loss percentage can change from one audio session to another, even if this parameter is kept unchanged. Here lies the main contribution of our work. The moving average algorithm we propose adjusts the playout delay from talkspurt to talkspurt, given a desired target of average loss percentage p. Our algorithm ensures that the average loss percentage we obtain during the session is close, if not equal, to the target value. At the same time, and in most of the cases, our algorithm realizes this target with a smaller average playout delay than the one we need to obtain the same average loss percentage with the algorithms proposed by Ramjee et al. For practical loss percentages, we validate our algorithm and those of Ramjee et al. using real packet audio traces. By using collected audio traces we can compare the algorithms under the same network conditions.

The remainder of this chapter is as follows. Section 4.2 provides some additional background and describes the packet delay traces used for validation, as well as the notation we use throughout this chapter. In Section 4.3, we describe some related work on playout delay algorithms. Section 4.4 describes the performance measures we consider to validate our algorithm and to compare it with the algorithms of Ramjee et al. We present in Section 4.5 our moving average algorithm. Section 4.6 describes, based on results from Section 4.5, that it is better to predict a function of the delay rather than the optimal delay itself. Section 4.7 compares the performance of an online hybrid implementation of

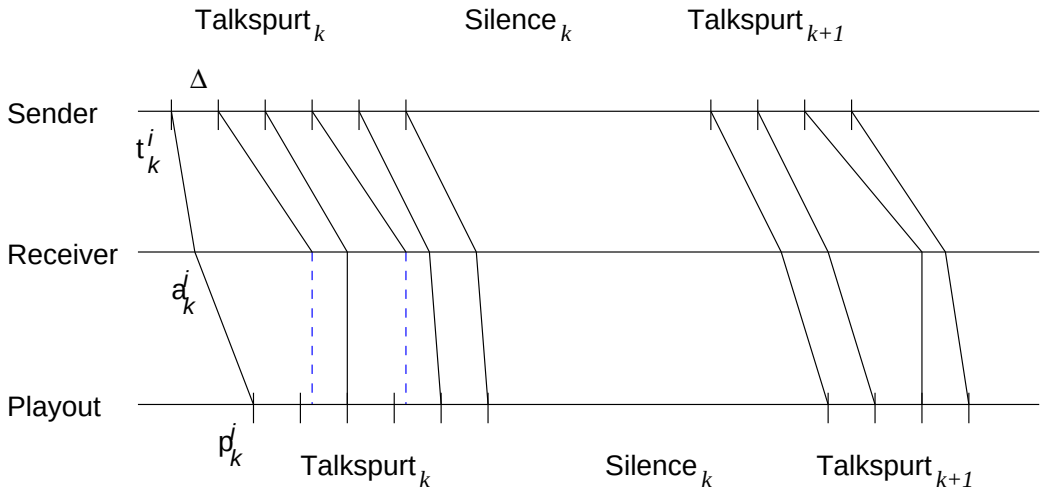


Figure 4.1: The timings between the transmission, reception and playout of packets.

the moving average algorithm with the Ramjee et al. algorithms. Finally, we conclude this chapter in 4.8.

4.2 Some background

Receivers use a playout buffer to smooth the stream of audio packets. This smoothing is done by delaying the playout of packets to compensate for variable network delay. The playout delay can be either constant through the whole session, or can be adjusted between talkspurts. Moreover, in a recent work [55], it has been shown that by using a technique called *packet scaling*, it is possible to change the playout delay from packet to packet while keeping the resulting distortion within tolerable levels. Here we only focus on the per-talkspurt playout delay adaptation approach.

Figure 4.1 shows the different stages incurred in an audio session. The i-th packet of talkspurt k is sent at time t_k^i, it arrives at the receiver at time a_k^i, and is held in the smoothing receiver's playout buffer until time p_k^i, when it is played out. Inside a talkspurt, packets are equally spaced at the sender by time intervals of length Δ seconds.

By delaying the playout of packets and dropping those that arrive after their deadline, we are able to reconstruct the original periodic form of the stream. This adaptation results in a regenerated stream having stretched or compressed silence periods compared to the original stream. These changes are not noticeable by the human ear if they are kept within tolerable small levels.

In Figure 4.1, a dropped packet due to a late arrival is represented by a dashed line. A packet is artificially dropped if it arrives after its scheduled deadline p_i^k. The loss percentage can be reduced by increasing the amount of time that packets stay in

Table 4.1: Description of variables.

Param.	Meaning
L	The total number of packets arriving at the receiver during a session.
N	The total number of talkspurts in a session.
N_k	The number of packets in talkspurt k.
t_k^i	The time at which the i-th packet of talkspurt k is generated at the sender.
a_k^i	The time at which the i-th packet of talkspurt k is received.
d_k^i	The variable portion of the end-to-end delay of the i-th packet in talkspurt k. $d_k^i = a_k^i - t_k^i - \min_{\substack{1\le k\le N \\ 1\le i\le N_k}} (a_k^i - t_k^i)$.
p_k^i	The time at which packet i of talkspurt k is played out.

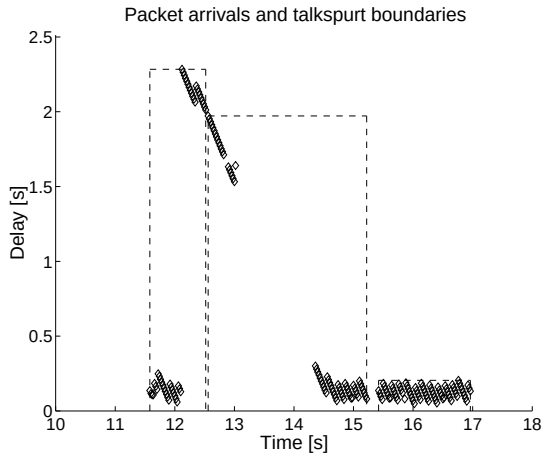
the playout buffer. An efficient playout adaptation algorithm must take into account the trade-off between loss and delay in order to keep both parameters as low as possible.

Throughout this chapter, we use the notation described in Table 4.1. For the validation of our algorithm, we consider the packet traces generated with the NeVoT audio tool that are described in [65]. NeVoT is an audio terminal program used to establish audio conversations in the Internet. The traces we use contain the sender and receiver timestamps of transmitted packets that are needed for the implementation of any playout adaptation algorithm. In these traces, one 160 byte audio packet is generated approximately every 20 ms when there is speech activity. A description of the traces (reproduced from [65]) is depicted in Table 4.2.

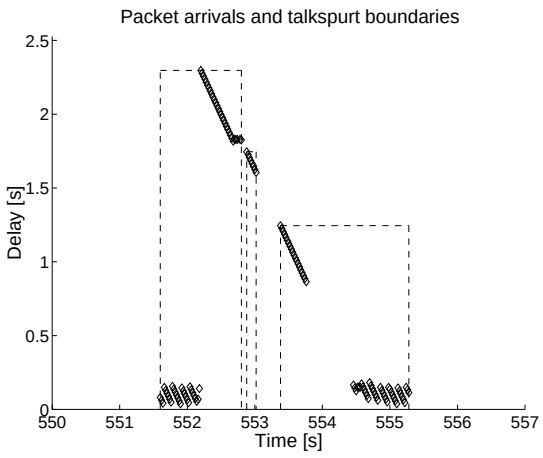
A typical sample of packet end-to-end delays is shown in Figure 4.2. A packet is represented by a diamond and talkspurt boundaries by dashed rectangles. The x-axis represents the time elapsed at the receiver since the beginning of the audio session. Only the variable portion of the end-to-end delay (d_k^i) is represented on the y-axis of Figure 4.2. To this end, the constant component of the end-to-end delay (mostly caused

Table 4.2: Description of the traces.

Trace	Start time	Length [s]	Talkspurts	Packets
1	08:41pm 6/27/95	1348	818	56979
2	09:58am 7/21/95	1323	406	24490
3	11:05am 7/21/95	1040	536	37640
4	09:20pm 8/26/93	580	252	27814
5	09:00pm 9/18/93	1091	540	52836
6	00:35am 9/24/93	649	299	23293



(a) A delay spike spanning through two consecutive talkspurts.



(b) A delay spike spanning through three consecutive talkspurts.

Figure 4.2: Delay spikes in end-to-end delay measurements.

by the propagation delay) is removed by subtracting from packet delays their minimum over all the corresponding trace. By considering the variable portion of the end-to-end delay, synchronization between sender and receiver clocks can be avoided[1] [65].

We observe in Figure 4.2 the presence of delay spikes. This phenomenon in end-to-end delay has been previously reported in the literature [12, 82]. Delay spikes represent a serious problem for audio applications since they affect the performance of playout delay adaptation algorithms. A *delay spike* is defined as a sudden large increase in the end-to-end delay followed by a series of packets arriving almost simultaneously, leading to the completion of the spike [82].

Delay spikes can be contained within a single talkspurt or can span over several talkspurts. Figure 4.2(a) shows a delay spike spanning through two consecutive talkspurts. Figure 4.2(b) shows a delay spike spanning over three talkspurts. Since the playout delay is generally changed between talkspurts, a playout algorithm behaves better when delay spikes span over more than one talkspurt. Only in this way, a playout algorithm can react adequately to the spike by setting the playout delay according to the experienced delay. If the spike vanishes before the end of a talkspurt, the playout algorithm will not have enough time to set the playout time accordingly.

In the next section, we briefly describe the algorithms proposed by Ramjee et al. Playout delay is adapted from talkspurt to talkspurt based on past statistics of the delay

[1]Later, when comparing the performance of different algorithms, all graphics consider the variable portion of end-to-end delays.

process. The playout delay of the first packet of each talkspurt is the basetime of the deadlines for subsequent packets in the same talkspurt. This principle is the basis for most of the existing playout adaptation algorithms [49, 65, 73, 82].

4.3 Related work

Extensive research work has been done in the area of adaptive playout mechanisms [30, 49, 55, 58, 65, 73, 82]. In [82], Ramjee et al. propose four algorithms for playout delay. All the four algorithms proposed in [82] compute an estimate of the average end-to-end delay and a measure of variability of delay similarly to the computation of round-trip-time estimates by TCP for the retransmission timer. We denote them as $\hat{d}_k^i$ and $\hat{v}_k^i$ respectively. These statistics are used to set the playout time for a talkspurt.

The algorithms that perform better in [82] are 1 and 4. We rename these algorithms as A and B respectively, and refer to them as such throughout the rest of the chapter.

To calculate $\hat{d}_k^i$ and $\hat{v}_k^i$, the packet's sender and receiver timestamps, t_k^i and a_k^i, are read from a trace file. Both algorithms differ only in the way they calculate $\hat{d}_k^i$ and $\hat{v}_k^i$. Algorithm A computes these statistics as follows:

$$\hat{d}_k^i = \alpha \hat{d}_k^{i-1} + (1-\alpha) d_k^i, \quad \text{and} \quad \hat{v}_k^i = \alpha \hat{v}_k^{i-1} + (1-\alpha)|\hat{d}_k^i - d_k^i|,$$

where $d_k^i = a_k^i - t_k^i$, and α has the default value of 0.998002.

Algorithm B is described in [82] but we also sketch it in Figure 4.3 for completeness. It operates in two modes: normal mode and spike (or impulse) mode. In normal mode, it behaves like Algorithm A but with different coefficients. When the difference between two consecutive delay values exceeds a given threshold, algorithm B triggers the spike mode. During this mode, the variable `var` is updated with an exponentially decreasing value that adjusts to the slope of the spike. The end of a delay spike is detected when `var` reaches an enough small value, and the algorithm returns to normal mode.

Once $\hat{d}_k^i$ and $\hat{v}_k^i$ are computed, the playout time of the i-th packet of talkspurt k is set by both algorithms as follows:

$$p_k^i = \begin{cases} t_k^i + \hat{d}_k^i + \beta \hat{v}_k^i, & \text{for } i = 1 \ . \\ p_k^1 + (t_k^i - t_k^1), & \text{for } 1 < i \leq N_k \ . \end{cases} \tag{4.1}$$

These values are computed for each packet but the playout time is changed only at the beginning of a talkspurt. By varying β one is able to achieve different loss probabilities and different average playout delays. In [82], β is set equal to the constant value of 4.

```
1.  d_k^i = a_k^i − t_k^i;
2.  if (mode == NORMAL) {
3.     if (|d_k^i − d_k^{i−1}| > 2|v̂_k^i| + 800) {
4.        var = 0; /* Detected beginning of spike */
5.        mode = SPIKE;
6.     }
7.  }
8.  else {
9.       var = var/2 + |(2d_k^i − d_k^{i−1} − d_k^{i−2})/8|;
10.      if (var ≤ 63) {
11.         mode = NORMAL; /* End of spike */
12.         d_k^{i−2} = d_k^{i−1};
13.         d_k^{i−1} = d_k^i;
14.         return;
15.      }
16.  }
17.  if (mode == NORMAL)
18.      d̂_k^i = 0.125d_k^i + 0.875d̂_k^{i−1};
19.  else
20.      d̂_k^i = d̂_k^{i−1} + d_k^i − d_k^{i−1};
21.  v̂_k^i = 0.125|d_k^i − d̂_k^i| + 0.875v̂_k^{i−1};
22.  d_k^{i−2} = d_k^{i−1};
23.  d_k^{i−1} = d_k^i;
24.  return;
```

Figure 4.3: Algorithm B

Larger values of β allow to obtain lower loss percentages due to late arrivals but at the cost of a longer average playout delay.

Algorithm A is slow in detecting delay spikes, but it maintains a good average playout delay over an audio session. Algorithm B reacts faster to delay spikes, but it underestimates the playout delay at the end of the spike [65].

To compare our moving average algorithm with algorithms A and B, we use the performance measures defined in [65]. For the clarity of the presentation, we redefine these measures in Sect. 4.4.

4.4 Performance measures

To assess the performance of a playout adaptation algorithm, we focus on the total number of packets that are played out during an audio session, as well as on the experienced average end-to-end delay. Suppose we are given a packet audio trace with the sender and receiver

timestamps of audio packets. Let p_k^i, N, L, N_k, t_k^i, and a_k^i be defined as in Table 4.1. As in [65], we define r_k^i to be a variable indicating if packet i of talkspurt k is played out or not. So, r_k^i is defined as:

$$r_k^i = \begin{cases} 0, & \text{if } p_k^i < a_k^i \\ 1, & \text{otherwise.} \end{cases}.$$

The total number of packets, T, played out in an audio session is thus given by:

$$T = \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i. \tag{4.2}$$

The average playout delay, D_{avg}, is equal to :

$$D_{avg} = \frac{1}{T} \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i [p_k^i - t_k^i]. \tag{4.3}$$

Finally, the loss percentage, l, is equal to :

$$l = \frac{L - T}{L} \times 100. \tag{4.4}$$

4.5 Moving Average prediction

Algorithms A and B are good in maintaining a low overall average playout delay and reacting to delay spikes. However, they lack a parameter allowing to have a direct control on the overall loss percentage during an audio session. It would be desirable to come up with an algorithm that sets the playout delay in a way to get a loss percentage p, given a trace of N talkspurts and L packets. By varying the parameter β in algorithms A and B, it is possible to obtain different loss percentages, but one is unable to have any particular control on this parameter. We describe in this section our moving average predictor (MA) for playout delay that takes as input the desired loss percentage per-session, p, and a packet delay trace. It returns a set of estimated playout delay values leading to an average loss percentage close, if not equal to the desired value p.

4.5.1 The Model

Let D_k be the optimal playout delay at the beginning of talkspurt k, and let p be the desired average loss percentage per-session. We mean by *optimal playout delay* the playout delay that makes the number of losses per talkspurt the closest to $p \times N_k$, N_k being the number

of audio packets received during the k-th talkspurt. By controlling the loss percentage per-talkspurt to p, we are sure that the overall loss percentage during the whole audio session is also close to p. We compute D_k as follows, let d_k^j be the variable portion of the end-to-end delay of the j-th packet in talkspurt k. For each talkspurt, $1 \leq k \leq N$, we sort in ascending order the packet end-to-end delay values to obtain N new ordered sets $\{d_{k_{sort}}^j\}$, with $1 \leq j \leq N_k$. We set the optimal playout delay of the k-th talkspurt to the following value:

$$D_k = d_{k_{sort}}^i, \quad i \leq N_k, \tag{4.5}$$

with $i = \texttt{round}((1-p)N_k)$. Thus, if $d_k^i \leq D_k$, the i-th packet of talkspurt k is played out, otherwise the packet is dropped due to a late arrival.

We present now our moving average predictor. Consider that we have a set of optimal delay values in the past $\{D_k, D_{k-1}, D_{k-2}, \ldots\}$, and that we want to predict the value of D_{k+1}. The predicted value of D_{k+1} is denoted by $\hat{D}_{k+1}$, and is taken as a weighted average of the last M values of the process $\{D_k\}$. Thus,

$$\hat{D}_{k+1} = \sum_{l=1}^{M} a_l D_{k-l+1}. \tag{4.6}$$

The coefficients a_l in (4.6) must be chosen in a way that minimizes the mean square error between $\hat{D}_k$ and D_k, i.e. $\mathbb{E}[(D_k - \hat{D}_k)^2]$. The desired coefficients are the solution of the set of the so-called normal equations [76]:

$$\sum_{m=0}^{M-1} a_{m+1} r_D(m-l) = r_D(l+1),\ l = 0, 1, \ldots, M-1. \tag{4.7}$$

In (4.7), $r_D = \mathbb{E}[D_k D_{k+l}]$ is the lag-l autocorrelation function of the process $\{D_k\}$. The exact form of the autocorrelation function is unknown, but it can be estimated using the past values of the process $\{D_k\}$. Suppose we have K values in the past, we can thus write

$$r_D(r) \simeq \frac{1}{K-|r|} \sum_{k=1}^{K-|r|} D_k D_{k+|r|}, \quad r = 0, \pm 1, \pm 2, \ldots, \pm(K-1). \tag{4.8}$$

The set of normal equations (4.7) can be solved using single matrix-vector operations. For large values of M (say $M > 100$), the well known Levinson-Durbin algorithm may be preferred [76].

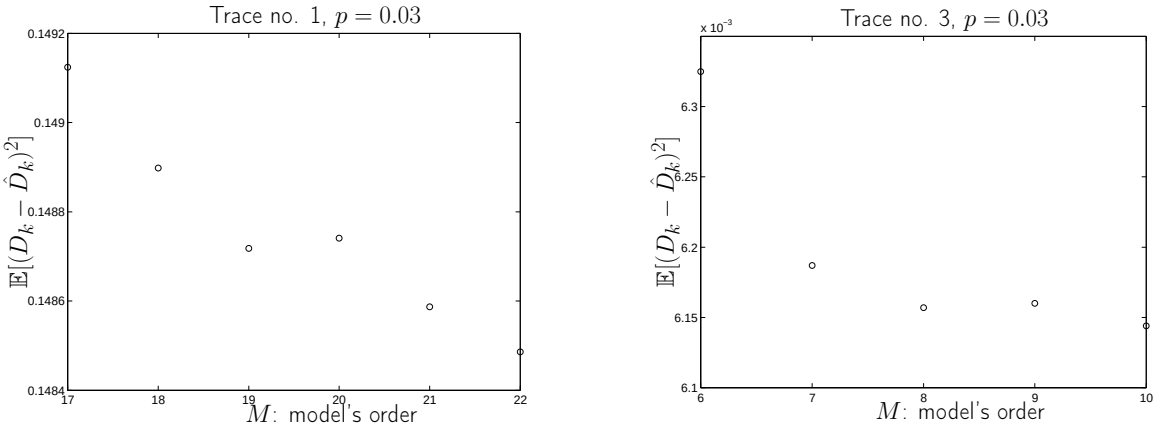


Figure 4.4: Model order selection for the MA algorithm.

M is called the *model's order*. Figure 4.4 illustrates how M is calculated. For a given packet trace, starting with $M = 1$, we compute all the values of $\{\hat{D}_k\}$ and estimate $\mathbb{E}[(D_k - \hat{D}_k)^2]$. Then, we increase M and we repeat the process. The model's order is taken equal to the lowest value of M preceding an increase in the mean square error. For example, for trace 1 the algorithm chooses M equal to 19, and for trace 3 it chooses M equal to 8. There exist different methods for selecting the model's order e.g. Double Sided t-test, Minimum Description Length, and Final Prediction Error. The reader is referred to [50] for a detailed description of these methods.

4.5.2 The moving average algorithm

We describe now our moving average algorithm for playout delay. The algorithm takes as input a packet delay trace with sender and receiver timestamps, and looks for $\{\hat{D}_k\}$, the estimates of optimal playout delays $\{D_k\}$. For each past talkspurt, the individual end-to-end delay values are sorted, and $\hat{D}_k$ is computed as in (4.6). $\hat{D}_k$ is calculated for each talskpurt as a weighted average of the last M talkspurts, for $k = M + 1, M + 2, \ldots, N$. Later, when evaluating the average playout delay and the loss percentage per-session, we discard in the computation the first M talkspurts.

Figure 4.5 depicts a pseudo-code version of the MA algorithm. The `getOptDelay()` function takes as input the whole set of end-to-end delay values, $\vec{d}$, and the desired loss percentage per-session p. Then, it applies (4.5) to return a set of optimal per-talkspurt delay values $\{D_k\}$. The first `for` loop solves the normal equations for a_l to compute $\hat{D}_{k+1}$ for each talkspurt, then it calculates $\mathbb{E}[(D_k - \hat{D}_k)^2]$ for different values of the model's order m, and holds the result in the vector $\overrightarrow{mse}$. Next, `getOrder`($\overrightarrow{mse}$) is called to find the model's order, M, by choosing the lowest value of M preceding an increase in $\overrightarrow{mse}$.

```
1.  D_k ← getOptDelay(d⃗,p);
2.  R ← autocorr(D_k, N);
3.
4.  for m = 1 to N {
5.      /* Get the weights */
6.      a⃗ = solve(R, m);
7.      /* Compute D̂_{k+1} for each talkspurt */
8.      D̂_{k+1} = Σ_{l=1}^{m} a_l D_{k-l+1};
9.      /* Update the mse vector for this value of m */
10.     mse(m) = E[(D_k − D̂_k)^2];
11. }
12.
13. M = getOrder(mse⃗);
14. a⃗ = solve(R, M);
15.
16. for k = M to N − 1 {
17.     D̂_{k+1} = Σ_{l=1}^{M} a_l D_{k-l+1};
18.     p^1_{k+1} = t^1_{k+1} + D̂_{k+1};
19.     20.  for j = 2 to N_{k+1}
21.              p^j_{k+1} = p^1_{k+1} + t^j_{k+1} − t^1_{k+1};
22. }
```

Figure 4.5: The MA algorithm.

