ADAPTIVE MEDIA STREAMING OVER MULTIPATH NETWORKS

THÈSE N^{O} 3908 (2007)

PRÉSENTÉE LE 26 OCTOBRE 2007

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

Laboratoire de traitement des signaux 4

SECTION DE GÉNIE ÉLECTRIQUE ET ÉLECTRONIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Dan JURCA

Licenced Engineer in Electronics and Telecommunications, Politehnica University of Timisoara, Roumania
et de nationalité roumaine

acceptée sur proposition du jury:

Prof. J. R. Mosig, président du jury
Prof. P. Frossard, directeur de thèse
Dr W. Kellerer, rapporteur
Prof. P. Thiran, rapporteur
Prof. M. van der Schaar, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2007

The mind is like an umbrella, it only works when it is open.

Heard it somewhere.....

Aknowledgment

First, I would like to thank my PhD advisor, Prof. Pascal Frossard, for giving me the opportunity to do research in his laboratory, and for making these last four years possible. If PhD is a learning experience, than I have learned a lot from him. His hard work, motivation and vision guided me through the PhD process.

I am thankful to the members of my thesis committee, Prof. Mihaela van der Schaar, Prof. Patrick Thiran, and Dr. Wolfgang Kellerer for the time spent in reviewing this manuscript, and for their helpful comments on improving its content. The work presented here has been financed through the grant PP-002-68737 offered by the Swiss National Science Foundation; I am grateful for this support.

I thank all my former and actual colleagues in the LTS4 laboratory for their support and for making the life in the lab enjoyable and fun. Special thanks go to Christophe de Vleeschouver, Jakov Chakareski and Jean-Paul Wagner for the interesting research discussions and collaborations during my PhD years. My kindest thanks go to my officemate, Ivana Radulovic, who has tolerated me over the years:). I thank the ITS staff, in particular Marianne Marion and Gilles Auric for making the administrative work seem easy, while a big "thank you" in French goes to Patricia Besson and Mathieu Lemay for helping me translate the abstract of this thesis.

I would like to thank Dr. Wolfgang Kellerer and Prof. Eckehard Steinbach, for giving me the opportunity of a four month research internship in DoCoMo Euro-laboratories Munich. I am also grateful to all my new friends, made during that period, which made my stay in Munich very enjoyable.

I thank all my friends from Switzerland and abroad, for their support and for all the good times we had together. Without giving names, I would only mention the "Romanian Group", the "Pre-Doctoral gang" and all the members of ITS. Without all of them, any of these would not have been possible.

Finally, I am indebted to my family, my mother Gabriela and my father Traian for their dedication towards me, during all my life. Many thanks go to my cousins, Oana, Razvan, Adriana and Dominique for making Switzerland seem closer to home. All my gratitude goes to my brother Radu, and his family, Carla and Lara, for their unconditioned love and support.

Abstract

With the latest developments in video coding technology and fast deployment of end-user broadband internet connections, real-time media applications become increasingly interesting for both private users and businesses. However, the internet remains a best-effort service network unable to guarantee the stringent requirements of the media application, in terms of high, constant bandwidth, low packet loss rate and transmission delay. Therefore, efficient adaptation mechanisms must be derived in order to bridge the application requirements with the transport medium characteristics.

Lately, different network architectures, e.g., peer-to-peer networks, content distribution networks, parallel wireless services, emerge as potential solutions for reducing the cost of communication or infrastructure, and possibly improve the application performance. In this thesis, we start from the path diversity characteristic of these architectures, in order to build a new framework, specific for media streaming in multipath networks. Within this framework we address important issues related to an efficient streaming process, namely path selection and rate allocation, forward error correction and packet scheduling over multiple transmission paths.

First we consider a network graph between the streaming server and the client, offering multiple possible transmission paths to the media application. We are interested in finding the optimal subset of paths employed for data transmission, and the optimal rate allocation on these paths, in order to optimize a video distortion metric. Our in-depth analysis of the proposed scenario eventually leads to the derivation of three important theorems, which, in turn represent the basis for an optimal, linear time algorithm that finds the solution to our optimization problem. At the same time, we provide distributed protocols which compute the optimal solution in a distributed way, suitable for large scale network graphs, where a centralized solution is too expensive.

Next, we address the problem of forward error correction for scalable media streaming over multiple network paths. We propose various algorithms for error protection in a multipath scenario, and we assess the opportunity of in-network error correction. Our analysis stresses the advantage of being flexible in the scheduling and error correction process on multiple network paths, and emphasizes the limitations of possible real systems implementations, where application choices are limited. Finally, we observe the improvements brought by in-network processing of transmitted media flows, in the case of heterogeneous networks, when link parameters vary greatly.

Once the rate allocation and error correction issues are addressed, we discuss the packet scheduling problem over multiple transmission paths. We rely on a scalable bitstream packet model inspired from the media coding process, where media packets have different priorities and dependencies. Based on the concept of data pre-fetch, and on a strict time analysis of the transmission process, we propose fast algorithms for efficient packet scheduling over multiple paths. We ensure media graceful degradation at the client in adverse network conditions by careful load balancing among transmission paths, and by conservative scheduling which transparently absorb undetected network variations, or network estimation errors.

The final part of this thesis presents a possible system for media streaming where our proposed mechanisms and protocols can be straightforwardly implemented. We describe a wireless setup where clients can access various applications over possibly multiple wireless services. In this setup,

we solve the rate allocation problem with the final goal of maximizing the overall system performance. To this end, we propose a unifying quality metric which maps the individual performance of each application (including streaming) to a common value, later used in the optimization process. We propose a fast algorithm for computing a close to optimal solution to this problem and we show that compared to other traditional methods, we achieve a more fair performance, better adaptable to changing network environments.

Keywords: multipath networks, rate allocation, path selection, load balancing, packet scheduling, forward error correction, network variability, network inter-operability.

Résumé

Les derniers développements en codage vidéo et le déploiement rapide des connections internet client à haut débit rendent les applications vidéo en temps-réel de plus en plus attractives tant pour les usages privés que professionnels. Cependant, internet, qui est un reseau faillible, se révèle toujours incapable de garantir les conditions strictes requises par les applications vidéo, que ce soit en terme de constance pour les haut débits, de perte de paquets ou de délais de transmission. Par conséquence, des mécanismes efficaces adaptatifs doivent être mis en place afin de mettre en adèquation les caractéristiques médium et les besoins propres de l'application.

Depuis peu, différentes architectures de réseau, telles que les réseaux cleint-à-client, les réseaux de distributions de contenus, ou encore les services sans-fil parallèles apparaissent comme des moyens potentiels de réduire les coûts de communication ou d'infrastructure, ou encore d'améliorer les performance de l'application. Cette thèse exploite les caractéristiques de cheminement divers propre à ces architectures afin de développer un nouveau cadre spécifique pour la transmission vidéo en réseaux à voies multiples. Dans ce nouveau cadre, nous abordons d'importantes questions liées à l'efficacité du processus de transmission vidéo, à savoir le choix du cheminement, l'allocation de taux, la correction des erreurs, et la planification de la transmission des paquets au travers de voies multiples.

Nous considérons d'abord une reprsentation du réseau qui offre la possibilités de voies multiples entre le serveur vidéo et le client. L'intérêt est de trouver le meilleur sous-ensemble de voies utilisés pour transmettre les données ainsi que le taux d'allocation optimal correspondant, afin d'optimiser une métrique de distortion vidéo. Nous nous sommes livrés à une analyse en profondeur du scénario proposé qui a conduit à l'énoncé de trois théorèmes importants. Ces derniers forment les bases d'un algorithme linéair optimal résolvant notre problème d'optimisation. Dans le même temps, nous proposons des protocoles distribués calculant la solution optimale, adaptée au cas de réseaux grande échelle pour lesquels une solution centralisée serait trop coûteuse.

Nous abordons ensuite le problème de la correction d'erreurs pour la transmission de videos redimensionnables à travers des réseaux à voies multiples. Différents algorithmes sont proposés pour la protection contre les erreurs dans un scenario à voies multiples. L'opportunité d'une correction d'erreurs insérée au réseau est aussi établie. Notre analyse souligne l'avantage de la flexibilité dans la gestion du processus de correction d'erreurs et de la planification de transmission de paquets dans les réseaux à voies multiples. Notre analyse met en avant les limites liées à l'implémentation de systémes réels pour lesquels les choix d'application sont contraints. Finalement, nous observons les améliorations apportées par le traitement de paquets inséré aux reseaux hétérogenes charactérisés par des variations inportantes de leurs parametres.

Une fois abordées les questions d'allocation de taux et de correction d'erreurs, nous discutons du problème de la planification de transmission de paquets au travers de réseaux à voies multiples. Notre approche repose sur un modèle de paquets vidéo redimensionnable inspiré du processus de codage vidéo, pour lequel les paquets vidéo ont différents priorités et dépendances. Nous proposons un algorithm rapide de planification de transmission efficace des paquets au travers des réseaux à voies multiples, basé sur le concept de pré-apport des données, et sur une analyse temporelle stricte du processus de transmission. Une dégradation vidée lente est assurée au client dans des conditions

de réseau défavorables, en veillant à charger les voies de transmission de manière équilibrée et en assurant une planification de transmission conservatrice qui absorbe de manière transparente les variations indécelables du réseau.

Dans la dernière partie, cette thèse propose un système destiné à la transmission vidéo où les mécanismes et protocoles proposés peuvent être directement implémentés. Nous décrivons une configuration sans-fil permettant aux clients d'accéder à de multiples applications par divers services sans-fil. Dans cette configuration, le problème du taux d'allocation est résolu en visant une maximisation des performance globales du système. Pour ce faire, nous proposons une métrique de qualité unifiante qui reporte les performances individuelles de chaque application (incluant la transmission vidéo) en une valeur commune utilisée ultérieurement dans le système d'optimisation. Une solution presque optimale est trouvée par un algorithme rapide. Nous démontrons que les performances ainsi obtenues sont plus équitables que celles obtenues par diverses méthodes traditionnelles, le système s'adaptant mieux aux environnements réseau changeants.

Mots-clefs: réseaux à voies multiples, allocation de taux, sélection de cheminement, équilibrage de charge, planification de transmission de paquets, correction d'erreurs, variabilité réseau, inter-opérabilité du réseau.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Streaming over the Internet

With the advances in audio-visual encoding standards and broadband access networks, multimedia communications are becoming increasingly popular. The continuing expansion of the Internet further stimulates the demand for multimedia services and applications. Standardization bodies (e.g., ITU-T), continuously work towards achieving better media encoding standards, which facilitate a more rapid penetration of media applications in the internet community. In the same time, new networking systems and solutions, like peer-to-peer networks or wireless services inter-operability, offer the end clients support for new, thrilling internet applications.

Media streaming applications over the internet are becoming popular, as they represent a fast and real-time method for delivering the desired remote content to the end client. In the general one-way streaming scenario, as represented in Figure 1.1, a streaming server must send stored or live media to the client. The information can be pre-encoded, or encoded in real-time into a bitstream, which is transmitted over the internet to the end user/client. The client must be able to consume the received media after an initial playback delay, without suffering interruptions or severe quality degradation.

The real-time nature of the streaming applications opens some questions whose answers lie at the intersection of networking and signal processing analysis. On one hand, the internet, as a transport medium only offers a best-effort forwarding of the data packets traversing it, without guaranteeing any quality of service. Only recently, mechanisms and protocols have been derived for the implementation of traffic priority, and accommodation of real-time traffic. However, such mechanisms are denied large scale deployment over the internet, due to high implementation costs and infrastructure failures. On the other hand, the media application requires fast and timely delivery of the media data, from the content server to the end client. Its stringent quality of service requirements (e.g., high bandwidth, low delays and loses, service stability and continuity during the client play-out time) can hardly be matched today by the available transport medium.

In this thesis, we present our novel approaches and solutions to these issues. We leverage on an

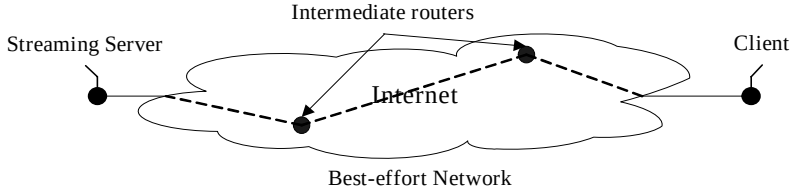


Figure 1.1: *General Media Streaming Scenario over the Internet.*

indepth analysis of the media encoding specifics and network characteristics in order to propose a new framework for media streaming applications over unreliable transport mediums. As path diversity is an inherent characteristic of the latest emerging network scenarios, (e.g., peer-to-peer networks, content distribution networks, wireless service inter-operability), we concentrate in our work on efficient streaming mechanisms for multipath networks.

1.2 Multipath Media Streaming

Peer-to-peer architectures, content distribution networks and inter-operable wireless networks are some of the latest architectures designed to either reduce the cost of the network infrastructure, enhance the application service guarantees, or increase user reachability. They rely on multiple available data transmission paths between sources and clients, in order to avoid some of the classic single path transmission scenario limitations. The benefits of these network architectures include aggregated bandwidth for resource-greedy applications, reduced latency for real-time applications, or extended network coverage for wireless users. In this context, multipath media streaming emerges as a natural research framework which offers the hope to overcome some of the lossy internet path limitations [1–3]. It allows for an increase in streaming bandwidth, by balancing the load over multiple network paths between the media server and the client. It also provides means to limit packet loss effects, when combined with error resilient streaming strategies and scalable encoding capabilities of the latest encoding standards [4–7], or reduce transmission delays.

However, this streaming framework requires extra efforts and resources for its management. Parallel route discovery and maintenance, sources coordination and efficient data scheduling, robustness in dynamic network conditions are just some of the issues that must be addresses in a successful multipath setup. Solutions to these problems have been proposed by the networking community. They usually adapt existing network algorithms and protocols to the new framework, with the final goal of optimizing the network performance. However, these solutions in general do not take into account the characteristics of the specific applications using the network infrastructure, possibly inducing a poor application performance [8].

While the streaming research community has given considerable attention to the modelling of the streaming application behavior in a multipath setup, it has mainly focused on the streaming process itself (media caching and scheduling aspects), starting from a given, fixed network scenario, failing to address the above-mentioned issues. Very little attention has been given to the idea of creating a joint application-network aware framework, optimal from the user perspective. Hence, important problems concerning the optimal construction and choice of transmission paths from a media perspective, packet error correction and scheduling on multiple paths, or streaming robustness in dynamic networks have not been thoroughly addressed so far.

In our thesis, we address the above mentioned issues from the perspective of a media streaming application. Our proposed framework for multipath media streaming offers solutions that take into account the specificity of the considered media application, along the underlying network context, in order to deliver optimal streaming performance as seen by the end client. We offer our ideas and solutions for media-aware path construction and selection, packet error correction and scheduling, and transmission robustness in multipath environments.

1.3 Problem Statement and Contributions

Efficient streaming solutions over the internet need to satisfy the stringent requirements of the media application, e.g., generally high transmission bandwidth, low packet delays, and network losses, low network variability and dynamics during medium to long periods of time, stable routes availability throughout the transmission process. However, even with the steady pace of internet expansion, and improved architectural design, the transport medium remains best-effort, incapable of offering any service guarantees to the traversing applications. Hence, adaptive techniques and algorithms must be derived in order to bridge the gap between the internet offered services and the

media application requirements, in order to improve the received media quality at the end client. In this thesis, we rely on the path diversity characteristic of the latest network architectures, in order to propose a new multipath framework for the analysis of media streaming applications. Within this framework, we offer an in-depth discussion of the most important issues concerning the envisioned streaming setup, which, in turn, allows us to derive novel mechanisms and algorithms for a more efficient streaming process. In particular, we address important issues like path selection and rate allocation, forward error correction and packet scheduling for video streaming in multipath transmission environments. We offer a theoretical analysis of the problems, we present and measure the performance of our proposed mechanisms and algorithms, and we discuss the system aspects related to possible implementations of our proposed tools in real systems.

Within a general network graph scenario, we first address the problem of **optimal path selection and rate allocation** for a media application. We define an optimization problem that relies on a media distortion metric in the optimization process. Our final goal is to select an optimal subset of transmission paths used by the application, along with the optimal rate allocation on these paths, in order to minimized the perceived media distortion at the client. Our theoretical analysis of the proposed general distortion metric finally leads to three important theorems which facilitate the choice of optimal transmission paths, and allows for the derivation of a fast path selection and rate allocation algorithm. We show that using the available network paths in increasing order of their loss probabilities is always optimal. The trade-off between adding extra bandwidth to the transmission/encoding process, hence increasing the streaming quality, and adding extra packet erasures by using network paths with higher loss probability, hence degrading the media reconstructed quality, offer a natural convergence point for our path selection algorithm. In the same time, we conclude, that, contrary to the common belief, utilizing all available network paths for media streaming is not necessarily optimal. Furthermore, we provide distributed protocols for path construction and selection in large scale network scenarios, based only on the local network information available at the client.

Next, we address the problem of **media forward error correction** in multipath networks. In a joint source-channel rate allocation framework, we investigate different FEC strategies and scheduling paradigms. Our analysis eventually leads to interesting insights on the optimal distribution of data and redundant packets over the multiple transmission paths, and our proposed algorithms compute efficient FEC rate allocation solutions in network environments with constrained resources. We show that flexible scheduling and FEC strategies can enhance the streaming process by better protecting the most important media packets, and by sending them over network paths affected by lower loss probabilities. We also asses the opportunity of in-network media flow processing in the case of active networks, where intermediate nodes can perform basic operations on the passing data flows, e.g., FEC decoding/re-encoding. We evidentiate the trade-off between transmission delays incurred due to intermediate node flow processing, and improved performance, and we show that in network scenarios with heterogeneous link parameters, such operations prove beneficial.

Media packet scheduling over multiple transmission paths is addressed next in our thesis. Based on the knowledge of media packets weights and dependencies in the bitstream, as generated by the media encoder, we propose a novel packet scheduling algorithm for efficient packet transmission over multiple network paths. Considering the total received media quality as dependent on the number and importance of the correctly received media packets, our algorithm proposes a load balancing technique over more network paths, which prioritizes the data packets that are more important for media reconstruction at the client. Furthermore, we increase the robustness of our algorithm to network variations, by a conservative timing analysis during the scheduling process. Compared to existing solutions, our approach adapts better to network rate variations, insuring a smooth quality degradation of the media in the case of adverse network conditions.

Finally, we describe a **possible real system** where our proposed mechanisms and conclusions can be applied in a straightforward manner. We envision a setup where multiple clients can access multiple data applications, including media streaming, over more available wireless services. With the help of a unifying quality metric, we map the performance of each type of applications as a function of allocated network resources. finally, we propose and solve an optimization problem

whose goal is to maximize the overall system performance. Our algorithm for network selection and rate allocation is performed iteratively in order to account for network variability and dynamics, and insures a more fair and adaptive behavior compared to other traditional methods.

Compared to previous work in streaming over multipath networks we bring the following important contributions:

- We define a general theoretical framework for the analysis of streaming media over multipath networks, in which we address several key issues of an efficient streaming system, e.g., path selection and rate allocation, forward error correction and packet scheduling;
- We provide optimization metrics based on both network characteristics and streaming sequence parameters. The metrics are later used in the defined optimization problems in order to achieve optimal solutions that maximize the end user perceived media quality;
- We address the issue of selecting an optimal subset of network paths out of an available set, and compute the optimal rate allocation on these paths, in order to optimize the client received media quality. Out theoretical analysis leads to the implementation of fast, optimal algorithms for the election of suitable transmission paths, along with their allocated transmission rate. For large scale media applications, we provide distributed algorithms for the computation of the optimal subset of end-to-end transmission paths, along with their corresponding rate allocation, based only on local views on the network, available at each intermediate node.
- We study the effect of forward error correction on multipath media streaming. We identify and compare different scheduling and FEC mechanisms for multipath error correction, and we provide fast algorithms that for the computation of suitable forward error correction strategies. We also analyze the opportunity of in-network media flow processing, by examining the advantages and disadvantages of intermediate nodes FEC operations. We identify the streaming scenarios where intermediate nodes FEC operations on the passing media flows increases the performance of the end-to-end streaming application;
- We address the problem of media packet scheduling on multipath networks. We leverage on the knowledge of the different packet weights and dependencies inside the media bitstream in order to provide fast scheduling algorithms which balance the data load over multiple transmission paths. Our algorithm achieves graceful media degradation at the client, in the case of adverse network conditions. We also study the robustness of our scheduling algorithm in the case of variable network scenarios. We provide an efficient scheduling method, based on a conservative timing analysis inside the scheduler, which transparently absorbs short-time, unpredictable network variations;
- We design a potential practical application scenario, where our proposed methods and techniques for efficient multipath streaming can straightforwardly be deployed. We envision a setup where multiple clients can access various applications over more available wireless services. In this setup, we formulate and solve an optimization problem whose final goal is to maximize the overall system's performance by a smart network selection and rate allocation for each individual client.

1.4 Road Track

We start by presenting an overview of the existing literature in multipath video streaming in Chapter 2. We discuss the relevant approaches and we position our ideas in front of these works and we emphasize the novelty brought by our approaches.

Next, we formulate our main path selection and optimization problem for media streaming over multipath networks in Chapter 3. Starting from a general video distortion model and a flow network architecture, we offer an in-depth theoretical analysis that allows us to derive some low

complexity rules guiding an efficient network resource allocation. We discuss practical implementation issues and distributed protocols for the path selection and rate allocation problem in large scale networks in Chapter 4.

We refine our distortion and network model for specific scalable streaming applications, and we discuss optimal ways for video packet protection in the face of transmission erasures, in Chapter 5. We identify a series of different FEC schemes and scheduling mechanisms that allow us to develop solutions for the optimal joint allocation of source rate channel protection rate in resource constrained multipath networks. Our approach to packet scheduling over multiple network paths is presented in Chapter 6. A careful timing analysis of the streaming process allows us to derive fast scheduling algorithms that take into account the network paths parameters along the characteristics of the encoded media stream. Furthermore, we present scheduling robustness mechanisms in front of unpredictable network variations in Chapter 7.

Finally, Chapter 8 discusses a possible practical system where our mechanisms could be deployed in a beneficial manner. We present a complete wireless system where clients can aggregate the resources of multiple wireless systems, and where streaming applications share the same transmission medium as other applications like voice conversations or data downloading. Our concluding remarks are given in Chapter 9.

Chapter 2

State of the Art

2.1 Introduction

With the advances in audio-visual encoding standards and broadband access networks, multimedia communications (MMC) is becoming quite popular. The continuing expansion of the Internet further stimulates the demand for MMC services. The existence of a multitude of clients for video streaming, e.g., Windows Media Player, Quicktime Player, or Real Player, and the increasing success of media internet services like Youtube, show the interest of the internet community for new video services and applications. However, as the transport medium for the media packets remains "best effort", these applications cannot guarantee any quality of service to their end users. Variable network rates and delays, packet loss and congestion, network reconfiguration and node dynamics are just some of the problems that must be addressed in order to provide optimal streaming services in today's internet [9]. The main concern of the streaming research community resides in providing efficient techniques and mechanisms for bridging the gap between the stringent and greedy QoS requirements of the multimedia application and the scarce available network resources. To this end, both application level (in the domain of video coding and compression), and transport and network level solutions are investigated.

In this chapter we make an overview of the efforts made by both the multimedia networking and coding communities to address the aforementioned problems. We start by presenting the networking proposals and advancements towards insuring some levels of service guarantee over the current best-effort internet. Then we present the main characteristics of the media applications (video), as resulting from the information encoding process. Finally, we address the recent works developed by the streaming community, with a special emphasis on the problems related to the issues addressed by our current work. We position our solutions in the context of previous works, and we discuss the novelty of our approaches throughout this chapter.

2.2 Networking Approach

2.2.1 Network Design and Monitoring

The networking community is spending a lot of effort in understanding and modeling the internet, with the goal of providing some necessary tools for the analysis of its performance. Based on these tools, further protocols and mechanisms can be implemented in order to go one step forward towards providing some guaranteed quality of service for the traversing applications.

A first framework for network modeling and analysis based on deterministic queuing theory is presented in [10]. The authors model the interaction between the application requirements and network services into a complete mathematical framework based on traffic flows. Within this framework, network elements are further analyzed and modeled in isolation for more accuracy [11].

Specific network modeling for real-time multimedia applications appears in [12]. The authors model the network packet loss and delay and their effects on multimedia transmission, while in [13], the authors discuss different packet loss metrics based on the behavior of the network in terms of loss burst length. Finally, [14] introduces a new network framework based on utility functions. The author discusses the trade-off between the additional benefit of allocating extra network resources to one application and the overall system performance based on a limited amount of resources.

The above-mentioned frameworks and modeling decisions can provide efficient means for application adaptation as long as they provide meaningful metrics. Values to these metrics can be obtained in real-time by efficient network monitoring, along with estimation and prediction mechanisms. An efficient method for robust monitoring of link delays and faults in IP networks is presented in [15], while the authors of [16] discuss a new and fast end-to-end bandwidth prediction protocol. Detecting shared congestion of flows via end-to-end measurements is addressed in [17], while [18] offers an example of a system for network traffic prediction. Internet path performance estimation from an application perspective is presented in [19].

2.2.2 Network Level Routing

Based on the monitoring of network metrics, efficient routing algorithms are derived in order to find suitable network paths for application data transmission. Regular optimization metrics for routing optimization refer to the number of hops to the destination, link delay, end-to-end bandwidth or loss probability. Depending on the application, one or more of these metrics are relevant in the routing process.

Numerous routing algorithms have been proposed to optimize a given network QoS metric [20]. More generally, routing with multiple metrics is the target of many works in QoS routing. But QoS routing with multiple constraints is, in general, an NP hard problem. An initial proof, for the case of at least two additive metrics is given in [21]. The authors propose heuristic algorithms for both source routing, and hop-by-hop routing, which find one path satisfying the QoS requirements of multimedia applications. Recent works in multi-constrained routing optimize a meaningful linear [22], respectively non-linear [23] relations between constraints, using low complexity algorithms.

Another way to improve the QoS of internet applications is to utilize multiple available network paths for data transmissions. Earlier efforts on using multiple transmission paths concentrate on aggregating the available bandwidths on different parallel packet routes between a server and a client. An overview of network striping techniques is presented in [24], while the authors of [25] provide a literature survey on traffic dispersion. More recently, the authors of [26] present a distance-vector algorithm for finding multiple paths, while the authors of [27] present a multipath extension of Direct Source Routing for wireless ad-hoc environments. The purpose of the algorithms is to achieve load balancing over multiple paths [28], and to simultaneously minimize delays. Algorithms for survivable networks construction are presented in [29].

Similarly, functions built on multiple path metrics are used in [30,31] to find multiple network paths for streaming. The authors of [32] discuss the problem of finding disjoint paths in single and dual link cost networks, while stability conditions for joint routing and rate control are derived in [33]. A theoretical study of loop-free conditions for multipath routing that should improve network performance is presented in [34], while [35] discusses the selection of paths for multipath network setting.

Data traffic distribution over multiple network transmission paths is optimized by solving packet scheduling and flow assignment problems. [36] presents an opportunistic traffic scheduling mechanism that works over multiple network paths, while traffic engineering for constrained multipath routing is addressed in [37,38]. Flow assignment problems have been addressed in [39] and [40]. The authors of the first paper are concerned with optimally splitting the data on multiple disjoint paths in order to avoid packet re-sequencing at the client. The second paper presents an algorithm that minimizes the end-to-end delay of data transmission while complying with an aggregated bandwidth constraint. The optimization of the network resource allocation in overlay multicast is discussed in [41,42], and packet splitting schedules for internet broadcast communications are introduced in [43].

Finally, network services can be enhanced by the active implication of some network elements in the transmission process. Adaptive buffer management, along with packet forward error technique are presented in [44, 45]. Nodes can actively participate to a more robust packet transmission in the framework of network coding [46] while new congestion control mechanisms [47] and adaptive sliding window strategies [48] offer better application quality and fairness in the network resources distribution. A survey of active network research is presented in [49]. QoS and multipath routing efforts have a direct applicability in wireless systems where the wireless medium offers the chance of nodes interconnection, or in peer-to-peer systems, when client peers connect to multiple sources in order to obtain the desired information.

2.2.3 Wireless Protocols and Advancements

As wireless technologies can offer the multipath network framework envisioned in our thesis, we discuss latest protocol advancements, especially towards interconnecting available wireless services. An overview of wireless communication and transmission principles is presented in [50], and specific 3G system specifications are detailed in [51–53]. [54–56] present mechanisms for capacity improvements to current wireless standards. Important statistics of a fading/shadowing channel for network performance analysis are analyzed in [57, 58]. The works explore the limitations of current wireless technologies, and offer possible directions of improvement.

The multipath advantage of ad-hoc wireless networks is discussed in [59]. The authors propose a cooperative packet caching and shortest multipath routing algorithm, while the authors of [60] present a slight modification to the network protocol stack in order to facilitate the connection of one WiFi wireless card to multiple home networks. Besides these service specific solutions, interworking several wireless services for multipath access is slowly emerging as a viable commercial solution in order to achieve a better end-user application quality, over unreliable wireless transmission mediums. While initial commercial products that manage multiple wireless service connectivity already exist [61], standardization efforts are paving the way towards more advanced products and services [62, 63]. The authors of [64] present handover possibilities between WLAN and cellular wireless systems and discuss the possible issues and problems. The possibility of future wireless network inter-connection for the provision of client multiple access is discussed in [65]. Also, future wireless network paradigms of trying to combine heterogeneous networks, both cellular, wireless hot spots and sensor networks are discussed in [63], while [66] discusses possible internet protocol properties for wireless services integration.

2.2.4 Positioning

While all these efforts are encouraging for the multimedia streaming community, as they offer the basis of network analysis and service guarantee provisioning, they do not explicitly address the application characteristics. Transport mechanisms are optimized mainly with the final goal of achieving better network utilization; they rely on algorithms that find the best transmission strategies given some established network metrics. While this may be optimal in terms of network utilization, it is however suboptimal from the point of view of the quality of service for the media streaming application. In 30-80% of the cases, the best paths found by classic routing algorithms are suboptimal from a media perspective [8].

In our work we derive mechanisms adapted to the specific streaming applications considered. Carefully looking at the media encoding specifics, we derive quality metrics that we later use as optimization metrics in our algorithms. Hence, we provide protocols for multipath selection and rate allocation, along with scheduling and error robustness mechanisms, starting from the needs of the streaming application, and we optimize the routing and packet scheduling accordingly. As we later show, the improvement brought by our methods for the streaming application is considerable, and justifies their use in the successful integration of media applications in future network systems.

2.3 Video Coding and Error Protection

2.3.1 Video Encoding Standards

The signal processing community is constantly directing its efforts towards creating new video encoding standards which achieve better compression of the media information, and offer a higher degree of scalability and robustness, helpful for transmission over an unreliable medium. The features of the latest video encoding standards like MPEG-4 [7] and H.264 [6] can be used by the streaming application in order to better cope with variable network conditions. Overviews of the coding principles laying at the foundation of these standards can be found in [67, 68].

Multiple works present an overview of video coding techniques that help the media application better cope with errors. Error resilient video encoding [69] and error concealment strategies at the client side [70, 71] are detailed. Error control mechanisms for video communication over the internet are presented in [72], while the specific principles behind the video redundancy coding in the H.263+ standard is presented in [73]. Further techniques for packet loss resilience based on video coding with optimal inter/intra mode switching appear in [74].

At the same time, application flexibility to network rate variations can be insured by scalable encoding of the video data. Spatial, temporal, SNR scalability, or any combination of the above, permits the application to adapt the streaming rate to the available network resources. Coarse encoding scalability can be obtained by encoding multiple video layers [75], or multiple descriptions (MDC) [76] of the same video sequence. In the case of video layers, the encoded video data is hierarchically organized into one base layer and a multitude of enhancement layers, such that each additional video layer brings a quality improvement to the previous, already decoded layers. On the other hand, MDC encoding creates multiple different, independent descriptions of the video data. Each description can be independently decoded, offering a basic reconstructed quality of the video sequence, while aggregating multiple descriptions results in improved quality. One possible technique for the creation of multiple descriptions via forward error correction is presented in [77].

Finer grained adaptation of the encoded stream to changing network conditions has been investigated as an extension of the existing, non-scalable video coding standards. In this case, the video data is encoded in one base layer, and one or more FGS layers that can be trunkated at byte level during the transmission process. The application can choose the optimal encoding rate or scale down the rate of a preexisting encoded sequence, according to network conditions, by maximizing a video quality metric [78]. The nonlinear representation of the total application quality as a function of total encoding rate is defined as a rate-distortion curve. An example of such a representation for scalable video encoding can be found in [79].

2.3.2 Error Correction in Video Streaming

While media encoding with redundancy and error robustness/concealment features at the encoder/decoder offer some protection for the application against transmission failures, further protection mechanisms can be employed for application robustness against network errors.

Network-layer error robustness strategies can be reactive or proactive [80–82]. In the case of reactive strategies, the system reacts to a discovered packet loss, usually by retransmission (ARQ). While being bandwidth efficient, such strategies incur large delays, as they require feedback from the media client to the transmission server. In the case of real-time multimedia applications, or streaming sessions where the client imposed playback delay is small, proactive strategies for error robustness are advisable, as they are much faster. Forward error correction (FEC) is the main technique to provide a more reliable packet transmission in erasure networks. FEC usually provides additional redundant packets, which are sent along the data packets to the client. As long as the client receives enough data and redundant packets, it is able to reconstruct all original data packets.

FEC strategies lower the error probability for the transmitted packets, at the expense of additional network resources. Depending on the model for network losses [83], the application can adapt the FEC strategy [84]. Such action can be modeled as a joint source channel coding op-

timization problem, whose purpose is to optimally allocate the network resources among media and redundant packets, so that the reconstructed quality of the media at the client is maximized. The authors of [85] deal with the optimal allocation of MPEG-2 encoding and media-independent forward error correction rates under the total given bandwidth. They define optimality in terms of minimum perceptual distortion given a set of video and network parameters. They compute the network error parameters after FEC decoding, and they derive the global set of equations that lead to the optimal dynamic rate allocation. A similar analysis is performed in [86]. An optimal partitioning between byte-level FEC and packet level FEC in the case of video multicast over wired and wireless networks is presented [87].

All these works consider the network conditions as known *a priori* (e.g., channel rate, probability loss rate and average burst length). They can be further extended to a more general JSCC (rate allocation) problem that takes into account intermediate active nodes or multiple existing paths between the server and the client. With this respect, intermediate peer nodes can be used by a streaming application to perform specific tasks on the passing flow in order to improve the streaming process. The authors of [88] present a multicast streaming architecture in which intermediate nodes perform FEC operations on the stream in order to better cope with packet losses on the network links. A scheme for overlay multihop FEC for video streaming over peer-to-peer networks can be found in [89].

Finally, making a distinction among the media packets that need to be protected, more advanced FEC strategies will add more redundancy for the most important packets of the stream, and less for the rest. Unequal error protection (UEP) has been proved to better utilize network resources, enhancing thus the perceived quality of the multimedia application. Network adaptive error control schemes for video streaming using hierarchical FEC are present in [90, 91].

2.3.3 Positioning

While these mechanisms offer the flexibility needed in order to cope with network channel errors and variations, their design is based on the knowledge of network parameters. Their functionality depends to some extent on the accuracy of the channel estimation, hence when these estimations are inexact, they are susceptible to failure. Intelligent scheduling on a packet level and adaptive rate allocation / error correction decisions can adapt the media streaming decisions in case of network parameter variability, and add an extra layer of flexibility in the wake of adverse network conditions (e.g., bandwidth shortage, or variable transmission delays and jitter).

In our work we present a study of different forward error correction techniques for multimedia streaming. We discuss the FEC technique in the case of scalable media streaming over multipath networks. We compare various algorithms that bring optimal results in a joint source channel coding framework, by exploiting the scalable media coding paradigm and error correction and scheduling flexibility. At the same time, we explore the trade-off between computational complexity and optimality of results, and propose simple and efficient algorithms for our optimization problem. We also explore the possible application of FEC codes in real systems where the choice of FEC modes is limited to a given set. Finally, we consider the case of in-network FEC processing, where intermediate nodes have decoding capabilities on the passing flows. We compare the end-to-end optimal FEC allocation problem, with the per-hop FEC allocation, and we identify the network scenarios where intermediate node processing of the passing flows brings a noticeable improvement for the overall streaming process.

2.4 Adaptive Video Streaming over the Internet

2.4.1 Adaptation Mechanisms

The flexibility offered by the application encoding and compression is exploited in the derivation of efficient transport and network mechanisms and protocols for media delivery. An overview of the main tendencies in network adaptive video streaming is presented in [92]. These tendencies

include robust transmission of media packets via error correction techniques, packet scheduling for optimal client received media quality and rate adaptation and path selection for the transmission process based on available network resources.

A first technique for the rate adaptation of smoothed variable bitrate video transmission is presented in [93]. The authors develop efficient techniques for transmitting video between two network nodes. They minimize the network bandwidth requirements by characterizing how the peak transmission rate varies as a function of the playback delay and the buffer allocation at the two nodes. A different approach is presented in [94], where the authors apply network calculus to obtain optimal multimedia smoothing in a deterministic framework.

Furthermore, adaptation between application requirements and network resources can be performed with the help of network elements, e.g., server, client or intermediate node buffers, or proxy use. The problem of buffer management and dimensioning in the case of parallel video servers is tackled in [95]. Using a generic buffer-pool model with worst case analysis, the author derives upper bounds on the server buffer requirements for a parallel server design with multiple disks per server. A system for proxy caching for media streaming over the internet is present in [96], while large-scale personalized video streaming systems with program insertion proxies appear in [97]. Network elements can facilitate video transmission between a server and a client, or can be helpful for in-network adaptation of the video stream, in order to match different client characteristics. The special case of video delivery from a streaming server to one or multiple clients through a proxy is presented in [98]. The authors address the problem of efficiently streaming a set of heterogeneous video streams from a remote server through a proxy to multiple asynchronous clients so that they can experience playback with low startup delays. Scalable proxy caching of video under storage constraints is also studied in [99]. The authors propose two different selective caching algorithms, appropriate for two different network scenarios, in order to increase the relevant overall performance metrics in each of the two cases.

2.4.2 Multipath Video Streaming

Wireless or peer-to-peer network systems inherently offer the media client multiple choices in terms of network streaming paths and streaming sources. The flexibility and advantages offered by multipath streaming come however at the expense of more complex mechanisms for path selection and rate allocation, packet scheduling and streaming robustness.

The benefits of multipath routing in multipath media streaming are presented in [100] and [1]. Among the main benefits of using multiple paths between a media server and a client we enumerate: (i) the reduction in correlation between packet losses, (ii) increased throughput, and (iii) ability to adjust to variations of congestion patterns on different parts of the network.

An overview of video streaming techniques for path diversity is presented in [101], while [102] discusses optimization and evaluation criterions for multimedia applications over multiple transmission paths. The authors of [103] implement and compare multipath streaming solutions at the transport and application layer. Multiple schemes are compared and the advantages and disadvantages of each one of them is presented.

Ongoing research is directed towards solving problems associated with multipath streaming scenarios, as presented in [5]. Efficient streaming mechanisms usually rely on scalable media delivery over multipath topologies. The authors of [3] address the problem of multiple description streaming over content delivery networks. They partially discuss the influence of joint and disjoint network paths between the server and the client, and offer general rules for efficient streaming. At the same time, the authors of [104] analyze a multiple path streaming scenario for the transmission of a video sequences encoded in multiple descriptions. They minimize an additive distortion metric, computed as the sum of the individual distortions of each of the independent descriptions. For complexity reasons, their analysis is reduced to a scenario comprising two encoded descriptions and two transmission paths.

Specific multipath streaming solutions for wireless WiFi networks are provided in [105–107], while the authors of [108] solve an optimization scheduling problem specific for wireless networks, using a partially observable MDP. Furthermore, multiple transmission paths can be used in cellular

wireless systems, in order to enhance media streaming applications [109]. The authors of [110] present a resource allocation framework based on service differentiation and analyze the capacity benefit achieved through service prioritization and dynamic rate adaptation.

2.4.3 Rate Allocation and Path Selection

The rate allocation and adaptation problem has been studied in simple one path streaming scenarios. The authors of [4] propose a novel rate allocation scheme to be used with FEC in order to minimize the probability of packet loss congested networks. They present a protocol suite (Transport Protocol, Loss and Bandwidth Estimation, Rate Allocation Algorithm and Packet Partition Algorithm) and compute the optimal rate allocation for the proposed distributed streaming model with FEC. Their work is later continued in [111] and [112].

Other server-driven strategies have been proposed to adapt to channel rate fluctuations. Frame discard strategies have been proposed in [113, 114]. These works address a network scenario consisting of a single path between the server and the client. When the available bandwidth is not sufficient, the streaming server finds the frames that can be discarded, in order to limit the degradation of the video quality. Branch and bound strategies for rate adaptation and packet selection have been recently proposed in [115] and [116]. The authors extend the work of [117] by providing faster algorithms for the analyzed rate-distortion optimization problem. Other packet selection algorithms for adaptive transmission of smoothed video can be found in [118] while advances in efficient resource allocation for packet-based real-time video streaming are reviewed in [119].

Furthermore, rate allocation problems in multipath network environments are addressed in the current literature. The advantage of user-level channel diversity is studied in [120] in terms of performance, fairness, robustness and cost. The authors of [121] solve the problem of finding the optimal set of network paths between the server and the client, which ensures a minimum startup delay. This work gives a detailed analysis of the multipath routing problem from the networking point of view. However, the authors do not take into account the specific characteristics of the envisioned application. The work presented in [122] addresses a similar problem of choosing the best path from a media perspective. However, the authors only address the question of path switching efficiency from the media application point of view, and do not investigate the benefits of multipath streaming.

2.4.4 Packet Scheduling in Video Streaming

Specific packet scheduling algorithms for streaming applications can serve as rate adaptation mechanisms inside the network, when nodes can decide to drop/forward the incoming packets as a function of the network status. At the same time they represent an efficient transmission tool, in the case of multipath streaming, when the scheduler decides which media packet is forwarded on each of the available paths, or a robustness mechanism against transmission errors, when the most important packets can be scheduled for transmission multiple times.

Packet scheduling decisions for multimedia streaming take into account the available network resources and the specific encoding of the media stream. Due to the predictive and scalable features of the encoder, different media packets have different weights in the reconstruction of the received bitstream. Hence, optimal scheduling strategies must take into account this feature in the transmission process. A simulation study of packet path diversity for media transport over the internet can be found in [123], while an optimal packet scheduling mechanism for multiple description coded video over lossy networks is presented in [124].

Informed scheduling decisions optimize the received media quality under network resources constraints. Specific scheduling algorithms for multimedia traffic either model the available network channel in a stochastic way, or rely on network information provided by estimation mechanisms. In the Rate-Distortion framework (RaDiO) presented in [117], the scheduling algorithm takes an optimal decision (transmission policy) for each media packet/set of packets, based on the stochastic parameters of the channel model. The optimal scheduling solution comes at the expense of

complex computations and large delays [125], [126]. More recent RaDiO works address the packet scheduling problem in distributed setups, where intermediate nodes can take independent decisions on packet droping/forwarding [127]. The framework can be extended to streaming scenarios with multiple available transmission paths [128], an example of which being ad-hoc wireless scenarios [129, 130]. Robustness to model inaccuracies can be obtained by repeated transmissions of the most important packets in the bitstream. More recently, work has been directed towards finding efficient video packet scheduling models in the RaDiO framework for multipath transmissions. Media packet scheduling with path diversity or server diversity is addressed in [131, 132]. Their sender-driven scheme enables the client to decide which packet to request at which instance of time and on which path/from which server, on a rate-distortion optimized way. The model is applied to other particular streaming setups in [133, 134].

On the other hand, [135, 136] base the packet scheduling decisions on prior information about the network obtained from network estimation algorithms. The multipath EDPF algorithm from [135] solves the packet scheduling problem by computing the earliest delivery time for each packet, on each of the available network paths. By sending each packet on the path that ensures the earliest delivery at the client, the authors minimize the packet reordering cost. Later, the same authors improve their algorithm with a selective frame discard strategy that drops less important frames in case the channel bandwidth is smaller than the encoded video rate [137]. While these algorithms are less complex and perform faster, they are vulnerable to channel prediction errors. Previous works [138], [139] enhance the robustness to channel prediction errors, by designing a new scheduling model, in which the packets/frames in a bitstream are rearranged. The most important parts of the bitstream are advanced ahead of the less important ones, so that they are scheduled for transmission with higher priority. [140] presents a delay-optimized robust transmission scheme for images, over multiple channels. Such mechanisms increase the probability of successful transmission of information necessary for correct decoding, however, they come at the expense of extra delays and occupied buffer space.

2.4.5 Wireless Streaming and Cross-layer Design

Wireless systems, because of their parallel presence and inter-operability possibilities represent a future platform for multipath streaming applications. The overview work of [141] gives a complete presentation of potential streaming systems in wireless networks and discusses the standardization efforts. Recent advances in wireless media delivery are presented in [142], while specific streaming applications for WiFi networks are discussed in [143]. The authors describe the general issues involved in integrating multiple description coding with layered video coding within a wireless multipath network environment and they compare the performance of the two encoding techniques under different path conditions. At the same time, efficient techniques for streaming over wireless networks that offer some QoS guarantees (e.g., UMTS networks [144]) are presented in [145]. Here, channel efficiency is improved by using the common UMTS channel for streaming, along with proactive hybrid ARQ protocols.

The cross layer design (CLD) paradigm emerged lately as a more efficient way to optimize the performance of multimedia applications over unreliable networks. It involves the communication and cooperation between the standard network layers in order to take informed application transmission decisions. To this end, the optimizer relies on the knowledge of system parameters from different layers of the network architecture when taking the optimal decision.

The authors of [146, 147] address the issue of cross-layer networking, where the physical and MAC layer knowledge of the wireless medium is shared with higher layers, in order to provide efficient methods of allocating network resources and applications over the internet. They provide an overview of the main challenges in matching the instantaneous radio channel conditions and capacity needs with the traffic and congestion conditions found over the packet-based world of the Internet. Relevant technical challenges of cross-layer design with a focus on video streaming over wireless networks are also present in [148]. They also address the impact the cross layer optimization strategy deployed at one client has on the multimedia performance of other stations.

A main interest of cross-layer design techniques is to adapt the streaming application parameters based on information taken for the wireless medium. A dynamic OFDMA-FDMA transmission system delivering MPEG4 video streams is presented in [149]. The authors of [147] propose a joint optimization of the application layer together with the data-link and physical layer of the protocol stack, using an application oriented objective function in order to maximize user satisfaction in Hyperlan systems. IEEE 802.11 based networks are discussed in [150]. The authors evaluate different error control and adaptation mechanisms available for robust video transmission, in different layers of the network architecture. Finally, the authors of [151] propose a cross layer design for the real time streaming of prerecorded video with prefetching to clients in wireless CDMA networks, while [152] address the same problem in UMTS systems.

While some of the required parameters from the different network layers do not have a direct meaning or equivalent in other layers, it is crucial for an effective system to construct realistic abstractions of these parameters. [153, 154] present a possible architecture for video delivery in a multi-user wireless environment, based on parameter abstraction at the physical, data link and application layer. Similar systems are presented in [155,156], while [157] uses the cross layer design paradigm in the context of multi-user, multi-application wireless networks. Finally more system or prototyping issues are raised in [158, 159].

A cautionary perspective on cross-layer design is offered in [160]. The authors contend that a good architectural design leads to proliferation and longevity, and explain this by means of examples. They also evidentiate the risk of unintended cross-layer interactions, with undesirable consequences on overall system performance, in the case of cross layer optimization.

2.4.6 Applications and Systems of Multipath Streaming

Depending on the envisioned application setup, more streaming scenarios can be considered. One-to-one network scenarios refer to the case of a single stream transmission between a server and a client. In one-to-many network scenarios more clients want to have access to the same content, leading to multicast systems as presented in [161] or [162], or tree-based peer-to-peer networks [163, 164]. Many-to-one and many-to-many scenarios refer to larger setups where one or more clients have access to different sources. Prominent examples of such scenarios are Content Distribution Networks (CDN) [165], large-scale peer-to-peer networks (multiple trees or mesh architectures), and large-scale multimedia streaming deployment architectures [166]. Finally, the network transport medium should be considered in all these scenarios. Special mechanisms are derived for the wireless networks, according to the characteristics of this medium [2, 167].

Wireless streaming scenarios and peer-to-peer streaming applications are two of the most prominent examples of application delivery setups with an inherent multipath topology. Peer-to-peer systems take advantage of the specific network architecture in order to offer cheap and robust transmission of application packets. An overview of design choices when creating a new peer-to-peer system (mesh, tree or multiple trees architectures) is presented in [168], while systems for multi-point to point communications are discussed in [169]. Latest advances in peer-to-peer technology and systems seem to spark the attention of the multimedia streaming community towards developing rich media streaming solutions on such distributed platforms. The latest success of the Skype [170] and BitTorrent [171] protocols demonstrate that such systems can provide sufficient average bandwidth for video streaming applications and are suitable for real time communication. Probably the first notable example of distributed video streaming is presented in [172]. The authors consider the cooperation between clients accessing a resource, in order to alleviate the load on the server. The work is set in the context of a traditional client-server framework, but relies on peer cooperation to distribute content, instead of dedicated servers that are geographically deployed (e.g., Content Distribution Networks [3]). Peer-to-peer systems like the ones proposed in [173–176] already propose basic multimedia streaming solutions.

Techniques for the optimization of multipath wireless ad-hoc streaming applications are discussed in [177]. Multi-stream coding, combined with multipath transmission, has been presented in [178] as a solution to fight against network errors in an ad-hoc network environment. Other works in distributed video streaming [179–181] deal with resource allocation and scheduling on

multiple, a priori chosen streaming paths, with the final goal of minimizing the overall distortion perceived by the media clients. All these works rely on a given set of transmission paths, and try to optimally exploit these network resources. However, none of them specifically targets the optimal choice of the streaming paths and the corresponding rate allocation problem.

2.4.7 Positioning

While a lot of works concentrate on the media streaming domain as presented above, we consider that numerous issues still remain unsolved or just partially addressed. We focus on a one-to-one streaming scenario defined by the transmission of a scalable encoded media sequence over a best effort network comprising multiple available paths between the server and the client. While this framework is promising in terms of future media delivery applications, considering the emerging network architectures, it also poses specific problems not thoroughly investigated yet. For example the joint optimization of application source rate, transmission path choice and path rate allocation remains unaddressed. In this thesis, we propose a mathematical model for analyzing this problem. Our analysis leads to the derivation of general rules and algorithms for efficient streaming in both centralized or distributed network scenarios.

At the same time, we shift the focus of our proposed solutions from the traditional optimization of network metrics towards application-oriented quality metrics. Our joint consideration of network resources and constraints on one side, and application-specific requirements on the other side, gives us new leverage during the transmission decision process. From this point of view, in our work we go one step beyond the state of the art solutions, in order to provide more efficient media scheduling solutions, with increased robustness against network shortages. Our methods generally guarantee smoother quality variations at the client compared to previous methods, while still being simple and requiring limited computational resources.

Finally, we provide a possible application scenario where media applications are integrated in a general service network. By fully exploiting the scalability properties of the latest media encoding standards, along with new application-oriented optimization metrics, we achieve better and more fair client perceived results.

Chapter 3

Media Flow Rate Allocation in Multipath Networks

3.1 Introduction

In this chapter, we address the problem of joint path selection and source rate allocation in order to optimize the media specific quality of service when streaming stored video sequences on multipath networks. An optimization problem is proposed in order to minimize the end-to-end distortion, which depends on video sequence dependent parameters, and network properties. An in-depth analysis of the media distortion characteristics allows us to define a low complexity algorithm for an optimal flow rate allocation in multipath network scenarios. In particular, we show that a greedy allocation of rate along paths with increasing error probability leads to an optimal solution. We argue that a network path shall not be chosen for transmission, unless all other available paths with lower error probability have been chosen. Moreover, the chosen paths should be used at their maximum available end-to-end bandwidth. Simulation results show that the optimal flow rate allocation carefully adapts the total streaming rate and the number of chosen paths to the end-to-end transmission error probability. In many scenarios, the optimal rate allocation provides more than 20% improvement in received video quality, compared to heuristic-based algorithms. This motivates its use in multipath networks, where it optimizes media specific quality of service, and simultaneously saves network resources at the price of a very low computational complexity.

The main contributions brought in this chapter can be briefly summarized as follows:

- We propose a general framework for streaming of pre-encoded media data in multipath networks, which encompasses network and media aware metrics;
- We perform the first theoretical media flow analysis on the optimality of number, and choice of network paths, in terms of end-to-end Quality of Service;
- We provide a linear time media aware routing algorithm that outputs the optimal set of network paths to be used in streaming pre-encoded video sequences, along with the corresponding flow rate distribution.

The chapter is organized as follows: Section 3.2 presents the streaming framework and formulates our optimization problem. The theoretical analysis of the streaming process is developed in Section 3.3 and Section 3.4 presents the routing algorithm. We discuss practical implementation scenarios and limitations in Section 3.5 and present our main results in Section 3.6. Finally we conclude in Section 3.7.

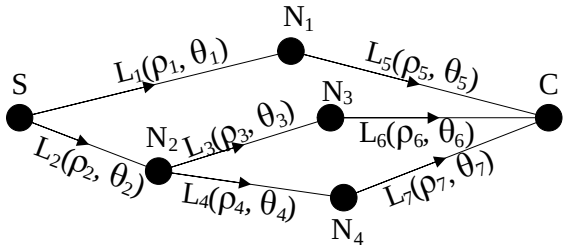


FIGURE 3.1: *Multipath Network Scenario.*

3.2 Distortion Optimized Multipath Media Streaming

3.2.1 Multipath Network Model

We consider a framework where the media streaming application uses a multipath network, which can be represented as follows. The available network between a media server S and a client C is modeled as a fully connected directed acyclic graph $G(V, E)$, where $V = \{N_i\}$ is the set of nodes in the network, and E is the set of links or segments (see Figure 4.11). Each link $L_u = (N_i, N_j) \in E$ connecting nodes N_i and N_j has two associated positive metrics:

- the available bandwidth $\rho_u > 0$ expressed in some appropriate unit (e.g., kbps), and,
- the average loss probability $\theta_u \in [0, 1]$, assumed to be independent of the streaming rate.

Let $\mathcal{P} = \{P_1, ..., P_n\}$ denote the set of available loop-free paths between the server S and the client C in G, with n the total number of non-identical end-to-end paths. A path $P_i = (S, N_i, N_j, ..., C)$ is defined as an ordered list of nodes and their connecting links, such that no node appears more than once, and that each link L_u between two consecutive nodes in the path belongs to the set of segments E. Let further b_i and p_i denote respectively the end-to-end bandwidth and loss probability of path P_i. We define the bandwidth of an individual path P_i as the minimum of the bandwidths among all links on the path (i.e., the "bottleneck bandwidth"). Hence, we have

$$b_i = \min_{L_u \in P_i} (\rho_u) \ . \tag{3.1}$$

Under the commonly accepted assumption that the loss process is independent on two consecutive network segments, and identically distributed on two or more flows traversing the same segment, the end-to-end loss probability on path P_i becomes a multiplicative function of the individual loss probabilities of all segments composing the path. It can be written as:

$$p_i = 1 - \prod_{L_u \in P_i} (1 - \theta_u) \ . \tag{3.2}$$

Finally, the media application sends data at rate r_i on path P_i, with a cost c_i. The cost represents the price to be paid by the streaming application, for using path P_i. As, in general, the underlying transport medium should be transparent for the application, we define the cost function as dependent only on the total flow rate r_i sent by the application on path P_i. A linear cost relation is simply expressed as follows :

$$c_i = \begin{cases} k \cdot r_i & \text{if } P_i \text{ is used, with } r_i \leq b_i \\ 0 & \text{if } P_i \text{ is not used} \end{cases} \ , \tag{3.3}$$

where k is a constant (i.e., the cost factor is identical for any path $P_i \in \mathcal{P}$). In this network model, efficient streaming strategies have to carefully allocate the rate between the different network paths. The goal of the next sections is to get the best out of the multipath network, both in terms of cost, and from a media-driven quality of service perspective.

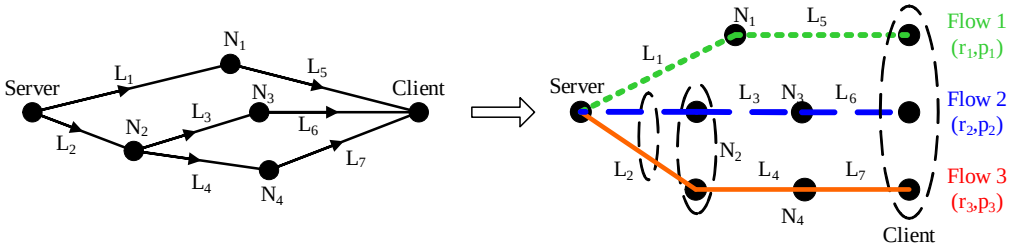


FIGURE 3.2: *Equivalent transformation between a network graph and a tree of paths between the server and the client.*

3.2.2 From Network Graph to Flow Tree

In order to study the flow rate allocation problem in multipath networks, we use a flow tree representation of the network graph G. The media server becomes the root of the tree, and each flow $\mathcal{F}_i$ represents the share of the overall media stream, which is sent on a network path P_i. The media stream is the composition of individual media flows, and the client is represented as a set of leaf nodes, with one leaf per flow. Note that several methods in graph theory have been proposed for constructing such trees, and we rather concentrate in our work on the rate allocation problem, among the branches of the tree. In this case, the rate allocation becomes a flow assignment problem.

Considering that there is (at most) one flow for each network path P_i, we can transform the original network graph G into a flow tree by duplicating any network edge and vertex that is shared by more than one network path, as represented in Figure 3.2. Since the transformation from paths to flows is bijective, each flow is characterized by a maximal end-to-end streaming rate, and an end-to-end loss probability, as computed in Section 3.2.1. The flow $\mathcal{F}_i$ on path P_i uses a streaming rate $r_i \leq b_i$, with a loss probability p_i, and a cost $c_i = k \cdot r_i$.

Due to the assumption of rate independent loss process, any two flows in the tree are independent in terms of loss probability. However, flows may be dependent in terms of aggregated bandwidth, since they may share joint bottleneck links. The flow tree representation allows us to explicit the constraints imposed on a valid rate allocation. These constraints are imposed by bandwidth limitation on the network links, and flow conservation in the network nodes. The necessary and sufficient conditions for the flow tree model to be a valid representation of the original network graph can finally be grouped into single flow, and multiple flow constraints and expressed as:

1. Single Flow Constraints:

   - path bandwidth limitations: $r_i \leq b_i, \forall P_i \in \mathcal{P}$;
   - flow conservation at intermediate nodes: for every node $N_j \in P_i$, $r_i^{in} = r_i^{out} = r_i$, where r_i^{in} and r_i^{out} are the incoming and respectively outgoing rates of $\mathcal{F}_i$ passing through node N_j.