Then, we compute the coefficients a_l with the value of M just found.

The last `for` loop computes $\hat{D}_{k+1}$ for each talkspurt and the playout times p_k^i. The playout time of the i-th packet of talkspurt k is set as follows:

$$p_k^i = \begin{cases} t_k^1 + \hat{D}_k, & \text{for } i = 1 \\ p_k^1 + (t_k^i - t_k^1), & \text{for } 1 < i \leq N_k. \end{cases} \tag{4.9}$$

The moving average algorithm requires the knowledge of the characteristics of the process $\{D_k\}$. Assuming the process $\{D_k\}$ is stationary during all the audio session, the best performance of this algorithm is obtained when it is run offline or after a large number of samples is collected.

4.5.3 The problem with low p

As our simulation results will show, the MA algorithm described in Sect. 4.5.2 deviates from the desired loss percentage p. See in (4.5) how the value of D_k depends on the

talkspurt size N_k, and on the desired loss percentage p. For a given value of p, (4.5) returns the delay value closest to $p \times N_k$. Thus, when computing $\hat{D}_k$, there will be a deviation of the overall perceived loss percentage from the one we desire. The highest deviation is for very low values of p. The algorithm leads to a loss percentage longer than the desired one. To deal with this deviation, for the range $0.005 \leq p \leq 0.02$, we allow our MA algorithm to slightly increase the playout delay by the following offset:

$$\Delta_{\hat{D}_k} = f(p)\sqrt{\mathbb{E}[(\hat{D}_k - D_k)^2]}. \tag{4.10}$$

In this way, the playout delay is increased as a function of the measured mean square error between D_k and $\hat{D}_k$, and as a function of p. Since the deviation of the measured loss percentage increases for small values of p, we set f to $f(p) = -\delta \times (\frac{p}{p_{\max}} - 1)$, where δ is a constant controlling how much we increase the playout delay as a function of the square root of $\mathbb{E}[(D_k - \hat{D}_k)^2]$. So, as p increases in the range $(0, p_{\max}]$, $\hat{D}_k$ converges to its original form (4.6), and if p decreases a longer offset is used. We set $p_{\max} = 0.02$ and $\delta = 0.5$. This allows to reduce considerably the deviation of the measured loss percentage from p, without impacting much the delay.

The following lines must be added to the pseudo-code shown in Figure 4.5 between lines 17 and 18:

if $p \leq 0.02$
  $\hat{D}_{k+1} \leftarrow \hat{D}_{k+1} + \Delta_{\hat{D}_k}$.

To see the gain obtained when applying (4.10), we plot in Figure 4.6, for $p \in [0.005, 0.02]$ the performance of the original MA algorithm before and after this change. The x-axis represents the total measured loss percentage due to late losses, l, and the y-axis plots the average playout delay, D_{avg}, for discrete values of p from 0.005 to $p_{\max} = 0.02$, with p increasing 0.005 at each step. We call this new algorithm *MA+offset*, and we refer to it as such during the rest of the chapter.

The deviation of the loss percentage for the MA+offset algorithm is much lower, while keeping the average playout delay within reasonable values. The gain we get compared to the basic MA algorithm is very clear. For trace 1, the MA+offset algorithm reaches loss percentages of 1.7% compared to 5.4% in the basic MA algorithm. The MA+offset algorithm is beneficial for all traces, and the gain is higher for trace 2 and trace 6, where the deviation of the desired loss percentage is now much lower compared to the original case.

Section 4.5.4 compares the performance of our MA+offset algorithm with algorithms A and B.

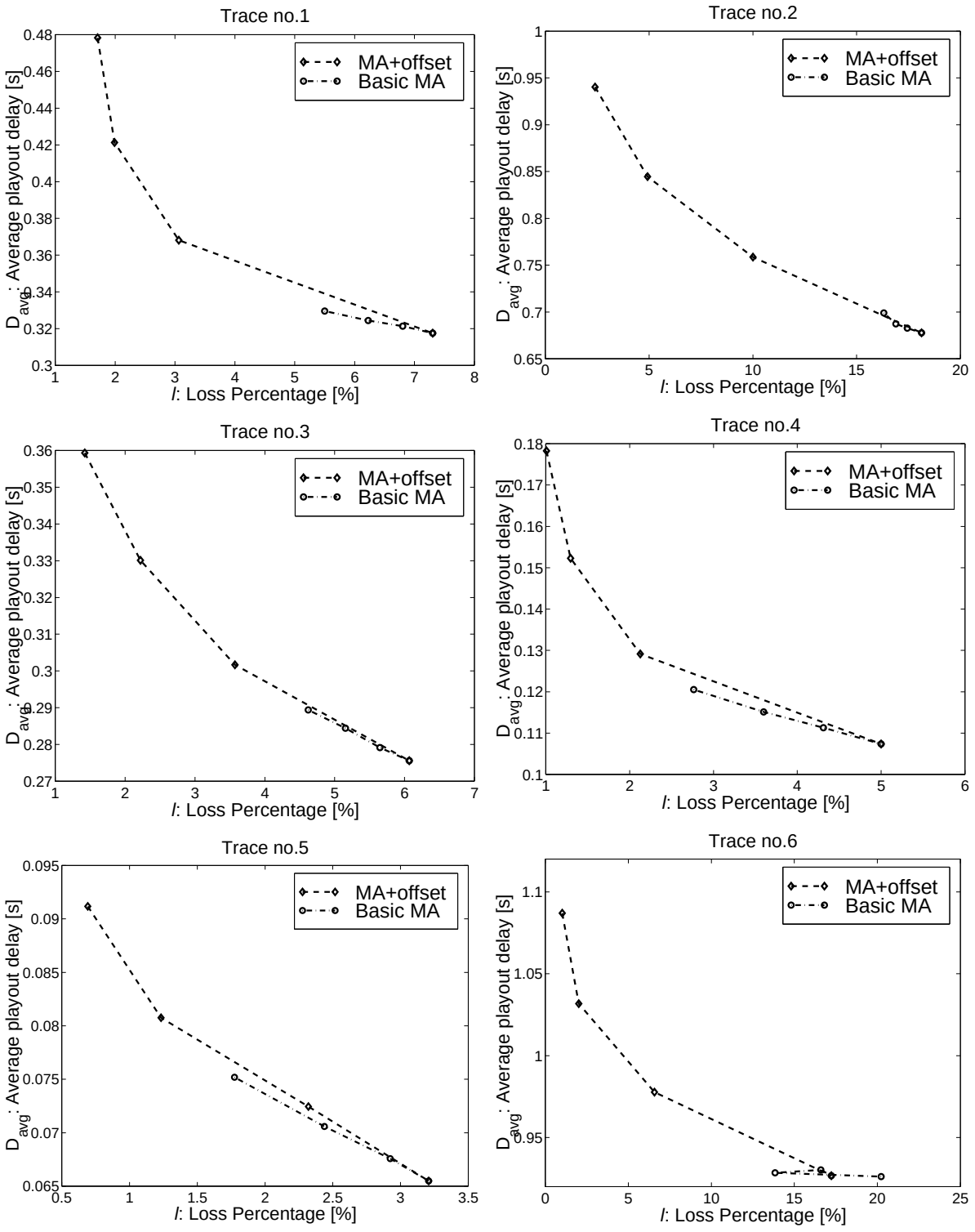


Figure 4.6: Performance of the MA algorithm before and after adding the offset to $\hat{D}_k$ for $p \in [0.005, 0.02]$.

4.5.4 Performance comparison

To evaluate each of the three algorithms, we use a simulator that reads in an input file containing the sender and receiver timestamps of each packet of an audio session. Then, each algorithm is executed, and we use expressions (4.3) and (4.4) to compute the average playout delay and the loss percentage during an audio session.

As pointed out above, algorithms A and B are unable to get a particular target of loss percentage p. Thus, to obtain different loss percentages in Algorithms A and B we vary β in (4.1) from 1 to 20; larger values of β allow to get lower loss percentages, at the expense of a higher average playout delay. We compare the performance of algorithms A and B with our MA+offset algorithm for $0.005 \leq p \leq 0.2$. Loss percentages smaller than 5% are rather desirable for interactive audio applications [45].

Figure 4.7 plots the corresponding results for each trace. The x-axis represents the perceived loss percentage per-session l, and the y-axis represents the average playout delay D_{avg}. Each execution of an algorithm gives a single point in the graphic. The plots in Figure 4.7 were obtained by connecting the discrete points returned by each approach.

In trace 1, we see how for loss percentages greater than 5%, the performance of the three algorithms is quite similar, with algorithms A and B having a slightly better performance than the MA+offset algorithm for loss percentages between 5% and 11%. For loss percentages lower than 5%, the performance of algorithms B and MA+offset remains similar but they outperform algorithm A with a maximum gain on average playout delay of about 40% of the MA+offset algorithm compared to algorithm A. Trace 2 is the only multicast session and has a large network loss percentage of about 50%, it has also long inactivity periods of up to 2 minutes. The MA+offset algorithm clearly outperforms algorithms A and B for the whole range of loss percentage and average playout delay, with a maximum gain on playout delay of about 200% compared to algorithm B. In trace 3, algorithms A and B remain close to the MA+offset algorithm, with the MA+offset algorithm giving better performance for the whole range of loss percentage and average playout delay. For traces 4 to 6, algorithms A and MA+offset remain close in both, loss percentage and average playout delay, outperforming algorithm B. For loss percentages lower than 3%, the MA+offset algorithm performs better than algorithms A and B. This difference in performance is clearly seen in trace 6, where the MA+offset algorithm shows a considerable gain over algorithms A and B.

Deviations of loss percentage persist in the MA+offset algorithm. The highest deviations in Figure 4.6 are for traces 2 and 6. Both are the shortest traces, they suffer from high variations on end-to-end delay, and high network loss due to congestion. Thus, the autocorrelation function does not have useful information about the process $\{D_k\}$, and consequently the estimated values $\{\hat{D}_k\}$ are inaccurate. Section 4.6 describes a transfor-

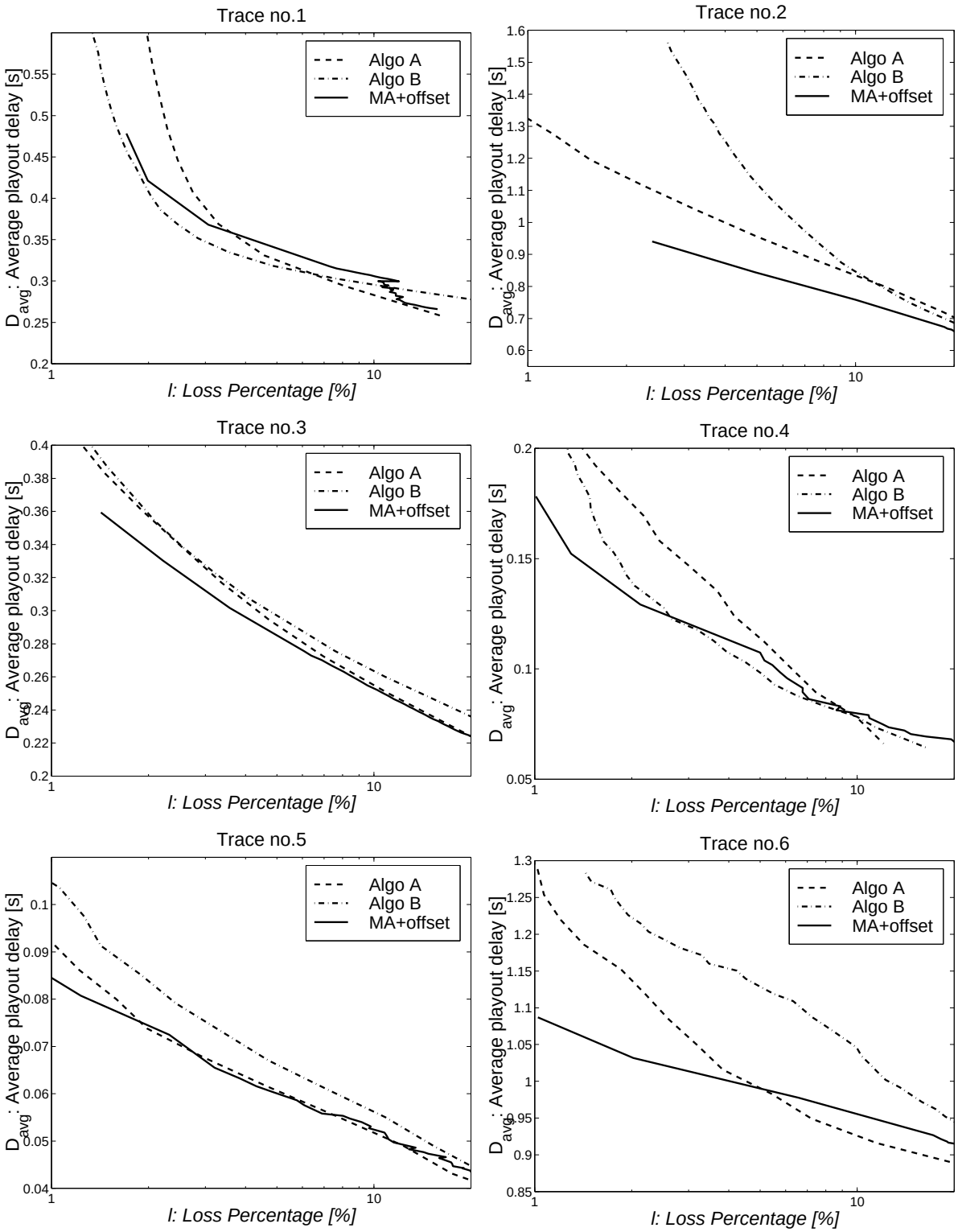


Figure 4.7: Performance comparison of algorithms A, B, and the MA+offset algorithm.

mation that can be done to reduce the deviations of the MA+offset algorithm.

4.6 Bias and Transformation

Our scheme is designed with the main objective to control the loss percentage to a certain value p, while minimizing the average playout delay. Here is the strength of our scheme compared to other schemes in the literature, where we do not have a direct control on the loss percentage. Our control on this parameter has been done till now by controlling the optimal playout delay per talkspurt D_k. But, the relationship between the playout delay and the loss percentage may not be linear. This may cause a deviation of the perceived loss percentage from the desired one. Technically said, our predictor is unbiased with respect to D_k, however it may be biased with respect to the loss percentage per talkspurt. We illustrate this bias by the following analysis.

Our moving average predictor of D_k ensures that $\mathbb{E}[D_k] = \mathbb{E}[\hat{D}_k]$. Let d_k^i, $1 \leq i \leq N_k$, be the delay of the i-th packet of talkspurt k. The way we define D_k also ensures that $\mathbb{E}[1\{d_k^i > D_k\}] = p$, with $1\{A\}$ being the indicator function. But, the average loss percentage we experience during the audio conversation is not $\mathbb{E}[1\{d_k^i > D_k\}]$, but rather $\mathbb{E}[1\{d_k^i > \hat{D}_k\}]$. We explain next why this experienced average loss percentage can be different from p, when the relationship between loss and delay is non-linear.

Let $F(x)$ be the complementary CDF of packet end-to-end delay, i.e., $F(x) = \mathbb{P}\{d_k^i > x\}$. It is easy to see that for $x = \mathbb{E}[D_k]$, $F(x)$ is equal to p, since $\mathbb{P}\{d_k^i > D_k\} = \mathbb{E}[1\{d_k^i > D_k\}]$ is equal to p by definition.

The average packet loss percentage we obtain with our scheme is equal to $\hat{p} = \mathbb{E}[1\{d_k^i > \hat{D}_k\}]$. We condition on the value of D_k. This leads to

$$\begin{aligned} \hat{p} &= \mathbb{E}\left[1\{d_k^i > \hat{D}_k\}\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[1\{d_k^i > \hat{D}_k\}\right] \mid D_k\right] \\ &= \mathbb{E}\left[F(\mathbb{E}[D_k] + \hat{D}_k - D_k)\right]. \end{aligned}$$

The last equality results from the fact that

$$\begin{aligned} \mathbb{E}\left[1\{d_k^i > D_k + y\}\right] &= \mathbb{E}\left[\mathbb{E}\left[1\{d_k^i > D_k + y\} \mid D_k\right]\right] \\ &= F(\mathbb{E}[D_k] + y). \end{aligned} \tag{4.11}$$

The proof of (4.11) is as follows. The objective is to compute the loss probability of a packet when the playout delay in a talkspurt is set y units far from the optimal playout delay D_k. In other words, we want to compute the loss probability of a packet, when the playout delay is set y units far from the playout delay that results in a packet loss percentage equal to p. But, the playout delay that results in a packet loss percentage equal to p (if we only look at the packet and not at the talkspurt to which the packet belongs) is simply $F(\mathbb{E}[D_k])$. Hence, the problem is equivalent to computing the loss probability of a random packet when the playout delay for this packet is set y units far from $\mathbb{E}[D_k]$, which is equal to $F(\mathbb{E}[D_k] + y)$.

Let ϵ_k be the prediction error for talkspurt k, i.e., $\epsilon_k = \hat{D}_k - D_k$. We write $\hat{p} = \mathbb{E}[F(\mathbb{E}[D_k] + \epsilon_k)]$. The bias of our predictor can be seen from this expression, when the function $F(x)$ in non-linear. $F(x)$ relates the packet loss probability of a packet to the playout delay. For example, if $F(x)$ is a convex function, we have by Jensen's inequality $\hat{p} > F(\mathbb{E}[D_k] + \mathbb{E}[\epsilon_k]) = F(\mathbb{E}[D_k]) = p$.

To correct this bias, some transformation of the process D_k can be used. Define $X_k = G(D_k)$. The prediction will be done on the process X_k instead of D_k, using a Moving Average predictor, i.e., $\hat{X}_{k+1} = \sum_{l=1}^{M} a_l X_{k-l+1}$. Once the estimate of X_k, denoted by $\hat{X}_k$ is obtained, we set the playout delay to $G^{-1}(\hat{X}_k)$. The average loss percentage becomes equal to

$$\hat{p} = \mathbb{E}\left[F(\mathbb{E}[D_k] + G^{-1}(\hat{X}_k) - G^{-1}(X_k))\right].$$

The function $G(x)$ must compensate for the non-linearity of the function $F(x)$. It must transform the error in setting the playout delay, so as to make $\hat{p}$ equal to p. Unfortunately, it is very difficult to find the expression of $G(x)$. Some approximations can be used. We give an example of a transformation that we use in this chapter. Our measurements show that the function $F(x)$ is convex, and close to exponential. We consider then as transformation the exponential function, with a decay coefficient α, that is, we take $G(x) = e^{-\alpha x}$. Hence, we predict $X_k = e^{-\alpha D_k}$ instead of predicting D_k.

4.6.1 Performance comparison

We apply the exponential transformation $G(x)$ to the process $\hat{D}_k$ in our MA algorithm. The MA algorithm remains the same, but we predict now the process $X_k = e^{-\alpha D_k}$ rather than directly predicting D_k. We call this new algorithm *transformed MA* algorithm.

When testing the transformed MA algorithm we found no significant differences for $10 < \alpha \leq 20$. For $\alpha < 10$ the performance degrades very slowly with decreasing α. We

thus set empirically the value of α to 10, and we use it when comparing with algorithms A and B in the next section.

To further improve the performance of the transformed MA algorithm, when transforming back $\hat{D}_k$ from $\hat{X}_k$, we implement the procedure described in Sec. 4.5.3 to reduce the deviations for small values of p. We call this variant *transformed MA+offset*.

Figure 4.8 compares the performance of our two MA+offset algorithms with algorithms A and B. In the subsequent figures, the transformed MA+offset is denoted as `MA+transf+offset`. Observe how the transformation applied on D_k considerably improves the performance of the MA algorithm. For trace 1, the transformed MA+offset algorithm clearly outperforms algorithms A, B, and the original MA algorithm with a gain of up to 50% for the whole range of loss percentage and average playout delay. For trace 2, the transformed MA+offset algorithm does not reach loss percentages lower than 5%. This is still due to the high jitter and network loss present in trace 2 which does not provide the autocorrelation function with useful information about the process D_k. However, the transformed MA+offset algorithm largely outperforms algorithms A and B for other values of p. For traces 3 to 5, we see clearly the benefit of applying the transformation $G(x)$. The transformed MA+offset algorithm outperforms all the other algorithms with a maximum gain on average playout delay of up to 80% in trace 4 for low loss percentages, compared to algorithms A and B. We notice an interesting behavior of the transformed MA algorithm in trace 6. This is the only trace for which the MA+offset algorithm behaves better than the transformed MA+offset version. Like trace 2, trace 6 also has a high end-to-end delay and a high network loss percentage; besides, trace 6 is one of the shortest sessions (in number of talkspurts). Thus, the MA+offset algorithm should be preferred when there are long congestion periods in the network and very high jitter.

We conclude that a moving average scheme is an attractive approach for playout delay control. The algorithms studied till now are offline algorithms. Section 4.7 presents an online hybrid algorithm combining algorithm B and the transformed MA+offset algorithm which gives a very good performance for most of the scenarios.