2. Multiple Flow Constraints:

   - link bandwidth limitations:
     $\sum_{P_i : L_u \in P_i} r_i \leq \rho_u, \forall L_u \in E$;
   - flow conservation at intermediate nodes: for every node $N_j \in V$:
     $\sum_{P_i} r_i^{in} = \sum_{P_i} r_i^{out} = \sum_{P_i} r_i, \forall P_i : N_j \in P_i$.

While the transformation between the network graph and the flow tree can be made for any type of graph, the choice of transmission paths in the flow tree may affect the total available resources of the network. Let path P_i be occupied by flow $\mathcal{F}_i$ characterized by its rate r_i, and let G' be the residual graph, after isolating flow $\mathcal{F}_i$. We define $f = maxflow(G(V,E))$ as the maximum flow rate sustained by the network graph G. For general network graphs the following relation is always true:

$$f \geq r_i + f', \tag{3.4}$$

where f' is the maximum flow of the residual graph G'.

We identify a special category of network graphs for which the previous relation always yields an equality, independent of our choice of $\mathcal{F}_i$. We call this graphs as **flow-equivalent graphs**. Flow-equivalent graphs contain every possible network graph that exhibits a single joint network segment, or multiple joint network segments belonging to independent network subgraphs. More general network graphs may also belong to the category of flow-equivalent graphs, depending on the network segment parameters. As flow-equivalent graphs map most common streaming scenarios and offer a simplified analysis of our optimization problem, they will be used in the rest of this chapter.

3.2.3 Media-Driven Quality of Service

The end-to-end distortion, as perceived by the media client, can generally be computed as the sum of the source distortion and the channel distortion. In other words, the quality depends on both the distortion due to a lossy encoding of the media information, and the distortion due to losses experienced in the network. The source distortion D_S is mostly driven by the source or streaming rate R and the media sequence content, whose characteristics influence the performance of the encoder (e.g., for the same bit rate, the more complex the sequence, the lower the quality). The source distortion decays with increasing encoding rate; the decay is quite steep for low bit rate values, but it becomes very slow at high bit rate. The channel distortion D_L is dependent on the average loss probability π, and the sequence characteristics. It is roughly proportional to the number of video entities (e.g., frames) that cannot be decoded correctly, and an increase in loss probability augments the channel distortion D_L. Overall, the end-to-end distortion can thus be written as:

$$D = D_S + D_L = f(R, \pi, \Gamma) \ , \tag{3.5}$$

where Γ represents the set of parameters that describe the media sequence. This generic distortion model is quite commonly accepted, as it can accommodate a variety of streaming scenarios. For example, when error correction is available, the total streaming rate has to be split between the video source rate that drives the source distortion D_S and the channel rate, which directly influences the video loss rate π [85].

The total streaming rate R, and the end-to-end loss probability π directly depend on the path selection and the flow rate allocation. In the multipath scenario described before, the media application uses rate allocation $\vec{R} = [r_1, ...r_n]$, where the flow rate r_i, with $0 \leq r_i \leq b_i$, represents the streaming rate on path $P_i \in \mathcal{P}$. The total media streaming rate R is expressed as:

$$R = \sum_{i=1}^{n} r_i \leq \sum_{i=1}^{n} b_i \ . \tag{3.6}$$

The overall loss probability π experienced by the media application can be computed as the average of the loss probabilities of the n paths:

$$\pi = \frac{\sum_{i=1}^{n} p_i \cdot r_i}{\sum_{i=1}^{n} r_i} \ . \tag{3.7}$$

The average end-to-end distortion model is a simple and general approximation, suitable for most common streaming strategies where the number of packets per frame is independent of the encoding rate. Note that the actual video loss process is likely to present a low correlation, due to the usage of multiple paths. Under the given network assumptions, the video distortion metric becomes quite insensitive to the actual link error model, and is mostly influenced by the average loss probability on the given network segment.

It is important to note that increasing R with the addition of a path reduces the source distortion. However, the addition of a path generally impacts the loss probability π, and may augment the channel distortion. The optimal flow rate allocation therefore results from a trade-off between increasing the streaming rate, and controlling the end-to-end loss probability. Finally, since paths may not be completely disjoint, $\vec{R}$ is a valid rate allocation on the network graph G, if and only if G can simultaneously accommodate the flow rates on all paths in $\mathcal{P}$. A necessary condition for the equality in the right-hand side of Eq. (3.6) to be verified requires that all bottleneck links of the n streaming paths are disjoint. Sufficient conditions for valid rate allocation are analyzed in the next section.

3.2.4 Multipath Rate Allocation: Problem Formulation

We consider the problem of the optimal routing and rate allocation strategy, for a given video stream that can be split into flows sent on different network paths between the streaming server, and the media client. The rate constraints are directly given by the network status, as shown before, and the overall streaming rate can be adapted by simple operations at the server (e.g., packet filtering). We can formulate the optimal multipath rate allocation problem as follows.

Given a network graph G, the optimization problem consists in jointly finding the optimal sending rate for a video packet stream, along with the optimal subset of network paths to be used for transmission, such that the end-to-end distortion is minimized. Equivalently, using the flow tree representation of the network graph proposed in Section 3.2.2, the optimization problem translates into finding the optimal rate allocation for each of the flows in the tree, such that the video distortion is minimized. It can be formulated as follows:

Multimedia Rate Allocation Problem (MMR): Given the network graph G, the number of different paths or flows n, the video sequence characteristics (Γ), and the total streaming budget Q, find the optimal rate allocation $\vec{R^*} = [r_1, ...r_n]^*$ that minimizes the distortion metric D:

$$\begin{aligned} \vec{R^*} &= \arg\min_{\vec{R}} D(r_1, ...r_n) \\ &= \arg\min_{\vec{R}} f(R, \pi, \Gamma) \end{aligned} \tag{3.8}$$

where $R = \sum_{i=1}^{n} r_i$ and $\pi = \frac{\sum_{i=1}^{n} p_i \cdot r_i}{\sum_{i=1}^{n} r_i}$, under the following constraints:

1. Budget Constraints: $\sum_{i=1}^{n} c_i \leq Q$;
2. Single Flow Constraints;
3. Multiple Flow Constraints.

In the next section, we present a detailed analysis of a typical distortion model for video sequences. While the non-convexity of the optimization metric does not permit an easy solution by integration of the constraints into a Lagrangian formulation, our analysis eventually allows us to define a simple algorithm, able to find the optimal rate allocation for flow-equivalent graphs, with linear time complexity.

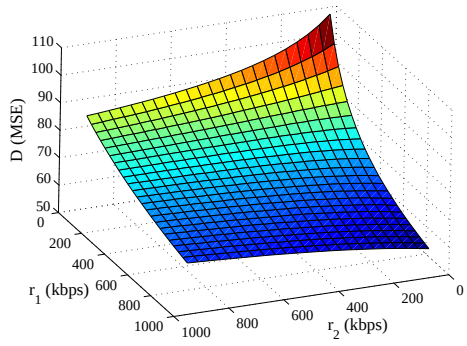


FIGURE 3.3: *Overall distortion measure for two network paths in function of available rates, $\alpha = 1.76 \cdot 10^5$, $\xi = -0.658$, $\beta = 1750$, $p_1 = 0.02$, $p_2 = 0.04$.*

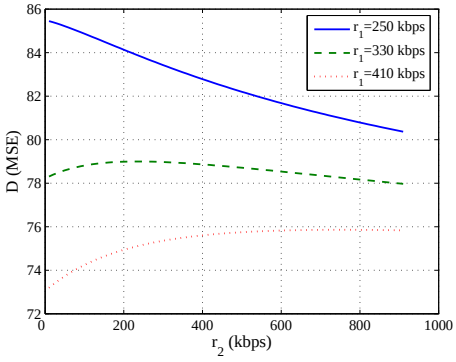


FIGURE 3.4: *Overall distortion behavior as a function of r_2, for various fixed values of r_1.*

3.3 Flow Rate Allocation Analysis

3.3.1 End-to-end Distortion Model

We introduce in this section a quite generic distortion model, which is able to capture the influence of the average encoding rate on the source distortion, as well as the impact of losses on the channel distortion. Recall that our objective is to find the best flow rate allocation on a multipath network with known average statistics. Hence, we are looking for an average distortion model that is able to estimate the video quality of service in a stationary regime.

In low to medium bit rate video streaming, it is commonly accepted that the source distortion is a decaying exponential function on the encoding rate, while the channel distortion is proportional to the number of lost packets (i.e., the packet loss probability, when the number of packet per frame is independent of the bit rate) [182]. Hence, we can explicitly formulate the Mean-Square Error distortion metric as:

$$D = \alpha \cdot R^{\xi} + \beta \cdot \pi \tag{3.9}$$

where α, $\beta \in \Re^+$ and $-1 \leq \xi \leq 0$ are parameters that depend on the video sequence. This distortion model is a simple and general approximation that follows closely the behavior of more sophisticated distortion measures, such as those proposed in [183–185]. Since it is suitable for most common streaming strategies where the number of packets per frame is independent of the encoding rate, we use the model of Eq. (3.9) in the remainder of this chapter. It can be noted that our simple model does not take into account the exact characteristics of the loss process, and that it mostly captures the effect of independent losses. We assume that bursts of losses on the video packet stream are quite unlikely due to the partitioning in multiple flows. Simple interleaving can also be applied to reduce the effects of bursts, if delay permits it. Finally, we should stress out that bursts of video packets losses are in general less penalizing for the channel distortion [83], so that our model has the advantage to provide a worst case estimate of the end-to-end distortion.

Before going deeper in the analysis of flow rate allocation, we propose a simple example to illustrate the behavior of the end-to-end video distortion in a multipath scenario. We consider a basic network scenario consisting of two disjoint network paths, P_1 and P_2, with bandwidth $b_1 = b_2 = 1000kbps$, and loss probabilities $p_1 = 2\%$ and $p_2 = 4\%$, respectively. Consider two independent flows $\mathcal{F}_1$ and $\mathcal{F}_2$ composing the same video stream, and traversing the two network paths with streaming rates $r_1 \leq b_1$, and $r_2 \leq b_2$. The evolution of the distortion function given in Eq. (3.9) is presented in Figure 3.3, for a test video sequence (i.e., Foreman CIF).

As expected, we observe that the decrease in distortion is larger if we increase the rate of flow

$\mathcal{F}_1$, than if we equivalently increase the rate of flow $\mathcal{F}_2$. This behavior is due to the lower loss probability that affects the path followed by the flow $\mathcal{F}_1$. At the same time, we observe that the distortion metric is always decreasing with the increase of r_1, hence it is optimal to fully utilize the bandwidth of the path with the smallest loss probability. In this case, for a given packet loss rate, it is better to increase the quality of each video frame by augmenting the rate r_1, as expected.

More interestingly, Figure 3.4 shows that the behavior of the distortion as a function of the rate r_2, depends on the value of the rate r_1. For high values of r_1, the distortion can even increase with growing rate r_2. Beyond a given value of the streaming rate on the most reliable network path, adding an extra flow can degrade the end-to-end quality of the media application since the packet loss rate increases. In this case, the negative influence of the error process on the second network path is greater than the improvement brought by additional streaming rate. Such a behavior is the key to explain why using all the paths to their full bandwidth does not necessarily result in an efficient strategy when streaming video data. Finally, the same type of behavior can be observed for stored video packet streams that are built on video packets and error control packets (e.g., Forward Error Correction). In this case, the sensitivity of the channel distortion is obviously lower for low error rates, but rapidly increases when the channel protection becomes insufficient.

3.3.2 Maximum or Null Flows

We now generalize the previous observations, and derive theorems that guide the design of an optimal rate allocation strategy for a given video packet stream in a flow equivalent network. This section shows that, in the optimal rate allocation, a flow is either used at its full bandwidth, or not used at all. Furthermore, the optimal rate allocation always chooses the lowest loss probability paths, i.e., a path shall not be selected, unless all other paths with a lower loss probability have been picked before. We start from an ideal streaming scenario with unlimited budget and disjoint network paths, and eventually add budget and flow constraints, which are however shown not to affect the initial findings.

Assume that the n disjoint network paths are represented into a tree of flows as explained in Section 3.2.2. Without loss of generality, we further assume that flows $\mathcal{F}_i$ with $1 \leq i \leq n$, are arranged in increasing order of the loss probability, i.e., $p_1 < p_2 < ... < p_n$. We note that, from the distortion metric point of view, any two flows $\mathcal{F}_i$ and $\mathcal{F}_j$, with rates r_i and r_j and traversing paths P_i and P_j with the same loss probability $p_i = p_j$, can be observed as a single flow affected by the same loss probability p_i, and having an aggregated rate $r_i + r_j$. Under these generic settings, we first claim that the optimal rate allocation either uses a network path to its full bandwidth, or does not use it at all.

Theorem 3.3.1 (On-Off Flows)**.** *Given a flow tree with independent flows $\mathcal{F}_i$ having rates $r_i \in [0, b_i]$ and a distortion metric as defined in Eq. (3.9), the optimal solution of the MMR problem when all the paths are disjoint, lies at the margins of the value intervals for all r_i. In other words, the optimal value of r_i is either 0 or b_i, $\forall i : 1 \leq i \leq n$.*

Proof. Deriving the distortion D given in Eq. (3.9) with respect to the rate r_i, $\forall i : 1 \leq i \leq n$, we obtain:

$$\begin{aligned} \frac{\partial D(r_1, ...r_n)}{\partial r_i} &= \alpha\xi(\sum r_i)^{\xi-1} + \beta \cdot \frac{p_i \sum r_j - \sum p_j r_j}{(\sum r_i)^2} \\ &= \alpha\xi(\sum r_i)^{\xi-1} + \beta \cdot \frac{\sum_j r_j \cdot (p_i - p_j)}{(\sum r_i)^2} \end{aligned}$$

Observe that the condition for an extremum, $\frac{\partial D(r_1,...r_n)}{\partial r_i} = 0$ for any r_i, implies:

$$\alpha \cdot \xi \cdot (r_i + \lambda)^{\xi+1} + \beta \cdot \mu = 0$$

where λ and μ stay constant in our proceeding. Since $0 \leq \xi + 1 \leq 1$, the equation has a single finite solution:

$$r_i^* = \sqrt[\xi+1]{\frac{\beta \cdot \mu}{-\alpha \cdot \xi}} - \lambda$$

At the same time, the derivative in any point $r_i < r_i^*$ is positive, while to the right of the optimal value, it is negative (since $\xi < 0$, and all other terms are positive). Hence r_i^* is a point of local maximum for the distortion function D, which means that only values at the margins of the value interval for r_i can minimize the objective function[1]. □

It can be further observed that, in the case of r_1, it holds that $\frac{\partial D}{\partial r_1} < 0$, for any positive value of r_1 (since $\xi < 0$, $\alpha, \beta > 0$ and $p_1 - p_j < 0$, $\forall j : 2 \leq j \leq n$). Hence the value $r_1 = b_1$ always minimizes the objective function, and is part of the optimal solution.

Corollary 3.3.1. *Given a flow tree with independent flows $\mathcal{F}_i$ having rates $r_i \in [0, b_i]$ and a distortion metric as defined in Eq. (3.9), the optimal solution of the MMR problem when all paths are disjoint, allocates $r_1 = b_1$, where the path P_1 is the path with the lowest loss probability.*

Theorem 3.3.1 greatly reduces the search space for an optimal solution to the MMR optimization problem. Hence we can rewrite the optimal streaming solution as a vector Φ of boolean values ϕ_i for each flow $\mathcal{F}_i$, where $\phi_i = 1$ means that path P_i is used with full rate $r_i = b_i$, and $\phi_i = 0$ denotes the fact that the path P_i is not used by the streaming application. The previous corollary further says that $\Phi = [\phi_1 = 1, \phi_2, ..., \phi_n]$ is part of the optimal solution.

For bounded intervals for all rates r_i, 2^{n-1} computations are sufficient for finding the optimal solution vector. For practical scenarios, with a limited number of available network paths between a server and a client, this number of computations is in general quite low. We can however further constrain the search space by considering that the optimal rate allocation always uses first the network paths with the smallest loss probabilities.

Theorem 3.3.2 (Parameter Decoupling)**.** *Given a flow tree with independent, disjoint flows $\mathcal{F}_i$ having rates $r_i \in [0, b_i]$ and a distortion metric as defined in Eq. (3.9), the structure of the optimal rate allocation is $\Phi^* = [1, 1, ..., 1, 0, 0, ...0]$.*

Proof. We prove the result by induction. Recall that the network paths/flows are arranged in increasing order of their loss probabilities p_i. We have already seen that $\Phi = [\phi_1 = 1, \phi_2, ..., \phi_n]$ is part of the optimal solution. Next we show that, for $n \geq 3$, $\Phi = [\phi_1 = 1, \phi_2 = 0, \phi_3 = 1, \phi_4, ..., \phi_n]$ cannot be part of the optimal solution.

For the sake of clarity, let us remove ϕ_i's with $i > 3$ from the notation, since they stay constant in our proof. By contradiction, assume that Φ is part of the optimal solution. It means that $D(b_1, 0, b_3) < D(b_1, 0, 0)$. Since the paths are ordered with increasing values of the loss probabilities and considered to be disjoint, we can always transfer part of the rate from $\mathcal{F}_3$ to $\mathcal{F}_2$, and improve the distortion. Let $r_2 = \min(b_2, b_3)$, and $r_3 = [b_3 - b_2]^+$. We have:

$$D(b_1, r_2, r_3) < D(b_1, 0, b_3) < D(b_1, 0, 0)$$

The first inequality comes from the definition of the distortion metric, the second one from the assumption that Φ is part of the optimal solution. We can further distinguish two cases:

- $b_2 \leq b_3$. Then, $r_2 = b_2$, and $r_3 \geq 0$. According to Theorem 3.3.1, there exists a solution $D(b_1, b_2, b_3 \cdot \phi_3^*) < D(b_1, b_2, r_3) < D(b_1, 0, b_3)$, with $\phi_3^* \in \{0, 1\}$. Φ cannot be part of the optimal solution since $\phi_2^* = 1$, which contradicts our assumption.

- $b_2 > b_3$. Then, $r_2 = b_3$ and $r_3 = 0$, and we have $D(b_1, b_3, 0) < D(b_1, 0, b_3) < D(b_1, 0, 0)$. From Theorem 3.3.1, there exists an even better solution where $r_2 = b_2$, leading to $\Phi^* = [110]$, which again contradicts our assumption.

[1] Since r_i^* is the only finite solution, this statement is valid even if r_i^* is not contained in $[0, b_i]$.

Next, we prove that $\Phi = [1...1, 0...0, 1...1, \phi_m, ..., \phi_n]$ cannot be part of the optimal solution. In other words, we prove that the optimal rate allocation Φ^* can only be a series of consecutive 1's, followed by a series of consecutive 0's. Let $\phi_j = 0$ and $\phi_k = 0$, with $j < m$, $k < m$, be the start and end of the series of consecutive 0's in Φ. Following the same reasoning as before, transferring rate from flows $\mathcal{F}_i$, with $k + 1 < i < m - 1$, to $\mathcal{F}_j$ can only improve the overall distortion. If $b_j \leq \sum_{i=k+1}^{m-1} b_i$, it directly leads to a solution with $\phi_j = 1$ that is better than Φ. Otherwise, it leads to a solution where $r_j = \sum_{i=k+1}^{m-1} b_i$ and $\phi_i = 0$ for $j < i < m$, which can further be improved by choosing either $r_j = b_j$ or $r_j = 0$ (from Theorem 3.3.1). Both cases exclude $\phi_j = 0$ and $\phi_i = 1$ for $j < i < m$ to be simultaneously part of the optimal solution. The proof can further be extended to the complete series of consecutive 0's in Φ. □

The previous theorems show that we can find the optimal solution for our optimization problem by iteratively searching all available network paths P_i, taken in ascending order of their loss probability p_i. Once we find a network path that can improve the overall distortion result, before using it, we have to make sure that all other network paths with better loss parameters are already used to their maximum available bandwidth. Hence, the search space is reduced to a maximum of n computations.

3.3.3 Non-Disjoint Network Paths

We now show that, relaxing the assumption on disjoint network paths in the original network graph does not change the general form of the optimal solution, in the case of flow-equivalent graphs. We assume that in the original flow-equivalent network graph G, there is at least one bottleneck link L_u, shared by at least two distinct network paths. Let $\mathcal{B}_u = \{P_k\}, \forall k : L_u \in P_k$, be the set of paths sharing the bottleneck link L_u. In this particular case, while using any of the paths P_k alone yields an available bandwidth $b_k \leq \rho_u$, using all of them in the same time results in an aggregated bandwidth $\sum_k b_k \geq \rho_u$. Note that L_u may or may not be a bottleneck link for any of the paths P_k treated independently. The paths P_k in $\mathcal{B}_u$ are called "joint paths". The following theorem regulates the sharing of bandwidth ρ_u among paths P_k:

Theorem 3.3.3 (Bottleneck Bandwidth Sharing)**.** *Let L_u be a bottleneck link for the set of paths $\mathcal{B}_u = \{P_k\}$ in the flow-equivalent graph G, the bottleneck link bandwidth ρ_u shall be shared among paths P_k in a greedy way, starting with the path affected by the lowest loss probability.*

Proof. As previously, let the paths $P_k \in \mathcal{B}_u$ be arranged in increasing order of their loss probabilities p_k. Let further $\vec{R}_u = \{r_k\}_{P_k \in \mathcal{B}_u}$ denote a valid rate allocation among the non-disjoint paths. Recall that a valid rate allocation has to satisfy the single flow constraints (i.e., $r_k \leq b_k, \forall k$), and the multiple flow constraints, $\sum_k r_k \leq \rho_u$. Let P_i be the path with the lowest loss probability in $\mathcal{B}_u$. If $r_i < b_i$ in $\vec{R}_u$, and $\sum_{k,k \neq i} r_i > 0$, one can always find a better rate allocation by transferring rate from other flows sharing the same bottleneck link, to the flow $\mathcal{F}_i$. Since the total rate stays constant, the rate transfer does not affect the source distortion, and does not violate the multiple flow constraints. It however reduces the channel distortion, resulting in improved overall performance. By induction, the proof can be extended to all non-disjoint paths in the flow-equivalent network. This shows that for any valid, but non-greedy rate allocation $\vec{R}_u = \{r_k\}_{P_k \in \mathcal{B}_u}$, there exists a better solution that uses in priority the lowest loss probability paths. □

Note that the previous theorem can easily be extended to any number of bottleneck links in $G(V, E)$ and to paths that belong to different sets $\mathcal{B}_u$ in the same time. The joint bottleneck link rate allocation procedure stays optimal as long as G belongs to the class of flow-equivalent network graphs. Theorem 3.3.3 permits to extend Theorem 3.3.2 to generic network graphs, with potentially non-disjoint paths, as long as G is a flow-equivalent graph. It results in the general rule that paths should be taken in the increasing order of their loss probability, and that all the flows should be used to their maximum capacity, which can be limited by joint bottleneck links, before considering an additional flow. Interestingly, any flow-equivalent network scenario can thus

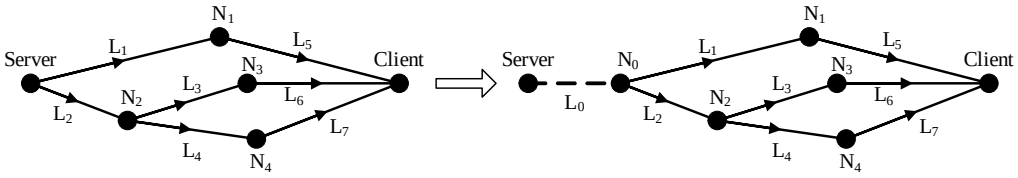


FIGURE 3.5: *Inclusion of budget or encoding rate constraints as a virtual network link in the original network graph.*

be transformed into a disjoint flow tree, by a greedy allocation of joint bottleneck bandwidths to flows affected by lower loss probabilities first. After this transformation, applying Theorem 3.3.1 and Theorem 3.3.2 will yield the optimal rate allocation for the given streaming scenario.

Finally, we can relax the assumption of independent flows in Theorem 3.3.1 by proper adaptation of the maximal bandwidth of all non-disjoint paths.

Corollary 3.3.2. *Given a flow-equivalent network with flows $\mathcal{F}_i$ ordered in increasing order of their loss probability, and a distortion metric as defined in Eq. (3.9), the optimal solution of the MMR problem lies at the margins of the value intervals for all r_i. In other words, the optimal value of r_i, $\forall i : 1 \leq i \leq n$, is either 0 or $b_i' = \min(b_i, w_i)$, where $w_i = \min_{u:L_u \in P_i} \{\rho_u - \sum_{k:L_u \in P_k \text{ and } p_k < p_i} b_k'\}$.*

Finally, multipath streaming applications may also have to respect a budget constraint $Q = \sum_i kr_i$, or a maximal encoding rate R_c in the case of pre-encoded media sequence. These constraints can be modelled as an additional virtual bottleneck link going out of the server. Figure 3.5 shows such a transformation, where link L_0 and node N_0 are added to the topology in order to incorporate the previous overall constraints. Link L_0 should not influence the loss process of the intermediate network, hence $\theta_0 = 0$. The bandwidth ρ_0 is established at $\rho_0 = \min(\frac{Q}{k}, R_c)$, where Q and R_c are simply set to ∞ in the case where there are no limitative factors on the total bandwidth. Applying Theorem 3.3.1, Theorem 3.3.2 and Theorem 3.3.3 on the new network graph $G' = (E, V, L_0, N_0)$ (which remains a flow-equivalent graph,a s long as $G(V, E)$ is a flow-equivalent graph), yields an optimal rate allocation for a stored packet stream, which fully takes into account the budget and encoding rate constraints.

3.4 Rate Allocation Algorithm

3.4.1 Linear Complexity Search Algorithm

The analysis proposed in Section 3.3 shows that a simple algorithm can find the optimal rate allocation by parsing all available network paths in ascending order of their loss probability. Denote $\Phi_i = [\phi_1, ..., \phi_n]$ a solution vector with $\phi_j = 1$, $\forall j \leq i$ and $\phi_j = 0$ otherwise. $R(\Phi_i) = \sum_{j=1}^{i} r_j$ becomes the cumulative rate of the first i flows, whose individual rates have been chosen according to Corollary 3.3.2. The overall loss probability of the first i flows, $\pi(\Phi_i)$, is then given by $\pi(\Phi_i) = \frac{\sum_{j=1}^{i} p_j \cdot r_j}{\sum_{j=1}^{i} r_j}$. The Search Algorithm iteratively computes $D(R(\Phi_i), \pi(\Phi_i))$, for $1 \leq i \leq n$, and the optimal rate allocation is the policy Φ^* that minimizes the distortion metric:

$$\Phi^* = \arg \min_{\Phi_i, 1 \leq i \leq n} D(R(\Phi_i), \pi(\Phi_i)) \tag{3.10}$$

The algorithm will be able to find the global optimal rate allocation only after parsing all available network paths. From the previous theorems, the optimal rate allocation solution Φ^*

takes the form of a consecutive series of 1's, followed by a consecutive series of 0's, hence requiring a maximum of n computations. We propose below a few conditions for early termination, which may avoid to test all possible solutions, while still ensuring a global optimal solution. These conditions represent an extra complexity reduction of the optimum search[2].

3.4.2 Conditions for Early Termination

The search algorithm has to iteratively compute $D(\Phi_i)$, for increasing values of i. A full search through n possible solutions may however be avoided, if any one of the following termination conditions is verified:

1. Distortion Limitation: If $D(\Phi_{i-1}) \leq \beta \cdot p_i$, then the optimal rate allocation contains $\phi_j = 0$, $\forall j \geq i$.

   It can be shown from the distortion function given in Eq. (4.1) that $\lim_{b_i \to \infty} D(\Phi_i) = \beta \cdot p_i$, when other rates b_j stay unchanged, $\forall j \neq i$. Hence, for a value of $D(\Phi_{i-1}) \leq \beta \cdot p_i$, adding another flow on path P_i will asymptotically increase the overall distortion metric to $\beta \cdot p_i$. Therefore, for any positive value of b_j, with $j \geq i$, and $p_j \geq p_i$, adding extra rate on path P_j will only increase the distortion measure in this case.

2. Path Bandwidth Limitation: Solving the equation $D(\Phi_{i-1}) = D(\Phi_i)$ for the variable r_i may provide, except the trivial solution $r_i = 0$, another positive, finite value for r_i, noted as $r_i^{'}$. This second solution happens in the case where $D(\Phi_{i-1}) \geq \beta \cdot p_i$ and $R(\Phi_{i-1}) \geq e^{\frac{ln(-\frac{\beta}{\alpha \cdot \xi}(p_i - \pi(\Phi_{i-1}))}{\xi}}$. The later value is obtained by solving $\frac{\partial D(\Phi_i)}{\partial r_i} \mid_{r_i=0} = 0$. It represents the minimum rate r_{i-1}, after which, adding an extra rate r_i could lead to an increase in distortion. In the case where $b_j \leq r_i^{'}, \forall P_j$ with $j \geq i$, adding another flow, will not decrease the overall distortion, since unused bandwidth is not sufficient anymore to compensate for the increase in loss probability in case an extra flow is added. In that case, according to Theorem 3.3.2 and to the definition of the distortion metric, $D(\Phi_j) \geq D(\Phi_{i-1}, r_i^{'})$, hence $D(\Phi_j) \geq D(\Phi_{i-1})$, $\forall j \geq i$.

Any of the above criteria represents a sufficient condition for search termination from the theoretical point of view, and can be applied at any stage of the optimal solution computation.

3.4.3 Rate Allocation Algorithm

This section presents a simple algorithm that computes the optimal rate allocation for the optimization problem. The previous theorems and conditions for termination represent the keys for a fast search through the flow tree. Assume that the server knows, or can predict the parameters of the intermediate network links, and the sequence-dependent distortion parameters. Initially, the network graph is transformed into a tree of flows $\mathcal{F}_i$, sorted along increasing values of the loss probabilities p_i, with greedy assignment of joint bottleneck link bandwidths. In case where two network paths have the same end-to-end loss probability, they are considered as a single path with aggregated bandwidth. The search for an optimal solution of the shape given by Theorem 3.3.2 is performed iteratively. At each step, the early termination conditions are verified. Once any of them is satisfied, or when the algorithm finishes the search of all flows, the algorithm stops and outputs the optimal multipath rate allocation strategy. `Algorithm 1` proposes a sketch of the rate allocation algorithm.

During the initialization process, `Algorithm 1` must compute all available paths between the streaming server S and the client C. This is a well-known problem in graph theory, and a solution

[2]Please note that the problem in general can be solved in less than linear time (e.g., $O(\log(n))$ computations). However, due to the limited number of paths chosen for transmission, as reflected by our simulation results, the linear time algorithm that parses the available network paths in ascending order of their loss probability, along with the conditions for early termination, achieve the optimal solution even faster.

Algorithm 1 Optimal Streaming Rate Allocation

Input:
Server S, Client C, Available Flow-Equivalent Network Topology $G(V, E)$, Budget Q, Maximum Encoding Rate R_c;
Output:
Optimal Rate Allocation Policy Φ^*;
Initialization:
Initial Rate Allocation $\Phi = [\phi_1, \phi_2, ...\phi_n] = [1, 0, ..., 0]$, according to Theorem 3.3.1;
Compute the set of available paths $P_i \in \mathcal{P}$, with their individual b_i and p_i;
Procedure RateAllocation
Address constraints Q and R_c as in Section 3.3.3;
Decouple joint paths according to Theorem 3.3.3;
Arrange the network paths is ascending order of their loss probabilities p_i and construct the Flow Tree;
for $i = 1$ to n **do**
  Compute $D(\Phi_i)$, where Φ_i represents a rate allocation with the first i flows used at their maximum bandwidth, and the other flows are omitted;
  if any of the termination conditions 'Distortion Limitation' or 'Path Bandwidth Limitation' is satisfied **then**
    break;
  end if
end for
Output $\Phi^* = \arg\min_{\Phi_i, 1 \leq i \leq n} D(R(\Phi_i), \pi(\Phi_i))$;

can be easily found by implementing a depth-first search (DFS) [186], for example. The algorithm then arranges the discovered network paths as a flow tree in ascending order of their end-to-end loss probabilities. Any sorting algorithm of complexity $O(n \log(n))$ can be used. After the flow tree is constructed, the core of the algorithm finds the optimal rate allocation with a complexity $O(n)$, at maximum.

3.5 Discussion

In this section we discuss the practical deployment of the mechanisms proposed above, and some of their limitations. The problem formulation and the methodology for the optimal flow rate allocation of a given video packet stream over multipath networks, are valid for numerous encoding scenarios, including off-line joint source and channel coding of media streams. We assume that the server is not able to perform complex coding operations in real-time, mostly for computational complexity and scalability issues. In such a scenario, adaptive streaming strategy mostly consist in finding the best routing strategy, and overall rate allocation, for the transmission of a given packet stream on a given multipath network. Additional benefits are offered when several versions of the same stream are available at the server. Due to the low complexity of our algorithm, the server could identify both the best transmission strategy, and the best stream to be sent, with an additional complexity that is only linear with the number of stored versions. Such a design choice is also beneficial in broadcast applications, where several clients are accessing the same stream. In such situations, fine adaptation of the packet stream to each individual client is impossible. Coupled with efficient packet partitioning strategy, our flow rate allocation solution however offers interesting perspectives in these scenarios.

In typical network infrastructures, bandwidth and loss rate are quite dynamic. However, they usually exhibit stable statistics on medium range timescales (i.e., in the order of few hundreds of milliseconds, to seconds). We assume that the server can estimate the average end-to-end bandwidth r_i and loss probability p_i of the available paths to the client, for such timeframes.

Additionally, we assume that each path is characterized by a total end-to-end delay δ_i, imposed on all packets traversing that path. Finally, the client imposes a maximum tolerable payback delay Δ, after which it starts playing the media file. Given the estimated parameters r_i, p_i and δ_i, the server chooses the optimal transmission strategy in order to maximize the received media quality. While the fastest estimation mechanisms on end-to-end scenarios provide accurate results on time frames of a few seconds [16], our rate allocation mechanism converges to the optimal solution in a very small number of computations. Since our algorithm has a low complexity, it can be run periodically, with updated network parameter estimates. It ensures the best transmission strategy for a stored video stream, given the accuracy of the periodic network parameter estimation.

We identify a few typical scenarios where optimal rate allocation between multiple stream paths can bring interesting benefits in terms of media quality. In each of these examples, the application of the algorithm proposed above is straightforward.

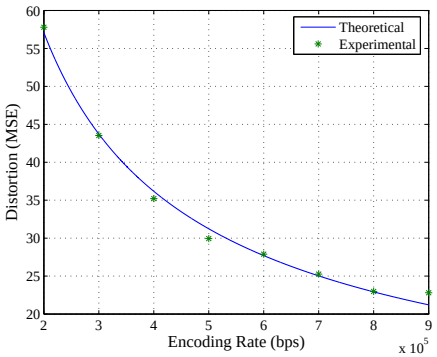
1. Wired Overlay Network Scenarios (e.g., Peer-to-Peer or Content Distribution Networks). The media information from a server/peer is forwarded towards the client by multiple servers/peers belonging to the same overlay network. The client consumes the aggregated media from multiple network paths, and the algorithm proposed above can be applied directly to find the optimal rate allocation.

2. Wireless Network Scenarios (e.g., WiFi Networks). A wireless client can aggregate the media information transmitted on multiple wireless channels. Interference among transmission channels can be minimized by choosing non-overlapping wireless channels (e.g., there are 8 non-overlapping channels according to the IEEE 802.11a standard specifications), and by optimizing the transmission schedule in the wireless network [187]. The authors of [60] test a protocol stack that allows one wireless network card to be simultaneously connected to, and switch between, multiple networks in a transparent way for the application. In the same time, the authors of [188] present a video system over WLANs that uses multiple antennas in order to aggregate the rate of multiple wireless channels.

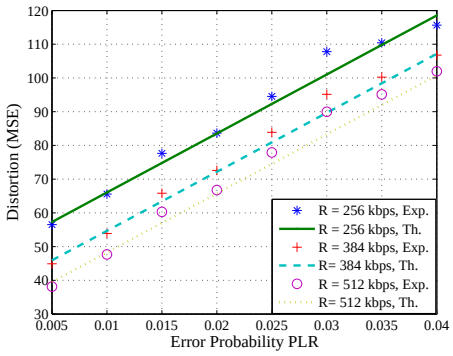
3. Hybrid Network Scenarios (e.g., UMTS/GPRS/WiFi Networks). A mobile client can simultaneously benefit from multiple wireless services in order to retrieve the media information from a server connected to the internet backbone. Existing commercial products [61] can already maintain connectivity to multiple wireless services (e.g., UMTS, EDGE/GPRS and WiFi hotspots), and transparently switch at any time to the service that offers the best channel performance, for a fixed subscription price. It is only a question of time before such commercial products will be able to aggregate the resources of multiple such services in order to enhance the user streaming experience, and telecommunications operators are actively working on such systems.

All these applications can be modelled according to Section 3.2.1, and the implementation of the proposed algorithm is generic and independent of any particular bandwidth and loss model, as long as the media flows can be considered independent in terms of losses. This assumption is valid in any disjoint path network scenario, since the media flows are independent in terms of both rate and losses. In generic network scenarios, our analysis still holds (namely the transformation between the network graph and the tree of flows in Section 3.2.2), as long as the predominant losses affecting the transmission process are independent among media flows (e.g., scenarios 2 and 3). An analysis of the rate allocation problem in general networks characterized by a Gilbert loss model (where the transformation in Section 3.2.2 can only be considered as an approximation) can be found towards the end of this chapter.

It can be noted that the applications mentioned above present in general a limited number of available network paths between the streaming server and the client. It is fairly easy for a server to continuously monitor these paths and to estimate their parameters. Based on these parameters, the execution of the proposed algorithm will output the optimal choice of paths and rates in terms of average media quality at the client. For very large network scenarios, it can be noted that the



(a) Encoding Rate Distortion Validation



(b) Loss Distortion Validation

FIGURE 3.6: *Distortion Model Validation with Video Streaming Experiments using the H264 encoder.*

assumption of full knowledge about the network can be relaxed in setting up a distributed version of the proposed algorithm as presented in the next chapter.

Finally, the network path selection and flow rate allocation problem does not consider media packetization and network scheduling issues. These issues are typically addressed at a lower and finer level. The packetized media stream can be split into packet flows corresponding to the chosen network paths, assuming a very simple scheduling algorithm. Given the estimated rates and delays on all the network paths, the server adapts the streaming rate to the available network bandwidth by simple operations on stored video packet stream. Then, it schedules the packets on the different paths according to the estimated arrival times at the client [189]. Network estimation errors and jitter can further be compensated at the client with the use of application dedicated buffers and conservative playback delay. Interleaving may also be implemented to fight against bursty loss processes when delays permits it.

3.6 Simulation Results

3.6.1 Simulation Setup

We test our optimal rate allocation algorithm in different network scenarios, and we compare its performance to heuristic rate allocation algorithms. We use an H.264 encoder, and the decoder implements a simple frame repetition error concealment strategy in case of packet loss. We concatenate the $foreman\_cif$ sequence to produce a 3000 frame-long video stream, encoded at 30 frames per second. The encoded bitstream is packetized into a sequence of network packets, each packet containing information related to one video frame. The packets are sent through the network on the chosen paths, in a FIFO order, following a simple earliest-deadline-first scheduling algorithm. We further consider a typical video-on-demand (VoD) streaming scenario, where the admissible playback delay is large enough (i.e., larger than the time required to transmit the biggest packet on the lowest bandwidth path). Hence, a video packet is correctly decoded at the client, unless it is lost during transmission due to the errors on the network links. Finally, since any budget/cost constraints can be easily integrates in the network setup as proven earlier, we do not consider them as a limiting factor in the following simulations.

Our simulations first validate the distortion metric proposed in Eq. (3.9). Then, the performance of our optimal rate allocation algorithm is compared to heuristic rate allocation algorithms, on a set of random network topologies. Finally, we carefully analyze the behavior of optimal rate allocation for a particular network scenario, and discuss optimal solutions.

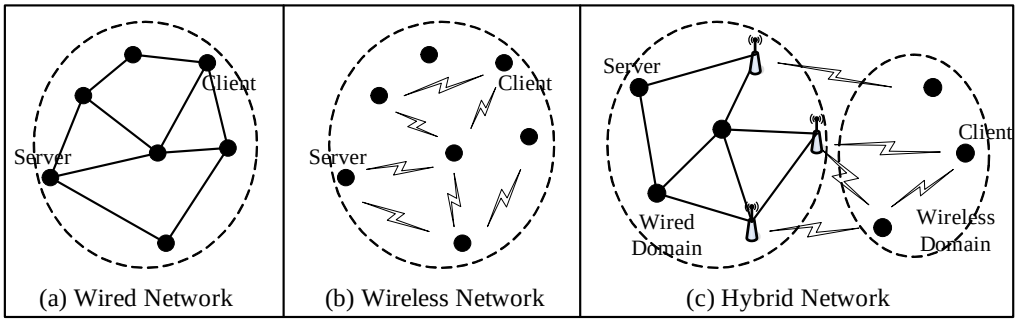


FIGURE 3.7: *Three Network Scenarios.*

3.6.2 Distortion Model Validation

The video sequence is encoded at rates between $200kbps$ and $1Mbps$, and the mean-square-error (MSE) between the original sequence and the decoded one is computed, in error-free scenarios. Simulation results are compared in Figure 3.6(a) to the distortion model values, whose parameters have been set to $\alpha = 1.7674 \cdot 10^5$, $\xi = -0.65848$, and $\beta = 1750$, respectively. We observe that the model distortion curve closely follows the experimental data, which validates the source distortion model.

In order to validate the loss distortion component D_L, random errors are introduced during the network transmission process, where each packet is lost with an independent loss probability PLR. Simulations are performed with different values of loss probabilities, and different encoding rates. We observe in Figure 3.6(b) that the theoretical model closely approximates the experimental data, where each experimental point is averaged over 10 simulation runs. Even if it stays quite simple, the distortion model used in our work closely fits the average behavior of lossy video streaming scenarios. Note that the sequence-dependent parameters may obviously have different values for other encoders or other video sequences. The evolution of the distortion function however stays the same, independently of the exact values of these parameters.

3.6.3 Rate Allocation Performance

We now present the performance of the proposed optimal rate allocation algorithm, in various random network scenarios. We simulate three different categories of network topologies:

1. *Wired* network graphs, in which the edges between nodes are characterized by high bandwidth and low error probability;
2. *Wireless* network graphs, with low bandwidth and high error probability for the intermediate links;
3. *Hybrid* network scenarios, where the server is connected to the wired infrastructure, and the client can access the internet via multiple wireless links.

The network scenarios are presented in Figure 3.7. In each of the three cases, we generate 500 random graphs, where any two nodes are directly connected with a probability γ. The parameters for each edge are randomly chosen according to a normal distribution, in the interval $[\rho_{min}, \rho_{max}]$, for the bandwidth, and respectively $[\theta_{min}, \theta_{max}]$ for the loss probability. The parameters for the wired and wireless scenarios are presented in Table 3.1. The hybrid scenario uses the parameters of both scenarios.

For each of the three types of scenarios, we compute the average end-to-end distortion when rates are optimally allocated, and we compare it to the results obtained by other simple rate allocation algorithms, namely, (i) a single path transmission scenario, which selects the best path in terms of loss probability (D_{PLR}), (ii) a single path transmission scenario (D_R), which uses the

TABLE 3.1: *Parameters for Random Graph Generation*

Parameter	Wired Scenario	Wireless Scenario
Nr. of Nodes	10	10
Connectivity Probability γ	0.4	0.6
ρ_{min}	$10^6 bps$	$10^5 bps$
ρ_{max}	$3 \cdot 10^6 bps$	$7 \cdot 10^5 bps$
θ_{min}	10^{-4}	10^{-3}
θ_{max}	$5 \cdot 10^{-3}$	$4 \cdot 10^{-2}$

TABLE 3.2: *Average Distortion Results (MSE)*

Scenario	D_{opt}	D_{PLR}	D_R	D_{2R}	D_{MF}
Wireless	91.2	99.74	122.861	143.79	108.52
Wired	16.7	20.47	23.4	23.27	17.62
Hybrid	63.4	73.809	83.97	92.533	72.57

best path in terms of effective bandwidth or "goodput" computed as $b_i\ (1 - p_i)$, (iii) a multipath transmission scenario (D_{2R}) that picks the best two paths in terms of goodput, and (iv) a multipath transmission scenario that uses the maximum available number of flows, denoted as D_{MF}. The results, averaged over 500 random graphs are presented in Table 3.2.

As expected, our algorithm provides the best average performance in the three considered scenarios. It has to be noted that, in each individual run of simulation, our algorithm never performs worse than any of the heuristic schemes. Also, we observe that, in the wireless scenario, the rate allocation that is the closest to the optimal strategy is the one offered by the use of the best single path in terms of loss rate. This can be explained by the high loss probabilities of the intermediate links, which cannot be compensated by extra rate added by subsequent flows. On the other hand, in the wired scenario, characterized by very small loss probabilities, the scheme that is the closest to the optimal solution is given by the greedy use of all available flows. In this case, the improvement brought by adding extra transmission rate outruns the losses suffered throughout the transmission process. The results for the hybrid scenario are situated, as expected, between the two extreme cases. The total streaming rates in the three scenarios are in average, $R = 4Mbps$ for the wired scenario, $R = 450kbps$ for the wireless scenario, and respectively $R = 800kbps$ for the hybrid one.

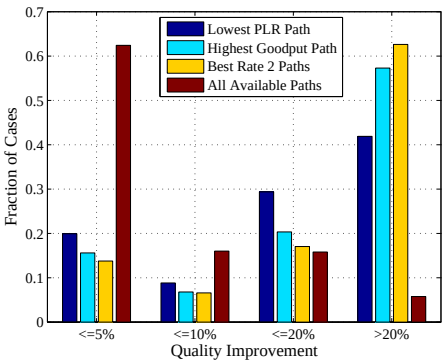


FIGURE 3.8: *Quality Improvement vs. Heuristic Rate Allocation Algorithms - Wired Scenario.*

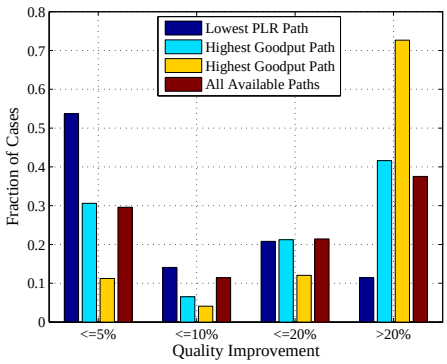


FIGURE 3.9: *Quality Improvement vs. Heuristic Rate Allocation Algorithms - Wireless Scenario.*

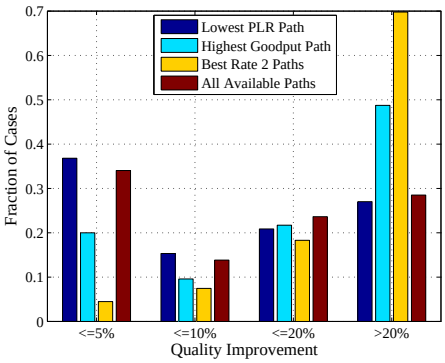


FIGURE 3.10: *Quality Improvement vs. Heuristic Rate Allocation Algorithms - Hybrid Scenario.*

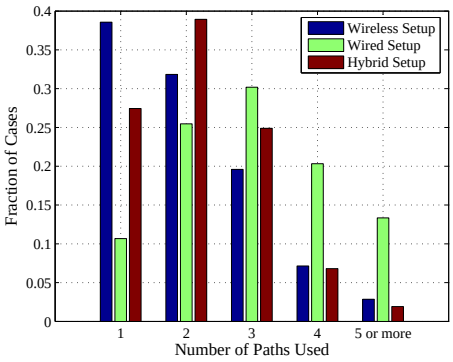


FIGURE 3.11: *Distribution of Optimal Number of Paths for the 3 Network Scenarios.*

TABLE 3.3: *Average Number of Paths*

Scenario	Optimal Nr.	Available Nr.
Wireless	2.04	5.04
Wired	3.049	4.856
Hybrid	2.17	4.419

Next, we study the benefit offered by optimal rate allocation, as compared to the simple heuristic schemes. The relevance of the optimal solution is measured by counting the number of simulation runs in which the optimal rate allocation brings an improvement of $[0-5\%]$, $[5-10\%]$, $[10-20\%]$ and above 20%, in terms of end-to-end video distortion, compared to the other streaming strategies. The results are presented in Figure 3.8, Figure 3.9, Figure 3.10.

We observe that, in more than half of the cases, network flooding represents a good approximation of the optimal solution in the wired scenario where losses are rare. However, we argue that it is still worth applying the proposed rate allocation algorithm, because it is of very low complexity, and can still save network resources. In the wireless scenario, the best approximation is presented in most of the cases by the lowest loss probability path streaming. Still, in almost 40% of the simulation runs, the optimal rate allocation improves the distortion result by more than 10%. Finally, in the hybrid scenario, the rate allocation algorithm provides significant quality improvements compared to all other heuristic approaches. It is also interesting to observe that the rate allocations based on the best goodput path, and best two goodput paths algorithms always provide the worst results.

We also compute the optimal average number of flows used in each simulation scenario, compared to the average number of available paths. The results are presented in Table 3.3. We observe that the wireless scenario uses the smallest number of flows, while the wired one has an average of no more than three flows, for a number of available paths that is far larger. From the multipath streaming point of view, it interestingly shows that, using a very large number of streaming paths does not contribute to an improvement of the video quality at the receiver. This is certainly interesting for the design of practical multipath streaming systems, where the number of paths that have to be synchronized, stays limited. The distribution of the number of flows used per simulation run, is presented in more details in Figure 3.11.

In summary, we observe that a small number of transmission flows is sufficient for an optimal video quality at the receiver, in all simulation scenarios. Paths with lower error probability should be preferred to higher bandwidth paths in wireless scenarios, while in all-wired scenarios with low error probability, adding high-rate flows can improve the overall video quality. In hybrid scenarios, a compromise between the two tendencies is expected to provide the best end-to-end distortion.

TABLE 3.4: *Parameter Values for the Links in $G(V, E)$*

Parameter	L_1	L_2	L_3	L_4	L_5	L_6	L_7
θ_i	0.02	0.01	0.035	0.01	0.015	0.035	0.01
ρ_u $(kbps)$	256	384	256	128	256	256	128

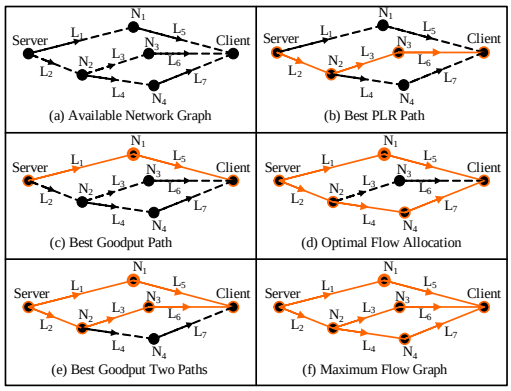


FIGURE 3.12: *One Network Topology Example - Optimal Flow Allocation and Other Heuristic Algorithms.*

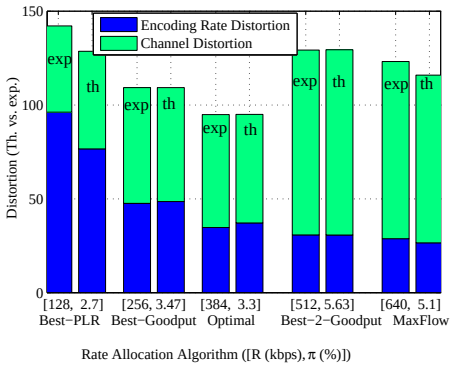


FIGURE 3.13: *Network Scenarios Computation: Theoretical Distortion Model vs. Experimentally Computed Distortion.*

3.6.4 A Case Study

This section proposes to analyze the performance of the optimal rate allocation algorithm in a given network scenario, illustrated in Figure 3.12. The network parameters are presented in Table 3.4. For each of the five rate allocation algorithms, we compute the distortion measure according to the theoretical distortion metric, and we validate it against experimental values, obtained from simulations with video sequences. Each experimental point is averaged over 10 simulation runs. Each video packet is scheduled on the network paths chosen by the given rate allocation algorithm, according to a simple first-available path first. In the same time, each video packet is affected by the individual loss process of each traversed network segment.

The R and π parameters, along with the model and experimental distortion values are presented in Figure 3.13, for each of the algorithms. It can be observed that the optimal rate allocation algorithm outperforms all other heuristic-based strategies. The optimal rate allocation reaches a balance between total used bandwidth, number of network paths, and error probability that affects the streaming process. The example clearly shows that it is not optimal to use only the best paths in terms of rate. In the same time, the greedy use of all available network resources does not provide better results. This clearly motivates the implementation of the proposed rate allocation algorithm, which optimizes the received video quality without wasting network resources. Finally, it can be noted again that the theoretical distortion model represents a very good approximation of the experimental setup.

3.7 Conclusions

In this chapter, we propose to use a flow model to analyze the opportunity of multipath media streaming over the internet. Based on an equivalent transformation between the available network graph and a tree of flows, we jointly determine the network paths and the optimal rate allocation for generic streaming scenarios represented by flow-equivalent graphs. A media specific performance metric is used, which takes into account the end-to-end network path parameters along with media aware parameters.

An in-depth analysis of the end-to-end distortion behavior, in the given network scenario,

drives the design of a linear time algorithm for optimal rate allocation. The form of the optimal rate allocation solution follows a simple greedy rule that always uses the paths with the lowest loss probability first. In particular, we show that extra network paths are either used at their maximum available bandwidth, if their value is large enough, or simply ignored. The overall rate allocation solution offers a careful trade-off between extra transmission rate and increase in the end-to-end error process. Even for large network scenarios, only a small number of paths should be used for transmission, and moreover, they should be chosen among the lowest loss probability channels.

The optimal rate allocation algorithm has been tested in various random network scenarios, and it significantly outperforms simpler schemes based on heuristic rate allocation strategies. In many cases, our algorithm even provides an end-to-end distortion improvement of more than 20%. Due to its low complexity, and important benefits in most streaming scenarios, the optimal rate allocation algorithm provides a very interesting solution to efficient media streaming over resource-constrained networks.

Chapter 4

Distributed Media Rate Allocation in Multipath Networks

4.1 Introduction

This chapter extends our work on media-specific rate allocation and path selection in multipath networks by considering practical implementations based on distributed algorithms. In common practical scenarios, it is difficult for the server to have the full knowledge about the network status. Therefore we propose here a distributed path selection and rate allocation algorithm, where the network nodes participate to the optimized path selection and rate allocation, based on their local view of the network. This eliminates the need for end-to-end network monitoring, and allows for the deployment of large scale rate allocation solutions. We design a distributed algorithm for optimized rate allocation, where the media client iteratively determines the best set of streaming paths, based on information gathered by network nodes. According to this rate allocation, each intermediate node forwards incoming media flows on the outgoing paths, in a distributed manner. The proposed algorithm is shown to quickly converge to the optimal rate allocation, and hence to lead to a stable solution. We also propose a distributed greedy algorithm that achieves close-to-optimal end-to-end distortion performance in a single pass. Both algorithms are shown to outperform simple heuristic-based rate allocation approaches for numerous random network topologies, and therefore offer an interesting solution for media-specific rate allocation over large scale multi-path networks.

We build on the work presented in the previous chapter, which provides a server-driven framework for the analysis of joint path and rate allocation in multipath streaming, based on media-specific quality metrics. We consider a network model composed of multiple flows between the client and the streaming server, which can moreover adapt the media source rate (by truncating of scalable streams, or packet filtering for example). The joint path selection and rate allocation performs iteratively, until all intermediate nodes converge to a (unique) optimal solution. Initially, the intermediate network nodes together report the resources available for the streaming session. Based on this information, the client determines the best path selection and rate allocation, and generates flow reservation requests to the intermediate network nodes and the streaming server. The client-based flow reservation is then accommodated within the network on a node-by-node basis.

The rest of this chapter is organized as follows. Section 4.2 describes in detail the streaming scenario considered, and presents the rate allocation optimization problem. We present our distributed solutions in Section 4.3 and we analyze the characteristics of the proposed algorithms in Section 4.4. Extensive simulation results are finally presented in Section 4.5, for numerous network topologies, and for a practical scenario that is analyzed in details.

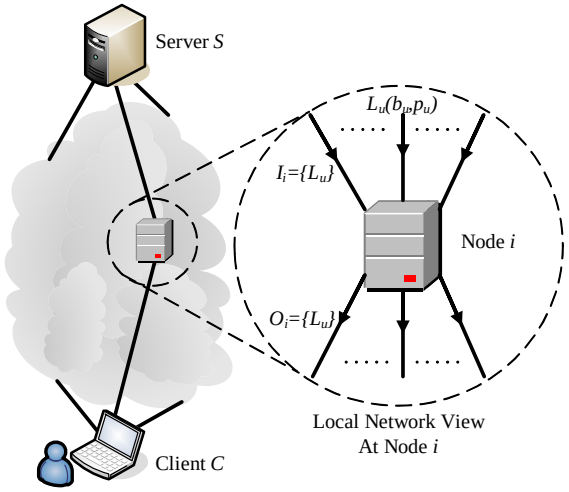


FIGURE 4.1: *Multipath Network Scenario and Network View at Node N_i.*

4.2 The Multipath Rate Allocation Problem

4.2.1 Network and Video Model

We consider that the media streaming application is deployed on a large scale network, modeled like in the previous chapter, as a flow-equivalent network graph $G(V, E)$, between the streaming server S and the client C (Figure 4.1). V is the set of nodes in the network, and E is the set of links. Each node $N_i \in V$ has a *local view* $\mathcal{N}_i = \{I_i, O_i\}$ of the network topology, where $I_i \subseteq E$ and $O_i \subseteq E$ represent the sets of incoming, and respectively outgoing network links to, and from node N_i. Each link $L_u \in E$ has two associated positive metrics: the available bandwidth $\rho_u > 0$, and the average packet loss probability $\theta_u \in [0, 1)$.

We define P_C^i, $1 \leq i \leq n$, as an end-to-end path between S and C in G, with parameters b_C^i and p_C^i being the end-to-end bandwidth and loss probability respectively, and n the total number of distinct paths. A flow[1] transmitted on path P_C^i has a streaming rate $r_C^i \leq b_C^i = \min_{L_u \in P_C^i} (\rho_u)$, and is affected by the loss probability $p_C^i = 1 - \prod_{L_u \in P_C^i} (1 - \theta_u)$.