4.7 A hybrid algorithm for playout delay

Moving average estimation has revealed to be an interesting approach for playout delay control. The transformed MA+offset algorithm described in the previous section gives in general better performance than any of the other algorithms we studied. This algorithm was run offline and the entire trace was used to compute the characteristics of $\{D_k\}$. We look now for an online version of the transformed MA+offset algorithm. During our simulations, the maximum model's order was never greater than 23. This means that we

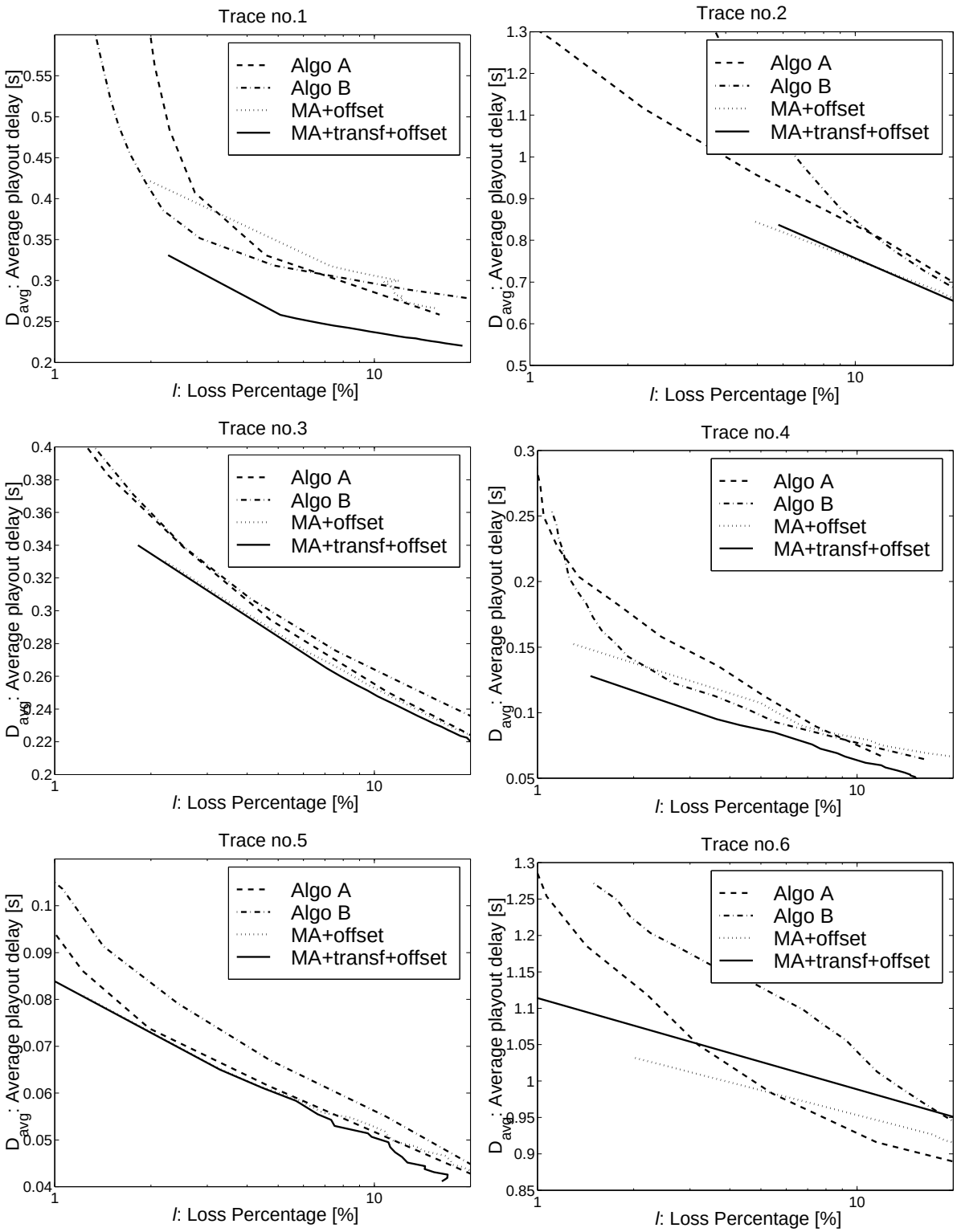


Figure 4.8: Performance comparison of algorithms A, B, and the transformed MA+offset algorithm.

do not need a large number of samples to compute a good moving average estimation. We propose in this section a combination of the transformed MA+offset algorithm and algorithm B, that we call *hybrid algorithm*.

The idea is quite simple and is sketched as pseudo-code in Figure 4.9. During the first talkspurts of an audio session, say `MAXTKSP` talkspurts, algorithm B is executed with $\beta = 4$. At the same time, we collect samples of $\{D_k\}$, we transform them to $\{X_k\}$, and we keep them in memory to be used later to compute the model's order and predictor coefficients. Then, starting from talkspurt `MAXTKSP+1`, the transformed MA+offset algorithm is executed and playout times are computed. The autocorrelation function is updated at each new talkspurt to account for the `MAXTKSP` most recent values of X_k. Since finding the model's order, M, is an exhaustive operation its value is computed only once and it is kept during the whole session. `MAXTKSP` is set equal to 100 for all the traces. The transformation applied to D_k is $X_k = e^{-\alpha D_k}$ and is denoted as $G(D_k)$.

4.7.1 Performance comparison

Figure 4.10 compares the performance of the hybrid algorithm with algorithms A and B. In trace 1, the hybrid algorithm outperforms algorithms A and B for almost all values of p. We observe an overall gain on playout delay of about 25% of the hybrid algorithm compared to algorithms A and B. We note again in trace 2 how the the hybrid algorithm does not reach loss percentages lower than 5%. In fact, since the number of D_k samples used to compute the autocorrelation functions is now small, the error introduced in $\hat{X}_k$, and consequently in $\hat{D}_k$, is larger in the hybrid algorithm than in the offline one. For traces 3 to 5, the performance of the hybrid algorithm and algorithms A and B is very similar, with the hybrid algorithm performing better than algorithm A for trace 4, and better than algorithm B for trace 5 in the loss range of interest ($p \leq 0.05$). Trace 6 has the highest session end-to-end delay and high network loss percentages (due to congestion), leading to a behavior similar to that of trace 2.

4.7.2 Delay spikes

Algorithm B detects delay spikes; when a delay spike occurs, the algorithm switches to spike mode and follows the spike. When the end of the spike is detected, the algorithm switches back to normal mode. We executed the MA+offset algorithm employing the spike detection approach of Algorithm B. When comparing the performance with no spike detection we found no significant differences. The MA+offset algorithm computes the autocorrelation function of the process $\{D_k\}$ to solve the system of normal equations and to calculate $\{\hat{D}_{k+1}\}$. D_k is the optimal per-talkpsurt playout delay. When delay spikes

```
1.  During the first MAXTKSP talkspurts {
2.      Execute Algo B;
3.      Compute playout times p_k^i;
4.      Collect statistics about D_k;
5.  }
6.  X_k ← G(D_k);
7.  R ← autocorr(X_k);
8.  M = findorder();
9.  /* For each talkspurt from k = 1 to MAXTKSP */
10.  X̂_k = Σ_{l=1}^{M} a_l X_{k-l+1};
11.  D̂_k = G^{-1}(X̂_k);
12.
13.  for k =MAXTKSP to N − 1 {
14.      X̂_{k+1} = Σ_{l=1}^{M} a_l X_{k-l+1};
15.      D̂_{k+1} = G^{-1}(X̂_{k+1});
16.      if p ≤ 0.02
17.          D̂_{k+1} ← D̂_{k+1} + (0.5 − 25p)√(E[(D_k − D̂_k)^2]);
18.
19.      p_{k+1}^1 = t_{k+1}^1 + D̂_{k+1};
20.      for j = 2 to N_{k+1}
21.          p_{k+1}^j = p_{k+1}^1 + t_{k+1}^j − t_{k+1}^1;
22.
23.      /* We recompute the autocorrelation function */
24.      /* for the MAXTKSP most recent values of X_k */
25.      R ← autocorr(X_k);
26.  }
```

Figure 4.9: The hybrid online algorithm for playout delay.

occur, the autocorrelation functions of $\{D_k\}$ account for them by definition.

4.8 Conclusions

In this chapter we describe a moving average algorithm that adaptively adjusts the playout delay at the beginning of talkspurts. To evaluate the performance of our algorithm, we compare it with existing schemes implemented in the NeVoT audio tool. Several variants of our moving average algorithm are studied. For small values of p, there is some deviation of the perceived loss percentage, and this deviation increases as p decreases. The MA+offset and the transformed MA+offset algorithms are proposed to reduce the deviation of the desired loss percentage. These variants allow to obtain a considerable gain compared to

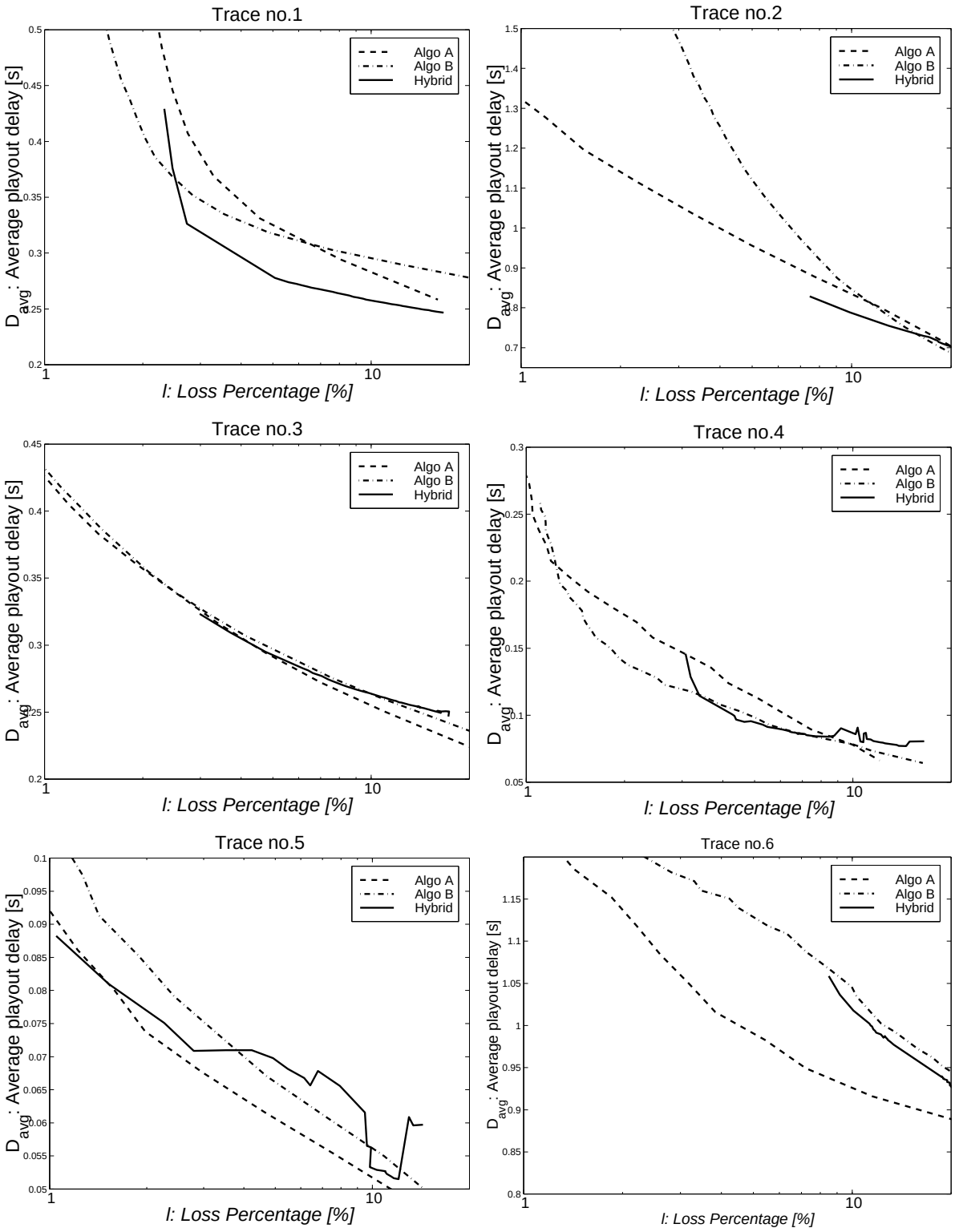


Figure 4.10: Performance comparison of algorithms A, B, and the hybrid online algorithm.

the original version while, at the same time, keeping the average playout delay within tolerable levels.

The strength of our scheme lies in the fact that we are able to tune the loss percentage p to a given desired value. When directly predicting the optimal playout delay, the desired loss percentage deviates from the desired one because the relation between the average playout delay and loss rate is not linear. We demonstrate that, by applying a transformation on D_k, the bias on the loss percentage can be reduced. Based on our measurements, we approximate this transformation by a negative exponential function. A mixture of algorithm B and our transformed MA+offset algorithm proves to be efficient in the loss percentages of interest. We call this algorithm *hybrid algorithm*.

Moving average estimation has revealed to be an efficient method for playout delay control. When network jitter and network loss are very high, as in traces 2 and 6, the MA algorithm do not reach loss percentages lower than 5%. Very high jitter decreases the correlation of the process D_k, leading to an inaccurate MA estimation.

Our algorithm predicts the optimal playout delay per-talkspurt, or a function of it, using the past history of the process. To reconstruct the periodic form of the stream of packets, the playout delay of packet in a talkspurt is based on the playout time of the first packet in the talkspurt. An interesting recent approach [55, 56] shows that it is possible to adapt the playout delay at each packet arrival, leading to a better performance than in a talkspurt basis. We are working on an extension of our MA approach that predicts the playout delay per-packet, allowing to change the playout delay during a talkspurt, and we expect our scheme to give better performance.

Part II

A model for AIMD protocols under variable delay

Chapter 5

TCP performance under variable delay

5.1 Introduction

In computing the throughput of long-lived AIMD connections (and of TCP connections in particular), existing analytical models do not take into account moments of the delay other than the first one. In a recent paper [24], it was observed however that the variability of this quantity impacts the throughput performance. In [24], a model is proposed which is partly analytical and partly empirical: it uses as parameters the probabilities of having a single, a double and a triple loss event, which should be inferred from the trace; it is through these parameters that the variability of delay is accounted for. Then in the rest of the derivation there, delays are replaced by their expectations.

Paths with high variability of delay are common in communication networks, and are typical to wireless networks [24]. Here are some examples of scenarios where the delays of packets can vary. First consider the High Data Rate (HDR) systems. The distribution of delays for these systems is described in [80]. HDR is a Qualcomm proposed CDMA air interface standard (3G1x-EVDO) for supporting high speed asymmetrical data. In HDR, the reason for variability of delay is the fact that the (link layer) packet to be transmitted is chosen dynamically among various connections, according to the channel that has the best state. Another source of delay variation on wireless links is ARQ (at the link level); ARQ can add considerable delay during retransmission, especially on geostationary satellite links where the propagation delay is large. The mobility of users in a mobile network is also an important source for delay variability. A situation that adds to variability in

the delays is when a TCP connection has lower priority with respect to other connections which have highly variable transmission rate. Such a situation occurs in UMTS where data packets are most frequently transmitted over shared channels (the FASH and RASH channels) in which higher priority is given to control packets. The delays of packets are also variable because of queuing time in routers. Generally speaking, the variability of delay is a common phenomenon in communication networks, and its consideration in the analytical models for AIMD protocols is important.

We propose in this chapter a model for the performance of a window-based AIMD mechanism in presence of variable delay. The model is based on stochastic difference equations. We provide closed-form expressions for the moments of the window size in steady state, as well as for the throughput of the mechanism. The model is validated with simulations using the TCP protocol, which has features of an AIMD policy in its steady state [42].

One of the key results of we obtain is that the performance we obtain when considering the variability of delay is better than the one we obtain when we assume the delay to be constant and thus replacing the delay by its average. This result is very important since it means that actual models for TCP, which only consider the mean delay, underestimate the performance of the protocol in environments where the packet delay is variable. Underestimating the performance of AIMD protocols may have a serious impact on the dimensioning of networks, and on the development of TCP-friendly multimedia applications.

In the next section we present our model, then we derive the expectation of its stationary window size in Section 5.3. The analysis of the throughput and of its dependence on delay variability follow in Sections 5.4 and 5.5. Section 5.6 validates our analytical results using ns-2 [67] simulations, and Section 5.7 ends this chapter with some concluding remarks.

5.2 The model

We consider systems for which the loss model at the packet level can be described by a Poisson process independent of the window size with a stochastic intensity. The average rate of loss events at the nth round-trip time is λ_n. A loss event in our case corresponds to the loss of a packet.

Our model studies a general window-based fluid AIMD mechanism. It applies to the TCP protocol when the window size is large enough so that the packet nature of TCP is diluted. The specification of our model to TCP will be described throughout the text. The TCP protocol will be used at the end of this chapter for the validation

of our results. Recall that a TCP connection in its steady state, and in the absence of timeouts and limitation on the throughput caused by the receiver window, can be seen as an AIMD protocol, where the congestion window increases by a constant factor every round-trip time, and where it is divided multiplicatively in presence of loss events [2, 42]. In particular, a TCP connection where the receiver acknowledges every packet, increases its congestion window by one packet every round-trip time, and divides its window by two when a packet loss is detected. The Reno version of TCP divides its window by two for every packet loss [29]. The Newreno and SACK versions divide their windows at most once by two in a round-trip time, regardless of the number of packet losses during the round-trip time [29]. We model both types of TCP behavior, while giving a particular attention to new versions as Newreno and SACK.

5.2.1 Stochastic recursive equations

We model the variable delay/rate path as follows. We consider some sequence of instants T_n and define the nth interval as $[T_n, T_{n+1})$. Let $R_n = T_{n+1} - T_n$ be its duration. R_n models the sequence of round-trip times seen by the AIMD control mechanism. We consider the window size W_n at the "end" of the nth interval. W_n is the window size just before instant T_{n+1}. (R_n, λ_n) is some stationary ergodic sequence of random variables. The window is a real number and is measured without loss of generality in terms of packets.

We consider some additive constant $\beta > 0$, and we assume that in the absence of loss, it takes R_n time to the AIMD protocol to increase the window size by β, i.e.,

$$W_{n+1} = W_n + \beta.$$

For example, on a long-lived TCP connection operating in congestion avoidance without the delayed ACK feature, we could take $\beta = 1$, in which case R_n would correspond to a round-trip time (as the window size of TCP increases by roughly one packet every round-trip time). Even though we frequently consider R_n as being the round-trip time, our model does not require that, and other definitions of R_n are possible. Another definition of R_n might be useful for congestion control mechanisms other than TCP.

We study now the dynamics of the window size when losses occur. Define $Z_n = 1$ if at least one loss occurred during $[T_n, T_{n+1})$ and $Z_n = 0$ otherwise, and set $\overline{Z}_n = 1 - Z_n$. Note that

$$P(Z_n = 0) = \mathbb{E}\left[\exp(-\lambda_n R_n)\right] := \mathcal{R}^*(\lambda),$$

λ is some parametric vector describing the intensity of the loss process and to be introduced later.

Consider now that at least one loss occurs during $[T_n, T_{n+1})$. Then the window size at the end of the $(n+1)$th interval is

$$W_{n+1} = \gamma_n W_n + \beta,$$

where γ_n is a random multiplicative factor that allows to account for cases where the multiplicative decrease of the AIMD mechanism is a function of the number of loss events that occur during the nth interval. Later, we will detail on this issue, and provide the expression for γ_n in the case of TCP.

Our dynamics can be interpreted as a simplified model for AIMD in which the multiplicative decrease occurs at the end of the round-trip time. Time interval R_n ends with a window size equal to W_n, and time interval R_{n+1} starts by a window size equal to $\gamma_n W_n$, to end with a window size equal to W_{n+1}. By dividing the window at the end of the round-trip time, we better model the fact that losses are not detected instantaneously. As for the window growth by β, we suppose that is done progressively during the round-trip time

Using the above two expressions for the case of no loss and for the case of at least one loss, we obtain the following dynamics:

$$W_{n+1} = A_n W_n + \beta, \tag{5.1}$$

where

$$\begin{aligned} A_n &= \overline{Z}_n + \gamma_n Z_n = 1 + Z_n(\gamma_n - 1) \\ &= \gamma_n + \overline{Z}_n(1 - \gamma_n). \end{aligned} \tag{5.2}$$

(5.1) is known as a stochastic recursive equation, see [2,9,17,21,35]. The loss process is assumed to be Poisson with an intensity dependent on the current round-trip time, and the process (R_n, λ_n) is assumed to be stationary ergodic. Moreover, for any integer n, the loss process after time T_n does not depend on $W_j, j \leq n$. Then, A_n is stationary ergodic.

Similar recursive equations have been used in the past to analyze the throughput of TCP and of TCP-friendly applications, see e.g. [3,112]. The special feature of our present model is that A_n is random and depends on R_n. Another feature is that the increase in the window size between instants $\{T_n\}$ is constant, and is independent of the duration of round-trip times.

Our model can be easily specified to TCP. The process R_n can be seen as the round-trip time of the TCP connection. A loss event corresponds to the loss of a TCP packet. The additive increase constant β is roughly equal to one packet. The multiplicative decrease

constant γ_n depends on the TCP version. Some versions as Reno divide their windows by two for every packet loss; in this case $\gamma_n = (1/2)^{N_n}$, where N_n is the number of loss events in the nth round-trip time. Other versions of TCP as Newreno and SACK divide their windows by two whether there is one or more loss events in a round-trip time. For these later versions, γ_n is constant equal to one half.

Our model does not account for some TCP mechanisms as the slow start phase, the timeouts, the receiver window, and so on. Our objective is not to provide an accurate model for TCP, but rather to illustrate the impact of delay variability on protocols implementing the AIMD mechanism. TCP is a typical example of such protocols. For accurate models of TCP that consider some of the non AIMD features (but that do not consider the variability of delay), we refer to [3, 69].

5.3 Stationary window size analysis

We begin by computing the stationary distribution and moments of the window size at times T_n. We also compute time average quantities. The throughput of the AIMD mechanism is then given in the next section for the case when the sequence $\{R_n\}$ models the round-trip times.

Concerning the process R_n, we consider two cases: $\{R_n\}$ are i.i.d. (independent and identically distributed), and $\{R_n\}$ are Markov correlated. In the i.i.d. case, we obtain closed-form expressions for the throughput and for the moments of the window size. In the Markov correlated case, we obtain linear equations that can be solved for the moments of the window size and for the throughput.

5.3.1 Window size at times T_n

Applying Theorem 2.A of [35] for which the conditions are easily checked, we get:

Theorem 4. *There exists a unique stationary ergodic process W_n^* that satisfies the same recursion (5.1) as W_n, and that is defined on the same probability space as (W_n, A_n). For any initial value W_0, $\lim_{n\to\infty} |W_n - W_n^*| \to 0$ and W_n converges to W_n^* in distribution. W_n^* has the explicit form:*

$$W_n^* = \sum_{j=0}^{\infty} \left(\prod_{l=n-j}^{n-1} A_l \right) \beta.$$

To simplify the exposition of the analysis, we consider hereafter that the system is in its stationary regime at time $t = 0$. First, we present the results for the case when the random variables $\{R_n\}$ are i.i.d. and $\lambda_n = \lambda$ are constant (do not depend on n). Then, we

explain how to compute the moments of W_0^* in the case the process $\{R_n, \lambda_n\}$ is Markov correlated. Through the analysis, we consider two values of γ_n:

A1.i: The AIMD protocol decreases the window size by a constant factor γ, independently of the number of losses during the round-trip time (provided there is at least one). This models the new versions of TCP as Newreno and SACK [29].