We define a similar video distortion model for the streaming application as in the previous chapter, consisting of the sum of the source distortion D_S and channel distortion D_L. The average end-to-end distortion can thus be written as:

$$D = D_S + D_L = \alpha \cdot R^{\xi} + \beta \cdot \pi, \tag{4.1}$$

where α, $\beta \in \Re^+$ and $\xi \in [-1, 0)$ are parameters that depend on the video sequence. In the above multipath streaming scenario, the streaming rate can simply be written as the sum of the rates of the different flows :

$$R = \sum_{i=1}^{n} r_C^i \ .$$

We assume that the streaming server can tune the media source rate to the transmission conditions (by scalable coding, or transcoding, for example). In the same time, when the loss processes on different paths are independent, the overall loss probability becomes :

$$\pi = \frac{\sum_{i=1}^{n} p_C^i \cdot r_C^i}{\sum_{i=1}^{n} r_C^i}.$$

[1] Throughout this chapter, the terms flow and end-to-end network path are used interchangeably.

The remainder of this section presents the distributed optimization problem, whose aim is to find the optimal flow rate allocation in order to maximize the received media quality at the client. We then present our solution to the optimization problem in the rest of the chapter. The assumption on full network status knowledge at a given node can therefore be released, and the need of end-to-end monitoring mechanisms eliminated.

4.2.2 Distributed Optimization Problem

We now formalize the distributed path selection and rate allocation problem addressed in this chapter. When no single node $N_i \in V$ (including S), is aware of the entire network topology G, we want to find the optimal path selection and flow rate allocation that minimizes the overall distortion D at the client. Under the assumptions that the streaming rate can be controlled (e.g., by scalable encoding, or packet filtering), and that packet loss rate is independent of the streaming rate, the server S adapts the video encoding rate to the aggregated rate of the available network paths used for streaming, and to the loss process experienced on these paths. The optimization problem can be formulated as follows:

Distributed Multimedia Rate Allocation Problem (DMMR): Given the flow-equivalent network graph $G(V,E)$ whose links L_u have a maximal bandwidth ρ_u and an average loss ratio θ_u, given the node local views $\mathcal{N}_i$, $\forall N_i \in V$ and given the video sequence characteristics ($\Gamma = (\alpha, \beta, \xi)$), find the complete set of end-to-end paths P_C^i, $1 \leq i \leq n$ and the optimal rate allocation $\vec{R}^* = [r_C^1, ...r_C^n]^*$ that minimizes the distortion metric D:

$$\vec{R}^* = \arg\min_{\vec{R}} D = \arg\min_{\vec{R}} (\alpha \cdot R^{\xi} + \beta \cdot \pi) \ , \tag{4.2}$$

under constraints:

$$\begin{aligned} r_C^i &\leq b_C^i, \ \forall P_C^i, \ 1 \leq i \leq n \\ \sum_{P_C^i : L_u \in P_C^i} r_C^i &\leq \rho_u, \ \forall u \text{ s.t. } L_u \in E \end{aligned}$$

where $\vec{R}$ represents the set of possible rate allocation on $G(V,E)$, $R = \sum_{i=1}^{n} r_C^i$ and $\pi = \frac{\sum_{i=1}^{n} p_C^i \cdot r_C^i}{\sum_{i=1}^{n} r_C^i}$.

4.3 Distributed Rate Allocation

4.3.1 Distributed Path Computation

We present in this section two algorithms for distributed path selection and rate allocation. The algorithms differ in the computation of the paths between the server S and the client C. Before describing in detail the distributed path computation and rate allocation strategies, we briefly introduce the notation and assumptions necessary to their presentation. Recall that every node $N_i \in V$ has only a local view of the network topology, denoted by $\mathcal{N}_i = \{I_i, O_i\}$. I_i and O_i are the sets of incoming and respectively outgoing links to/from N_i. We assume that N_i possesses an estimate of the bandwidth ρ_u and loss probability θ_u on the outgoing links (i.e., $\forall L_u \in O_i$).

Let P_i^k denote a path connecting the node N_i to the server. In addition to maximal bandwidth b_i^k and loss probability p_i^k, a path is characterized by two decision flags that are used by the distributed rate allocation algorithms. The flag f^k is a path reservation flag that can only be set or reset by the client C, respectively the server S, and the flag d^k is a decision flag that can be updated by any intermediate node on the path P_i^k. While f^k is used to advertise the network flows

requested by the client C, d^k is used to signal the feasibility of a requested flow at an intermediate node.

We denote by $\Pi_i = \{P_i^k\}$ the set of all distinct paths between the server S and the node N_i. Note that two distinct paths P_i^k and P_i^l may not necessarily be fully disjoint, as they may share one or more network links. Without loss of generality, we assume that the paths in Π_i are ordered according to the increasing value of the path loss probabilities p_i^k. Let finally $\Pi_i^u \subseteq \Pi_i$ be the set of distinct paths between the server S and the node N_i, which share the incoming link $L_u \in I_i$.

End-to-end paths between the server and the client are then built in a distributed manner, since no node has the full knowledge of the network status. These paths are computed by path extension, which is performed independently at each network node. We define $\rightarrow$ as the path extension operator that adds a link $L_u \in O_i$ leaving node N_i, to an incoming path $P_i^k \in \Pi_i$. In other words, if link L_u connects nodes N_i and N_j, we can write $P_j^l = P_i^k \rightarrow L_u$, with $P_j^l \in \Pi_j^u$ and $P_i^k \in \Pi_i$. We can compute the bandwidth and loss probability parameters for the extended path $P_j^l = P_i^k \rightarrow L_u$ respectively as $b_j^l = \min(b_i^k, \rho_u)$, and $p_j^l = 1 - (1 - p_i^k)(1 - \theta_u)$.

We propose two different methods for distributed path computation (employed by the two proposed algorithms), which respectively constructs all the possible paths, or builds them in a greedy manner with respect to their loss process. Formally, the two path extension rules can be stated as follows.

Rule 4.3.1. *Each incoming path $P_i^k \in \Pi_i$ at node N_i is extended towards all the outgoing links $L_u \in O_i$.*

If the set of outgoing links directly connect N_i to several nodes N_j, the set of extended paths at node N_i can be written as $\Omega_i = \{P_j^l = P_i^k \rightarrow L_u \mid P_i^k \in \Pi_i, L_u \in O_i\}$. The subset of the extended paths that borrow the particular outgoing link L_u is written as $\Omega_i^u = \{P_j^l = P_i^k \rightarrow L_u \mid P_i^k \in \Pi_i\}$. All paths with null bandwidth are obviously omitted. It is easy to see in this case that $|\Omega_i^u| = |\Pi_i|$, and that $|\Omega_i| = |\Pi_i||O_i|$, where $|X|$ represents the cardinality of X. The size of the set is multiplicative in the number of incoming flows and in the number of outgoing links [190]. It has to be noted that resource allocation for flows in Ω is constrained by the available bandwidth on joint bottleneck links, and that all the paths may not be used simultaneously at their full transmission bandwidth.

Rule 4.3.2. *The incoming paths $P_i^k \in \Pi_i$ at node N_i, taken in order of increasing loss probability p_i^k are extended towards the outgoing links $L_u \in O_i$, taken in decreasing order of reliability. Similarly to a water-filling algorithm, the total outgoing bandwidth is greedily allocated to the set of incoming paths, until all the incoming paths are extended, or until no more bandwidth is available.*

When the sets of outgoing links, and the incoming paths are both ordered along increasing values of loss probability, the set of extended paths at node N_i can be written as:

$$\Gamma_i = \{P_j^l = P_i^k \rightarrow L_u \mid \sum_{\mu=1}^{u} \rho_\mu > \sum_{\nu=1}^{k-1} b_i^\nu \text{and} \sum_{\mu=1}^{u-1} \rho_\mu < \sum_{\nu=1}^{k} b_i^\nu\}.$$

The subset of the paths in Γ_i that borrow the outgoing link L_u is denoted Γ_i^u. Note that in this case, simultaneous resource allocation for all flows in Γ_i is feasible on G.

Based on the distributed path computation that follows either Rule 1, or Rule 2, we now describe the rate allocation strategy and present the optimal and greedy algorithms for multipath media streaming.

4.3.2 Distributed Path Selection and Rate Allocation

The distributed path computation and rate allocation algorithms proceed first by determining the paths available between the server and client, and then by reserving paths according to the optimal allocation computed by the client. They proceed in two phases, the path discovery and the path

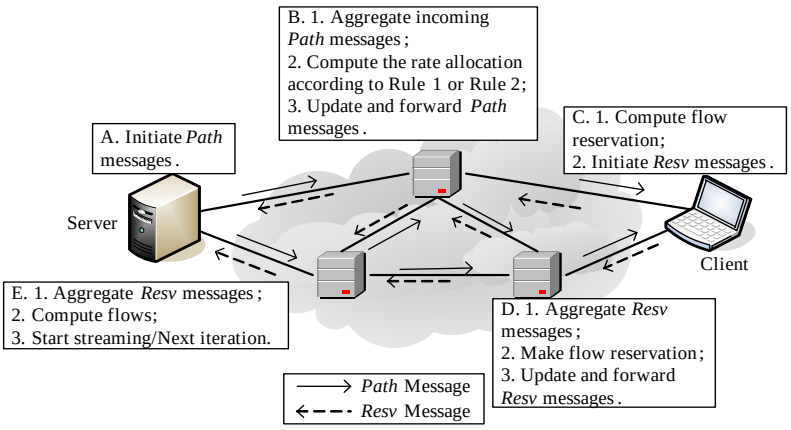


FIGURE 4.2: *Distributed path selection and reservation.*

reservation phases, respectively. To this aim, control messages are exchanged between the server S and the client C, and forwarded by the intermediate nodes, as illustrated in Fig. 4.2. We assume the existence of a bidirectional control channel between any two nodes in G that are connected by a network segment L_u. In order to derive exact bounds on the performance of our algorithms, we assume that the control channel is reliable, and that nodes are synchronized, i.e., any node receives all dedicated control packets in a bounded time interval. Note that these assumptions are not crucial to the design of the proposed algorithms, which can work with looser synchronization. Loose node synchronization can be achieved by employing separate synchronization protocols [191]. Most works addressing decentralized systems [168] generally assume loose node synchronization in order to derive bounds on protocol performance.

The server sends on all outgoing links path discovery messages, $Path^u$, which are forwarded by the intermediate nodes on the control channel associated with link L_u. At each intermediate node, the $Path$ messages contain the information b_i^k and p_i^k related to every possible flow between the server and node N_i, along with potential information related to previously successfully reserved flows. The node then extends the path according to Rule 1 or Rule 2 (in the case of Algorithm 1 or Algorithm 2 respectively), and forwards path discovery message $Path^u$ that contains information about the paths that borrow links L_u. Depending on the path extension strategy, the client will eventually receive information about all possible paths, or only a subset of them that are computed in a greedy manner, based on decreasing reliability.

Upon reception of path discovery messages, the client C computes the optimal path selection Π_C^* using the Theorems 3.3.1 to 3.3.3, and the information it gets from the nodes about end-to-end paths. It should be noted that these theorems greatly simplify the rate allocation, since they state that paths should be either used at their full bandwidth, or simply dropped. The client then initiates path reservation messages, $Resv^u$, which are forwarded by the network nodes to the server, on the backward control channel associated with link[2] L_u. A path reservation message $Resv^u$ contains information about the path(s) that should be reserved on link L_u for the streaming session (e.g., requested rate b_C^k, end-to-end loss probability p_C^k and flags f^k and d^k, which are both set to 1 by C). However, there is no guarantee that all paths in Π_C^* can be accommodated simultaneously. Once all $Resv$ messages are received at node N_i (one for each outgoing link), the node N_i attempts to greedily allocate the bandwidth for the requested flows ($d^k = f^k = 1$) on the outgoing links, following the order of increasing loss probability p_C^k. It eventually marks the flows that cannot be reserved at the requested rate b_C^i, by setting the flag $d^k = 0$. Once a valid subset of paths $\Pi^* \subseteq \Pi_C^*$ is successfully reserved by S (i.e., all d^k flags are set to 1), the nodes update their local view of the network, $\mathcal{N}_i' = \mathcal{N}_i \setminus \Pi^*$, and new path discovery messages are issued. The

[2]Due to practical implementation considerations, an empty $Resv$ message should be sent even on links that do not contain any reserved flow. Alternatively, timeouts should be implemented at each intermediate node.

Algorithm 2 Distributed Path Selection and Rate Allocation Algorithms

server S:	node N_i:
upon receive $Resv^u$, $\forall L_u \in O_S$:	upon receive $Resv^u$, $\forall L_u \in O_i$:
1. compute Π_C^* based on flags f^k;	1. $\forall$ paths $P_i^k \in \{P_i^k\} \vert P_i^k \rightarrow L_u \in Resv^u \setminus \Pi^*$:
2. update Π^* based on flags d^k;	set $d^k = 0$ if $b_C^k > \rho'_u$,
3. if $\Pi^* = \emptyset$ or $\Pi^* = \Pi_C^*$, return Π^*.	where the available output bandwidth ρ'_u
4. else update network view $\mathcal{N}'_S$	is updated according to a greedy allocation;
send $Path^u$, $\forall L_u \in O_S$.	2. send $Resv^v$, $\forall L_v \in I_i$.
node N_i:	client C:
upon receive $Path^u$, $\forall L_u \in I_i$:	upon receive $Path^u$, $\forall L_u \in I_C$:
1. update network graph $\mathcal{N}'_i$	1. compute the set of available paths Π_C;
2. compute available paths Π_i according to $\mathcal{N}'_i$;	2. compute the optimal allocation Π_C^* from Π_C;
3. compute extended paths Ω_i, resp. Γ_i, $\forall L_v \in O_i$,	3. $\forall P_C^k \in \Pi_C^*$, set $f^k = d^k = 1$;
acc. to Rule 1 (Alg. 1) or Rule 2 (Alg. 2)	
4. send discovery messages $Path^v$, $\forall L_v \in O_i$.	4. send reservation messages $Resv^v$, $\forall L_v \in I_C$.

client aggregates information about the residual network resources, and updates the path selection Π_C^* accordingly. The process is iterated until convergence to the optimal rate allocation, which is reached when all flows reserved by C can be accommodated by the network at the requested rate b_C^k.

The distributed path selection and rate allocation algorithms illustrated in Fig. 4.2 are finally summarized in Algorithm 2, where the left-hand side, and right-hand side columns respectively correspond to the path discovery, and path extensions phases. Initially, both algorithms start at the server side, with Step 4. The algorithms differ in the path extension rule (step 3 in the bottom left block). For the sake of clarity, we call Algorithm 1, resp. Algorithm 2, the distributed path allocation and rate allocation solutions that rely on Rule 1, resp. Rule 2 for path extension.

The path extension rule directly controls the convergence to the stable rate allocation, but also the quality of the rate allocation. Comprehensive information about end-to-end paths as created by Rule 1 allows to reach an optimal rate allocation, but at the expense of possibly several iterations of the path reservation schemes. The algorithm however converges in a small number of rounds to a feasible solution, given the network graph G. The Rule 2 constructs only a limited subset of end-to-end network paths, given a greedy forwarding solution at each intermediate node N_i. It allows for a quicker computation of the solution, which may however be suboptimal. Both algorithms are analyzed in Section 4.4 and their performance is compared in Section 4.5.

4.4 Analysis and Discussion

4.4.1 Properties

This section proposes an analysis of the path selection and rate allocation algorithms introduced in the previous section. Under the assumption that the network is stable during the execution of our algorithms, we derive hard bounds on the convergence of the rate allocation towards the optimized solution. Observe that one iteration of the algorithms requires one complete message exchange between S and C, on the available paths. Hence, the time required by one round is in the order of the round trip time (RTT) of the slowest paths in the network. The computations at intermediate nodes and at S and C are trivial and their duration can be neglected. The assumption about the stability of the network in terms of average bandwidth and loss probability of the network links is therefore generally valid since the rate allocation algorithms converge in a very small number of steps, as shown in the next section. Since the total number of paths is quite small in general [192], the algorithms reach a stable solution after a convergence time that corresponds to only a few RTTs, during which the average link characteristics are likely to stay unchanged.

We consider first the Algorithm 1, which uses Rule 1 for path extension, so that the client has

a complete view of end-to-end paths to compute the path selection. We show that the Algorithm 1 converges in one round if paths are disjoint. Then, we show that in the worst case, one round of the algorithm reserves at least the path with the lowest loss probability. Consequently, the Algorithm 1 terminates in a finite number of rounds. We now formally prove these three properties.

Property 1. If the paths requested by C do not share any bottleneck joint link L_u, Algorithm 1 converges in one round.

Proof. Let Π_C be the set of available paths between S and C discovered by Algorithm 1, and let $\Pi_C^* = \{P_C^1 \ldots, P_C^m\}$ be the optimal set of paths chosen by C for transmission, according to Theorems 1 to 3. If b_C^k represents the available rate of on requested path $P_C^k \in \Pi_C^*$, we have $b_C^k \leq \rho_u, \forall L_u \in P_C^k$. Since, by hypothesis, the chosen paths P_C^k do not contain any joint bottleneck link L_u, we have $\rho_u \geq \sum_{k:L_u \in P_C^k} b_C^k, \forall L_u \in P_C^k$ and $\forall P_C^k \in \Pi_C^*$. This means that any node N_i, upon the reception of reservation packets, $Resv$, can allocate the requested bandwidth on the outgoing links for all requested flows. Therefore, no flow is marked with $d^k = 0$, and the server S can compute the optimal allocation $\Pi^* = \Pi_C^*$, after one round of the protocol. □

Property 2. Let the network graph that corresponds to the available resources at one stage of the algorithm be denoted $G' = \bigcup_{i:N_i \in V} \mathcal{N}_i'$. During each round, Algorithm 1 reserves in G' at least the end-to-end flow P_C^i between S and C that is affected by the smallest loss probability p_C^i.

Proof. Let $P_C^i \in \Pi_C^* \setminus \Pi^*$ be the lowest loss probability path requested by C but not yet reserved by our algorithm. Observe that P_C^i is the lowest loss probability path in the residual graph G', and also in the local view $\mathcal{N}_i'$ observed by each node N_i. Hence, at every node N_i traversed by P_C^i, the flow P_C^i will have priority during the greedy reservation phase of Algorithm 1.

Indeed, from the path extension operation we have $b_C^i \leq \rho_u, \forall L_u \in P_C^i$. Hence, P_C^i is successfully reserved at each intermediate node N_i on the path. Finally, the flow P_C^i reaches S with the $Resv$ packets with both flags $d^i = f^i = 1$, hence the server S integrates the flow into the set of successfully reserved paths: $\Pi^* = \Pi^* \cup P_C^i$. □

Property 3. Algorithm 1 converges and terminates in at most m rounds, where m is the number of allocated flows, which is moreover not larger than the total number of available distinct paths in G.

Proof. This result is a direct consequence of Property 2. At each round, the algorithm reserves at least one flow, and the available rate of the links in the residual network decreases. Hence, on subsequent rounds of the algorithm, the client C will not be able to request an infinite number of flows. □

The previous properties show that Algorithm 1 converges to the optimal path selection in a limited number of rounds, no more than the total number of available end-to-end paths between S and C. Moreover, in the case of disjoint network paths, our protocol manages to reserve the optimal set of flows needed for transmission in a single round. And in general networks, the algorithm secures at least one transmission flow from the optimal allocation.

We now concentrate on the second algorithm, and demonstrate that it converges in a single iteration. Moreover, we show that the solution offered by Algorithm 2 is actually identical to the optimal solution provided by Algorithm 1 if each network node has only one outgoing link.

Property 4. Algorithm 2 converges after one round of path discovery and selection phases.

Proof. Let Π_C be the set of available paths between S and C, as discovered in the path discovery phase of Algorithm 2, based on path extension Rule 2. Let further $\Pi_C^* = \{P_C^1 \ldots, P_C^m\}$ be the optimal set of paths chosen by C for transmission according to Theorems 3.3.1 to 3.3.3, based on the information received from the network nodes. Let finally b_C^k be the rate of the requested path $P_C^k \in \Pi_C^*$, with $b_C^k \leq \rho_u, \forall L_u \in P_C^k$. The greedy rate allocation in the path extension given

by Rule 2 ensures that, at any node N_i, and $\forall L_u \in O_i$, we have $\sum_{k:L_u \in P_C^k} b_C^k \leq \rho_u$. This means that any node N_i, upon the reception of reservation packets, can allocate the bandwidth on the outgoing links for all requested flows. Therefore, no flow is marked with $d^k = 0$, and the server S can compute the optimal allocation $\Pi^* = \Pi_C^*$ after one round of the protocol. □

Property 5. Algorithm 2 provides the same solution as Algorithm 1 if the outdegree of every intermediate node N_i is equal to 1.

Proof. In this particular type of networks, we observe that the rate allocation operations during path extension in the path discovery phase becomes identical for both Algorithms 1 and 2. Since the rest of the algorithms is totally identical, they will provide the exact same solution, which is moreover optimal. □

4.4.2 Practical Implementation

We discuss here the practical implementation of the proposed algorithms, and propose a few examples for deployment in real network scenarios. In large scale networks, monitoring end-to-end paths between any two given nodes becomes highly complex and costly. Nor active neither passive monitoring solutions scale well in terms of execution time, accuracy and complexity with a growing number of intermediate nodes and network segments [193]. Since full knowledge about network status cannot be achieved in large scale networks, distributed path computation solutions are certainly advisable. They additionally allow to release the computational burden of a single node/server, and distribute it among several intermediate nodes [190]. Networking protocols have been proposed to organize large scale random network graphs into DAGs [38], or sets of multiple end-to-end paths [26] and even to ensure special network properties like path disjointness and survivability [29].

In this chapter, we address the decentralized path computation and rate allocation problem, from the perspective of a media streaming application. The forwarding decisions are taken in order to maximize the quality of service of such specific applications, in particular to minimize the loss probability and aggregate enough transmission bandwidth. Our algorithms present a low complexity in terms of message passing and execution time. In variable network scenarios, where the link parameters change slowly over time, our algorithms can be run periodically in order to adapt the streaming process to a dynamic network topology. Observe that the fastest network parameter estimation algorithms offer good results on timescales of a few seconds [16], while the execution of our path-computation algorithms takes one, or a few round-trip times. Hence, running our algorithm periodically, on timescales equal to the network estimation intervals ensures the optimal transmission decision, with the latest estimation about the network state. Finally, the control overhead can be limited to two packets on each link of the network, for each iteration of the distributed algorithms. For most typical scenarios, the overhead stays very low compared to the streaming rate. It typically depends on the periodicity chosen for the computation of the distributed rate allocation.

Our framework for path selection and rate allocation can be applied in a straightforward manner to a multitude of large scale network scenarios, e.g., overlay network scenarios (Content Distribution Networks or Peer-to-peer networks), wireless network scenarios, or hybrid interworked wireless setups.

For the case of shared network resources in many-to-many setups, simple modifications to our algorithms can yield good resource allocations among clients, given an optimization metric. Consider Φ as the resource sharing policy implemented at an intermediate node i. Φ is designed according to the final optimization metric of the overall system, e.g., maximizing system quality [79]. Fairness and congestion control mechanisms on the end-to-end discovered paths can also be successfully applied [47]. Finally, simple distributed resource sharing and packet prioritization schemes can be implemented based on the different importance of the simultaneous sessions [127]. Based on Φ, each node i can take an appropriate decision on how to allocate its resources, (namely

the bandwidth of the outgoing links) among the concurrent applications, based on pre-defined utility functions for example. While the design of truly fair distribution of resources between concurrent sessions is outside the scope of our work, our generic framework allows to limit the bandwidth offered to a single session, and therefore permits the implementation of independent congestion control solutions.

4.5 Simulations

4.5.1 Simulation Setup

We analyze the performance of our path computation algorithms in different network scenarios, and we compare them to simple heuristic-based rate allocation algorithms. Results are presented in terms of convergence time, and video quality performance. We first study the average behavior of the algorithms in random network graphs, and we eventually discuss in details a specific, realistic scenario, implemented in ns2 [194] in the presence of cross traffic.

In all simulations, the test image sequence is built by concatenation of the *foreman* sequence, in CIF format, in order to produce a 1500-frame video stream, encoded in H.264 format at 30 frames per second (equivalent to 50 seconds of video). The encoded bitstream is packetized into a sequence of network packets, where each packet contains information related to at most one video frame. The size of the packets is limited by the size of the maximum transmission unit (MTU) on the underlying network. The packets are sent through the network on the chosen paths, in a FIFO order, following a simple scheduling algorithm [189]. The video decoder finally implements a simple frame repetition error concealment strategy in case of packet loss. A video packet is correctly decoded at the client, unless it is lost during transmission due to the errors on the network links, or unless it arrives at the client past its decoding deadline. We consider typical video-on-demand (VoD) streaming scenarios, where the admissible playback delay is large enough, i.e., larger than the time needed to transmit the biggest packet on the lowest bandwidth path.

4.5.2 Random Network Graphs

We generate two types of network topologies: (i) typical *Wireless* network graphs, with low bandwidth and high error probability for the network links; and (ii) *Hybrid* network scenarios, where the server is connected to the wired infrastructure (high rate, low loss probability), and the client can access the internet via multiple wireless links, which have a reduced bandwidth, and a higher loss probability. For both scenarios, we generate 500 random graphs, with 10 nodes each. Any two nodes are directly connected with a probability γ. The parameters for each link are randomly chosen according to a normal distribution, in the interval $[R_{min}, R_{max}]$ for the bandwidth, and respectively $[p_{min}, p_{max}]$ for the loss probability. The parameters for the wired and wireless links are presented in Table 4.1.

Table 4.1: *Parameters for Random Graph Generation*

Parameter	Wired Links	Wireless Links
Connectivity Probability γ	0.4	0.6
R_{min}	$10^6 bps$	$10^5 bps$
R_{max}	$3 \cdot 10^6 bps$	$7 \cdot 10^5 bps$
p_{min}	10^{-4}	10^{-3}
p_{max}	$5 \cdot 10^{-3}$	$4 \cdot 10^{-2}$

First we analyze the number of rounds in which Algorithm 1 converges to the optimal rate allocation given by a centralized algorithm, as proposed in [192]. The results for both network scenarios are presented in Figure 4.3. We observe that the great majority of the cases require less than three iterations in order to reach the optimal rate allocation. This shows that our algorithm

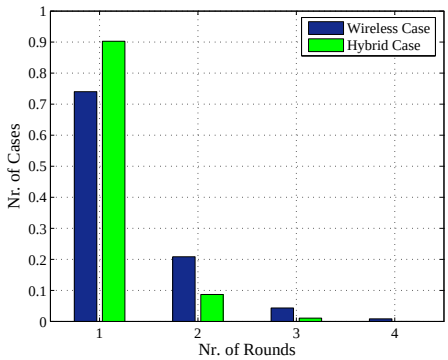


FIGURE 4.3: *Number of rounds of the iterative rate allocation, necessary to converge to optimal solution of Algorithm 1.*

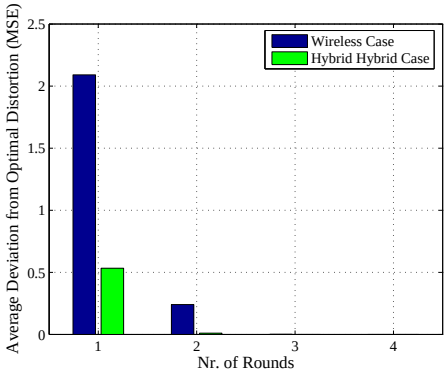


FIGURE 4.4: *Convergence of Algorithm 1, measured in terms of video distortion (MSE) as compared to the optimal solution.*

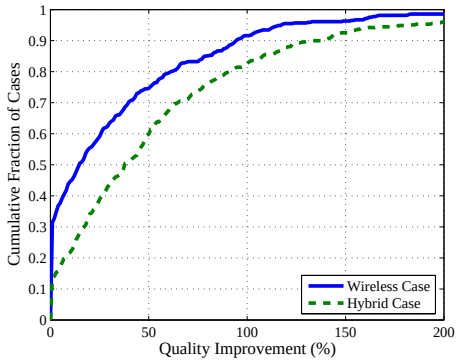


FIGURE 4.5: *Cumulative density function for the improvement in quality offered by Algorithm 1 vs. a Heuristic Rate Allocation Algorithm.*

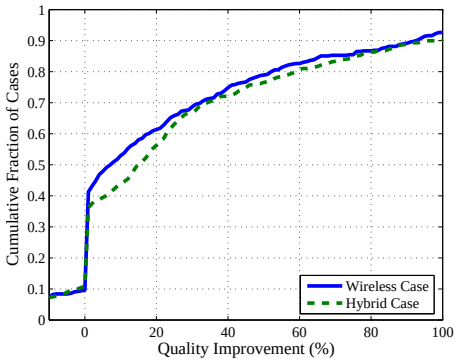


FIGURE 4.6: *Cumulative density function for the improvement in quality offered by Algorithm 2 vs. a Heuristic Rate Allocation Algorithm.*

performs very fast and needs only a very small number of control messages to converge to the optimal rate allocation.

Next, we propose to examine in Figure 4.4 the convergence of Algorithm 1, computed in terms of video distortion, as compared to the quality of the stream achieved with the optimal rate allocation. We observe that the distortion due to Algorithm 1 rapidly decreases, and that the partial solutions are very close to the optimal one, even after the first round of the iterative rate allocation strategy. It clearly illustrates that the proposed distributed algorithm converges very fast to the optimal solution, and that the most critical paths in terms of video quality are already allocated by the very initial rounds of the distributed solution.

In both Figure 4.3 and Figure 4.4, we can observe that Algorithm 1 performs better in the *Hybrid* network scenario than in the *Wireless* case. This is due to the fact that this network scenario has in average less bottleneck links. Please observe that in this simulated scenario, the bottleneck links are usually the wireless links, since the rates of the wired links are much higher. Therefore, Algorithm 1 is expected to converge faster to the optimal solution in the *Hybrid* scenario, where paths are less likely to share bottleneck links. This is in accordance with the properties of this algorithm presented in the previous section.

Then we analyze the performance of the proposed algorithm, in terms of video quality obtained with the rate allocation solution. We compare the results obtained with Algorithm 1, to

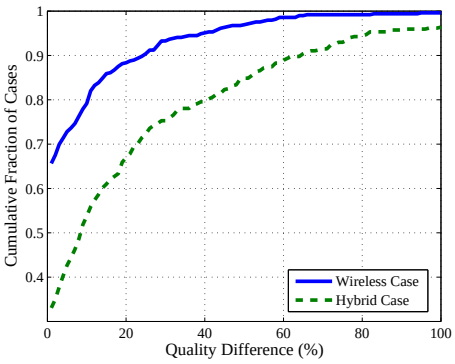


FIGURE 4.7: *Cumulative density function of the relative difference in quality, for Algorithm 1 vs Algorithm 2.*

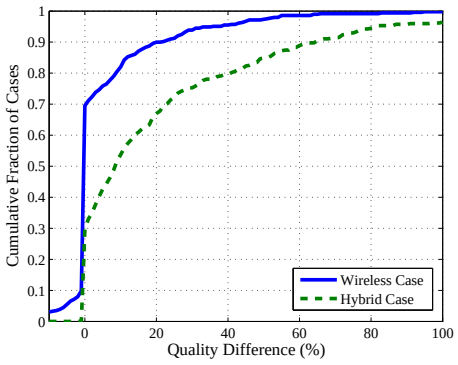


FIGURE 4.8: *Cumulative density function of the relative difference in quality, for Algorithm 1 limited to one iteration only, vs Algorithm 2.*

the ones obtained by a simpler distributed heuristic which forwards the incoming network flow at each intermediate node on the best outgoing link in terms of loss probability (e.g., single best-path streaming). We compute the distribution of the penalty in quality suffered by the heuristic scenario, for 500 different network graphs. The cumulative density function is represented in Figure 4.5, which illustrates the probability for the improvement in quality to be within a predefined range $[0, x]$. We observe that, for both network scenarios, our algorithm obtains significantly better results in more that 70% of the cases. This motivates the extra control overhead introduced by Algorithm 1, which is needed to reach the optimal rate allocation. A similar behavior is shown in Figure 4.6, where we observe that Algorithm 2 also performs much better than the single best path strategy in a large fraction of the cases considered, and for both network scenarios.

Algorithms 1 and 2 are compared in Figure 4.7 and Figure 4.8. Figure 4.7 represents the cumulative density function of the difference incurred by Algorithm 2, with respect to the optimal allocation offered by Algorithm 1. A similar representation is proposed in Figure 4.8, except that the quality provided by Algorithm 1 is computed based on the rate allocation obtained after the first round of the iterative algorithm, as opposed to the optimal allocation that is used in Figure 4.7. From both figures, we see that, for the *Wireless* scenario, the performance of the greedy scheme is equal to the optimal solution in almost 65% of the cases. Algorithm 2 is even better, when compared to the execution of the optimal algorithm after the first round (70% of the cases providing equal or better results). This is due to the very small number of paths chosen for transmission, and to the fact that link parameters in the *Wireless* scenario are quite homogeneous. In the pathological case where all network links would have the same parameters, the performance of the two algorithms would be identical. Good results are also observed for the *Hybrid* network scenario. However, in this case we observe that the greedy algorithm offers bad results in a significant number of cases, since quality attains only 50% of the optimal solution in almost 20% of the cases. This is mainly due to the heterogeneity of the network links parameters in hybrid scenarios.

Finally, we compare Algorithms 1 and 2 in terms of number of flows chosen for the streaming application. The results for the *Wireless* and *Hybrid* network scenarios are presented in Figure 4.9 and Figure 4.10, respectively. We observe that in general Algorithm 2 uses a smaller number of flows for transmission. This can be explained by the greedy allocation of paths, when Rule 2 is used for path extension. Similar results can be observed when the average streaming rate is computed for the solutions provided by both algorithms, for each type of networks. Table 4.2 shows that Algorithm 2 generally results in a smaller transmission rate. However, the performance in terms of received video quality is very close to the optimal one, since the paths with the lowest loss probability are prioritized in both algorithms. In addition, the particular network setup used in

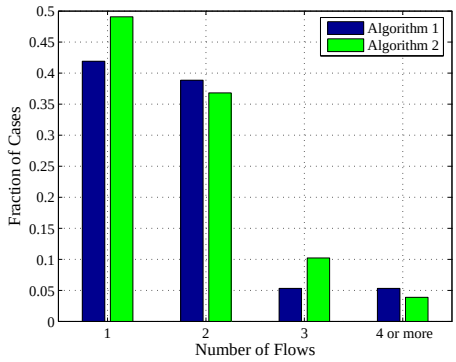


FIGURE 4.9: *Average Number of Flows used by Algorithms 1 and 2 in the Wireless Network Case.*

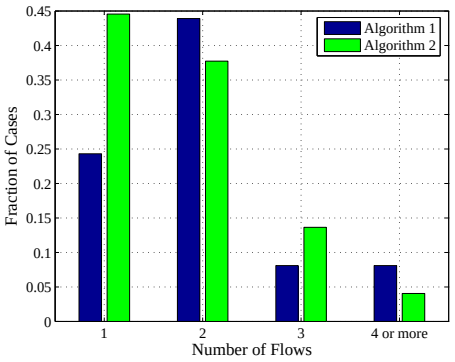


FIGURE 4.10: *Average Number of Flows used by Algorithms 1 and 2 in the Hybrid Network Case.*

TABLE 4.2: *Average transmission rates chosen by Algorithms 1 and 2*

	Wireless	*Hybrid*
Algorithm 1	531*kbps*	797*kpbs*
Algorithm 2	473*kbps*	591*kpbs*

the simulation allows for average streaming rates that already offer a good encoding quality, where the rate-distortion gradient is not very large.

Overall, the previous results show that Algorithm 1 represents a fast path computation solution in most types of networks that present a low number of bottleneck links. On the other side, Algorithm 2 offers a viable, lower complexity alternative for very large network scenarios with homogeneous link parameters, where convergence time is an issue (e.g., in networks characterized by quickly varying parameters).

4.5.3 Sample Network Scenario

We now compare the performance of the two path computation algorithms presented, in a specific network scenario that represents a practical case study. We send the *foreman* sequence, encoded at 375*kbps* and 550*kbps* over a network as presented in Figure 4.11 (a). The network scenario is reproduced in the ns2 simulator, and the path computation mechanisms are implemented as extensions to the simulator. On each of the network paths from the server to the client, we simulate 10 background flows. These flows are generated according to an On/Off source model with exponential distribution of staying time, and average rates between 100 and 300*kbps*. The instantaneous rate available to the streaming application is considered to be the difference between the total link bandwidth, and the instantaneous rate of the aggregated background traffic. We generate two network cases, one with low average link rates and high transmission error probability

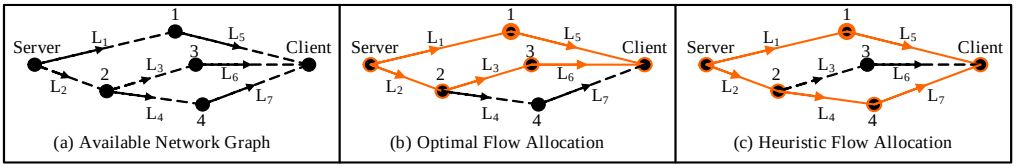


FIGURE 4.11: *Network scenario: a) Available network graph; b) Flow allocation chosen by Algorithm 1; c) Flow allocation chosen by Algorithm 2.*

TABLE 4.3: *Parameter values for the network links in Figure 4.11*

	L_1	L_2	L_3	L_4	L_5	L_6	L_7
Case 1: Loss (%)	2.0	1.0	2.0	1.5	1.5	0.5	2.5
Case 1: Rate (*kbps*)	325	225	225	225	325	225	225
Case 2: Loss (%)	1.5	1.0	1.0	0.75	1.0	0.5	1.5
Case 2: Rate (*kbps*)	450	300	300	300	450	300	300

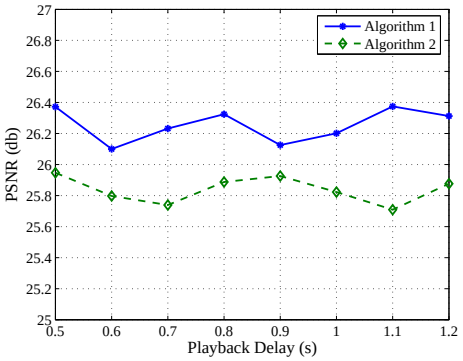


FIGURE 4.12: *Performance evaluation of Algorithms 1 and 2 as a function of playback delay (Network Case 1, no FEC).*

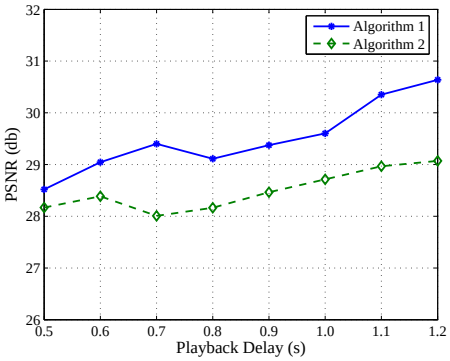


FIGURE 4.13: *Performance evaluation of Algorithms 1 and 2 as a function of playback delay (Network Case 1, with FEC).*

(i.e., end-to-end loss probability higher than 6%), and a second case with higher average link rates and average transmission error probability (i.e., end-to-end loss probability of about 3%). The average bandwidth, and loss probabilities are presented in Table 4.3, for the two cases under consideration. The network MTU is set to 1000 bytes worth of video data. Finally, we also consider cases where the video stream is sent along with forward error protection. Overhead packets are sent in addition to the video packets for packet loss recovery. FEC blocks of 20 packets are formed by adding two redundant packets for each set of 18 video packets in the first network case. In the second case, one FEC packet is added to each group of 19 video packets. Therefore, all video packets can be recovered if at least 18, respectively 19 packets are correctly received in a block of 20 packets. Note that in this specific scenario, both strategies result in an overall streaming rate that is smaller than the average aggregated bandwidth available on the network. Distortion is mostly caused by packet losses, or late arrival due to bandwidth fluctuations.

Figure 4.11 b) and c) first show the path selection provided by Algorithm 1 and 2, respectively. Both network cases result in the same allocation, and the application packets and the control messages of our algorithms share the same network links. Simulations are then run according to these path allocations, and each simulation point is averaged over 10 simulation runs. Figure 4.12 and Figure 4.13 present the performance of Algorithms 1 and 2 as a function of the playback delay imposed by the client, respectively in absence or presence of FEC protection. Recall that the server performs a simple round-robin packet scheduling strategy, for a given set of streaming path. Hence, the playback delay influences the scheduling performance, and larger playback delays allows to pay smaller penalty due to the scheduler choices. The video distortion values incorporate the source distortion due to the low encoding rate of the sequence, along with the loss distortion due to packet transmission losses, and late arrivals at the client. We observe that, even if the choice of transmission paths differs between the two algorithms, the performance is similar, since the end-to-end paths are disjoint, and quite homogeneous in the network case under study. It can be noted that the influence of the playback delay is similar for both schemes. In the same time, it can be observed that using even a minimum error protection strategy unsurprisingly improves the final results, while using no transmission protection at all greatly emphasizes the quality degradation due to network losses in comparison to other streaming parameters, e.g., playback delay. Very

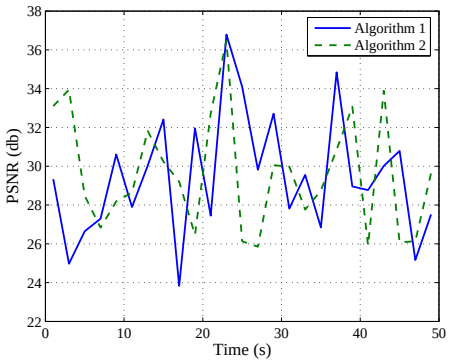


FIGURE 4.14: *Temporal evolution of the video quality (Network Case 2, no FEC).*

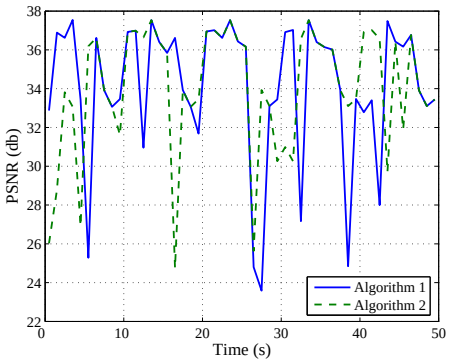


FIGURE 4.15: *Temporal evolution of the video quality (Network Case 2, with FEC).*

similar results can be observed for the second network case with the 500 kbps video bitstream, but they are omitted here due to space constraints.

Finally, we pick one of the simulation runs for each algorithm, and analyze the temporal evolution of the quality. The reconstructed video quality is measured at the receiver averaged for each group of 30 pictures, in the absence or presence of FEC, respectively. Results are presented in Figure 4.14 and Figure 4.15 for the second network case, where the playback delay imposed by the client is set to one second. It can be seen that both algorithms again perform similarly in the presence of network losses and cross traffic. The quality fluctuations are mostly due to packet losses, and basic FEC protection already helps to improve the decoded quality. It confirms the results presented above, and positions both algorithms as efficient solutions for distributed media-specific rate allocation in multipath networks.

4.6 Conclusions

This chapter has addressed the problem of decentralized path computation for multimedia streaming applications in large scale networks. When end-to-end monitoring at the media server becomes intractable and expensive, distributed mechanisms need to be derived in order to optimize the streaming process in terms of media quality. We present two such mechanisms for path computation that differ in the construction of available paths between the streaming server and the client on a node-by-node basis. The first algorithm provides a comprehensive view of the set of end-to-end paths, which leads to optimal rate allocation, at the price of a small convergence time. The second algorithm only offers partial information about the available paths, which results in a lower complexity solution. However, thanks to a greedy allocation that favors the most reliable paths, the performance of the second algorithm stays close to the optimal performance in most of the cases.

In both algorithms, each node is responsible for a rate allocation decision for all incoming flows, on the outgoing links. Hence, the available set of transmission paths to the client is created only from the original local network views at each individual intermediate node. It allows to release the assumption of full network knowledge at any single node in the network and eliminates the need for expensive path monitoring mechanisms. Both solutions therefore represent interesting alternatives for media specific path selection in large scale networks. In particular, extensive simulations demonstrate that the optimal algorithm converges very fast, in particular in networks that present a small number of bottleneck links. In the same time, the greedy algorithm represents a viable and low complexity solution in very large network scenarios with homogeneous link parameters, and stringent limitations on the convergence time of the algorithm.

Chapter 5

Forward Error Correction for Multipath Media Streaming

5.1 Introduction

In this chapter we address the problem of joint optimal rate allocation between media source rate and error protection rate in lossy multipath networks. In lossy network scenarios, where media packets are prone to transmission erasures it is important to chose the right amount of redundancy, and the proper distribution between the source and channel rate, in order to guarantee successful decoding at the end client. Based on a general distortion model for layered encoding video streams, which takes into account possible packet transmission losses, we formulate a general optimization problem that achieves an optimal balance between video source rate and forward error correction rate, given a constraint on total network resources. The optimal solution for our general problem differs with the choice of FEC and scheduling schemes. Hence, based on the most common FEC and scheduling techniques, we propose several concrete instances of this problem and we compute the optimal achieved solutions. In particular, we address the equal and unequal forward error correction schemes, along prioritized or un-prioritized scheduling techniques for layered video coding. At the same time, we offer fast heuristic algorithms that provide good results for our problem with a minimum computational effort. We compare the different instances based on the obtained results. Our results confirm the conclusions drawn in the previous chapters, namely that it is always best to stream on the best network paths first, and that fully utilizing the network resources is not always optimal in terms of average media quality. In the same time, we show the benefits of unequal error protection and we identify the tradeoff between rate allocation optimality and service granularity in real systems.

Furthermore, we address the same problem of optimal channel rate allocation for media streaming in active networks, where intermediate nodes are able to perform basic FEC decoding/encoding operations. FEC performance is analyzed in the case of hop-by-hop FEC protection, and compared with an end-to-end FEC scenario, in order to demonstrate the benefits of FEC operations in the intermediate nodes. FEC operations in intermediate nodes are shown to become especially useful when the network segments on the streaming path have quite heterogeneous characteristics.

The rest of this chapter is organized as follows: Section 5.2 introduces the network, video and FEC models. We discuss possible FEC and scheduling schemes for our proposed setup in Section 5.3 and Section 5.4, and we formulate the optimization problem in Section 5.5. The proposed algorithms are presented in Section 5.6, and evaluated in Section 5.7. Finally, we discuss the case of active networks when intermediate nodes can perform basic FEC operations in Section 5.8, and we conclude the chapter in Section 5.9.

5.2 Multipath Streaming System

5.2.1 Network Model

As in the previous chapter, we consider a framework where the multimedia streaming application uses a multipath network. The available network between the server S and the client C is modeled as a flow-equivalent graph $G(V, E)$, where $V = \{N_i\}$ is the set of nodes in the network, and E is the set of links or segments. Each link $L_u = (N_i, N_j) \in E$ connecting nodes N_i and N_j has three associated positive metrics: the available bandwidth ρ_u and loss probability θ_u as in the earlier chapters, and the propagation delay $t_u \geq 0$, considered as static.

Finally, let $\mathcal{P} = \{P_1, ..., P_N\}$ denote the set of available loop-free paths between the server S and the client C in G, with N the total number of non-identical end-to-end paths. $\mathcal{P}$ can be computed according to the network flow transformation and theorems presented earlier in this thesis. A distinct path $P_i \in \mathcal{P}$ is characterized by the end-to-end bandwidth b_i and loss probability p_i, computed as in the previous chapters.

In addition, we consider the end-to-end propagation delay of path P_i, τ_i, computed as the sum of the intermediate links delays:

$$\tau_i = \sum_{L_u \in P_i} t_i \ . \tag{5.1}$$

Server S uses the available network paths for media packet transmission to the client. After initiating the media request, the client waits for a limited playback delay Δ before starting the playout.

5.2.2 Video Model

We represent the end-to-end distortion, as perceived by the media client, as the sum of the source distortion, and the channel distortion. In other words, the quality depends on both the distortion due to a lossy encoding of the media information (D_S), and the distortion due to losses experienced in the network (D_L). Overall, the end-to-end distortion can thus be written as in the previous chapters:

$$D = D_S + D_L = f(R, \pi, \Gamma) \ , \tag{5.2}$$

where Γ represents the set of parameters that describe the media sequence. This generic distortion model is quite commonly accepted, as it can accommodate a variety of streaming scenarios [85]. For example, when error correction is available, the total streaming rate has to be split between the video source rate that drives the source distortion D_S and the channel rate, which directly influence the video loss rate π.

We assume the video sequence to be layered encoded into L separate layers, each layer $l \leq L$ being characterized by its encoding rate r_l. Video layers are transmitted starting with the base layer, and then adding subsequent enhancement layers, if the network conditions permit it. We assume that a video layer can either be fully transmitted or dropped from an encoder/sender point of view. Hence the total encoding rate of the video stream can be expressed as the sum of the rates of all layers that are transmitted from S to C:

$$R = \sum_{j=1}^{l} r_j, \tag{5.3}$$

where l is the number of transmitted video layers, as decided by the streaming application.

A commonly accepted model for the source rate distortion is a decaying exponential function on the encoding rate, while the channel distortion is proportional in average to the number of lost pixels/video elements. Under the common assumption that network packets contains data referring to the same amount of video information (e.g. one frame, one slice, one encoded video layer of a frame), the channel distortion is proportional to the number of lost packets, and is

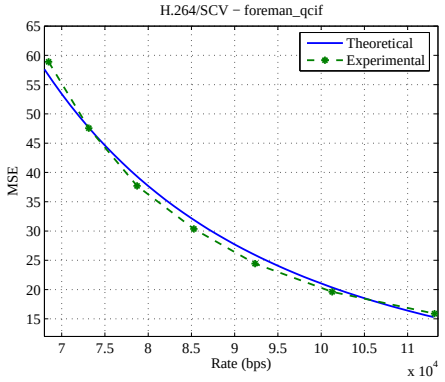


FIGURE 5.1: *Video Model Validation - Source Distortion: H264/SVC encoder, foreman_qcif, 30 fps, one BL and one EL,* $\alpha = 4.41 \cdot 10^4$, $\xi = -1.34515$.

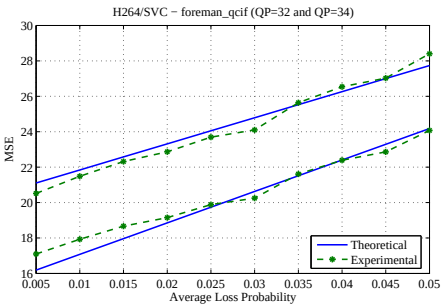


FIGURE 5.2: *Video Model Validation - Loss Distortion: H264/SVC encoder, foreman_qcif, 30 fps, one BL and one EL,* $\beta = 147$.

differentiated by the importance of the video layer containing the lost packets. For video encoding instances where higher video layer cannot be decoded unless all lower video layers are present at the decoder, we build on the general distortion model presented in the previous chapter, and explicitly formulate the distortion metric as:

$$D = \alpha \cdot (\sum_{j=1}^{l} r_j)^{\xi} + \beta \cdot \pi_1 + \sum_{j=2}^{l} (\pi_j \cdot (D_{j-1} - D_l) \cdot \prod_{s=1}^{j-1} (1 - \pi_s)) \tag{5.4}$$

where α, ξ and β are sequence dependent parameters. D_j represents the source distortion of the first j layers of the video stream, and $\pi = \{\pi_j | \forall j : 1 \leq j \leq l\}$ is the set of average loss rates experienced during the transmission process by the video packets of each layer j. π_j depends on the loss probabilities p_i of the subset of network paths used for the transmission of the packets of video layer j, and on the eventual error protection scheme employed for protecting the video packets. Notice that our model for the loss distortion D_L separates the packet losses in the base layer (seen as more severe, because of frame loss and the activation of error concealment strategies at the decoder) and the losses in the enhancement layers (seen as affecting only the total quality of the given frame). In our framework, we consider the packetized bitstream, with one network packet per frame and per video layer. Depending on network available resources, the server decides the number of video layers that can be transmitted to the client. A video layer can either be fully transmitted or dropped.

We validate the distortion model with streaming experiments. We encode the *foreman_qcif* sequence (300 frames, 30 frames per second) in one base layer (BL) and one enhancement layer (EL), with the help of the H.264/SVC encoder. The total rate of the encoded sequence is varied, by encoding at different quantization parameters (QP) for the BL. The chosen encoder implementation always uses a QP for the EL, 6 points below the QP of the BL. On the sequence of packets we are inflicting transmission packet losses according to an independent loss probability $p \in [0, 0.05]$, and we compare the decoded video quality with the original one, by averaging over 100 simulation runs. Results for the validation of the source distortion are presented in Figure 5.1, while Figure 5.2 presents the validation of the loss distortion model. We observe that the model closely follows the experimental results[1].

[1] For a complete validation of the video distortion model, please see [195].

5.2.3 Forward Error Correction

Among all error correction techniques, packet-level FEC is generally preferred in the case of multicast-like or delay sensitive streaming scenarios, especially when packet losses are expected to affect the transmission process. Generically, a FEC block of n packets contains k media packets and $n-k$ FEC packets. In the case of Reed-Solomon codes (RS), the receiver can fully reconstruct the original k data packets as long as it correctly receives at least k packets of the FEC block.

We assume that the server S can protect each media layer against transmission errors, with one systematic forward error correction schemes $FEC(n,k)$. The loss probability for each video layer, protected by $FEC(n,k)$ can be computed starting form the total error probability p, affecting the transmission process of that layer. Let π_j be the error probability affecting video layer j, after FEC decoding. It can be computed as the average probability of loosing exactly i video packets from the FEC block ($1 \leq i \leq k$), and at least $\lfloor n-k-i+1 \rfloor$ redundant packets.

$$\pi_j = \frac{1}{k} \cdot \sum_{i=1}^{k} i \cdot p_i(n,k), \tag{5.5}$$

where $p_i(n,k)$ is the probability of losing at least $n-k+1$ packets from the FEC block, out of which, exactly i packets are video packets. For an iid loss process, $p_i(n,k)$ can be easily computed:

$$p_i(n,k) = \binom{k}{i} p^i (1-p)^{k-i} \sum_{l=\lfloor s+1-i \rfloor}^{s} \binom{s}{l} p^l (1-p)^{s-l}, \tag{5.6}$$

where $s = n-k$.

Given the network and video models presented above, an upper bound on n can be easily computed as:

$$n \leq f \cdot \min_{P_i \in \mathcal{P}} (\Delta - \tau_i) \tag{5.7}$$

where f is the encoded video sequence frame rate, and Δ is the maximum playback delay imposed by the client. Knowing that the FEC performance in general increases with the increase in block size, we consider the maximum block size allowed by the network, e.g., $n = f \cdot \min_{P_i \in \mathcal{P}} (\Delta - \tau_i)$ as the FEC block size[2].

5.3 FEC Schemes

5.3.1 Equal Error Protection Scheme

We investigate two separate forward error correction schemes. First we address the simple Equal Error Protection scheme (EEP), in which all video layers are protected by the same FEC scheme $FEC(n,k)$.

Assume that, according to the scheduling mechanism utilized, each video layer $j \leq l$ is affected by the loss process p_j before FEC decoding at the client. The final loss probability π_j affecting each video layer after FEC reconstruction is computed based on n, k and p_j, according to Eq. (5.5) and Eq. (5.6). At the same time, the total rate of the video stream becomes:

$$R = \sum_{j=1}^{l} r_j \cdot \frac{n}{k} \tag{5.8}$$

and is constrained by the total network available rate $\sum_{i=1}^{N} b_i$.

[2]While the complexity of the RS coding process grows as a quadratic function of n, in delay sensitive streaming scenarios, we expect n to be generally small, hence limiting the required coding execution time.

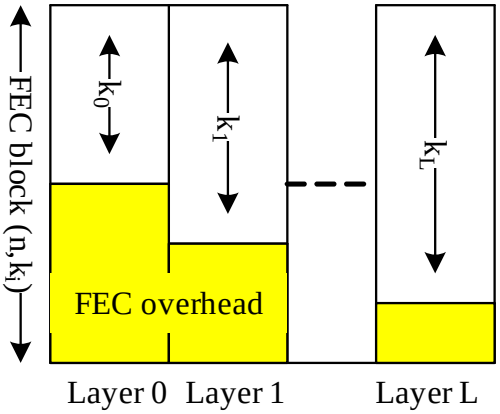


FIGURE 5.3: *Unequal Error Protection per Video Layer: each video layer is protected by different FEC parameters, no matter the allocated transmission paths.*

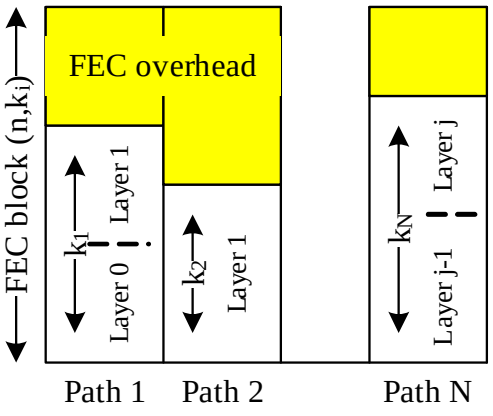


FIGURE 5.4: *Unequal Error Protection per Network Path: each network path offers different FEC parameters for the protection of the passing data, no matter to which video layer it belongs.*

5.3.2 Unequal Error Protection Scheme

Next, we consider the case on unequal error protection (UEP) when different video layers traversing different paths in the network can be protected by individual FEC schemes. Different UEP schemes can refer to individual transmitted video layers, case in which each layer $j \leq l$ is protected by a separate FEC scheme $FEC(n, k_j)$ (Figure 5.3), or to individual network paths, case in which all video data traversing a particular network path P_i is protected by a separate FEC scheme $FEC(n, k_i)$ (Figure 5.4).

In the first case, the total rate of video layer j becomes $r_j \cdot \frac{n}{k_j}$, and depending on the scheduling mechanism utilized, will be affected by the end-to-end loss process after FEC decoding π_j, computed starting from p_j, n and k_j.