A1.ii: The AIMD protocol decreases its window by a constant factor γ for every loss, which gives $\gamma_n = \gamma^{N_n}$. N_n denotes the number of packets lost in round-trip time R_n. This can be assumed to model old versions of TCP as Reno [29]. In contrast to new versions, the Reno version of TCP can divide its window by more than two in a round-trip time depending on the number of packets lost and the location of these losses in the congestion window.

5.3.1.1 The i.i.d. case

Taking expectation in (5.1), we get in the stationary regime:

$$\begin{aligned} \mathbb{E}\left[W_0^*\right] &= \frac{\beta}{1-\mathbb{E}\left[A_0\right]} \\ &= \frac{\beta}{1-\mathbb{E}\left[\gamma_0 + \overline{Z}_0(1-\gamma_0)\right]}. \end{aligned} \tag{5.3}$$

This is the average window size sampled just before time T_1 (end of time interval R_0). We give in the following the expression of $\mathbb{E}\left[W_0^*\right]$ for the two particular values of γ_0 cited in A1.i and A1.ii. Under A1.i we have

$$\mathbb{E}\left[A_0\right] = \gamma + \mathcal{R}^*(\lambda)(1-\gamma).$$

Hence,

$$\mathbb{E}\left[W_0^*\right] = \frac{\beta}{1-\mathbb{E}\left[A_0\right]} = \frac{\beta}{1-\gamma} \times \frac{1}{1-\mathcal{R}^*(\lambda)}, \tag{5.4}$$

whereas under A1.ii we have, for $k = 0, 1, 2, ...,$

$$P(A_0 = \gamma^k) = P(N_0 = k) = E\left[\frac{(\lambda R_0)^k}{k!}\exp(-\lambda R_0)\right],$$

so that

$$\mathbb{E}\left[A_0\right] = E\left[\sum_{k=0}^{\infty} \frac{(\gamma\lambda R_0)^k \exp(-\lambda R_0)}{k!}\right] = \mathcal{R}^*(\lambda(1-\gamma)).$$

Thus

$$\mathbb{E}\left[W_0^*\right] = \frac{\beta}{1 - \mathcal{R}^*(\lambda(1-\gamma))}.$$

5.3.1.2 The correlated case

We model here the correlation that may exist among $\{(R_n, \lambda_n)\}$. We explain how to compute the moments of the window size in presence of such correlation.

Consider an N-state ergodic Markov chain $\zeta(n)$ embedded at times T_n, with transition probabilities P_{ij} and with steady state probabilities π_j, $j = 1, ..., N$. Assume that the distribution of the couple (R_n, λ_n) is only a function of the state of the Markov chain at time T_n and not of the previous history. Hence, given the state of the Markov chain at times T_n and T_m, $m > n$, the coefficients A_n and A_m are independent. We denote by $(R(i), \lambda(i))$ the values of this couple when the Markov chain is at state i.

Our goal is to compute $\mathbb{E}\left[W_0^*\right]$. Later, this will serve to compute the throughput of the AIMD mechanism. Define $w_j = \mathbb{E}\left[W_0^* 1\{\zeta(0) = j\}\right]$, and define $a_j = \mathbb{E}\left[A_0 | \zeta(0) = j\right]$. By using the recurrence (5.1), we have

$$\begin{aligned} w_j &= \mathbb{E}\left[W_1^* 1\{\zeta(1) = j\}\right] \\ &= \sum_{i=1}^{N} \mathbb{E}\left[A_0 W_0^* 1\{\zeta(0) = i\} 1\{\zeta(1) = j\}\right] + \pi_j \beta \\ &= \sum_{i=1}^{N} a_i w_i P_{ij} + \pi_j \beta. \end{aligned}$$

We obtain a system of linear equations,

$$(I - \mathcal{A})\underline{w} = \mathcal{B} \tag{5.5}$$

where $\mathcal{A}_{ij} = a_i P_{ij}$, $\mathcal{B}_i = \pi_i \beta$ and $\underline{w} = \{w_i\}$. We may obviously assume that at least for some i, $a_i < 1$ (for either A1.i or A1.ii). Hence $\mathcal{A}$ is a strictly sub-stochastic matrix and its largest eigenvalue is strictly smaller than one. Hence, Equation (5.5) has a unique solution $\underline{w}$, and finally we obtain $\mathbb{E}\left[W_0^*\right] = \sum_{i=1}^{N} w_i$.

Denote by $\mathcal{R}_i^*(s)$ the LST of the round-trip time R_n given that the Markov chain is in state i, i.e., $\mathcal{R}_i^*(s) := \mathbb{E}\left[\exp(-sR_0) | \zeta(0) = i\right]$. Then under A1.i we have,

$$a_i = (\gamma + \mathcal{R}_i^*(\lambda(i))(1-\gamma)). \tag{5.6}$$

Under A1.ii we have,

$$a_i = \mathcal{R}_i^*(\lambda(i)(1-\gamma)). \tag{5.7}$$

5.3.2 Window size at random time

For the completeness of the study, we compute in this section the expectation in the stationary regime of the process $W(t)$ of the window size. The time average window size is equivalent to the average number of packets in the network at random time. Using Palm calculus, this is given by

$$\mathbb{E}[W(t)] = \frac{\mathbb{E}[S_1]}{\mathbb{E}[R_1]}, \text{ where } S_1 = \int_{T_1}^{T_2} W(s)ds.$$

To sum the window size between T_1 and T_2 (beginning and end of round-trip time R_1), we make the following assumption:

A2: The window size grows linearly during the interval $[T_n, T_{n+1})$ with rate β/R_n, and only at the end of the interval, the window size will decrease if there has been a loss during the interval.

Under A2 we have,

$$S_1 = R_1\left(A_0W_0 + \frac{\beta}{2}\right).$$

5.3.2.1 The *i.i.d.* case

We consider the case where $\{R_n\}$ are i.i.d. and λ is constant. Although there is a dependence between W_n and R_{n-1}, R_{n-2}, ..., there is no dependence between W_n and R_n, R_{n+1}, ... Thus,

$$\mathbb{E}[W(t)] = \frac{\beta}{2} + \mathbb{E}[A_0]\,\mathbb{E}[W_0^*],$$

where $\mathbb{E}[W_0^*]$ is given by (5.4), and

$$\mathbb{E}[A_0] = \begin{cases} \gamma + \mathcal{R}^*(\lambda)(1-\gamma), & \text{under A1.i.} \\ \mathcal{R}^*(\lambda(1-\gamma)), & \text{under A1.ii.} \end{cases}$$

5.3.2.2 The correlated case

We consider the same model for R_n as that in Section 5.3.1.2. Our problem is to compute $\mathbb{E}[R_1A_0W_0]$. We condition on the state of the Markov chain at T_0. This gives,

$$\mathbb{E}\left[R_1 A_0 W_0\right] = \sum_{i=1}^{N} w_i a_i \sum_{j=1}^{N} P_{ij} \mathbb{E}\left[R_1 | \zeta(1) = j\right].$$

The w_i can be obtained by solving the system of N linear equations in Section 5.3.1.2. The a_i are given in (5.6) and (5.7) for cases A1.i and A1.ii. We put everything together, which gives

$$\mathbb{E}\left[W(t)\right] = \frac{\beta}{2} + \frac{1}{\mathbb{E}\left[R_1\right]} \sum_{i=1}^{N} w_i a_i \sum_{j} P_{ij} \mathbb{E}\left[R_1 | \zeta(1) = j\right].$$

5.4 Throughput and square-root formula

We compute in this section the throughput of the AIMD mechanism. Consider in what follows the case of i.i.d. round-trip times and λ_n constant equal to λ. This will allow a nice closed-form relating the throughput of an AIMD mechanism like TCP to the packet loss ratio p, and that accounts for the variability of the delay, not simply its average value as in previous models. In the case of correlated round-trip times, we compute the throughput numerically.

A window-based flow control mechanism transmits a window size of packets in every round-trip time. If we look at our model, W_n packets are transmitted in the interval $[T_n, T_{n+1})$. The connection's throughput (in fact it is the sending rate) is simply equal to the average window size $\mathbb{E}\left[W_0^*\right]$ computed in Section 5.3.1 divided by the average round-trip time $\mathbb{E}\left[R_0\right]$. Denote by X the throughput of the connection (in packets per second). Therefore, $X = \mathbb{E}\left[W_0^*\right] / \mathbb{E}\left[R_0\right]$.

Consider in what follows the i.i.d. case, which will allow a nice closed-form expression of the throughput. The Markov correlated case is more complex and requires numerical analysis. We have in the i.i.d. case:

$$X = \frac{\mathbb{E}\left[W_0^*\right]}{\mathbb{E}\left[R_0\right]} = \frac{\beta}{\mathbb{E}\left[R_0\right]\left(1 - \mathbb{E}\left[A_0\right]\right)}. \tag{5.8}$$

It is clear from (5.8) that the throughput of an AIMD mechanism changes with the variability of the round-trip time. This change is caused by the term $\mathbb{E}\left[A_0\right]$ in the denominator of X. In the next section, we will study the relation between this term and the variability of the delay, and in consequence the relation between the variability of the delay and the throughput.

The following analysis holds for A1.i, which is the case when the AIMD mechanism divides its window independently of the number of packets lost in the round-trip time. Under A1.i, we have

$$X = \frac{\beta}{1-\gamma} \times \frac{1}{\mathbb{E}\left[R_0\right](1-\mathcal{R}^*(\lambda))}. \tag{5.9}$$

Let us establish the relation between the throughput and the packet loss ratio. Such relation is usually used while modeling the TCP protocol. It is well known in the networking community that the throughput of a long-lived TCP connection (at least in the steady phase where there are no timeouts, no slow-start phases, and no limitation caused by the receiver window) is inversely proportional to the square-root of the packet loss ratio, and to the average round-trip time [2, 69]. We show here what this relation becomes when delay is variable.

Let p denote the packet loss ratio. As before, λ denotes the average rate of loss events. The expressions of TCP throughput in the literature are derived in the case when a packet loss results in a division of the window size. p is not the packet loss probability, but rather the probability that a packet loss results in a division of the window size. Recall that we are working under A1.i. This requires to compute the average rate of loss events that result in a division of the window size. Denote this average rate by $\lambda' \leq \lambda$. This average rate is equal to,

$$\lambda' = \frac{P(Z_0 = 1)}{\mathbb{E}\left[R_0\right]} = \frac{1-\mathcal{R}^*(\lambda)}{\mathbb{E}\left[R_0\right]}.$$

Another expression of λ' is $\lambda' = pX$. By equating these two expressions, we obtain

$$1 - \mathcal{R}^*(\lambda) = \mathbb{E}\left[R_0\right] pX.$$

We substitute this expression in (5.9), which yields

$$X = \frac{1}{\mathbb{E}\left[R_0\right]} \sqrt{\frac{\beta}{(1-\gamma)p}}.$$

This relation is very interesting since it tells us that in an environment where the delay is variable, the throughput is always inversely proportional to the average round-trip time and to the square root of p. The impact of delay variability on the throughput figures in p. This probability represents how many times the window of the AIMD mechanism is divided. The average loss rate λ represents how many packets are lost. The mapping between packet losses and window divisions is dependent, not only on the average round-trip time, but also on its variability. This issue has been addressed in [24], and the

probability p has been computed empirically. An interesting result of our model is that it provides a closed-form expression for computing p, without doing measurements. Indeed, if we know the rate of loss events λ and the distribution of round-trip times, and if round-trip times are i.i.d., we can compute p as

$$p = \frac{\lambda'}{X} = \frac{1-\gamma}{\beta}(1-\mathcal{R}^*(\lambda))^2.$$

This expression of p only holds for A1.i, which models new versions of TCP (Newreno, SACK) that do not divide their windows more than once per round-trip time, regardless of the number of packets lost. It does not hold for A1.ii. Indeed, under A1.ii, every packet lost results in a division of the congestion window, and so $p = \lambda/X$. By substituting in (5.8), we get

$$X = \frac{\beta}{\mathbb{E}\left[R_0\right](1-\mathcal{R}^*(pX(1-\gamma)))}.$$

This is an implicit equation in X. The square root relation between X and p does not hold in this context.

The square root formula for TCP throughput we present in this section suggests one more thing. If the probability p with which a TCP packet causes a division of the congestion window is constant independent of the round-trip time, the throughput of new TCP versions as Newreno and SACK will depend on the round-trip time only through its average, and so the existing models in the literature hold in this case. The existing models do not hold in the other cases since the distribution of round-trip times is to be considered.

5.5 Dependence of throughput on delay variability

The analysis we present in this section is done under A1.i for i.i.d. round-trip times and $\lambda_n = \lambda$. It also holds under A1.ii for i.i.d. round-trip times. At the end of the section, we comment on the validity of the result for any stationary ergodic process of round-trip times.

We study here the impact of variability of delay on the expected window size and on the throughput. We consider the expectation of the window size just before times T_n. As we saw above, the throughput of the AIMD mechanism, X, is proportional to the expected window size, so studying one of the two quantities is equivalent to studying the other.

Using our above results, we conclude the following:

Theorem 5. *Consider two AIMD systems having the same loss process and the same average delay. Both systems are identical. Let R_n and $\overline{R}_n$ be their round-trip times and suppose them to be i.i.d. Denote $\mathcal{R}^*(\lambda) = \mathbb{E}\left[\exp(-\lambda R_n)\right]$ and $\overline{\mathcal{R}}^*(\lambda) = \mathbb{E}\left[\exp(-\lambda \overline{R}_n)\right]$. Assume that*

$$\mathcal{R}^*(\lambda) \geq \overline{\mathcal{R}}^*(\lambda). \tag{5.10}$$

Let W_n (resp. $\overline{W}_n$) be the window process corresponding to R_n (resp. $\overline{R}_n$). The throughputs of the two systems are X and $\overline{X}$. We have the following,

$$\mathbb{E}\left[W_0^*\right] \geq \mathbb{E}\left[\overline{W}_0^*\right], \qquad X \geq \overline{X}.$$

Proof: The proof easily follows from (5.4) and (5.9). □

We explain now the relation between (5.10) and the variability of round-trip times. A popular measure of variability of random variables is the convex increasing stochastic order. We say that the variable R_n is larger than the variable $\overline{R}_n$ in the convex increasing order (or more variable) if for any convex increasing function h, we have $\mathbb{E}\left[h(R_n)\right] \geq \mathbb{E}\left[h(\overline{R}_n)\right]$. We denote this by $R_n \geq_{\text{conv}} \overline{R}_n$. R_n is greater than $\overline{R}_n$ in the increasing convex order if and only if there exists a joint probability space such that $\overline{R}_n \leq \mathbb{E}\left[R_n | \overline{R}_n\right]$. For more details see [10, Chp.4 (2.3.2)].

Since the function $g(R) := \exp(-\lambda R)$ is convex in R, we then obviously have

Remark 6. *Either one of the following is a sufficient condition for (5.10):*
(i) $R_n \geq_{\text{conv}} \overline{R}_n$, or
(ii) Let $\overline{R}$ be a constant and let $\overline{R} = \overline{R}_n \leq \mathbb{E}\left[R_n\right]$.

The first property ensures that the variability of R_n is larger than that of $\overline{R}_n$. If it is the case, the condition (5.10) is satisfied and the throughput of the AIMD mechanism in presence of R_n is larger than the one in presence of $\overline{R}_n$. The second property, combined with Theorem 5, implies that if we replace delays by their expectations, the throughput of an AIMD mechanism decreases. Our main result is then: *the larger the variability of the delay, the better the throughput of AIMD mechanisms in general, and of TCP in particular.* A related result has been found in [3], but in another context. [3] shows that the larger the variability of times between loss events, the better is the throughput of TCP.

Consider yet the case of i.i.d. round-trip times. When the average time between packet losses is large compared to the average round-trip time, the i.i.d. case converges to

the constant round-trip time case. Indeed, in (5.9), the term $\mathcal{R}^*(\lambda)$ can then be approximated by the first two terms of the Taylor expansion, i.e. by $1 - \lambda\mathbb{E}[R_0]$. This leads to a throughput $X = \beta/((1-\gamma)\lambda\mathbb{E}^2[R_0])$, which equals what we obtain in the constant round-trip time case. The variability of the round-trip time in this case has almost no impact on the throughput, and the round-trip time can be safely substituted by its average. Thus, models assuming constant round-trip times hold in the case of i.i.d. round-trip times that are on average small compared to times between loss events. In all other cases where the delay varies on time scales of the order of the average time between packet losses, a model as the one we propose is required.

5.6 Numerical results

5.6.1 Simulated scenario

Instead of creating an artificial model for a variable delay, we preferred to simulate a realistic scenario that induces a large variability in the delay. We use the ns-2 simulator [67] to study the scenario depicted in Figure 5.1. Each simulation is run for 2000 simulation seconds. The TCP source starts at time $t = 0$ and the ON-OFF source starts at time $t = 3$ secs. A Poisson ON-OFF source is attached to node n_1, and a Newreno TCP source is attached to node n_2. The lengths of ON and OFF periods are exponentially distributed and are set to the same average value. During ON periods, packets are sent according to a Poisson process.

The link between nodes n_2 and R_1, which is called *lossy link*, drops packets according to a Poisson process. The packet loss rate in the lossy link is assumed to be fixed and equal to λ packet-losses/second. The SimpleIntServ queue at the input of node R_1 is a 2-level priority queue, where packets from the ON-OFF source get higher priority. The scenario is configured in a way such that there are no congestion losses in the network (this is done by setting a large buffer at the input to the bottleneck link). Our objective is to validate the model in presence of a loss process independent of the window size.

5.6.2 Results

The results we present in what follows consider only the case where $\lambda_n = \lambda$. Figure 5.3 compares the performance of the simulations for three analytic models that correspond to three different assumptions on the round-trip time (RTT) process:

– Only the **mean RTT** value is considered when computing TCP throughput. This means in particular that we ignore the variability and correlation of round-trip time samples. Since our simulations are run with the Newreno version of TCP, this

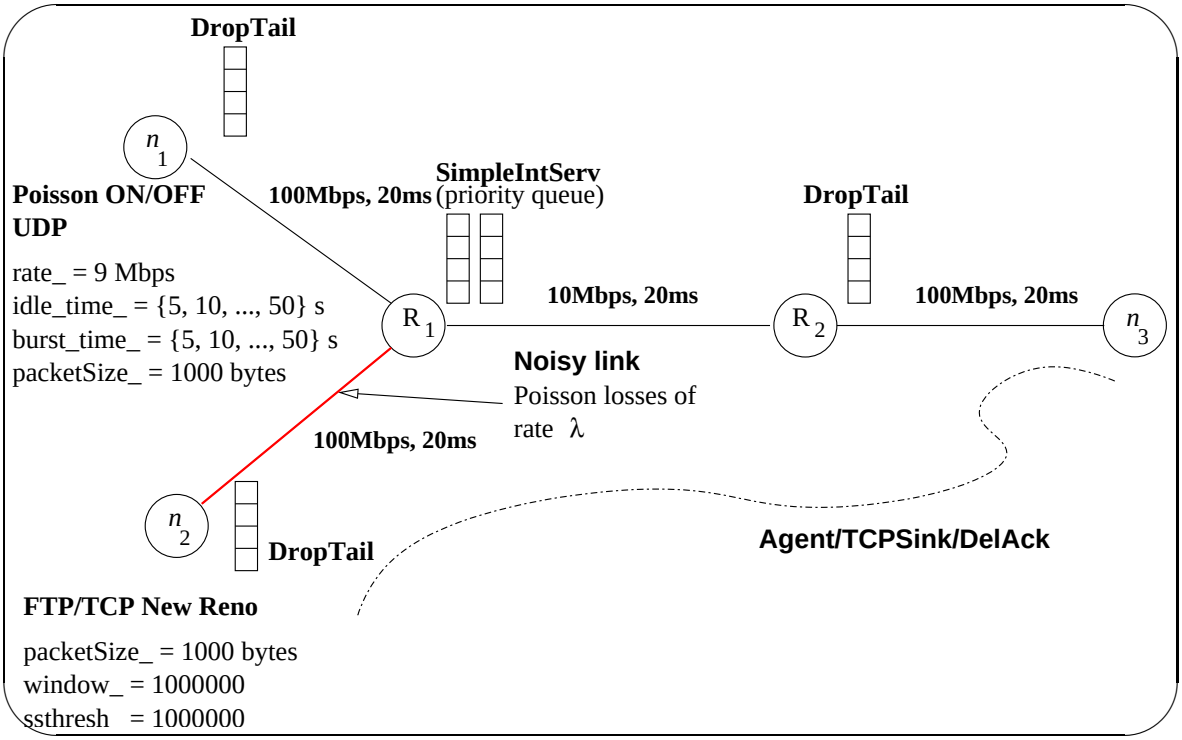


Figure 5.1: The simulated topology.

corresponds to case $A1 - A2.i$. So, throughput is computed using the following expression:

$$X = \frac{\beta}{1-\gamma} \times \frac{1}{\mathbb{E}\left[R_0\right]\left(1 - e^{-\lambda \mathbb{E}[R_0]}\right)},$$

with $\beta = 0.5$ (since the delayed ACK feature is used) and $\gamma = \frac{1}{2}$.

– The **whole RTT distribution** is considered when computing TCP's throughput, but the correlation is ignored. RTTs are thus considered to be i.i.d. Equation (5.9) is used for computing the throughput and the Laplace-Stieltjes transform in this equation is evaluated using all RTT measurements instead of just using the mean RTT.

– **The correlated case** or Markov case. The throughput is computed in the following way.

  – We first use a simple empirical approach to model RTT as being modulated by a two state (ON-OFF) Markov chain. This is done by associating large RTT values to an ON state and small values to an OFF state, and then computing

the empirical transition rates between the states of the modulating chain. More precisely, we sort in ascending order the RTT values, then choose a sample subset to be the OFF state (the lowest RTT values) and the rest of the samples are considered the ON state. The ON state is chosen from the point where the ordered RTT process increases very fast (generally, the ON state is taken above the first 85% samples of the ordered RTT process). Figure 5.2 shows a typical RTT trace and the corresponding ordered RTT process (right). So, in this case, the RTT process is considered to be in the ON state starting from about the sample number 9600 on the ordered RTT process.

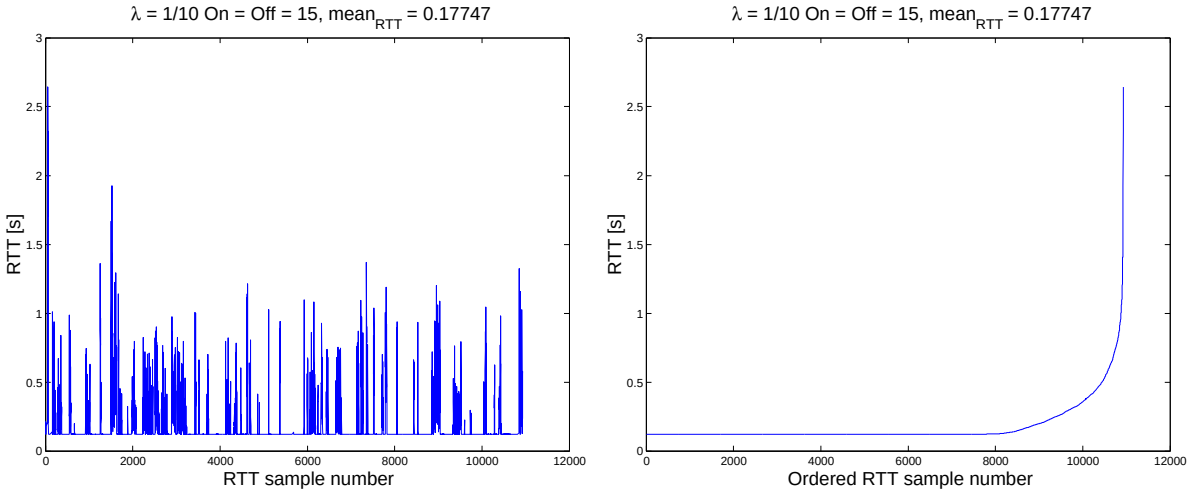


Figure 5.2: A typical RTT process and the ordered RTT process for deciding ON and OFF states.

- By analyzing the original RTT process we obtain the vector R_i, where $i \in \{\text{on, off}\}$ represents the state of the Markov chain underlying the RTT process.
- We get the transition probabilities of the Markov chain modulating the RTT process as follows:

$$p_{ij} = \frac{N_{ij}}{Ni} \quad \text{and} \quad p_{ii} = 1 - p_{ij}\,,$$

where $\{(i,j), i \neq j\} \in \{\text{on, off}\}$, N_{ij} is the number of transitions between on-off (resp. off-on) states, and N_i is the total number of RTT samples occurred during state i.

- The steady-state probabilities are computed as:

$$\pi_i = \frac{N_i}{N_{\text{RTT}}}\,,$$

Table 5.1: Confidence intervals for the Markov approach when $\lambda = \frac{1}{10}$.

ON-OFF [s]	$\mathbb{E}[X_{\text{ON-OFF}}]$	Confidence interval
5	2.3542×10^6	$\pm 0.0237 \times 10^6$
10	2.4471×10^6	$\pm 0.0201 \times 10^6$
15	2.4850×10^6	$\pm 0.0254 \times 10^6$
20	2.4318×10^6	$\pm 0.0257 \times 10^6$
25	2.6546×10^6	$\pm 0.0177 \times 10^6$
30	2.6344×10^6	$\pm 0.0185 \times 10^6$
35	2.3678×10^6	$\pm 0.0319 \times 10^6$
40	2.4649×10^6	$\pm 0.0270 \times 10^6$
45	2.4993×10^6	$\pm 0.0266 \times 10^6$
50	2.6316×10^6	$\pm 0.0125 \times 10^6$

where $i \in \{\text{on, off}\}$, N_i is the number of samples of the RTT process during state i, and N_{RTT} is the number of samples of the RTT process.

– Finally, we compute $\underline{w}$ as in (5.5), then we calculate the throughput in [bps] as:

$$X = \frac{8B \sum_i w_i}{E[R_0]},$$

where $B = 1000$ bytes is the size of TCP packets and, as above, $i \in \{\text{on, off}\}$.

To compute each point in the following figures, the same simulation is run M times with different seeds, the throughput is computed for each run, and it is finally averaged over M. We consider a value of $M = 50$ in our simulations. As an example, we show in Table 5.1 the 95% confidence intervals for the Markov approach for $M = 50$. Since confidence intervals are small enough, $M = 50$ is well justified.

Figure 5.3 shows the throughput results for $\lambda = \frac{1}{15}$ losses/s and $\lambda = \frac{1}{20}$ losses/s. Each point in the plot corresponds to the average throughput computed for the corresponding ON-OFF period lengths. The x-axis represents the average duration of ON and OFF periods. For all cases, the ON and OFF periods are set to the same average durations. The plot labelled as "Fixed" represents the case when the mean RTT value is only considered for throughput computation. The plot labelled as "Variable" represents the case when the whole RTT distribution is considered but not the correlation, and the plot labelled as "Markov" graphs the correlated case. For all cases, the Newreno version of TCP was considered.

The first thing we conclude from our simulation results is that considering delay variability (Variable and Markov cases) leads to a higher throughout than in the case when only the average delay is considered (the Fixed case). The second remark is that

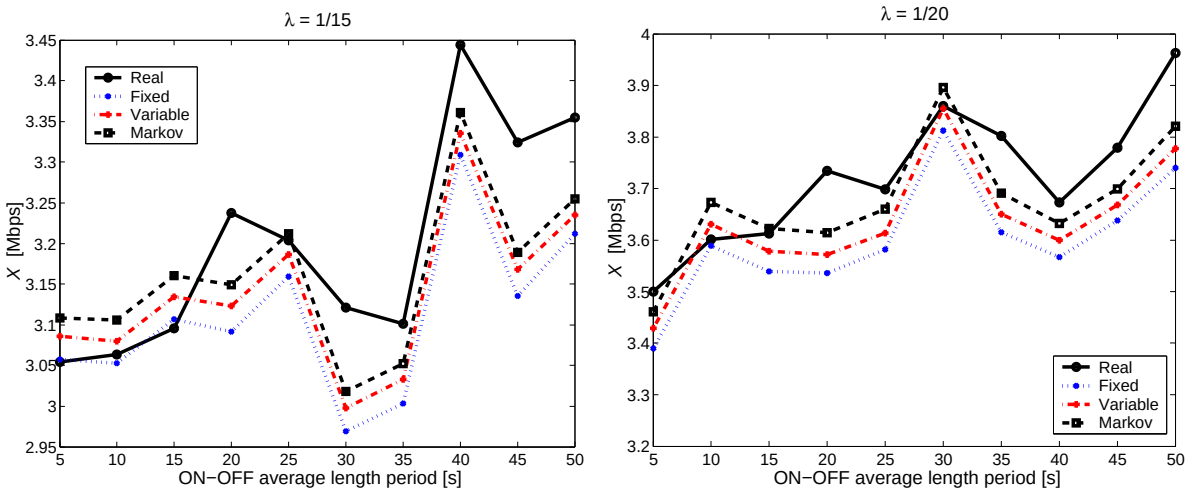


Figure 5.3: Throughput computed when considering delay variability.

the real TCP throughput computed from measurements is closer to the Variable and Markov cases than the Fixed case. This indicates that our model is able to provide a better approximation of TCP throughput than existing models which only consider the mean delay. Note how, starting from ON-OFF period lengths of 20 seconds, the Markov approach is the nearest to the real measured throughput.

5.7 Conclusions

We presented a model for an AIMD mechanism that accounts for delay variability. The model is based on stochastic difference equations. We solved this model for the moments of the window size in the stationary regime as well as for the throughput for the case of i.i.d. and Markov correlated round-trip times. We then studied the dependence of the throughput on delay variability. Our main analytical result was that the throughput of an AIMD mechanism increases when the delay becomes more variable.

In our analysis, we covered two AIMD versions. The first one divides its window in a lossy round-trip time by a constant which is independent of the number of losses. This mechanism models the new versions of TCP as Newreno and SACK. We also considered another AIMD mechanism that divides its window by a constant for each loss event. This latter mechanism models the Reno version of TCP.

We validated our analytical results with ns-2 simulations. We can summarize our conclusions from the numerical validation as follows:

– A model that only considers the mean RTT underestimates the total throughput of TCP.

– A model that accounts for the distribution of RTT but not the correlation (i.e. Equation (5.9)) is more accurate than the model that replaces RTT by its mean.

– The Markov model that takes into account also the correlation is the most accurate among the class of models we considered. We may expect further accuracy to be obtained by using a higher order Markov chain (with more than two states).

Chapter 6

Conclusions and perspectives

6.1 Conclusions

This thesis was organized in two parts. In the first part we have investigated the factors impacting the quality of interactive real-time multimedia applications. Though we focus on audio applications, our results are valid for any kind of real-time applications using the schemes we studied. In the second part we have proposed a model for the performance of AIMD-like protocols in presence of variable delay; the model is based on stochastic difference equations. We obtain a closed-form expression for the throughput illustrating the impact of delay variability.

In Chapter 2, we have investigated the performance of a simple FEC scheme implemented in audio tools like Freephone [34] and Rat [79]. Redundancy, which is compressed with a higher compression ratio than the original information, is carried ϕ packets apart from the original packet. We modeled the path between sender and receiver as a single bottleneck node in the network, and we supposed that the bottleneck router is dedicated only to the audio flow. Our assumption holds when, for example, all flows in the network implement the FEC scheme we studied. We used an $M/M/1/K$ queue to model the loss process of audio packets in the bottleneck router. The Poisson assumption of interarrival times is justified by the random delay added to packets by routers located upstream the bottleneck. Our main objective was to find a simple mathematical expression giving us some insights on the gain from using this FEC scheme on audio applications. The expression we obtained expresses the audio quality at the receiver as a function of the amount

of FEC redundancy and the offset between the redundancy and the original information. Our main result is that, even for the upper bound of audio quality, $\phi \to \infty$, adding FEC with this scheme always leads to a quality deterioration, which is mainly caused by an important increase in the network load.

In Chapter 3 we addressed the question of when and how this FEC scheme can lead to a quality improvement in audio applications. We considered the case of a single audio flow sharing the bottleneck with an exogenous traffic not implementing FEC. We also relaxed the linear utility function we derived in Chapter 2, and we considered three different non-linear utility functions. This assumption is based on previous observations for multimedia applications [96], where it is shown that this kind of applications are typically non-linear, being convex around zero and concave after a certain rate, with this rate depending on the type of multimedia application. For example, for the case of audio applications, the reconstruction of a packet from a copy with a compression ratio $\alpha < 1$ may give a quality close to that of the original packet. For the case of multiplexing an audio flow and other flows not implementing FEC, we modeled the other flows as a single exogenous flow of constant rate and with an exponentially distributed packet size. The arrival process at the bottleneck of the audio and exogenous flow was modeled as a Poisson process. We also assumed that audio packets arrive at the bottleneck according to a Poisson process. We modeled our system as an $M/G/1/K$ queuing system where service times are independent and identically distributed. We solved numerically our expressions of audio quality for different queue loads, and for different amounts of FEC. We showed that it is possible to get a gain with the FEC scheme we studied if the intensity of the audio flow is low compared to the exogenous flow not implementing FEC. When the intensity of the flow implementing FEC increases the gain in quality decreases, and it would dissapear when most of the flows start implementing FEC. With this results in mind we also conclude that a linear utility function is not an adequate assumption for assessing the quality of this kind of applications. We confirmed this result when we studied different non-linear utility functions at the end of Chapter 3.

We looked in Chapter 4 at algorithms for playout delay control. These mechanisms reduce the effects of jitter at the receiver by trading loss for delay. We identified in algorithms currently implemented in audio tools like NeVoT [91], the lack of a parameter allowing to control the loss rate in an audio session. Controlling this parameter is a key characteristic for any playout adaptation algorithm. Our main objective in this chapter of the thesis was to propose an algorithm for playout delay with tunable loss rate. Playout algorithms estimate the arrival time of future packets for computing their corresponding playout deadline. We proposed a set of Moving Average (MA) algorithms for playout delay with tunable loss rate: an offline MA algorithm, an offline MA algorithm with a

transformation of the optimum playout delay process, and an online hybrid algorithm. To prove the performance of our algorithms we used measurements of packet end-to-end delay of audio sessions done with NeVoT. We compare the performance of our approach with algorithms currently implemented in the NeVoT audio tool proposed by Ramjee et al. in [82]. We compared the performance of our algorithms with those of Ramjee by using a simulator that reads the trace files of end-to-end delay. Our algorithms gave a better performance than the Ramjee's algorithms for most of the cases in the loss range of interest for interactive audio applications, with the main strength of our scheme being the ability to control the loss percentage seen by an audio session.

Finally, in the second part of this thesis, we studied in Chapter 5 the performance of AIMD-like protocols under variable delay. Several accurate models exist for this kind of protocols; however, they ignore delay variability. Moreover, they compute the throughput by modeling the round-trip time as being constant, and thus replacing it by its expectation. We proposed, at the best of our knowledge, the first analytical model for AIMD protocols accounting for delay variability while computing the throughput. Our model is based on stochastic difference equations. We obtained closed form expressions for the throughput and the window size in steady state. We validated our model by analysis and simulation with the NS-2 simulator [67]. To this end, we modeled two versions of the TCP protocol: Newreno and SACK, and TCP Reno. We considered two different cases while modeling round-trip times (RTT): the case when RTT are independent and identically distributed random variables, and the case when RTT are Markov correlated. Our main result is that, when the packet loss process is constant and when RTT are i.i.d., the throughput of an AIMD mechanism increases with delay variability. This is a very important result since it means that current models for AIMD protocols, which only consider the first moment of round-trip delays, underestimate the performance when packet delay is variable. For the Markov case, we considered a simple two-state (on-off) Markov chain for modeling round-trip time. We observed that, for mean ON-OFF RTT periods from 25 seconds the Markov model is the most accurate than the other models we considered.

6.2 Perspectives

The set of MA algorithms we proposed adapt the playout delay at the beginning of each talkspurt during an audio session. Recent works on playout delay control algorithms propose mechanisms for adapting the playout delay in a packet fashion rather than in a talkspurt fashion. The advantage of these mechanisms is that they are more aggressive in deciding the playout deadline for a packet, reducing in this way the average time that packets stay in the playout buffer. Besides, in a recent paper [100], Sun and Ifeachor propose a

talkspurt-based adaptation algorithm for playout delay based on the predicted subjective quality given by a method similar to PESQ. We think that per-packet adaptation and buffer adaptation can be used in a highly dynamic way to achieve a better performance. So, our future work in this direction will focus on per-packet adaptation playout algorithms. We will evaluate different buffer adaptation methods based on subjective quality prediction, and we will study the feasibility of their implementation on audio tools.

Besides, there exists an increasing interest on adaptive schemes for audio over packet-switched networks. In [19], Boutremans and Le Boudec propose a scheme that allows an application to control its sending rate (and inherently the amount of redundancy it injects into the network) based on the network conditions. The sending rate is thus a function of the loss and delay processes of the network. The E-model is used to find functions relating the encoding rate with the R-factor, and the loss rate and the end-to-end delay with the utility seen by an application. The authors validate their scheme numerically and by simulation.

The scheme proposed by Boutremans uses traditional talkspurt-based playout algorithms. We think that a per-packet playout algorithm using subjective quality prediction can be used together with the model of Boutremans to give better performance. We will focus our future research on this direction.

Ballot theorem

In this appendix, we cite the Ballot theorem that we have used to solve the problem for case $1 \leq \phi \leq K_\alpha$. The reader is referred to [101] for details.

Theorem 7. *Suppose that an urn contains n cards marked with nonnegative integers $k_1, k_2, \ldots, k_n$, respectively, where $k_1 + k_2 + \cdots + k_n = k \leq n$. All the n cards are drawn without replacement from the urn. Denote by ν_r, $r = 1, 2, \ldots, n$, the number of the card drawn at the rth drawing. Then,*

$$P\{\nu_1 + \cdots + \nu_r < r \quad \textit{for} \quad r = 1, \ldots, n\} = 1 - \frac{k}{n}, \tag{1}$$

provided that all possible results are equally probable.

Part III

Présentation des Travaux de Thèse

Annexe

Introduction

Depuis sa création, l'Internet est en évolution permanente. Ceci est surtout dû à la simplicité de son architecture basée sur IP. Le protocole IP fournit un simple service " best-effort " basé sur la transmission de datagrammes, sans aucune garantie dans l'ordre de livraison desdits datagrammes, leur intégrité, ou l'arrivée de chaque datagramme à sa destination [77]. Cette simplicité permet d'interconnecter une grande diversité de réseaux et de former ce qu'on connaît comme l'Internet.

Pour palier à la simplicité et aux inconvénients du protocole IP, le protocole TCP (Transmission Control Protocol) fournit un service fiable de transmission des données à la couche d'application [78]. Les fonctions principales du protocole TCP sont : le contrôle de flux de bout en bout, le contrôle des erreurs, et le contrôle de congestion. Le contrôle de flux de bout en bout signifie qu'un émetteur ne peut injecter dans le réseau plus de données que ce que le récepteur peut accepter. La fonction de contrôle des erreurs consiste à récupérer chaque perte des paquets dans le réseau ; cette tâche est possible grâce à une procédure de retransmission basée sur des numéros de séquence, des acquittements positifs, et un temporisateur de retransmission. Le contrôle de congestion permet d'adapter le débit de transmission d'une source en fonction de la charge du réseau [42]. Le protocole TCP réalise ces fonctions de bout en bout et sans aucune modification au réseau ; c'est de cette façon que la transmission des données dans l'Internet est réalisée d'une façon fiable.

Le concept d'un réseau à intégration de services est d'un grand intérêt [46, 98]. D'un côté, un nouveau réseau vertébrale téléphonique fournirait plusieurs niveaux de qualité de service (QoS) pour transporter différents types de trafic. De l'autre côté, l'Internet actuel

serait utilisé pour transmettre du trafic multimédia, c'est-à-dire, de la voix, de la vidéo, et des données en même temps.

L'Internet a été conçu initialement pour transporter du trafic de données en utilisant le protocole TCP sur IP. La transmission de trafic temps-réel est un défi bien plus grand car ceci pose des contraintes très strictes de délai et de qualité de service. Utiliser TCP pour transmettre ce type de trafic n'est pas en général adéquat puisque celui-ci introduit de la variabilité du délai dû à la retransmission et au reordonnancement. De plus, les mécanismes de contrôle de congestion de TCP ajoutent de la variabilité dans le débit. Les applications temps-réel sont sensibles au délai et au débit ; en revanche, elles peuvent tolérer un certain taux de pertes et ne nécessitent pas une fiabilité à toute épreuve.

Plusieurs standards existent pour transmettre du trafic multimédia sur des réseaux à commutation de paquets. Le standard le plus répandu pour le trafic multimédia est H.323. Celui-ci utilise de l'encapsulation RTP pour l'audio sur UDP/IP [107]. Le protocole RTP (Real-time Transport Protocol) fournit des fonctions de transport de bout en bout pour les applications multimédias temps-réel, et supporte des transmissions multipoint et point à point. [92] Les protocoles RTCP (real-time transport control protocol) et RTP sont généralement utilisés ensemble pour fournir des statistiques à la couche d'application, telles que le taux de pertes, la gigue, etc. Ces statistiques sont d'utilité aux applications multimédias pour implementer des mécanismes équitables avec d'autres flux partageant le même lien [18,19,52,64]. Ainsi, il est possible de construire des applications implementant des mécanismes adaptatifs similaires à ceux de TCP. Ce type de mécanisme est connu sous le nom de " TCP-amical " (TCP friendly en anglais).

De nos jours, l'un des services temps-réel qui commence à être amplement déployé est celui de la transmission de voix sur l'Internet. Ceci est aussi connu sous le nom de Téléphonie sur Internet, ou tout simplement VoIP (Voice over IP en anglais) [39, 102]. Transmettre de la voix sur Internet présente plusieurs avantages. L'un des avantages le plus important est le coût de transmission. Le coût d'un appel sur un réseau à commutation de paquets comme l'Internet est bien plus bas que son équivalent sur un réseau téléphonique traditionnel. D'autres avantages que l'on peut citer sont le partage de fichiers et de tableaux, la possibilité de passer des appels cryptés, identification de l'émetteur, etc. Utiliser l'Internet pour transporter du trafic de voix présente aussi plusieurs avantages du point de vue du fournisseur de services. Par exemple, l'intégration de différents types de trafic serait bien plus simple qu'avec le réseau téléphonique traditionnel, la commutation serait moins coûteuse, et les plateformes de gestion seraient plus robustes.

Dans le modèle de base de la téléphonie sur Internet, les utilisateurs ont accès à un terminal multimédia (notament un ordinateur) connecté à l'Internet. Les utilisateurs peuvent y être connectés par des nombreux moyens : une connexion par modem classique,

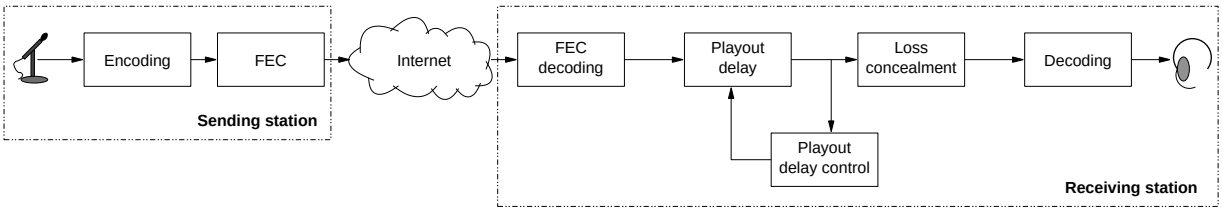


Figure 1: Scénario général d'une session audio.

une ligne ADSL, un réseau local (LAN), ou même un réseau sans fil. Si plus de deux utilisateurs participent à une conversation, une connexion multipoint est souvent établie.

La Figure 1 illustre les opérations générales lors d'une session audio. A l'émetteur, la voix est échantillonnée et numérisée, pour être alors transmise à l'autre bout. Alors, l'application réceptrice stocke les paquets reçus dans un tampon, pour les jouer ensuite après une certaine deadline. Pour réduire le taux de pertes au récepteur, les paquets transmis sont souvent encodés avec des codes correcteurs d'erreurs (FEC).