In the second case, we can recompute the relevant end-to-end parameters of each path P_i in the network model (bandwidth $b_i^{'}$ and loss process $p_i^{'}$), as seen after applying the FEC scheme $FEC(n, k_i)$ and decoding the data accordingly. The available bandwidth for video packet transmission on path P_i becomes:

$$b_i^{'} = b_i \cdot \frac{k_i}{n}, \tag{5.9}$$

and the new loss process $p_i^{'}$ affecting video packets on path P_i can be computed starting from the FEC parameters n and k_i, and actual packet loss process p_i. Performing this transformation for every individual path $P_i \in \mathcal{P}$, we obtain a new set of available network paths $\mathcal{P}^{'}$ for video streaming (e.g., same set of paths, but with different parameters). The new path parameters $b_i^{'}$ and $p_i^{'}$ will affect the video flows according to the scheduling mechanism employed.

5.4 Scheduling Mechanisms

5.4.1 Equivalent Network Model

We address two different scheduling mechanisms that help us transmit the video information over the network paths. Initially, we present a simple earliest deadline first scheduling mechanism that is unaware of the characteristics of the network paths or of the specifics of the video encoding structure. The scheduling algorithm forwards the incoming media and FEC packets in a FIFO order, on the first available network path, according to the respective rates and propagation

delays. Using this scheduling mechanism in the long run, the multimedia application will perceive the available network between S and C as one equivalent end-to-end network path with average equivalent parameters.

We can easily compute the parameters of the equivalent network end-to-end path, starting from the initial parameters of each individual network path P_i. Let b be the total bandwidth of the equivalent network model. As we have seen in the previous chapters, the network graph $G(V, E)$ can be modelled as a network of disjoint flows/path, as perceived by the media application. In this case we can compute:

$$b = \sum_{i=1}^{N} b_i. \tag{5.10}$$

The average loss probability p of the end-to-end equivalent network link can be computed as the average of the loss probabilities affecting each individual network path in $G(V, E)$:

$$p = \frac{\sum_{i=1}^{N} b_i \cdot p_i}{\sum_{i=1}^{N} b_i}. \tag{5.11}$$

Finally, an upper bound on the propagation delay can be computed for the end-to-end equivalent network link as:

$$\tau = \max_{i:1 \leq i \leq N} \tau_i. \tag{5.12}$$

Considering this scheduling mechanism, the transmitted video layers will experience the network as a single equivalent network path with the equivalent parameters as computed above. The maximum possible FEC block size n can be computed starting from the end-to-end propagation delay τ and Δ, while the error probability π_j affecting each video layer j, protected by a specific FEC code, can be computed starting from the loss probability p of the network link. Finally, the total source coding rate and FEC rate are upper bounded by the total available bandwidth of the equivalent network link b.

5.4.2 Priority Scheduling

Next we address a scheduling algorithm that takes into account the different parameters of the network paths, and the relative importance of the video layers. As seen in the previous chapters, it is always best to fully utilize the network paths in ascending order of their loss probability p_i. Hence we adopt a scheduling strategy that maps the video layers, including the accompanying FEC rate, in increasing order of their importance, on the best available network paths in terms of loss probability.

Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ be the ordered set of available network paths, according to their loss probabilities (e.g., $p_1 < \ldots < p_N$). In the previous chapter, we have seen that network paths P_i and P_j with equivalent error processes $p_i = p_j$ can be considered by the media application as a single network path with aggregated bandwidth $b_i + b_j$ and equivalent propagation delay $\max(\tau_i, \tau_j)$.

At the same time, let the l transmitted video layers be ordered according to their importance (e.g., layer 1 corresponds to the base layer, layer 2 corresponds to the first enhancement layer, ...), and let $FEC(n, k_j)$ be the forward error correction scheme employed for protecting video layer $j \leq l$. For simplicity reasons we assume that the maximum FEC block size is computed in the same way as before. The total network rate required for the transmission of video layer j will be $r_j \cdot \frac{n}{k_j}$. We assume that layer j is mapped according to the gradual filling algorithm described above on network paths $P_s, \ldots, P_t$ with reserved rates $c_s, \ldots, c_t$, where $c_s \leq b_s$, $c_t \leq b_t$, and $c_i = b_i, \forall i : s < i < t$. We observe the following rate equality:

$$r_j \cdot \frac{n}{k_j} = \sum_{i=s}^{t} c_i. \tag{5.13}$$

	EEP	UEP Layer	UEP Path
FIFO Sch.	EqEEP	EqLayer	EqPath
Priority Sch.	SchEEP	SchLayer	SchPath

TABLE 5.1: *Different Optimization Algorithms for the Problem Instances, based on the possible combinations of scheduling and FEC strategies.*

while the total error probability p_j affecting layer j before FEC decoding can be computed as:

$$p_j = \frac{\sum_{i=s}^{t} c_i \cdot p_i}{\sum_{i=s}^{t} c_i}. \tag{5.14}$$

Based on p_j we can now compute the final error process affecting layer j after FEC decoding, π_j, according to Eq. (5.5) and Eq. (5.6). Please observe that, compared to the previous scheduling case, where all transmitted video layers are affected by the same loss probability p, we schedule now the most important video layers on the best paths, hence we have $p_1 <, \ldots, < p_l$.

5.5 Optimization Problem

We consider the problem of optimal rate allocation strategy, for a given video stream that can be split into flows sent on different paths from the streaming server S and the client. Given the network rate constraints and path status in terms of propagation delays and loss probability, we are interested in finding the optimal rate split between source encoding rate and forward error protection rate, in order to maximize the received video quality. Hence, we can formulate the optimization problem as follows:

Joint Multimedia - FEC Rate Allocation Problem (JMFR): Given the flow-equivalent network graph G, the number of different paths or flows n, the video sequence characteristics (Γ) and the total number of encoded video layers L, find the optimal number of transmitted video layers l^*, and the optimal forward error protection scheme $FEC(n, k_j^*)$ for each layer $j \leq l^*$, such that the perceived video distortion D at the client is minimized:

$$\{l^*, k_j^*\} = \underset{l \leq L; k_j \leq n; 1 \leq j \leq l}{\arg\min} D(R, \pi, \Gamma), \tag{5.15}$$

under the network rate constraint:

$$\sum_{j=1}^{l^*} r_j \cdot \frac{n}{k_j^*} \leq \sum_{i=1}^{N} b_i. \tag{5.16}$$

Given the different scheduling strategies for the multipath data transmission, and the various FEC schemes for the protection of the layered video data, the optimization problem will present multiple instances, each one having an optimal solution. The following sections present our proposed algorithms for solving the instances of the optimization problem and discuss in details their performance and opportunity.

5.6 Optimization Algorithms

5.6.1 Optimal Full Search Algorithms

In the previous section we have presented different scheduling and FEC mechanisms and we have computed in each case the parameters necessary for solving the proposed optimization problem. Now we present the algorithms we use in order to search for the optimal solution, and discuss their performance and complexity.

Depending on the scheduling mechanism and the FEC scheme employed we can identify six different types of algorithms as defined in Table 5.1. Each of the algorithms employs one FEC and one scheduling strategy, from the ones presented above.

We are utilizing the full search algorithms as a benchmark for performance. For the sake of clarity we present in `Algorithm 3` the pseudo-code for one of these algorithms. Slight variations in the code will lead to the implementation of full search algorithms for all other streaming strategies.

Algorithm 3 EqLayer Full Search Algorithm.

Input:
Flow-equivalent network graph $G(V,E)$, network paths $\mathcal{P} = \{P_i(b_i, p_i, \tau_i)/\forall i : 1 \leq i \leq N\}$, encoded video bitstream parameters Γ, video layers rates r_l, $\forall l : 1 \leq l \leq L$, frame rate f, playback delay Δ.
Output:
Optimal joint rate allocation $\{l^*, k_j^*\}$.
Initialization:
Compute equivalent network link bandwidth: $b = \sum_{i=1}^{N} b_i$;
Compute equivalent network link loss process: $p = \frac{\sum_{i=1}^{N} b_i \cdot p_i}{\sum_{i=1}^{N} b_i}$;
Compute equivalent network link propagation delay: $\tau = \max_i \tau_i$;
Compute maximum FEC block size: $n = f \cdot (\Delta - \tau)$;
Procedure Compute optimal JMFR solution:
for Every number of video layers $l \leq L$ and every $k_j \leq n$, $1 \leq j \leq l$ **do**
  Check rate constraint:
  if $\sum_{j=1}^{j} r_j \cdot \frac{n}{k_j} \leq b$ **then**
    Compute π_j, $\forall j : 1 \leq j \leq l$, starting from p and k_j;
    Compute $D = D(R, \pi, \Gamma)$ according to the Equivalent Network Scheduling and UEP per Video Layer schemes;
  end if
end for
Output $\{l^*, k_j^*\} = \underset{l \leq L; k_j \leq n; 1 \leq j \leq l}{\arg\min} D(R, \pi, \Gamma)$.

The algorithm finds the optimal solution for the optimization problem, by parsing every feasible rate allocation between source video rate and error correction rate. It outputs the optimal number of video layers to be transmitted, along the optimal FEC strategy for each transmitted layer, such that the media distortion as perceived by the client is minimized.

While the algorithm outputs the optimal result for every network scenario, the computational resources needed are rather high. During the full search for the optimal parameters, the algorithm needs to compute one distortion value for every feasible value of $k_j \leq n$, for every video layer $j \leq L$. Hence, the total complexity of the algorithm is $O(n^L)$. Similarly, the FEC strategy that allocates one FEC code per each individual network path requires a total of $O(n^N)$ computations, with N being the number of distinct available network paths. The exponential complexity of these algorithms will prohibit their use in large scale scenarios with a large number of available network paths and finer granularity in the video encoding. Therefore, we introduce now heuristic algorithms that achieve similar results with a much lower computational complexity.

5.6.2 Utility-based Heuristic Algorithms

In this section we introduce our heuristic approach towards solving the optimization problem. We build on the utility framework introduced in [33], and present algorithms that iteratively take a stepwise locally optimal decision.

Let each algorithm start from an initial feasible solution where only the video base layer, without any FEC protection, is scheduled for transmission, according to the employed scheduling

mechanism. Let also $\mathcal{F}^s = \{l, \{k_j\}; 1 \leq j \leq l\}$ be a feasible solution obtained by our algorithms at iteration s.

We associate to this solution, the total video rate $R^s = \sum_{j=1}^{l} r_j$, satisfying the total network rate constraint $\sum_{j=1}^{l} r_j \cdot \frac{n}{k_j} \leq \sum_{i=1}^{N} b_i$. We can also compute the values $\pi^s = \{\pi_j; 1 \leq j \leq l\}$ representing the loss process observed by every transmitted video layer on the network. Based on these values we can compute the perceived client distortion $D^s = D(R^s, \pi^s, \Gamma)$. Let B_s be the residual available network rate after transmitting all data packets related to solution $\mathcal{F}_s$.

At the next algorithm iteration, $s+1$, we can either attempt the transmission of an extra video layer $l+1$, in case $l+1 \leq L$, or change the FEC parameter k'_j of any of the already scheduled video layers $j \leq l$. Let the new distortion measures associated to each of these actions be D^a_{s+1}, where a identifies the specific action taken. We define the utility of an action a as the ratio between the perceived video quality improvement by performing this action, and the amount of network resources δr^a, necessary for implementing the action:

$$U_a = \frac{D_s - D^a_{s+1}}{\delta r^a}. \tag{5.17}$$

δr^a can be easily computed as r_{l+1} in case a new video layer is scheduled for transmission, or as the extra necessary network rate in order to change the FEC parameters of video layer j from k_j to k'_j, e.g., $\delta r^a = r_j n \frac{k_j - k'_j}{k_j k'_j}$. Any of the actions a is feasible as long as $\delta r^a \leq B_s$. In the same time, action a brings an improvement in quality if $U_a > 0$.

Algorithm 4 SchPath Utility Algorithm.

Input:
Flow-equivalent network graph $G(V, E)$, network paths $\mathcal{P} = \{P_i(b_i, p_i, \tau_i)/\forall i : 1 \leq i \leq N\}$, encoded video bitstream parameters Γ, video layers rates r_l, $\forall l : 1 \leq l \leq L$, frame rate f, playback delay Δ.
Output:
Optimal joint rate allocation $\{l^*, k^*_j\}$.
Initialization:
Compute maximum FEC block size: $n = f \cdot \min_i(\Delta - \tau_i)$;
$\mathcal{F}_1 = \{1, k_1 = n\}$;
Compute $B_1 = \sum_{i=1}^{N} b_i - r_1$;
Compute the ordered set $\mathcal{P} = \{P_i : 1 \leq i \leq N\}$, s.t. $p_1 <, \ldots, p_N$.
Procedure Compute heuristic JMFR solution:
Iteration s=1;
while 1 **do**
  for every feasible action a **do**
    Compute updated distortion value D^a_{s+1} according to the Priority Scheduling mechanism and UEP scheme;
    Compute utility function U_a;
  end for
  if no feasible action a exists, or $U_a \leq 0$, $\forall a$ **then**
    Break;
  end if
  Compute new solution: $\mathcal{F}_{s+1} = \arg\ \max_a U_a$;
  Update available network bandwidth B_{s+1};
  Update iteration: $s = s + 1$.
end while
Output $\mathcal{F}_s$.

The algorithm, at each iteration s will chose the next solution $\mathcal{F}_{s+1}$ by performing the action that maximizes the utility value among all feasible actions. The algorithm stops either when there

Average Encoding Bitrate per video layer [kb/s]	QP=30	QP=34	QP=38
Base Layer	328.8335	233.1850	159.1450
Enhancement Layer 1	482.6697	244.9661	145.1805
Enhancement Layer 2	546.5654	342.1015	201.5870

TABLE 5.2: *Average encoding rate per video layer for encodings with different quantization parameters using H.264/SVC.*

are no more feasible actions, e.g., the network rate has already been totally utilized, or there are no more actions that bring a positive improvement to the current solution. Depending on the FEC and scheduling mechanisms employed, six different algorithms can be derived. `Algorithm 4` presents the pseudo code of one of them, the modifications towards all the others being straightforward.

For a complete search over the FEC parameter space, during each action a, the parameter k'_j becomes $k'_j = k_j - 1$. In real system implementations, where only a limited amount of FEC schemes are available, k'_j should be chosen as the next smaller parameter from the feasible set of schemes after k_j.

During each iteration, the algorithm needs at most L computations, while the maximum number of iterations is $n \cdot L$. Hence the total complexity of the proposed algorithm is $O(n \cdot L^2)$. In the following sections we asses the performance of our heuristic method compared to the optimal full search.

5.7 Experimental Results

5.7.1 Setup

We test the proposed mechanisms in various network setups with various encoded bitstreams. We use a concatenated version of the $foreman\_cif$ sequence (3000 frames), encoded at 30 frames per second using the scalable encoder H.264/SVC. We encode the sequence in three video layers, one base layer and two enhancement layers, at different encoding rates given by the chosen quantization parameters (QP). Our specific encoder generates the desired number of enhancement layers starting from the given QP value for the base layer, and decreasing it by 6 for each additional layer. The obtained data rates for the video layers encoded at different QPs are presented in Table 5.2. We assume that the video layers cannot be decoded unless all lower layers are available at the decoder.

We use a multipath network scenario that offers a variable number of end-to-end transmission paths to the media application. Our results are obtained for network scenarios with two, three or four network paths. Each network path is characterized by a random iid loss process uniformly drawn in the interval $[1-25]\%$, and a propagation delay randomly drawn in the interval $[50-100]ms$. The end-to-end bandwidth of each path is randomly assigned in intervals that are meaningful for each experiment. Finally we assume that the client imposes a fixed playback delay $\Delta = 700ms$, after which it starts playing the received video data. Any packets arriving at the client after their decoding deadline are considered as lost for the application and discarded.

Within the presented framework we compare the performance obtained by the proposed algorithms for optimal joint source-FEC rate allocation, representing the different FEC schemes and scheduling mechanisms presented above. Our results are averaged over 100 simulation runs for each network scenario and each transmitted bitstream. In particular, we emphasize the better performance brought by the UEP error correction scheme and the priority scheduling mechanism. Finally, we discuss real system implementations with constraints on the available set of FEC parameters.

5.7.2 EEP vs. UEP

First we compare the EEP and UEP forward error correction schemes in the case of full search algorithms. We identify five network scenarios, ranging from very low end-to-end loss probability

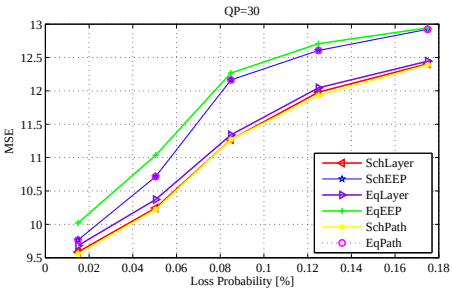


FIGURE 5.5: *FEC schemes comparison for various scheduling mechanisms, video base layer encoding: QP=30.*

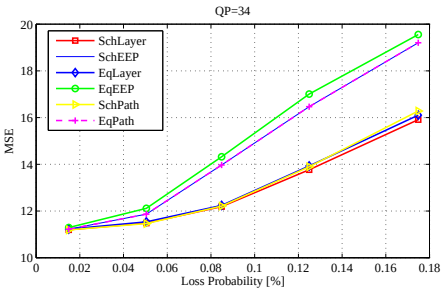


FIGURE 5.6: *FEC schemes comparison for various scheduling mechanisms, video base layer encoding: QP=34.*

Average Loss Probability [%]	SchLayer	EqLayer	SchPath
Base Layer	0.059	0.056	0.06
Enhancement Layer 1	4	3.46	2.1
Enhancement Layer 2	11.52	-	9.3

TABLE 5.3: *Average Loss Probability after FEC decoding for each video Layer, for the algorithms based on UEP.*

to very high one, and we set the end-to-end available bandwidth to be lower than the total encoded rate of the transmitted video bitstream. Each algorithm runs on the network scenario and optimizes the encoding FEC rate allocation in order to maximize the video distortion measure. They decide how many video layers to transmit and how much error protection should be added to each layer, given the total network resource constraints.

Results for the three encoded bitstreams are presented in Figure 5.5, Figure 5.6 and Figure 5.7. We observe that for every bitstream and every range of network losses, the UEP scheme performs better than the EEP scheme. While the improvement is minimal for very low error network scenarios, it becomes increasingly visible as the network conditions get worse. These results clearly evidentiate the importance of flexible error protection in the case of scalable video transmission over lossy networks. The UEP scheme protects differently the video layers, according to their overall importance to the final distortion measure, being able to better utilize network resources. On the other hand the EEP scheme overprotects the higher layers of the video stream, hence wasting the available bandwidth.

Table 5.3 provides a different representation of the same results. Here we show the total error process associated with each transmitted video layer after FEC decoding at the client in the case of the UEP scheme. We observe that, while the base layer is very well protected, ensuring practically zero losses, the higher layers are gradually less protected, as the application can tolerate a higher amount of losses with lower impact on the reconstructed media quality. On the other hand the EEP scheme does not offer this flexibility, hence leading to a suboptimal performance.

5.7.3 Equivalent Network Model vs. Priority Scheduling

Next, we compare the two proposed scheduling mechanisms. Due to the coarse granularity provided by the used video encoder, in this subsection we hand-pick the network total bandwidth, such that we emphasize the conceptual differences between the two scheduling mechanisms[3]. We choose network scenarios with total end-to-end bandwidth that can easily accommodate the first two video layers of each bitstream without error protection (but not three layers), while we randomly choose the error rates of each path as presented before.

[3]Please note that with fully scalable encoding systems, e.g., FGS encoders, the difference between the scheduling mechanisms would always be visible.

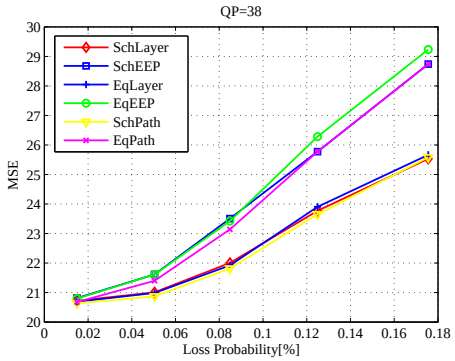


FIGURE 5.7: *FEC schemes comparison for various scheduling mechanisms, video base layer encoding: QP=38.*

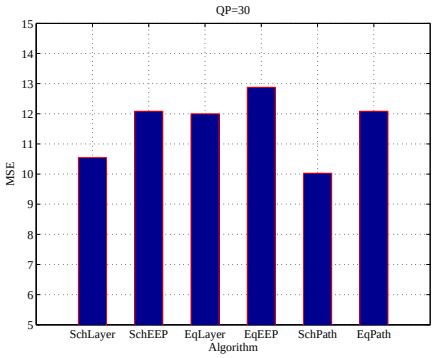


FIGURE 5.8: *Scheduling mechanisms comparison for various FEC strategies, video base layer encoding: QP=30.*

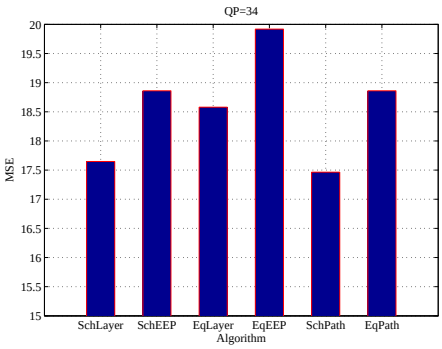


FIGURE 5.9: *Scheduling mechanisms comparison for various FEC strategies, video base layer encoding: QP=34.*

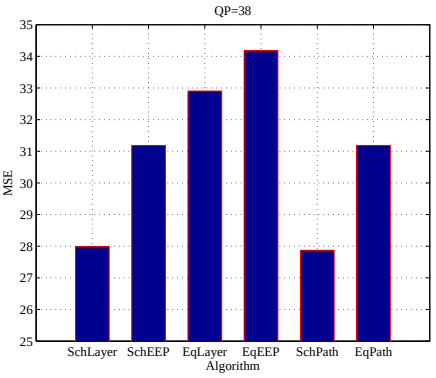


FIGURE 5.10: *Scheduling mechanisms comparison for various FEC strategies, video base layer encoding: QP=38.*

Figure 5.8, Figure 5.9 and Figure 5.10 present the obtained MSE results for the proposed algorithms. We observe that in general the Priority Scheduling with UEP performs better that the Equivalent Network scheduling, for all tested bitstreams. It can also be noted that all algorithms based on UEP outperform the EEP scheme, which corresponds to the results presented in the previous section. The difference in performance between the two scheduling mechanisms can be explained by the better resources utilization of the priority scheme. As the Priority Scheduling scheme sends the most important video layers on the better network paths in terms of error probability, it requires less rate for the error protection, hence being able to send more video layers. On the other hand, the Equivalent Network scheduling scheme considers the network as a single equivalent link with equivalent error parameters, hence it requires more rate for the error protection of the most important layers. In turn, this leaves less resources for transmitting extra video layers. Table 5.4 presents the average number of video layers transmitted by each of the algorithms utilizing UEP. We observe that, in general, the Priority Scheduling mechanisms manage to transmit more video information than the Equivalent Network mechanism on similar network setups.

Average Number of Transmitted Layers	SchLayer	EqLayer	SchPath
Four Paths Scenarios	1.6	1.15	1.62
Three Paths Scenarios	1.55	1.23	1.53
Two Path Scenarios	1.6	1.18	1.55

TABLE 5.4: *Average number of transmitted video layers for UEP-based algorithms in various network scenarios.*

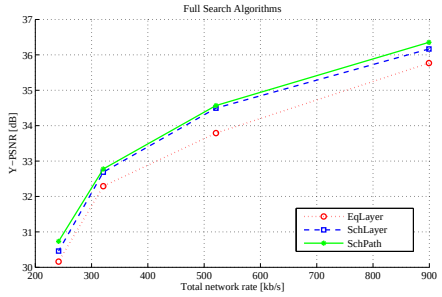


FIGURE 5.11: *Full search algorithms performance for different video encoding rates.*

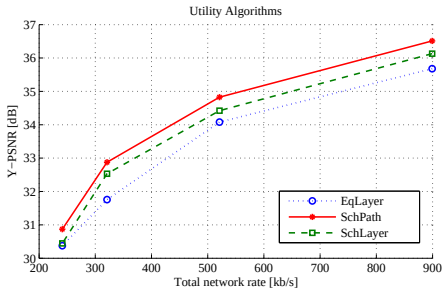


FIGURE 5.12: *Utility algorithms performance for different video encoding rates.*

5.7.4 Full Search vs. Utility algorithms

Finally, we compare the performance of the proposed heuristic algorithms based on utility, to the full search ones. On the same network setups, we run both the full search and utility algorithms for bitstreams encoded at various bitrates. Figure 5.11 and Figure 5.12 present the averaged PSNR results for the Priority Scheduling mechanisms. We observe that the heuristic utility based algorithms have a performance that is similar to the one of the full search, while they require a much smaller computation effort.

The good performance of the heuristic algorithms is naturally motivated by the assumptions we made on the encoding format (e.g., video layers are decoded in a sequential manner, and higher layers cannot be decoded unless previous layers have already been decoded), and the previous results showing the optimal unequal error protection based on the importance of each video layer.

Finally, we consider the performance of real systems where the choice of FEC codes is limited to a finite available set. Let the sender be able to access any of the following FEC codes: RS(20, 16), RS(20,12) and RS(20,8) in order to protect the transmitted media packets. We test the utility based algorithms constrained by the available set of FEC codes, and we compare the obtained results to the optimal ones found by the full search. Table 5.5 summarizes the results averaged over 100 simulation runs for one video bitstream.

Compared to previous results we observe a slight degradation in algorithm performance compared to the optimal full search results. This is explained by the lack in flexibility in the FEC mode choice. An the same time, we observe that full utilization of network resources is no longer optimal. Depending on the algorithm, only a fraction of the network bandwidth is utilized in order to achieve the optimal result. Hence, flooding the network with data and redundant packets in not

	SchLayer	EqLayer	SchPath
Full Search Distortion (MSE)	9.34	11.046	9.35
Constrained Utility Distortion (MSE)	10.898	11.987	14.118
Constrained Utility Resource Utilization (%)	76%	74%	89%

TABLE 5.5: *Algorithm performance in systems scenarios with limited choice of FEC parameters, and percentage of total network resources utilized.*

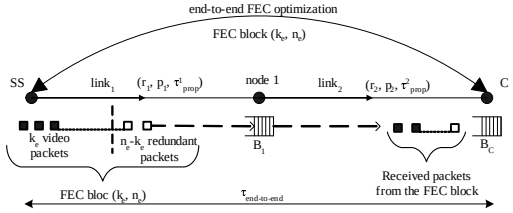


FIGURE 5.13: *End-to-end FEC scenario*

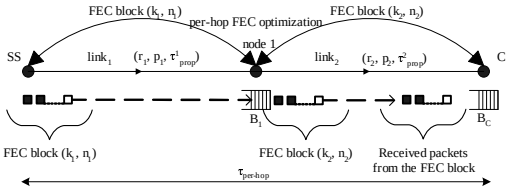


FIGURE 5.14: *Per-hop FEC scenario*

optimal, unless the designed system has full flexibility in the choice of FEC and scheduling strategies. This observation is in line with our previous results on path selection and rate allocation, presented in the previous chapters.

5.8 Active Networks

In this section we address the same joint source-channel rate allocation problem in active networks where intermediate nodes are able to perform basic FEC decoding/encoding operations. FEC performance is analyzed in the case of hop-by-hop FEC protection, and compared with an end-to-end FEC scenario, in order to demonstrate the benefits of FEC operations in the intermediate nodes.

We consider a simplified network model consisting of one path between the server and the client made of multiple links L_u that connect intermediate nodes $i-1$ and i. The intermediate nodes are able to perform FEC encoding/decoding operations. The intermediate nodes i and the client have buffers assumed to be large enough to prevent overflow, and the server S is aware of the parameters of all the links L_u along the path to the client C. Within this context, two scenarios are studied, where the intermediate nodes either transparently forward packets, or provide simple FEC operations. These scenarios are represented in Figure 5.13 and Figure 5.14, respectively.

Given the single path network model, we consider a simplified version of the end-to-end distortion model in Eq. (5.4), which takes into account the total media encoding rate and the network packetization effects over a single transmission path. It can be written as :

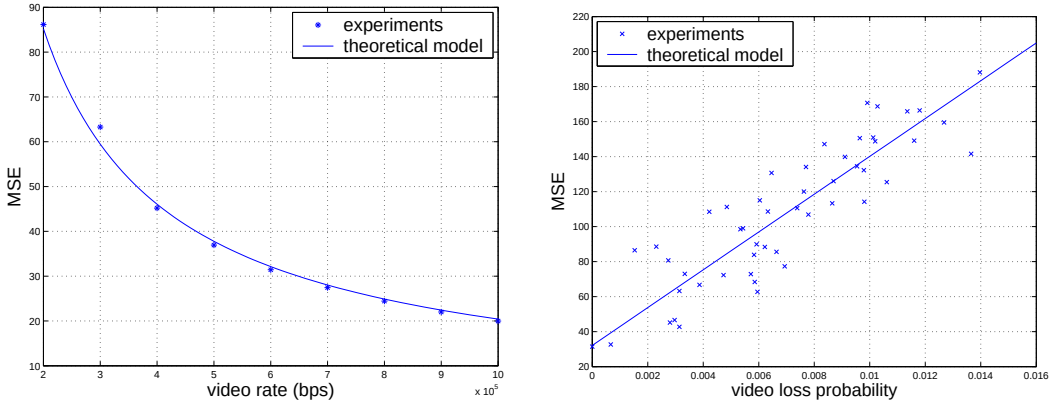
$$D = \alpha R^{\xi} + \beta R \pi,$$

where the first term of the sum represents the source distortion D_S, and the second term is the loss distortion D_L.