La Figure 2 montre le scénario de base de la téléphonie sur IP. Dans ce cas, une session audio est établie entre deux terminaux multimédia. Les opérations que l'on a cité précédemment sont réalisées par lesdits terminaux. Ce scénario de base a évolué en deux autres scénarios illustrés dans les Figures 3 et 4. Dans la Figure 3, un terminal multimédia est connecté à l'Internet et transmet des paquets de voix à un téléphone traditionnel. Le téléphone est connecté au commutateur centrale téléphonique, et celui-ci à son tour est connecté à l'Internet au moyen d'un gateway de téléphonie IP. Dans la Figure 4, deux téléphones sont utilisés pour faire un appel sur Internet. Dans ce cas, l'utilisateur qui place l'appel contacte le gateway de téléphonie IP, puis se connecte au système avec un identificateur d'appel, pour pouvoir alors composer le numéro du terminal qu'il souhaite appeller. Une session H.323 est alors initiée avec l'adresse IP du gateway du récepteur, qui est capable de contacter le téléphone du récepteur. C'est à ce moment que la session de voix commence et que les deux gateways échangent des paquets IP. Dans ce cas, les opérations de paquetisation, encodage, décodage, et de reordonnancement des paquets se

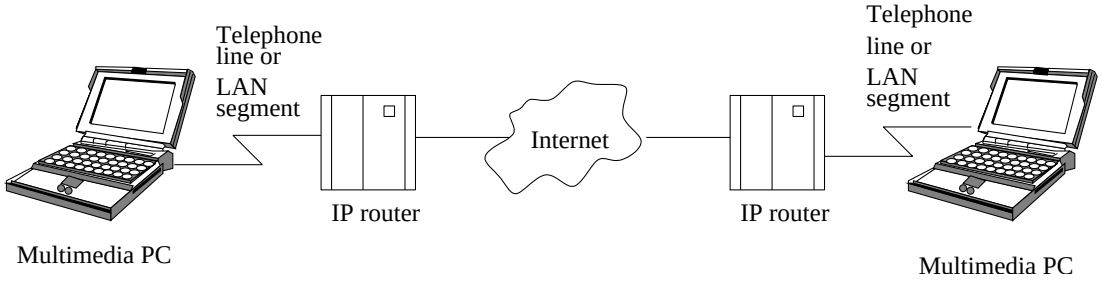


Figure 2: Scénario de base de la téléphonie sur IP.

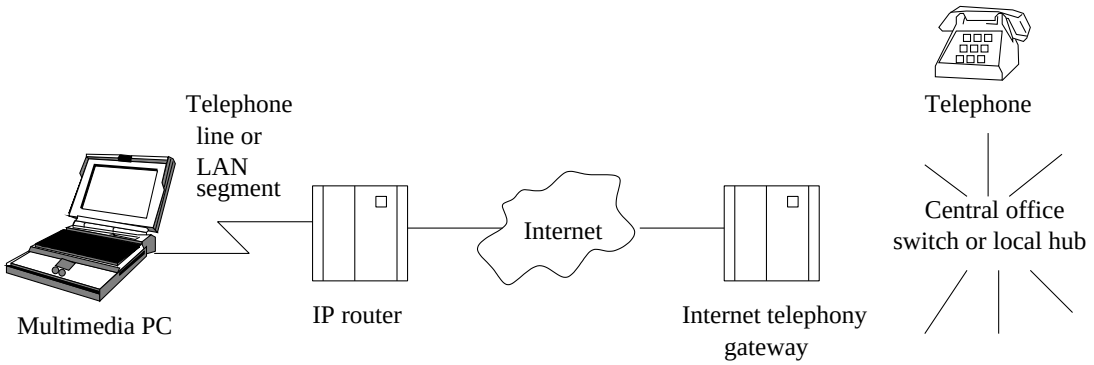


Figure 3: Architecture terminal multimédia–téléphone.

font dans les gateways.

Les architectures que nous venons de décrire imposent deux grands défis :

1. Le premier défi est celui de **l'implementation** d'un système VoIP. Un terminal VoIP demande beaucoup plus de ressources qu'un téléphone traditionnel puisqu'il doit supporter plusieurs codecs. Un autre problème associé à la transmission numérique de la voix est la annulation de l'écho. Lorsque l'architecture de VoIP comporte au moins un téléphone, la conversion de quatre à deux fils induit de l'écho dans la communication. Dans ce cas, l'écho est souvent réduit avec des mécanismes de filtres adaptatifs. Dans le cas où il y a au moins un ordinateur dans une session audio, l'absence d'écouteurs peut faire que l'un des participants entende sa propre voix. L'écho dans ce cas est facilement annulé en utilisant des dispositifs adaptés aux conversations multimédias [14].

   L'architecture d'un système VoIP pose deux autres problèmes : le support de la transmission en tones multifréquences (DTMF: Dual Tone Multifrequency), et la synchronisation des horloges. DTMF se réalise souvent par une indication d'entrée

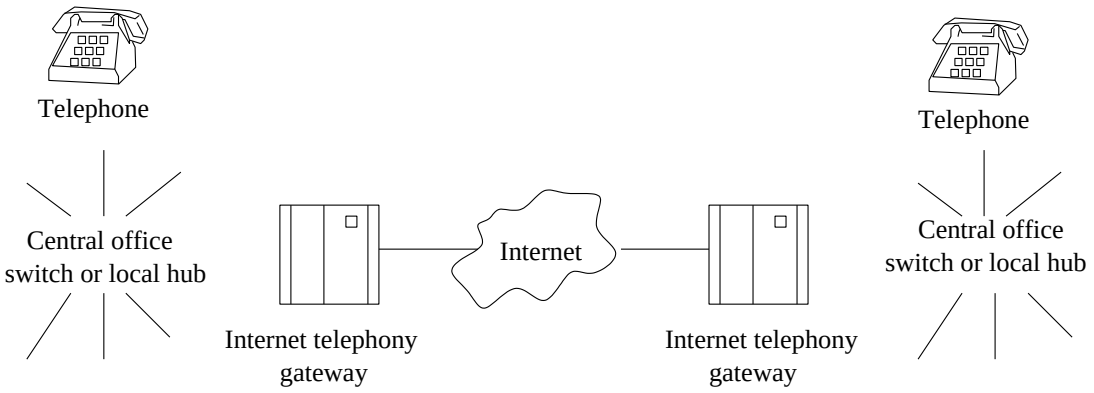


Figure 4: Architecture téléphone–téléphone.

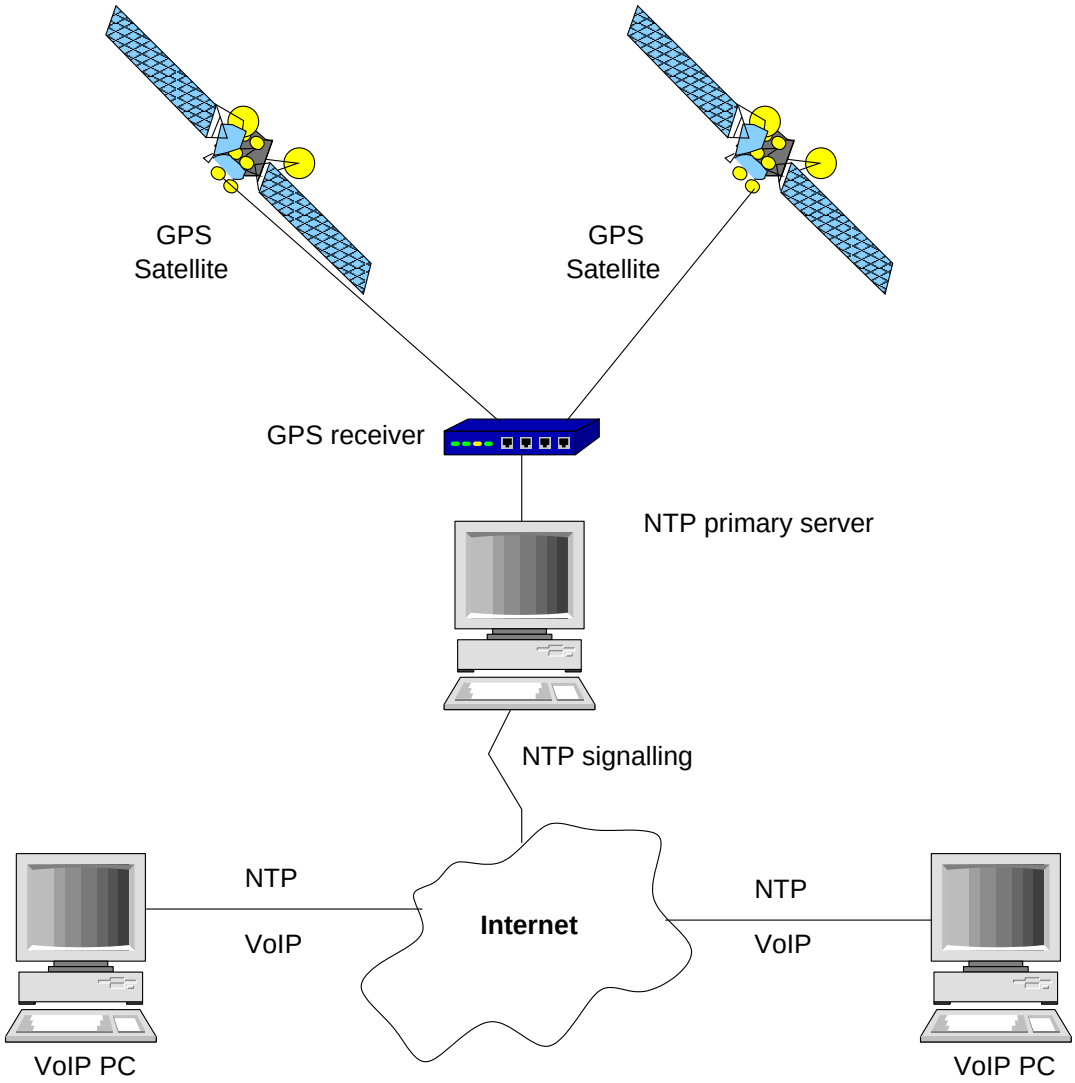


Figure 5: Synchronisation NTP.

(H.245) avec H.323, ou plus récemment avec SIP (Session Initiation Protocol) [93]. La synchronisation des horloges est une caractéristique souhaitable dans la voix sur IP puisqu'elle permet d'éliminer la déviation des horloges entre l'émetteur et le récepteur [41,62,66,70,116]. Le protocole NTP (Network Time Protocol) est généralement utilisé pour synchroniser les horloges dans les applications VoIP, il est souvent utilisé avec GSM (Global System for Mobile communications) pour avoir une plus grande précision, mais l'utilisation de ce dernier reste coûteuse. La Figure 5 montre la plateforme utilisée dans [62] pour synchroniser les horloges de deux terminaux en utilisant NTP. Un serveur de temps GPS communique le temps de base au serveur NTP (au milieu).

La synchronisation des horloges est bénéfique aussi pour les performances des algorithmes de contrôle de diffusion (playout delay en anglais). Nous étudierons ces algorithmes en détail dans le Chapitre 4.

2. Le second grand défi posé par la téléphonie sur Internet, et sans doute le plus difficile à résoudre, est de fournir de **la qualité de service (QoS)**. Puisque l'Internet est un réseau " best-effort ", transmettre de la voix numérisée sur celui-ci est un problème très complexe. Les trois facteurs qui affectent la qualité de la voix dans une session VoIP sont :

   Bande passante. Un utilisateur qui participe à une session VoIP doit avoir un minimum de bande passante pour pouvoir placer un appel avec une qualité acceptable. Un utilisateur qui se connecte à l'Internet avec un modem de 33kbps serait plus limité qu'un utilisateur qui s'y connecte avec une liaison ADSL, ou un réseau local (LAN).

   Codecs. Puisque la bande passante est une ressource variable, il est souhaitable que le partage de bande passante soit équitable entre les applications VoIP et d'autres applications. Alors, une application VoIP doit, de préférence, être " amicale " dans le sens de TCP (TCP friendly). Ceci signifie que le débit de transmission doit être adaptatif en fonction de la bande passante disponible dans le réseau. Une application VoIP peut varier son débit grâce à différents codeurs. Si à un instant donné, la bande passante réduit, une application VoIP change " à la volée " son débit de transmission en changeant sa transmission avec un codeur de plus bas débit [19, 61, 68].

   Délai. Deux types de délai jouent un rôle dans la transmission de voix sur Internet : le délai fixe et le délai variable. D'abord, la partie constante du délai est composée des opérations de quantification, paquetisation, transmission, et de décodage. Ce type de délai dépend du codeur et de la taille des paquets. Des implementations matérielles, telles que les téléphones IP, permettent de réduire cette partie constante du délai. La partie variable du délai est composée du délai de bout en bout et du délai de diffusion dans le récepteur [82]. Le délai de bout en bout est une fonction de l'état du réseau. Quand le réseau est encombré, la variabilité du délai de bout en bout augmente. Cette variabilité est connue sous le nom de " gigue " (jitter en anglais). La Figure 6 illustre les différents types de délai dans une session VoIP.

   Pour réduire la gigue, des mécanismes de contrôle du délai de diffusion sont utilisés [1,28,32,54,65,83,97,103]. Le délai de diffusion est le temps d'attente des paquets dans le tampon du récepteur. La fonction principale d'un algorithme

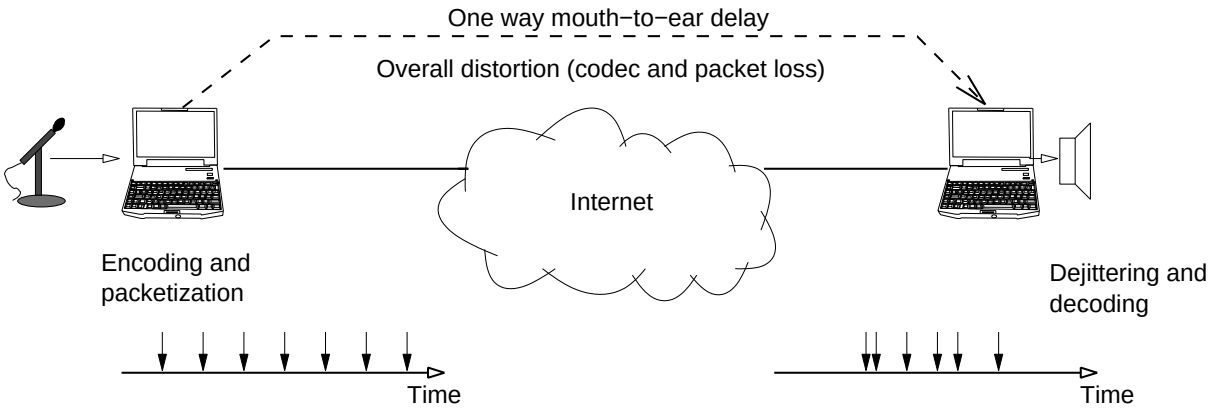


Figure 6: Différentes composantes du délai dans une session VoIP.

de contrôle du délai de diffusion est de faire un compromis entre le taux de pertes tardives et le délai, tout en réduisant la gigue. La Figure 7 montre la performance des algorithmes proposés dans [82]. L'axe des y représente le temps écoulé entre l'arrivée des paquets, et l'axe des x représente les numéros de séquence au cours de la session audio. Les paquets avec une gigue supérieure à la ligne de deadline sont considérés comme perdus et ne sont pas joués. Ceci augmente le taux de pertes mais améliore l'interactivité de la session. Dans le Chapitre 4 nous étudions en détail ce type d'algorithmes, et nous proposons une nouvelle approche permettant de contrôler le taux de pertes moyen vu par une application audio.

Pertes. Pour réduire les effets des pertes, des mécanismes proactifs et réactifs sont souvent utilisés par une application de VoIP. Le mécanisme le plus utilisé est une approche proactif de correction d'erreurs (FEC Forward Error Correction) [14, 39, 72]. Ces mécanismes transmettent d'abord les paquets originaux, et puis leur redondance dans des paquets séparés. Ainsi, si un paquet est perdu, il peut toujours être récupère et joué en attendant sa redondance. Un autre type de mécanisme FEC utilise l'encodage de parité [31]. Dans ce cas, les paquets sont rassemblés dans de groupes d'une taille donnée, et puis une opération XOR est réalisée sur chaque groupe. Le paquet résultant est transmis avec le groupe suivant et il est placé dans le premier paquet. Ce mécanisme produit un paquet deux fois plus grand que la taille des autres paquets dans son groupe. Il permet de récupérer une seule perte.

Les deux types de mécanismes FEC que nous venons de décrire permettent de réduire le taux moyen de pertes au dépit d'un délai additionnel. Par ailleurs, afin de réduire au maximum le taux de pertes, des mécanismes implementés dans le récepteur sont souvent utilisés. Quelques exemples de ces mécanismes

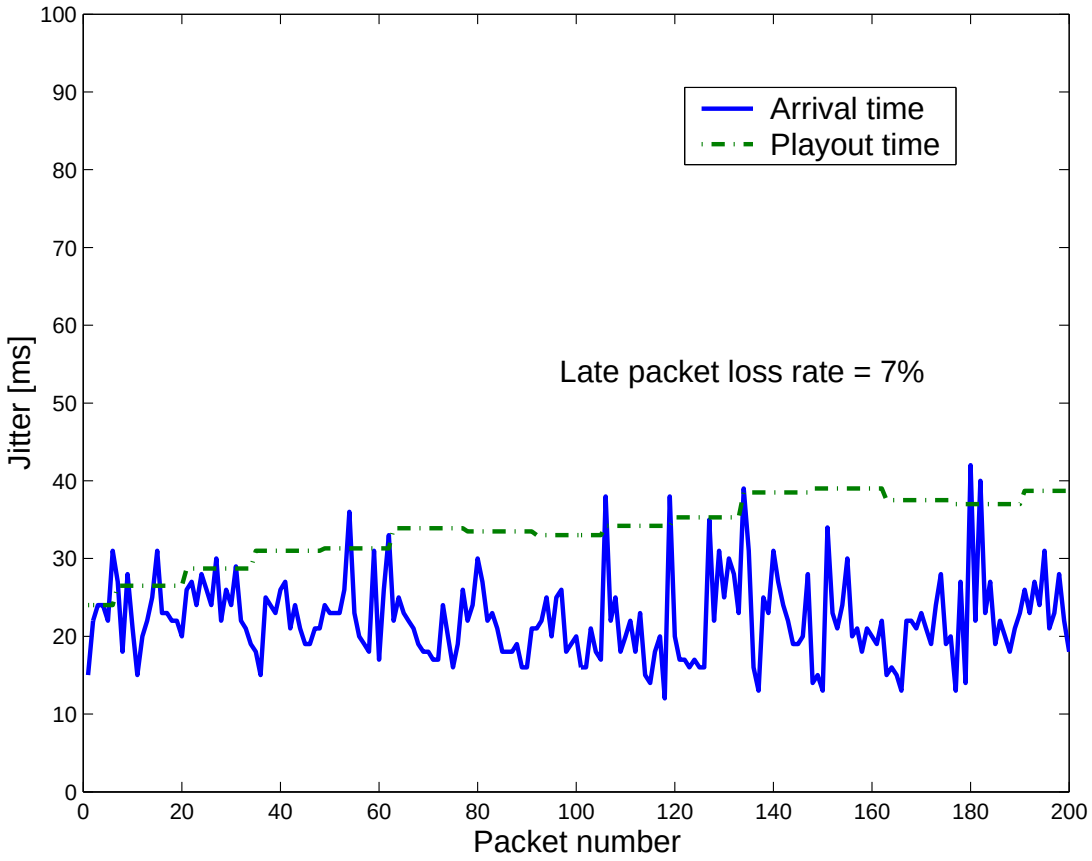


Figure 7: Opération de base d'un algorithme de contrôle du délai de diffusion.

sont l'insertion de silence, l'interpolation, et la régénération des paquets [38, 72, 88–90].

Dans le Chapitre 2, nous proposons un modèle de files d'attente pour analyser les performances d'un mécanisme FEC proactif, qui est implementé dans des applications audio telles que Freephone [34] ou Rat [79].

Le délai, les pertes, et la gigue sont inévitables dans l'Internet actuel. Les techniques que nous venons de décrire doivent être utilisées afin de réduire leur effet sur la qualité des applications VoIP, ou sur la qualité de n'importe quelle autre application multimédia utilisant ces mécanismes. Par ailleurs, la variabilité du processus de pertes et celle du délai rendent encore plus difficile la tâche de fournir une qualité de service acceptable par ce type d'application.

Au cours des dernières années, plusieurs efforts de recherche ont été réalisés sur chaque aspect des applications multimédias. On identifie ensuite quelques axes de recherche dans ce domaine :

Pertes et délai de bout en bout. Le processus de pertes et celui du délai sont souvent

corrélés. Si le paquet i est perdu, il est très probable que le paquet $i+1$ soit aussi perdu. Les pertes en rafale affectent la qualité des applications multimédias. La dépendance entre le processus de pertes et celui du délai a été largement étudiée, et plusieurs mécanismes ont été proposés pour réduire leurs effets dans ces applications [11, 15, 16, 37–39, 52, 63]. Dans [15], les auteurs supposent que l'impact du trafic Internet sur un flux de paquets émis à des intervalles réguliers peut être modélisé par un flux de Bernouilli en batch. Ils proposent un mécanisme de FEC, pour l'implementer dans ce qui a été le premier outil audio capable d'adapter son débit en fonction de l'état du réseau [34]. Plus récemment, Jiang et Schulzrinne montrent dans [47] que, lorsque les applications multimédias subissent des pertes en rafale, le processus de pertes final après le décodage de la FEC et après le contrôle du délai de diffusion est toujours affecté par les pertes en rafale. Le mécanisme FEC proposé dans [15] a été étudié par analyse et simulation, et il fait toujours l'objet d'une ample recherche.

Algorithmes de délai de diffusion. Les algorithmes de contrôle du délai de diffusion proposés par Ramjee et al. dans [82] sont la référence la plus citée dans cet axe de recherche. Ils proposent quatre algorithmes pour adapter le délai de diffusion au début de chaque phrase d'une session audio. D'ailleurs, ils identifient des pics dans le processus du délai de bout en bout, et ils proposent un algorithme particulier pour réduire les effets de ce phénomène. Depuis, d'autres algorithmes ont été proposés pour améliorer les performances des mécanismes de Ramjee [49, 65, 73]. L'idée générale est de mettre à jour régulièrement la distribution du délai de bout en bout, et d'adapter le délai de diffusion d'après un taux de pertes souhaité. Une classe différente d'algorithme a été proposée dans [55, 57–59, 113]. Ces algorithmes sont très dynamique car ils mettent à jour le délai de diffusion à l'arrivée de chaque paquet. Ainsi, les paquets peuvent être joués presque immédiatement après leur arrivée. Ceci est possible grâce à une technique de traitement de signaux appelée " modification de l'échelle du temps " [99, 108, 114].

Protocoles. Il existe plusieurs protocoles pour transmettre du trafic multimédia sur Internet. Des protocoles de transmission, des protocoles de recherche de gateway [85], des protocoles de signalisation [84], des protocoles qui collectent de statistiques de QoS [92], et des protocoles de synchronisation. Ils sont actuellement utilisés et largement déployés. Par exemple, les protocoles de recherche de gateway et les protocoles de signalisation sont actuellement des sujets de recherche très actifs. Si un terminal IP est utilisé pour appeler un téléphone traditionnel, le terminal IP doit connaître l'adresse d'un gateway approprié. Ceci est réalisé par un *protocole de recherche de*

gateway. Un mauvais choix du gateway peut entraîner une mauvaise qualité d'appel. Donc, ces protocoles sont censés trouver le gateway optimum pour placer des appels.

L'un des objectifs les plus importants de la la téléphonie sur Internet est l'interopérabilité avec le réseau téléphonique traditionnel. Pour ce faire, des protocoles de signalisation et d'interconnexion tels que H.323 et SIP ont été déployés. H.323 est utilisé depuis longtemps, mais SIP commence a gagner sa place parce qu'il est moins complexe que H.323, plus facile à apprendre, son temps de latence d'établissement de connexion est faible , et il est plus interopérable que H.323.

Évaluation de la QoS. Comme on a décrit précédemment, les facteurs qui affectent la qualité des applications multimédias sont les pertes, le délai, et la gigue. Il est crucial de comprendre le comportement et l'interdépendance de ces phénomènes pour construire des modèles permettant d'offrir la meilleure qualité de service possible.

Comment peut-on mesurer la qualité de service (QoS) d'une session multimédia ? Deux approches générales existent. Les applications multimédia essaient de réduire au maximum les pertes et la gigue. La première approche pour mesurer la qualité consiste alors à mesurer le taux de pertes moyen, ou le délai de diffusion moyen au cours d'une session multimédia. On dit dans ce cas, que la QoS est mesurée d'une façon **objective** car une quantité physique mesurable (le taux de pertes, ou le délai moyen) est reportée par l'application pour indiquer la qualité perçue au cours d'une session [36, 65]. La deuxième approche pour mesurer la qualité se focalise sur les utilisateurs des applications multimédias. Dans ce cas, la session est d'abord enregistrée. Il y a alors deux signaux : le signal transmis et le signal reçu. Le signal d'origine est connu comme le signal de référence. Alors, un groupe d'utilisateurs compare les deux signaux d'une façon *subjective* en écoutant d'abord le signal de référence et puis le signal reçu. Le signal reçu est noté dans l'intervalle $[1 \ldots 5]$ en fonction de sa qualité subjective. Une note de 5 est donnée pour indiquer la meilleure qualité (pas de différence entre le signal de référence et le signal reçu), 4 pour une haute qualité, 3 pour acceptable, et ainsi de suite. Une note égale à 1 est donnée quand la qualité est évaluée comme inacceptable. Cette façon de mesurer la qualité est connue sous le nom de Évaluation Moyenne des Opinions (MOS : Mean Opinion Score, en anglais) [23, 60, 86, 95, 115].

Plusieurs efforts ont été faits pour standardiser une façon de mesurer la qualité. Dans le cas de l'évaluation subjective, il faut calculer une moyenne MOS sur un grand nombre d'utilisateurs ; ceci est une tâche difficile. Donc, l'idée générale est de *prédire* la qualité subjective en fonction des paramètres de transmission [48]. Par exemple, dans la téléphonie sur Internet deux standards ont été proposés : le modèle E [105],

et PESQ (Perceptual Evaluation of Voice Quality) [106]. Les suppositions faites ces deux standards pour mesurer la qualité sont la différence principale entre eux. Par exemple, le modèle E ne considère que les paramètres statiques de transmission. PESQ considère l'adaptation du délai de diffusion mais n'inclut pas le délai absolu dans son évaluation. PESQ mesure la qualité d'un appel uniquement en considérant la qualité de la voix. Néanmoins, on a tendance à comparer la qualité d'un appel de VoIP avec la qualité d'un appel téléphonique traditionnel. Le modèle E prend ceci en compte à différence de PESQ. Un autre facteur que le modèle E prend en compte et que PESQ ignore est que les sources externes non corrélées qui affectent la qualité de la voix, sont considérées dans une note psychologique.

Le modèle E a été proposé à l'origine pour évaluer les appels téléphoniques traditionnels, mais des efforts de recherche sont en cours pour l'utiliser dans la téléphonie sur IP. Dans [27], Cole et Rosenbluth identifient les paramètres de transport les plus importants dans VoIP et les adaptent dans l'expression du facteur R. Le facteur R représente la qualité d'une session audio dans le modèle E. Celui-ci capture les effets du bruit, de l'écho, de la quantification, du délai, des équipements spéciaux, et d'autres facteurs concernant l'utilisateur final. Plus récemment, d'autres travaux ont été proposés pour simplifier le modèle E pour l'appliquer à des réseaux de commutation de paquets [43, 44, 109–111].

Tarification. La recherche dans cette direction se focalise surtout sur des mécanismes de tarification pour de différents types de qualité de service. Par exemple, dans une application multimédia, un utilisateur voudrait payer moins pour une session d'une bonne qualité et constante que pour une session de qualité variable. La caractéristique best-effort de l'Internet rend plus intéressant cet axe de recherche. Dans [22], Caesar et al. comparent plusieurs stratégies de tarification : plat (FL), sensitive à la congestion (CS), sensitive à la QoS (QoSS), et un système hybride sensitive à la congestion et à la QoS (CSQoSS). L'analyse se base sur un modèle avec deux classes d'utilisateurs. Les utilisateurs de type I paient n'importe quel prix pour la meilleure QoS. Les utilisateurs de type II demandent la meilleure QoS et payent un coût inférieur à un prix maximum donné. Dans [33], Ganesh et Laevens proposent un mécanisme dans lequel un routeur place un coût social dans chaque paquet. Les utilisateurs changent leurs demandes en fonction de ce prix. Le modèle maximise une fonction d'utilité du prix qu'un utilisateur donne à la bande passante. Les auteurs donnent une définition du prix de congestion en fonction des données arrivant sur le lien qui leur est alloué. Les auteurs montrent par simulation que le débit de transmission et le prix convergent.

Contributions de la thèse

Cette thèse est organisée en deux parties. Dans la première partie nous étudions les performances des applications temps-réel interactives. Dans la deuxième partie, nous analysons les performances des protocoles de type AIMD avec des conditions de délai variable.

Première partie : Applications temps-réel interactives.

Des applications telles que NeVoT, Vat, Freephone, ou Rat ont été largement utilisées pour faire des audio et des vidéo conférences sur Internet, aussi bien pour des sessions multipoint que pour des sessions point à point. Les dépendances entre les pertes et le délai ont été étudiées au cours des dernières années, et des mécanismes pour réduire leurs effets dans les applications multimédias ont été proposés et implementés. L'un des objectifs de cette thèse est d'évaluer les performances de ces mécanismes dans les applications temps-réel interactives. Nous étudions dans le Chapitre 2 la performance d'un mécanisme FEC implementé dans Freephone et Rat. Dans ce mécanisme, le paquet i porte l'information originale, et une version redondante encodée généralement à plus bas débit, est transmise ϕ paquets après. Par exemple, le paquet i est encodé avec PCM de loi μ, et sa redondance avec GSM ou LPC. Si le paquet i est perdu dans le réseau, il peut toujours être joué au récepteur si sa copie redondante dans le paquet $i + \phi$ arrive. Le paquet redondant est espacé du paquet original pour essayer de réduire les effets des pertes en rafale qui sont typiques dans l'Internet. On suppose que la qualité de la copie redondante est une fonction linéaire du taux de compression par rapport au paquet original. Alors, avec une analyse de files d'attente, nous montrons que la quantité d'information utile au récepteur diminue pour n'importe quelle quantité de redondance, et pour n'importe quelle valeur de ϕ [5]. Ceci est un résultat très important car cela signifie qu'utiliser de la FEC détériore la qualité d'une application de VoIP.

Dans le Chapitre 3, nous généralisons notre modèle de files d'attente en considérant deux aspects qui peuvent améliorer la qualité: multiplexage avec d'autres flux, et des fonctions d'utilité non linéaires. Cette dernière supposition est basée sur l'observation que la qualité des applications multimédia est souvent mesurée subjectivement, et qu'elle dépend d'un large nombre de paramètres. Des fonctions d'utilité similaires à celles que nous proposons ont été utilisées précédemment dans le cas des applications multimédias [96]. De plus, nous généralisons notre modèle en utilisant une file d'attente $M/G/1/K$ avec un flux audio et un flux exogène. Nous trouvons des conditions pour lesquelles l'utilisation de FEC est bénéfique pour la qualité [4,6]. À noter que, bien que nous nous focalisons sur des applications de VoIP, nos modèles sont valables pour n'importe quelle autre application

utilisant ce type de mécanisme.

Dans le Chapitre 4 nous abordons le problème du contrôle du délai de diffusion. Dans les applications temps-réel, les paquets sont encodés pour alors être transmis périodiquement. Cette périodicité est perdue à cause de la gigue introduite lorsque les paquets traversent les files d'attente dans les routeurs dans le chemin entre l'émetteur et le récepteur. Les algorithmes de contrôle du délai de diffusion permettent de réduire la gigue en faisant un compromis entre les pertes tardives et le délai. La plupart des algorithmes actuels ne permettent pas de contrôler le taux de pertes perçu par une application audio. Nous identifions ce problème et nous proposons un ensemble d'algorithmes de moyenne mobile pour le délai de diffusion permettant de contrôler le taux de pertes lors d'une session audio [81]. Pour ce faire, nous évaluons et comparons nos algorithmes grâce à des traces réelles. Nous comparons nos algorithmes avec ceux actuellement implementés dans l'application NeVoT. Nous montrons que les performances de nos algorithmes sont meilleures que celles de NeVoT pour un grand nombre de scénarios.

Deuxième partie: Un modèle pour les protocoles AIMD avec des conditions de délai variable

Dans la deuxième partie de cette thèse nous proposons un modèle pour analyser les protocoles de type AIMD avec des conditions de délai variable. Le meilleur exemple de ce type d'algorithmes est le protocole TCP. Les applications multimédias non interactives utilisent souvent TCP car leurs contraintes de délai ne sont pas aussi strictes que celles des applications interactives. Les modèles traditionnels pour les protocoles AIMD calculent le débit en modélisant le délai d'aller-retour comme étant une constante, pour alors le remplacer par sa moyenne. Néanmoins, on a observé récemment dans [24] que la variabilité du délai affecte le débit des protocoles AIMD. La variabilité du délai est un phénomène courant dans les réseaux de communication. D'ailleurs, avec l'utilisation croissante des réseaux sans fil, il est très important d'étudier les effets de la variabilité du délai sur les performances des protocoles de type AIMD.

Notre contribution dans cette partie de la thèse est le premier modèle analytique (à notre connaissance) prenant en compte la variabilité du délai pour calculer le débit des protocoles AIMD. Le modèle est basée sur des équations différentielles stochastiques. Il fournit une expression close pour le débit et pour la taille de la fenêtre dans l'état d'équilibre. Nous considérons deux cas pour modéliser le temps d'aller-retour : des variables aléatoires indépendantes et identiquement distribuées (i.i.d.), et des variables corrélées Markov. Pour valider notre modèle, nous utilisons deux versions du protocole TCP : SACK et/ou Newreno, et Reno.

Nous montrons par analyse et par simulation, avec le simulateur NS [67], qu'une

augmentation de la variabilité du délai améliore les performances de ce type de protocole. Le résultat le plus important, es que lorsque le processus de pertes est constant et les RTT sont i.i.d., le débit d'un mécanisme AIMD augmente avec la variabilité du délai. Ceci signifie que les modèles actuels pour ce type de protocole, qui ne considèrent que le premier moment des délais d'aller-retour, sous-estiment les performances quand le délai des paquets est variable [7]. Nous décrivons en détail notre modèle dans le Chapitre 5.

Finalement, nous concluons cette thèse dans le Chapitre 6, et nous discutons aussi quelques perspectives pour notre travail futur.

Annexe

Conclusions et perspectives

Conclusions

Cette thèse est organisée en deux parties. Dans la première partie, on a étudié les facteurs qui affectent la qualité des applications multimédias interactives. Bien qu'on se focalise sur des applications audio, nos résultats sont applicables à n'importe quelle application multimédia temps réel qui utilise les mécanismes a étudiés. Dans la deuxième partie de la thèse, nous avons proposé un modèle pour analyser les performances des protocoles de type AIMD avec des conditions de délai variable. Notre modèle est construit avec des équations différentielles stochastiques. Il fournit une expression close pour le débit et montre l'impact de la variabilité du délai sur ce type de protocoles, dont TCP est le meilleur exemple.

Dans le Chapitre 2 nous avons étudié les performances d'un mécanisme de correction d'erreurs (FEC) qui est implementé dans des applications audio telles que Freephone [34] ou Rat [79]. La redondance, qui est généralement encodée avec un débit plus bas que l'information originale, est transmise ϕ paquets après la transmission du paquet original. Le chemin entre l'émetteur et le récepteur a été modélisé comme un goulot d'étranglement dans le réseau, et on a supposé que celui-ci était dédié uniquement au flux audio. Cette supposition est valide, par exemple, quand tous les flux dans le réseau implementent le mécanisme FEC étudié. On a utilisé une file d'attente $M/M/1/K$ pour modéliser le processus de pertes des paquets audio dans le goulot d'étranglement. La supposition du processus de Poisson est justifiée par le délai aléatoire introduit par les routeurs se trouvant avant

le goulot d'étranglement. Notre but était de trouver une expression mathématique simple permettant d'explorer l'impact de l'utilisation des mécanismes FEC sur les applications audio. L'expression qu'on a obtenue donne la qualité de l'audio dans le récepteur en fonction de la quantité de redondance introduite par la FEC, en fonction de la distance entre la redondance et l'information originale. Le résultat le plus important que nous en avons tiré est que, même pour la limite supérieure de la qualité de l'audio, $\phi \to \infty$, l'utilisation de FEC a toujours un impact négatif sur la qualité de l'audio. Ceci est principalement dû à une augmentation importante de la charge du réseau.

Dans le Chapitre 3, nous avons étudié les conditions sous lesquelles ce mécanisme FEC améliore la qualité de l'audio. On a considéré le cas d'un flux audio partageant le goulot d'étranglement avec un flux exogène n'implementant pas de FEC. On a aussi considéré des fonctions d'utilité non linéaires. Des travaux précédents [96], ont montré que la qualité dans les applications multimédias a souvent une forme non-linéaire. Typiquement, la fonction d'utilité de ce type d'application est convexe près de zéro et concave après un certain débit, où le débit dépend du type d'application multimédia. Pour le cas des applications audio, le fait de reconstruire un paquet à partir d'une copie dont le taux d'encodage est $\alpha < 1$, pourrait donner une qualité très proche à celle du paquet d'origine. Pour le cas du multiplexage d'un flux audio et de flux audio n'implementant pas de FEC, on a modélisé ces flux (à l'exception du flux audio) comme étant un seul flux exogène de débit constant et dont la taille des paquets suivait une loi exponentielle. Le processus d'arrivée du flux audio et du flux exogène dans le goulot d'étranglement, a été modélisé par un processus de Poisson. On a utilisé une file d'attente $M/G/1/K$ pour modéliser le réseau, et on a supposé que les temps de service étaient indépendants et identiquement distribués. On a fourni des solutions numériques pour la qualité de l'audio avec de différentes charges dans la file d'attente, et pour différents taux de redondance de FEC. On a montré qu'il est possible d'obtenir un gain en qualité avec ce mécanisme FEC si l'intensité du flux audio est plus petite que celle du flux exogène. Quand l'intensité du flux implementant de la FEC augmente, le gain en qualité diminue. Si la plupart des flux implementaient de la FEC, alors le gain en qualité pourrait devenir nul. D'après ces résultats, nous concluons aussi qu'une fonction d'utilité non-linéaire n'est pas une supposition adéquate pour mesurer la qualité de ce type d'application. Nous avons confirmé ce résultat quand on a étudié d'autres fonctions d'utilité non-linéaires à la fin du Chapitre 3.

Dans le Chapitre 4 nous avons abordé le problème du contrôle du délai de diffusion (playout delay en anglais). Les mécanismes de contrôle du délai de diffusion réduisent la gigue dans le récepteur en faisant un compromis entre les pertes tardives et le délai de diffusion. On a observé que les algorithmes implementés actuellement dans des applications audio telles que NeVoT [91], ne permettent pas de contrôler le taux de pertes perçu par

une application audio. Ceci est une caractéristique essentielle dans un algorithme de contrôle du délai de diffusion. Dans ce chapitre de la thèse, notre objectif principal a été de proposer un algorithme de délai de diffusion capable de contrôler le taux de pertes vu par une application. Pour ce faire, on a proposé un ensemble d'algorithmes de moyenne mobile (MM) capables d'accepter comme entrée un taux de pertes souhaité par l'utilisateur d'une application audio. Cet ensemble d'algorithmes est composé d'abord par : (a) un algorithme MM " offline ", (b) un algorithme MM offline qui transforme le processus du délai optimum afin de réduire les déviations d'estimation observées pour les cas où le taux de pertes souhaité était très petit, et (c) un algorithme hybride MM " online ". Les performances de nos algorithmes ont été évaluées grâce à des traces audio réelles obtenues avec l'outil NeVoT. Nous avons comparé les performances de nos algorithmes avec celles des algorithmes proposés par Ramjee [82], et qui sont impleméntés dans NeVoT. En utilisant des traces réelles, on a évalué nos algorithmes avec des conditions identiques du réseau. Pour la plupart des cas, nos algorithmes ont montré de meilleures performances sur une grande variété de scénarios. Surtout, la caractéristique la plus intéressante de nos algorithmes étant de permettre de contrôler le taux de pertes souhaité dans une session audio.

Finalement, dans la deuxième partie de cette thèse, nous avons présenté un modèle pour les protocoles de type AIMD avec des conditions de délai variable. Plusieurs modèles pour ce type de protocoles ont déjà été proposés ; pourtant, ils ignorent la variabilité du délai. De plus, ils modélisent le délai d'aller-retour comme étant une constante, et ils le remplacent par son espérance lors du calcul du débit d'une connexion. Dans cette partie de la thèse, nous avons proposé le premier modèle analytique pour les protocoles de type AIMD prenant en compte la variabilité du délai lors du calcul du débit. Notre modèle est basé sur des équations de différences stochastiques. On obtient des expressions closes pour le débit et pour la taille de la fenêtre dans l'état d'équilibre. On a validé notre modèle par analyse et par simulation avec le simulateur NS-2 [67]. Pour ce faire, on a modélisé deux versions du protocole TCP : Newreno (ou SACK) et TCP Reno. On a considéré deux cas différents pour modéliser le temps d'aller-retour (RTT) : le cas où les RTT des paquets sont des variables aléatoires indépendantes et identiquement distribuées (i.i.d.), et le cas où les RTT sont correlées-Markov. Le résultat le plus important que nous avons obtenu est que, lorsque le taux de pertes des paquets est constant et lorsque les RTT sont i.i.d., le débit d'un mécanisme AIMD augmente avec la variabilité du délai. Ceci est un résultat très important, puisque cela signifie que les modèles actuels pour ce type de protocoles, qui ne considèrent que le premier moment des RTT, sous-estiment les performances du protocole quand le délai est variable. Pour le cas Markovien, on a considéré une simple chaîne de Markov à deux états (on-off) pour modéliser les RTT. On

a observé qu'en général, l'estimation du débit avec le modèle Markovien est plus précise que dans les autres cas.

Perspectives et travaux futurs

L'ensemble d'algorithmes MM que nous avons proposés adapte le délai de diffusion au début de chaque phrase dans une session audio. Des travaux récents sur ce sujet proposent des mécanismes pour mettre à jour le délai de diffusion à l'arrivée de chaque paquet plutôt qu'au début de chaque phrase. L'avantage de ces mécanismes est qu'ils sont plus dynamiques au moment de décider la deadline pour jouer un paquet, et de cette façon, ils réduisent le temps moyen de séjour des paquets dans le tampon du récepteur. Par ailleurs, dans un article récent [100], Sun et Ifeachor proposent un algorithme basé sur une estimation subjective de la qualité qui utilise une méthode similaire à PESQ. Nous pensons qu'une méthode qui adapte le délai paquet par paquet, et qui estime la qualité subjective au cours d'une session audio, peut avoir de meilleures performances que n'importe quelle autre méthode existante actuellement. Alors, notre travail futur se focalisera sur cette idée tout en étudiant la possibilité de l'implementer dans un outil audio réel.

Par ailleurs, il y a actuellement dans ce domaine un grand intérêt à construire des mécanismes adaptatifs pour l'audio. Dans [19], Boutremans et Le Boudec proposent un mécanisme permettant de contrôler le débit d'émission d'une application audio (et en conséquence, la quantité de redondance injectée dans le réseau) en se basant sur les conditions du réseau. Le débit d'émission est alors une fonction du taux de pertes et du processus de délai dans le réseau. Ils utilisent le E-model, et proposent des fonctions qui expriment le débit en fonction du facteur R, et l'utilité d'une application audio en fonction du taux de pertes et du délai de bout-en-bout. Les performances de leur mécanisme sont évaluées numériquement et par simulation.

Les algorithmes proposés par Boutremans et Le Boudec utilisent des algorithmes traditionnels de contrôle du délai de diffusion. Nous pensons qu'un algorithme qui adapte le délai de diffusion paquet par paquet, avec une estimation de la qualité subjective peut être utilisé avec le modèle de Boutremans et Le Boudec pour obtenir de meilleures performances. Une partie de notre travail futur se focalisera sur cette idée.

Bibliography

[1] Prathima Agrawal, Jyh-Cheng Chen, and Cormac J. Sreenan. Use of statistical methods to reduce delays for media playback buffering. In *International Conference on Multimedia Computing and Systems*, pages 259–263, 1998.

[2] Eitan Altman. Stochastic recursive equations with applications to queues with dependent vacations. *Annals of Operations Research*, 112(1):43–61, April 2002.

[3] Eitan Altman, Konstantin Avratchenkov, and Chadi Barakat. A stochastic model of TCP/IP with stationary random losses. In *Proceedings of the ACM Sigcomm*, pages 231–242, Stockholm, Sweden, August 2000.

[4] Eitan Altman, Chadi Barakat, and Víctor M. Ramos R. On the utility of FEC mechanisms for audio applications. In *Proceedings of the International Workshop on Quality of Future Internet Services (QofIS)*, Coimbra, Portugal, September 2001.

[5] Eitan Altman, Chadi Barakat, and Víctor M. Ramos R. Queueing analysis of simple FEC schemes for IP telephony. In *Proceedings of the IEEE Infocom*, Anchorage, Alaska, USA, April 2001.

[6] Eitan Altman, Chadi Barakat, and Víctor M. Ramos R. Queuing analysis of simple FEC schemes for voice over IP. *Computer Networks*, 39:185–206, 2002.

[7] Eitan Altman, Chadi Barakat, and Víctor M. Ramos R. Analysis of AIMD protocols over paths with variable delay. In *Proceedings of the IEEE Infocom*, Hong Kong, March 2004.

[8] Eitan Altman and Alain Jean-Marie. Loss probabilities for messages with redundant packets feeding a finite buffer. *Journal on Selected Areas in Communications*, 16(5):779–887, 1998.

[9] Venkat Anantharam and Takis Konstantopoulos. Stationary solutions of stochastic recursions describing discrete event systems. *Stochastic Processes and Their Applications*, 68:181–194, August 1997. Correction published in vol. 80, no. 2, pp. 271–278, April 1999.

[10] François Baccelli and Pierre Brémaud. *Elements of queueing theory: Palm-Martingale calculus and stochastic recurrences*. Springer-Verlag, 1994.

[11] Jean Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Journal of High Speed Networks*, 2(3):305–323, December 1993.

[12] Jean Bolot. End-to-end Internet packet dynamics. In *Proceedings of the ACM Sigcomm*, pages 189–199, San Francisco, California, August 1993.

[13] Jean Bolot, Hugues Crépin, and Andrés Vega García. Analysis of audio packet loss in the Internet. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 163–174, April 1995.

[14] Jean Bolot, Sacha Fosse-Parisis, and Don Towsley. Adaptive FEC-based error control for interactive audio in the Internet,. In *Proceedings of the IEEE Infocom*, March 1999.

[15] Jean Bolot and Andrés Vega García. Control mechanisms for packet audio in the Internet. In *Proceedings of the IEEE Infocom*, March 1996.

[16] Jean Bolot and Andrés Vega García. The case for FEC-based error control for packet audio in the Internet. In *ACM Multimedia Systems*, 1997.

[17] Alexander Alekseevich Borovkov and Serguei Foss. Stochastically recursive sequences and their generalizations. *Siberian Advances in Mathematics*, 2(1):16–81, 1992.

[18] Catherine Boutremans and Jean-Yves Le Boudec. Adative delay aware error control for Internet telephony. In *Proceedings of the IP Telephony Workshop*, New York, USA, April 2001.

[19] Catherine Boutremans and Yves Le Boudec. Adaptive joint playout buffer and FEC adjustement for Internet telephony. Technical report, EPFL, May 2002.

[20] Onno J. Boxma. Sojourn times in cyclic queues – the influence of the slowest server. In *Computer Performance and Reliability*, pages 13–24. Elsevier Science Publishers B.V. (North-Holland), 1988.

[21] A. Brandt. The stochastic equation $Y_{n+1} = A_n Y_n + B_n$ with stationary coefficients. *Advances in Applied Probability*, 18, 1986.

[22] M. Caesar, S. Balaram, and D. Ghosal. A comparative study of pricing strategies for IP telephony. In *Proceedings of the IEEE Globecom*, San Francisco, CA, USA, November 2000.

[23] Gregory Cermak. Subjective quality of speech over packet networks as a function of packet loss, delay, and delay variation. *International Journal of Speech Technology*, 1:65–84, January 2002.

[24] Mun Choon Chan and Ramachandran Ramjee. TCP/IP performance over 3G wireless links with rate and delay variation. In *Proceedings of Mobicom*, pages 71–82, Atlanta, Georgia, USA, September 2002.

[25] Israel Cidon, Asad Khamisy, and Moshe Sidi. Analysis of packet loss process in high-speed networks. *IEEE Transactions on Information Theory*, IT-39(1):98–108, January 1993.

[26] Jacob Willem Cohen. *The Single Server Queue*. North-Holland, 1969.

[27] R. Cole and J. Rosenbluth. Voice over IP performance monitoring. *ACM Computer Communication Review*, 31(2), April 2001.

[28] Phillip DeLeon and Cormac Sreenan. An adaptive predictor for media playout buffering. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2097–3100, March 1999.

[29] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. *ACM Computer Communication Review*, 26(3):5–21, July 1996.

[30] Nikolaus Farber, Yi Liang, Bernd Girod, and Balaji Prabhakar. Adaptive playout and TCP window control for voice over IP in best-effort networks. Technical report, Stanford University, Information Systems Laboratory, Department of Electrical Engineering, March 2001.

[31] Daniel Ratton Figueiredo and Edmundo de Souza e Silva. Efficient mechanisms for recovering voice packets in the Internet. In *Proceedings of the IEEE Global Telecommunications Conference Globecom*, pages 1830–1837, 1999.

[32] Kouhei Fujimoto, Shingo Ata, and Masayuki Murata. Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications. In *Proceedings of the IEEE Globecom*, 2002.

[33] A. Ganesh and K. Laevens. Congestion pricing and user adaptation. In *Proceedings of the IEEE Infocom*, Anchorage, Alaska, USA, April 2001.

[34] Andrés Vega García and Sacha Fosse-Parisis. FreePhone audio tool. http://www-sop.inria.fr/rodeo/fphone. High-Speed Networking Group, INRIA Sophia Antipolis.

[35] Paul Glasserman and D. D. Yao. Stochastic vector difference equations with stationary coefficients. *Journal of Applied Probability*, 32:851–866, 1995.

[36] Olof Hagsand, Kjell Hanson, and Ian Marsh. Measuring Internet telephony quality: Where we are today? In *Proceedings of the IEEE Global Telecommunications Conference Globecom*, pages 1838–1842, 1999.

[37] Jeong-Soo Han, Seong-Jin Ahn, and Jin-Wook Chung. Study of delay patterns of weighted voice traffic of end-to-end users on the VoIP network. *International Journal of Network Management*, 2(5):271–280, 2002.

[38] Vicky Hardman, Angela Sasse, Mark Handley, and Anna Watson. Reliable audio for use over the Internet. In *Proceedings of the INET*, Oahu, Hawaii, 1995.

[39] Vicky Hardman, Martina Angela Sasse, and Isidor Kouvelas. Successful multiparty audio communication over the Internet. *Communications of the ACM*, 45(5):74–80, May 1998.

[40] Omar Ait Hellal, Eitan Altman, Alain Jean-Marie, and Irina A. Kurkova. On loss probabilities in presence of redundant packets and several traffic sources. *Performance Evaluation*, 36-37:486–518, 1999.

[41] Orion Hodson, Colin Perkins, and Vicky Hardman. Skew detection and compensation for Internet audio. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, New York, NY, USA, July 2000.

[42] Van Jacobson and Michael J. Karels. Congestion avoidance and control. In *Proceedings of the ACM Sigcomm*, pages 314–329, August 1988.

[43] Jan Janssen, Danny De Vleeschauwer, Maarten Buchli, and Guido H. Petit. Assessing voice quality in packet-based telephony. *IEEE Internet Computing*, May-June 2002.

[44] Jan Janssen, Danny De Vleeschauwer, and Guido H. Petit. Delay and distortion bounds for packetized voice calls of traditional PSTN quality. In *IP Telephony Workshop*, pages 167–172, Berlin, Germany, April 2000.

[45] N. Jayant. Effects of packet loss on waveform coded speech. In *Proceedings of the International Conference on Computer Communications*, pages 275–280, October 1980.

[46] Wenyu Jiang, Jonathan Lennox, Sankaran Narayanan, Henning Schulzrinne, Kundan Singh, and Xiaotao Wu. Integrating Internet telephony services. *IEEE Internet Computing*, May-June 2002.

[47] Wenyu Jiang and Henning Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proceedings of Nossdav*, 2000.

[48] Nils Olof Johannesson. The ETSI computation model: A tool for transmission planning of telephone networks. *IEEE Communications Magazine*, pages 70–79, Jan 1997.

[49] Aman Kansar and Abhay Karandikar. Jitter-free audio playout over best effort packet networks. In *ATM Forum International Symposium*, New Delhi, India, 2001.

[50] D. G. Kleinbaum, L. L. Kupper, and K. E. Muller. *Applied Regression Analysis and Other Multivariable Methods*. PWS-Kent, Boston, 1988.

[51] Leonard Kleinrock. *Queuing systems*. John Wiley, New York, 1976.

[52] Isidor Kouvelas and Vicky Hardman. Overcoming workstation scheduling problems in a real-time audio tool. In *Proceedings of the Usenix*, January 1997.

[53] Isidor Kouvelas, Orion Hodson, Vicky Hardman, and Jon Crowcroft. Redundancy control in real-time Internet audio conferencing. In *Proceedings of Audio–Visual Services over Packet Networks (AVSPN)*, Aberdeen, Scotland, UK, September 1997.

[54] Nikolaus Laoutaris and Ioannis Stavrakakis. Intrastream synchronization for continuous media streams: A survey of playout schedulers. *IEEE Network*, 16(3), 2002.

[55] Yi J. Liang, Nikolaus Farber, and Bernd Girod. Adaptive playout scheduling and loss concealment for voice communications over IP networks. *IEEE Transactions on Multimedia*, April 2001.

[56] Yi J. Liang, Nikolaus Farber, and Bernd Girod. Adaptive playout scheduling using time-scale modification in packet voice communications. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[57] Wen-Tsai Liao, Jeng-Chun Chen, and Ming-Syan Chen. Adaptive recovery techniques for real-time audio streams. In *Proceedings of the IEEE Infocom*, Anchorage, Alaska, USA, April 2001.

[58] Fang Liu, Jong Won Kim, and C.C. Jay Kuo. Adaptive delay concealment for Internet voice applications with packet-based time-scale modification. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Salt Lake City, May 2001.

[59] Fang Liu, Jong Won Kim, and Jay Kuo. Adaptive delay concealment for Internet voice applications with packet-based time-scale modification. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2001.

[60] Athina P. Markopoulou, Fouad A. Tobagi, and Mansour J. Karam. Assessing the quality of voice communications over Internet backbones. *IEEE/ACM Transactions on Networking*, 11(5):747–760, October 2003.

[61] Johnny Matta, Christine Pépin, Khosrow Lashkari, and Ravi Jain. A source and channel rate adaptation algorithm for AMR in VoIP using the E-model. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 92–99, Monterey, CA, USA, June 2003.

[62] Hugh Melvin and Liam Murphy. Time synchronization for VoIP quality of service. *IEEE Internet Computing*, May-June 2002.

[63] Art Mena and John Heidemann. An empirical study of real audio traffic. In *Proceedings of the IEEE Infocom*, 2000.

[64] Arnaud Meylan and Catherine Boutremans. Realisation of an adaptive audio tool. In *Proceedings of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS)*, Enschede, The Netherlands, October 2000.

[65] Sue B. Moon, Jim Kurose, and Don Towsley. Packet audio playout delay adjustment: Performance bounds and algorithms. *ACM/Springer Multimedia Systems*, 6:17–28, January 1998.

[66] Sue B. Moon, Paul Skelly, and Don Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of the IEEE Infocom*, New York, NY, USA, March 1999.

[67] University of California at Berkeley. Network simulator v.2. Available via http://www-nrg.ee.lbl.gov/ns-2.

[68] Chinmay Padhye, Kenneth J. Christensen, and Wilfrido Moreno. A new adaptive FEC loss control algorithm for voice over IP applications. In *Proceedings of the IEEE International Performance, Computing, and Communication Conference*, pages 307–313, February 2003.

[69] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of the ACM Sigcomm*, pages 303–314, 1998.

[70] Vern Paxson. On calibrating measurements of packet transit times. In *Measurement and Modeling of Computer Systems*, pages 11–21, 1998.

[71] C. Perkins and O. Hodson. Options for repair streaming media. *Request for Comments 2354*, June 1998.

[72] Colin Perkins, Orion Hodson, and Vicky Hardman. A survey of packet-loss recovery techniques for streaming audio. *IEEE Network*, pages 40–48, September–October 1998.

[73] Jesús Pinto and Kenneth J. Christensen. An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*, pages 224–231, October 1999.

[74] Matthew Podolsky. *Transmission of Real-time Multimedia Over the Internet*. PhD thesis, University of California, Berkeley, 1999.

[75] Matthew Podolsky, Cynthia Romer, , and Steven McCanne. Simulation of FEC-based error control for packet audio in the internet. In *Proceedings of the IEEE Infocom*, pages 505–515, April 1998.

[76] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing: Principles, algorithms, and aplications*. Prentice-Hall Inc., 1996.

[77] Darpa Internet Program. RFC 791: Internet Protocol. ftp://ftp.rfc-editor.org/in-notes/rfc791.txt, September 1981.

[78] Darpa Internet Program. RFC 793: Transmission Control Protocol. ftp://ftp.rfc-editor.org/in-notes/rfc793.txt, September 1981.

[79] T.M. Project. Rat: Robust audio tool. Multimedia Integrated Conferencing for European Researchers. University College London.

[80] Qualcomm. Delays in the HDR system, June 2000.

[81] Víctor M. Ramos R., Chadi Barakat, and Eitan Altman. A moving average predictor for playout delay control in VoIP. In *Proceedings of the International Workshop on Quality of Service (IWQoS)*, June 2003.

[82] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the IEEE Infocom*, pages 680–688, 1994.

[83] Jonathan Rosenberg and Henning Schulzrinne. Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet. In *Proceedings of the IEEE Infocom*, pages 1705–1714, March 1999.

[84] Jonathan Rosenberg, Henning Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 2543: SIP Session Initiation Protocol. Available at ftp://ftp.rfc-editor.org/in-notes/rfc3261.txt, June 2002.

[85] Jonathan Rosenberg and Henning Shulzrinne. Internet telephony gateway location. In *Proceeedings of the IEEE Infocom*, San Francisco, CA, USA, March 1998.

[86] Lopamudra Roychoudhuri, Ehab Al-Shaer, Hazem Hamed, and Gregory B. Brewster. Audio transmission over the Internet: Experiments and observations. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2003.

[87] Mohammad Reza Salamatian. *Transmission Multimédia Fiable Sur Internet*. PhD thesis, Université Paris Sud UFR Scientifique d'Orsay, December 1999.

[88] Henning Sanneck. Concealment of lost speech packets using adaptive packetization. In *International Conference on Multimedia Computing and Systems*, pages 140–149, June 1998.

[89] Henning Sanneck and Nguyen Tuong Long Le. Speech property-based FEC for Internet telephony applications. In *Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference*, pages 38–51, San José, California, USA, January 2000.

[90] Henning Sanneck, A. Stenger, K. Ben. Younes, and Bernd Girod. A new technique for audio packet loss concealment. In *Proceedings of the IEEE Global Internet*, pages 554–557, November 1996.

[91] Henning Schulzrinne. Voice communication across the Internet. Technical report, University of Massachusetts, Amherst, 1992.

[92] Henning Schulzrinne, Steven Casner, R. Frederick, and Van Jacobson. RFC 3550: RTP: A transport protocol for real-time applications. Available at ftp://ftp.rfc-editor.org/in-notes/rfc3550.pdf, July 2003.

[93] Henning Schulzrinne and S. Petrack. RFC 2833: RTP payload for DTMF digits, telephony tones and telephony signals. Available at ftp://ftp.rfc-editor.org/in-notes/rfc2833.txt, May 2000.

[94] John Scourias. Overview of the global system for mobile communications. University of Waterloo.

[95] Cristina Hristea Seibert and Fouad Tobagi. Assessing the user-perceived quality of packet voice in networks with mobile users. In *Proceedings of the ACM MSWiM*, 2003.

[96] Scott Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13:1176–1188, September 1995.

[97] Cormac J. Sreenan, Jyh-Cheng Chen, Prathima Agrawal, and B. Narendran. Delay reduction techniques for playout buffering. *IEEE Transactions on Multimedia*, pages 88–100, June 2000.

[98] William Stallings. *ISDN and Broadband ISDN with Frame Relay and ATM*. Prentice Hall, 4/e edition, 2001.

[99] A. Stenger, K. Younes, R. Reng, and B. Girod. A new error concealment technique for audio transmission with packet loss. In *EUSIPCO*, 1996.

[100] Lingfen Sun and Emmanuel Ifeachor. New models for perceived voice quality prediction and their applications in playout buffer optimization for VoIP networks. In *Proceedings of the International Conference on Communications (ICC)*, 2004.

[101] Lajos Takács. *Combinatorial Methods in the Theory of Stochastic Processes*. John Wiley and Sons, 1967.

[102] Guy Thomsen and Yashvant Jani. Internet telephony: going like crazy. *IEEE Spectrum*, 37(5):52–58, May 2000.

[103] Po Tien and Maria Yuang. Intelligent voice smoother for silence suppressed voice over Internet. *IEEE Journal on Selected Areas in Communications*, 17(1):29–41, 1999.

[104] Thomas E. Tremain. The government standard linear predictive coding algorithm: LPC-10. *Speech Technology*, 1:40–49, April 1982.

[105] International Telecommunication Union. The E-model, a computational model for use in transmission planning. ITU-T Recommendation G.107, July 2000.

[106] International Telecommunication Union. Perceptual evaluation of speech quality, an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. ITU-T Recommendation P.862, February 2001.

[107] International Telecommunications Union. Recommendation H.323: Packet-based multimedia communication systems. Available at www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-H.323, February 1998.

[108] Werner Verhelst and Marc Roelands. An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 554–557, Minneapolis, USA, April 1993.

[109] D. De Vleeschauwer, J. Janssen, and G.H. Petit. Delay bounds for low bit rate voice transport over IP networks. In *Proceedings of the SPIE Conference on Performance and Control of Network Systems III*, volume 3841, pages 40–48, Boston, Massachusetts, September 1999.

[110] D. De Vleeschauwer, J. Janssen, and G.H. Petit. Voice over IP in access networks. In *Proceedings of the Seventh IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks*, pages 28–30, Antwerp Belgium, June 1999.

[111] D. De Vleeschauwer, J. Janssen, G.H. Petit, and F. Poppe. Limites de la qualité pour le transport de la voix en mode paquet. In *Revue des Télécommunications d'Alcatel*, pages 19–24. Alcatel, January 2000.

[112] Milan Vojnovic and Jean-Yves Le Boudec. On the long-run behavior of equation-based rate control. In *Proceedings of the ACM Sigcomm*, pages 103–116, August 2002.

[113] Benjamin Wah and Dong Lin. Transformation-based reconstruction for real-time voice transmissions over the Internet. *IEEE Transactions on multimedia*, 1(4):342–351, December 1999.

[114] Ondria J. Wasem, David J. Goodman, Charles A. Dvorak, and Howard G. Page. The effect of waveform substitution on the quality of PCM packet communications. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(3), March 1988.

[115] Anna Watson and Angela Sasse. Measuring perceived quality of speech and video in multimedia conferencing applications. In *Proceedings of ACM Multimedia*, pages 55–60, September 1998.

[116] Li Zhang, Zhen Liu, and Cathy Honghui Xia. Clock synchronization algorithms for network measurements. In *Proceedings of the IEEE Infocom*, 2002.

[117] Yin Zhang, Nick Duffield, Vern Paxson, and Scott Shenker. On the constancy of Internet path properties. In *Internet Measurement Workshop (IMW)*, Marseille, France, November 2002.

Résumé

Dans cette thèse, nous proposons des modèles pour évaluer les performances des applications multimédias temps-réel. De plus, nous proposons un modèle pour les protocoles de type AIMD. Le premier sujet étudié est un mécanisme de correction d'erreurs (FEC). Premièrement, nous utilisons une file d'attente $M/M/1/K$ pour modéliser le réseau. Nous considérons que la qualité de la voix varie linéairement par rapport au taux de redondance de la FEC. La redondance du i-ème paquet est portée dans le paquet $i + \phi$. Notre analyse montre que, même pour le cas $\phi \to \infty$, ce mécanisme n'améliore pas la qualité de l'audio. Deuxièmement, nous modélisons notre système par une file d'attente $M/G/1/K$. Nous considérons deux aspects que peuvent contribuer à améliorer la qualité de l'audio : (a) multiplexer l'audio avec un flux exogène, et (b) des fonctions d'utilité non-linéaires. Sous ces contraintes, on montre qu'il est possible d'améliorer la qualité de l'audio avec la méthode FEC étudiée. Le deuxième sujet traité concerne les mécanismes de contrôle du délai de diffusion. Nous proposons un ensemble d'algorithmes de moyenne mobile permettant de contrôler le taux de pertes dans une session audio. Les performances de nos algorithmes ont été évaluées et comparées grâce à des traces réelles. Le troisième sujet abordé concerne les protocoles de type AIMD. Nous proposons un modèle analytique prenant en compte la variabilité du délai. Notre modèle utilise des équations différentielles stochastiques. Il fournit une expression close pour le débit et pour la taille de la fenêtre. Nous montrons, par analyse et par simulation, qu'une augmentation de la variabilité du délai améliore les performances d'un protocole AIMD.

Mots clés: Correction d'erreurs (FEC), fonction d'utilité, théorème de ballot, contrôle du délai de diffusion, estimateur de moyenne mobile, TCP, variabilité du délai, équations différentielles stochastiques.

Abstract

In this thesis, we propose models to evaluate the performance of real-time multimedia applications. Besides, we propose a model for AIMD protocols. The first subject we study is a simple error correction (FEC) mechanism. We first model the network as an $M/M/1/K$ queuing system. We assume a linear utility function relating the audio quality and the amount of redundancy. The redundancy of packet i is carried by packet $i + \phi$. Our analysis shows that, even for the case $\phi \to \infty$, this simple FEC scheme always leads to a quality deterioration. Next, we model the bottleneck router as an $M/G/1/K$ queuing system. We consider two cases that may contribute to a quality improvement: (a) multiplexing the audio flow with an exogenous flow, and (b) considering non-linear utility functions. Under these assumptions, we show that this FEC scheme can lead to a quality improvement. The second subject investigated is about playout delay algorithms. We propose a set of moving average algorithms allowing to control the average loss rate in an audio session. We study and compare the performance of our algorithms by simulation with real packet traces. The third subject we study is about the performance of AIMD protocols. We propose, at the best of our knowledge, the first analytical model for this kind of protocols accounting for delay variability. The model is based on stochastic difference equations. It provides a closed-form expression for the throughput and for the window size in steady state. We show by analysis and simulation that an increase in delay variability improves the performance of AIMD protocols.

Keywords: Forward Error Correction (FEC), utility function, ballot theorem, playout delay control, moving average estimation, TCP, delay variability, stochastic recursive equations.