We validate the distortion model for the particular case of the MPEG-4 video streaming, where the decoder implements basic error concealment functions. The *foreman.cif* sequence (300 frames) is encoded at 30 *fps* with an interval of 15 frames between I-frames, and the packet size is set to 500 Bytes. Figure 5.15(a) presents the comparison between our theoretical model and the experimental results in the case of no loss, while Figure 5.15(b) shows the distortion as a function of the packet loss probability for a given video rate. It can be seen that the experimental data fits quite well the analytical values, and similar behavior has been observed for different video rates.

Under the FEC assumptions presented in Section 5.3, the scenario under consideration becomes the following. A streaming media server S sends live or stored media content to a receiver C. The media (e.g., video) is encoded and sent through the network in blocks of packets. The video packets are protected with FEC packets, forming FEC blocks. All packets (media and FEC) have an average size of M bytes, and the encoding format allows each data packet to be decoded independently from the others, possibly with some distortion (i.e., we use all received video packets).

The end-to-end quality optimization problem becomes the following: Given (i) the characteristics (ρ_u, θ_u and t_u) of all links L_u, and (ii) a maximum end-to-end delay Δ in the transmission of one video packet, find the optimal transmission scenario S^*, or equivalently the optimal FEC parameters $\vec{k^*}$ and $\vec{n^*}$, that minimize the end-to-end distortion D :



(a) RD-curve, $p = 0$ (b) video rate =600kbps

FIGURE 5.15: *Validation of the theoretical distortion model for* $\alpha = 4.3214 \cdot 10^6$, $\xi = -0.8876$ *and* $\beta = 18 \cdot 10^{-3}$

$$(\vec{k^*}, \vec{n^*}) = \arg \min_{\vec{k},\vec{n}} \left(\alpha R(\vec{k}, \vec{n})^{\xi} + \beta R(\vec{k}, \vec{n}) \pi(\vec{k}, \vec{n}) \right), \tag{5.18}$$

under the constraint $R \leq \min(\rho_u)$ and a maximum transmission delay below Δ. $(\vec{k}, \vec{n})$ represent the vectors of FEC parameters for the links in the streaming path.

The next subsection presents an analytical study of the loss probabilities and transmission delays in the two streaming policies, that will eventually allow to solve the optimization problem. It concentrates on a simple network topology where the path from the server to the client consists of two links and one intermediate node. However, the study can easily be generalized to any topology with multiple hops.

5.8.1 FEC Performances

5.8.1.1 End-to-End FEC Protection — In the case of end-to-end FEC protection in a topology like the one in Figure 5.13, the server sets the parameters (k, n) based on its knowledge about the network status. The intermediate node acts as a simple router and transparently forwards the received packets on the second link. Hence, the media rate is equivalent to: $R = \frac{k}{n} \min(\rho_1, \rho_2)$ and the transmission delay becomes:

$$\tau(k, n) = t_1 + t_2 + \frac{nM}{\min(\rho_1, \rho_2)} ,$$

where $\frac{nM}{\min(\rho_1, \rho_2)}$ represents the transmission time of a complete n-packet FEC block. Without loss of generality, we assume here that the time required for FEC coding can be neglected.

The video loss rate π, as seen by the receiver after FEC recovery is expressed as:

$$\pi = \frac{\sum_{i=1}^{k} i p_i(k, n)}{k},$$

where $p_i(k, n)$ is the probability of losing i video packets on the two links, after FEC recovery. It is computed as the probability of losing i video packets and at least $\lfloor n-k-i+1 \rfloor$ FEC packets, on either the first or the second link. For a uniform and independent loss process, it yields :

$$p_i(k,n) = \sum_{a=0}^{i} \binom{k}{a}\binom{k-a}{i-a}\theta_1^a\theta_2^{i-a}(1-\theta_1)^{k-a}(1-\theta_2)^{k-i}$$
$$\sum_{j=0}^{n-k}\sum_{b=\lfloor c-i+1\rfloor}^{c}\binom{n-k}{j}\binom{c}{b}\theta_1^j\theta_2^b(1-\theta_1)^c(1-\theta_2)^{c-b},$$

where $\theta_{1,2}$ respectively represent the loss probability on the first and second link, and $c = n-k-j$.

The extension of the two links case to the more general case of N links and $N-1$ intermediate routers is straightforward. The media rate is equivalent to : $R = \frac{k}{n}\min(\rho_1, ..., \rho_N)$ and the transmission delay becomes:

$$\tau(k,n) = \sum_{u=1}^{N} t_u + \frac{nM}{\min(\rho_1, ..., \rho_N)}.$$

The expression of $p_i(k,n)$ can be easily computed in an iterative way, but is omitted here due to the lack of a closed form expression.

5.8.1.2 Hop-by-hop FEC Protection — In the case of hop-by-hop FEC protection, the losses can be isolated on the various links, at the price of a possible larger end-to-end delay. The server and the intermediate nodes can set different FEC parameters (k_u, n_u), individually for each link L_u (see Figure 5.14). The sizes of the FEC blocks are however constrained by a maximum end-to-end delay. The media rate is given by $R = \min(\frac{k_1}{n_1}\rho_1, \frac{k_2}{n_2}\rho_2)$, and the total delay can be written as :

$$\tau(k_1,n_1,k_2,n_2) = t_1 + t_2 + \frac{M}{R}(k_1+k_2) + \tau_w^1,$$

where $\frac{M}{R}(k_1+k_2)$ represents the transmission time of the FEC blocks (k_1,n_1) and (k_2,n_2) on the first and respectively second link. If the loss probability on the first link is larger than 0, there is a non-zero probability that the intermediate node waits forever before it receives enough media packets to fill in k_2 slots in the n_2-packet FEC block. To avoid such a scenario, a limit is set in the intermediate node, that will send available data after τ_w^1. We set this limit to be equivalent to the average waiting time in the intermediate node, $\tau_w^1 = \lfloor\frac{k_2}{k_1(1-\pi^1(k_1,n_1))}\rfloor$. Experiments have shown that this value is in general sufficient to absorb the packet losses on the first link. In the very low probability case where the waiting time is larger than τ_w^1, the FEC parameters on the second link can be slightly different than (k_2,n_2), with a small impact on the hop-by-hop FEC performance.

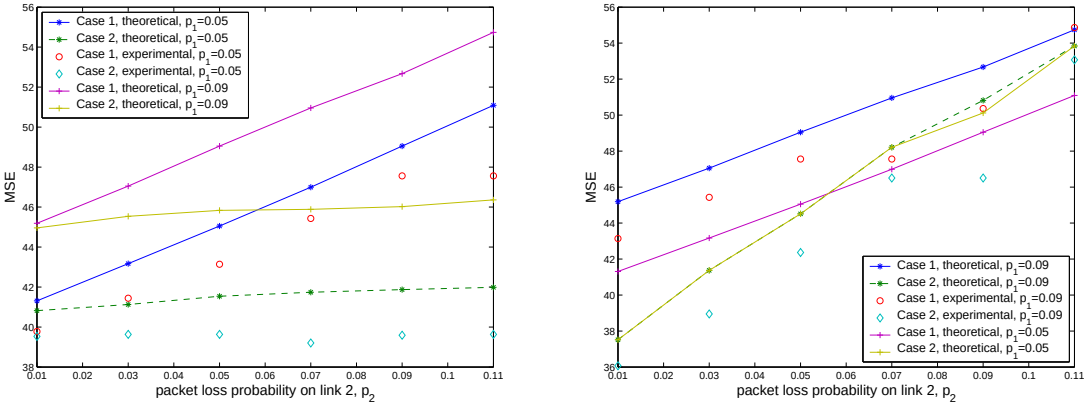
Since the loss processes on the two links are isolated due to the FEC decoding/encoding operations at the intermediate node, the overall media loss rate, as seen by the receiver can be expressed as:

$$\pi(k_1,n_1,k_2,n_2) = \pi^1(k_1,n_1) + \pi^2(k_2,n_2)(1-\pi^1(k_1,n_1)),$$

where $\pi^1(k_1,n_1)$ and $\pi^2(k_2,n_2)$ are the video loss rates after FEC recovery on each individual link. They are given by $\pi^u(k,n) = \frac{\sum_{j=1}^{k} j p_j(k,n)}{k}$, where $p_j(k,n)$ is the probability of losing j media packets out of the FEC block (k,n) after FEC recovery, and can be computed individually for each link u as in Section 5.3.

For the general case of N links and $N-1$ intermediate nodes the media rate is: $R = \min(\frac{k_1}{n_1}\rho_1, ..., \frac{k_N}{n_N}\rho_N)$ and the total delay becomes:

$$\tau(\vec{k},\vec{n}) = \sum_{u=1}^{N}(t_u + \frac{M}{R}k_u) + \sum_{u=1}^{N-1}\lfloor\frac{k_{u+1}}{k_u(1-\pi^u(k_u,n_u))}\rfloor.$$



(a) $r_1 = 600\ kbps$, $r_2 = 1\ Mbps$ (b) $r_1 = 1\ Mbps$, $r_2 = 600\ kbps$

FIGURE 5.16: *Minimal distortion in end-to-end and hop-by-hop FEC scenarios*

In the same time, the overall media loss rate can be expressed as:

$$\pi(\vec{k},\vec{n}) = \pi^1(k_1,n_1) + \sum_{u=2}^{N} \pi^i(k_u,n_u) \prod_{j=1}^{u-1} (1-\pi^j(k_j,n_j)).$$

Having the expressions for $R(\vec{k},\vec{n})$, $\pi(\vec{k},\vec{n})$ and $\tau(\vec{k},\vec{n})$, we can solve the optimization problem. Knowing that FEC performs better with the increase in the block size, we can implement an efficient search algorithm for the optimal solution by limiting the feasible search space for the $(\vec{k},\vec{n})$ parameters. The search space of the $\vec{n}$ parameters is greatly reduced based on the delay constraint Δ, while the search space for the $\vec{k}$ parameters is limited knowing the loss probabilities θ_i on all the links. Results are presented in Section 5.8.2.

5.8.2 Results

link 1		link 2		Case 1		Case 2			
r_1	p_1	r_2	p_2	k	n	k_1	n_1	k_2	n_2
700	6	400	1	15	20	9	17	8	9
500	2	700	7	18	25	13	16	7	12
800	5	800	5	29	40	15	21	13	18
1000	9	600	3	20	30	12	25	11	14
600	5	1000	9	19	30	17	22	6	12

TABLE 5.6: *Optimal $(\vec{k},\vec{n})$ for end-to-end (Case 1) and hop-by-hop (Case 2) FEC protection, as a function of r_i [kbps] and p_i [%].*

For the same simulation setup as used for validating the distortion model, we now solve the optimization problem given from Eq. (5.18), and find the optimal $(\vec{k},\vec{n})$ parameters for the hop-by-hop FEC protection policy. They are then compared to the optimal parameters for the end-to-end FEC scenario. Table 5.6 presents the optimal values in the two cases for different parameters of the streaming path segments, where the maximal end-to-end delay has been set to $\tau_{max} = 0.2s$ and the propagation delays have been neglected. It can be seen that the FEC blocks are in general much smaller in the end-to-end case because of the end-to-end delay constraint. Also, the optimal FEC construction greedily uses all the available bandwidth on the highest rate links, in order to limit as much as possible losses on this particular link. Loss therefore occurs almost exclusively on the smaller rate segment.

Figure 5.16 compares the optimal performance of both FEC techniques in terms of average MSE distortion for different link parameters. As expected, the hop-by-hop protection performs much better than the end-to-end FEC policy. This is especially true for segments with very different characteristics, and the performance becomes similar when the path becomes homogeneous. Also, for stringent end-to-end delay constraints, it can happen that the end-to-end FEC protection performs better thanks to the increased flexibility in building longer blocks. It can be noted finally that the experimental results are slightly better than the theoretical ones. This phenomenon is due to a so low effective loss probability (thanks to the very good FEC protection), that even a high number of simulations can hardly reproduce. In the very small probability case where a FEC block cannot be decoded, the distortion becomes however high enough for the average behavior to jump on the theoretical curve.

5.9 Conclusions

In this chapter we address the problem of optimal joint source-channel rate allocation for multimedia streaming applications over lossy multipath networks. Based on different FEC and scheduling strategies for layered encoded video streaming we derive algorithms for the efficient computation of the source rate and forward error protection rate, with the final goal of optimizing the client perceived video quality. In a lossy multipath scenario with limited network resources we find optimal to perform a prioritized scheduling of the video layers according to their importance, on the best network paths first. In the same time, unequal error protection strategies that protect better the most important video information are shown to be more efficient. Our results confirm our results on path selection and rate allocation, presented in previous chapter. We also discuss real system implementations when the optimization problem is solved only on an available set of video rates and FEC strategies. We show that in such a case, flooding all available network paths is no longer optimal in terms of reconstructed media quality. Moreover, we discuss the same optimization problem in the context of active networks when intermediate nodes can perform basic operations on the passing data flow, e.g. FEC decoding and re-encoding. We show the benefit of in-network flow processing especially in the case of heterogeneous networks, when different network segments belonging to the same end-to-end network path have different network parameters.

Chapter 6

Media Packet Scheduling for Multipath Streaming

6.1 Introduction

In previous chapters we have discussed the problem of multipath streaming in flow networks, and we have provided efficient solutions for path selection and rate allocation. We have seen that in general, due tot the error-prone nature of the transmission medium, only a small number of network paths (hence limited streaming rate) are used for the streaming application. At the same time, we have discussed the efficient distribution of network resources between effective streaming rate and forward error correction rate for increased tolerance to network erasures. The efficiency of multipath video streaming is however tied to the packet transmission strategy, which aims at offering an optimal quality of service in delay-constrained video applications.

This chapter addresses the problem of video packet scheduling in multipath network scenarios, under playback delay and buffer constraints. It aims at efficiently distributing the video information on the available network paths, while judiciously trading off playback delay and distortion at the receiver. We consider the selection of inter-dependent video packets to be transmitted (or equivalently the adaptive coding of the video sequence), and their scheduling on the available network paths, in order to minimize the distortion experienced by the end-user. The complex distortion optimization problem is a priori NP-complete, and no method can solve it in polynomial time [196]. With help of heuristics from constrained multipath streaming scenarios, we propose a polynomial complexity algorithm for efficient video scheduling in practical scenarios.

Assuming a simple streaming model, which captures the unequal importance of video packets and their dependencies, we propose a detailed analysis of timing constraints imposed by delay sensitive streaming applications. This analysis allows us to identify sets of valid, or feasible transmission policies, which compete for the distortion optimized multipath streaming solution. The optimal strategy is computed based on a modified branch and bound algorithm [186] that applies search and pruning methods specific to the multipath streaming problem. The method greatly reduces the complexity of the computations compared to a full search over the policy space, and still provides an optimal solution. However, there is no guarantee that it performs in polynomial time for every instance of the problem, and we rather use it as a benchmark for other streaming algorithms. Hence, we propose a heuristic-based approach to the optimization problem, based on load-balancing techniques, which leads to a polynomial time algorithm. This fast scheduling algorithm is finally adapted with sliding window mechanisms, to the case of real time streaming where the server only has a partial knowledge about the packet stream. Simulation results demonstrate close to optimal performances of the fast scheduling solution, for a large variety of network scenarios. Compared to state-of-the-art algorithms, it offers smaller quality variations on dynamic bandwidth channels, and preserves a minimal quality level by improved scheduling.

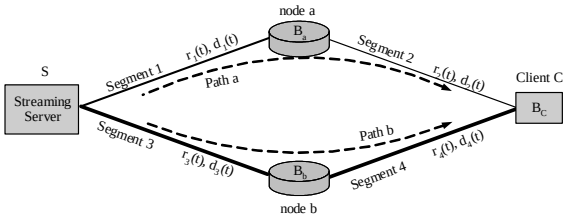


FIGURE 6.1: *Multipath Streaming Scenario. The client accesses the streaming server simultaneously through two different paths, each one composed of two segments with intermediate buffers.*

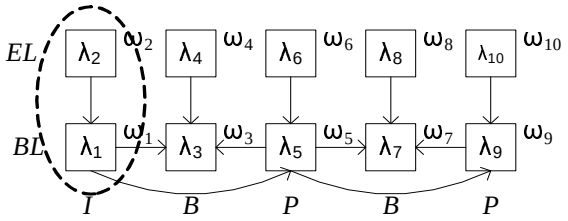


FIGURE 6.2: *Directed acyclic dependency graph representation for a typical MPEG layered-encoded video sequence (one network packet per layer, with IPBPB format).*

Interestingly enough, the performance of the real time scheduling algorithm stays quite consistent, even for small video prefetch windows, and for low accuracy in the channel bandwidth prediction. This extends the validity of our algorithm to multipath live streaming systems with stringent delay constraints, and simple bandwidth prediction methods.

The main contributions presented in this chapter are threefold. First, we study video packet scheduling in a rate-distortion multipath streaming scenario, taking into account possible buffer constraints in each intermediate network nodes. Since congestion is the main cause of loss, it certainly becomes primordial to respect the buffer constraints in network nodes, in order to design efficient streaming systems. Second, we propose an optimal solution for the distortion optimization problem, which takes into account the non-stationary nature of the video sequence, the packet dependencies introduced by the encoding algorithm, and the network status. This optimal solution allows to bound the performance of scheduling algorithms. Finally, we present a novel polynomial time algorithm that provides performances similar to the optimal streaming strategy. This algorithm is eventually adapted to real time scenarios, with more restrictive delays, and to cases where the accuracy in the prediction of the channel status is reduced. It still offers interesting performances in such cases, and thus provides a very efficient solution for multipath video streaming applications.

This chapter is organized as follows. Section 6.2 describes our multipath streaming model and introduces the notation used in the distortion optimization problem. The packet scheduling problem is analyzed in detail in Section 6.3. Based on this timing analysis, we propose both optimal and fast heuristic-based algorithms to solve the distortion optimization problem in Section 6.4. Simulation results are presented in Section 6.5, and we conclude in Section 6.6.

6.2 Multipath Video Streaming

6.2.1 General Framework

We consider the simple multipath network topology represented in Figure 6.1. The client C requests a media stream from a streaming server S, which transmits the requested bitstream via two disjoint paths. Each network path consists in two segments connected through an intermediate node that simply forwards, after a possible buffering delay, incoming packets from the first segment,

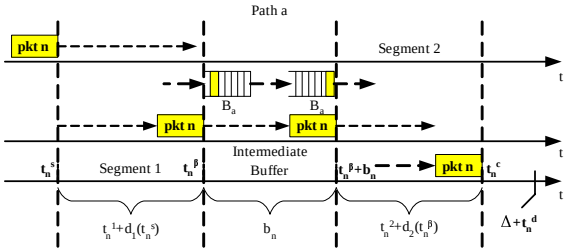


FIGURE 6.3: *Time diagram for packet λ_n sent on path a.*

towards the client on the second segment. The intermediate nodes represent network streaming proxies, edge servers or peers, for example. The streaming server is connected to the channels through buffer interfaces, which are modelled as FIFO queues. Thus, the channels drain the packets from the buffers, in the same order in which the server places them into the buffers. The network channels between the server and the client are represented as variable bandwidth, lossless links. The variable nature of the bandwidth implies that the rate at which the channels drain data placed in the server's buffers, changes as a function of time. At the other end, the client waits for an initial playback delay Δ after its request for a stream has been acknowledged. It then starts decoding the media stream, and plays it continuously.

During the streaming session, the server selects a subset of the pre-encoded media packets to communicate to the client, taking into account the available bandwidth on the different network paths, and buffer fullness in the nodes, or at the receiver. The segment bandwidth, latency and intermediate buffer fullness can be estimated at the server, or reported by various methods (e.g., as in [16]). The work presented in this chapter rather addresses the selection of the packets that should be communicated to the client, as well as the network path they need to follow. It actually does not even require an exact knowledge of the channel bandwidth, but accurate network information yet increases the performance of the streaming system. Finally, the network topology could present several disjoint paths, and several nodes on each path. However, for the sake of clarity, we consider in the problem formulation only the two-path scenario presented in Figure 6.1. The extension to scenarios with a larger number of paths, is straightforward.

6.2.2 Streaming Model and Notations

In the multipath streaming topology represented in Figure 6.1, each network segment i is characterized by an instantaneous rate $r_i(t)$ and an instantaneous latency $d_i(t)$. The rate $r_i(t)$ is the total bandwidth allocated to the streaming application on segment i at time instant t. Equivalently, we denote the cumulative rate on segment i, up to time instant t, by $R_i(t) = \int_0^t r_i(u)du$. Additionally, the streaming server assumes that no packet is lost on the network segments, except those induced by late arrivals or buffer overflows, and that the order of the packets is not changed between two successive nodes. These assumptions are quite realistic in most of today's wired streaming networks. The intermediate nodes $\{a, b\}$ have buffers of capacity B_a and respectively B_b, which are available for the streaming session. The client has a playback buffer of capacity B_c. We first assume that all segment rates and latencies along with intermediate buffer capacities are accurately predicted by the server at all time instants, possibly with feedback of the overlay nodes. We will eventually relax that assumption to consider realtime streaming scenarios.

The video sequence is encoded into a bitstream using a scalable (layered) video encoder. The bitstream is then fragmented into network packets under the general rule stating (i) that each network packet contains data relative to at most one video frame, and (ii) that an encoded video frame can be fragmented into several network packets. Let $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_N\}$ be the chronologically ordered sequence of N network packets, after fragmentation of the encoded bitstream. Each network packet λ_n is characterized by its size s_n in bytes, and its decoding timestamp t_n^d. From the client viewpoint, all the video packets are not equivalently valuable, due to the non-stationary

nature of the video information. Therefore, each network packet can be characterized by a weight ω_n, which represents the reduction in the distortion perceived by the client, in the case where packet λ_n is successfully decoded. We refer to a successfully decoded packet as a network packet that is received and correctly decoded by the client before its decoding deadline.

Additionally, in most video encoding schemes, packets generally have dependencies between them. In other words, the successful decoding of one packet λ_n is contingent on the successful decoding of some other packets, called *ancestors* of λ_n. The successful decoding of one packet may depend on the correct decoding of several ancestors, and we denote by A_n, the set of ancestors of packet λ_n. Such dependencies can be represented by a directed acyclic dependency graph [117], as shown in Figure 6.2. The nodes in the graph represent the network packets and are characterized by their individual weights, and directed edges represent dependencies between packets and their ancestors.

We denote by $\pi = (\pi_1, \pi_2, ..., \pi_N)$ the transmission policy adopted by the streaming server, and by Π be the set of all the feasible policies π. The policy π_n used for packet λ_n consists in a couple a variables $[q_n, t_n^s]$ that respectively represent the path q_n chosen for packet λ_n, and its sending time t_n^s. It completely characterizes the server behavior with respect to packet λ_n under the general policy vector π. In the multipath network scenario presented above, the server can decide to send packet λ_n on paths a or b, or simply to drop the packet without sending it. Therefore, the action imposed on packet λ_n can be written as:

$$q_n = \begin{cases} a & \text{if packet } \lambda_n \text{ is sent on path } a \\ b & \text{if packet } \lambda_n \text{ is sent on path } b \\ 0 & \text{if packet } \lambda_n \text{ is dropped.} \end{cases}$$

Let Π be the set of all the feasible policies π, in the network scenario under consideration. Remember that packets are sent sequentially on a path, and that the streaming strategy aims at avoiding buffer overflows that would result in packet loss.

Finally, in our streaming model, a packet is decoded by the receiver only if its arrival time, t_n^c, is smaller than its decoding deadline, i.e., if $t_n^c \leq t_n^d + \Delta$ where t_n^d represents the decoding timestamp of packet λ_n, and Δ is the playback delay at client. We assume here, without loss of generality, that the client request has been sent at time $t = 0$, and that the decoding timestamp of the first packet p_1 is set to 0. The processing time at the receiver is further neglected. Under these assumptions, and taking into account packet dependencies, the successful decoding of a packet λ_n under the streaming strategy $\pi \in \Pi$, can be represented by the binary variable $\varphi_n(\pi)$, where $\varphi_n(\pi)$ is equal to 1 if the packet arrives on time at the decoder, and if all its ancestors have been successfully decoded. We further take into account the difference between frame order in the bitstream and the decoding order of the frames at the client. This impacts, for example, the scheduling of a B frame that is placed in the bitstream after the future P frame it depends on. In other words, we can write:

$$\varphi_n(\pi) = \begin{cases} 1 & \text{if } \begin{cases} q_n \neq 0 \\ t_n^c \leq t_n^d + \Delta \\ \varphi_m(\pi) = 1, \forall \lambda_m \in A_n \\ \text{ at time } t_n^d + \Delta \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

The overall benefit Ω of the streaming strategy $\pi \in \Pi$, which is equivalent to the quality perceived by the receiver, can now simply be expressed as the sum of the weights ω_n of all successfully decoded packets. We assume that packets whose $\varphi_n(\pi) \neq 1$ are simply discarded at the client, hence the overall benefit can be written as $\Omega(\pi) = \sum_{\forall n:\varphi_n(\pi)=1} \omega_n$.

6.2.3 Distortion Optimization Problem

Given the abstraction model of the encoded video bitstream, the distortion optimization problem consists in an efficient selection of the subset of video packets to be transmitted, jointly with

their streaming policy. We assume a server-driven scenario in which the server is aware of, or can estimate the network conditions (i.e., $r_i(t)$ and $d_i(t)$), at each time instant. The server then only schedules for transmission packets that can arrive at the client before their decoding deadline. The streaming server considers that the transmission links are lossless, and that packet loss only happens due to buffer overflow, or late arrival.

The distortion optimization problem can be stated as follows: *Given* Λ, the packetized bit-stream of an encoded video sequence, Δ, the maximum playback delay imposed by the client, and the network state, *find* the optimal transmission policy $\pi^* \in \Pi$ that maximizes the overall quality measure Ω. The optimization problem translates into finding $\pi^* \in \Pi$ s.t.:

$$\Omega(\pi^*) = \max_{\pi \in \Pi} \sum_{\forall n:\varphi_n(\pi)=1} \omega_n.$$

The optimization problem can be easily reduced to the more general case of optimal scheduling problems. This family of problems proves to be NP-complete [196] and an optimal algorithm that solves them in polynomial time does not exist. Hence, we still propose an optimal algorithm that efficiently finds the distortion minimal streaming strategy for long video sequences, to be used as a benchmark for faster, sub-optimal methods. We then design a heuristic-based algorithm that provides close to optimal performance, but in polynomial time, and we eventually apply it to realtime streaming scenarios.

6.3 Packet Scheduling Analysis

6.3.1 Unlimited Buffer Nodes

This section proposes an in-depth analysis of the scheduling of packets in the streaming model described above, and computes the parameters necessary to solve the distortion optimization problem. Our approach represents a segment-by-segment analysis of the network behavior, including intermediate nodes buffers. This approach is a first step towards a more comprehensive analysis of network behavior related to the specificities of video streaming applications. In general, the particular characteristics of media packet streams, like timing issues or unequal importance of data, prevent the application of general end-to-end analysis like [14] in such scenarios.

We consider first the case where buffering space in the network nodes and the client is not constrained, i.e., $B_a = B_b = B_c = \infty$. The server has the knowledge of N video packets, where N can be the total number of network packets of the video stream (in the case of stored video), or simply the number of packets contained in the prefetch window in real-time streaming. The server is able to transmit network packets simultaneously on the two network paths. Under the assumption of unlimited buffer space, the server can send packets on each of the paths at the maximum rates of the first segments ($r_1(t)$ for path a or $r_3(t)$ for path b, see Figure 6.1).

Under a given policy π, the sending time t_n^s of each packet λ_n can thus be easily computed. Suppose that λ_n is sent on path a (i.e., $q_n = 1$). Let $S_n^a(\pi) = \sum_{m<n, q_m=1} s_m$, $S_n^a(\pi)$ represent the cumulative size of all the packets that need to be sent on path a before λ_n, under the policy π. Under the assumption that the available bandwidth is fully utilized by the streaming application, t_n^s is the shortest time t at which the cumulative rate $R_1(t)$ is larger than S_n^a :

$$t_n^s(\pi) = \arg\min_t |R_1(t) - S_n^a(\pi)|. \tag{6.1}$$

In other words, the packet λ_n can only be sent when all the previous packets scheduled on the same path have been transmitted. It will then arrive at the client after a certain delay, caused by the transmission delays (t_n^1 and t_n^2) on the 2 segments that compose path a, the latencies introduced by the two links ($d_1(t)$ and $d_2(t)$) and the queuing time at the node b_n. Therefore, the time instant at which packet λ_n enters the node buffer can be expressed as $t_n^\beta = t_n^s + t_n^1 + d_1(t_n^s)$.

The arrival time of packet λ_n at the client, can be written as $t_n^c = t_n^\beta + b_n + t_n^2 + d_2(t_n^\beta)$. The timing representation of the transmission of packet λ_n is provided in Figure 6.3.

The transmission delays t_n^1 and t_n^2 represent the time needed to send packet λ_n, at the bandwidth available on path a. They have to verify:

$$R_1(t_n^s + t_n^1) - R_1(t_n^s) = R_2(t_n^\beta + t_n^2 + b_n) - R_2(t_n^\beta + b_n) = s_n,$$

and can be computed similarly to Eq. (6.1). The queuing time b_n corresponds to the time needed to transmit the $B(t_n^\beta)$ bits present in the buffer, at time t_n^β when packet λ_n enters the buffer. The buffer fullness can be computed recursively as

$$B(t_n^\beta) = \max[B(t_{n-1}^\beta) + s_{n-1} - R_2(t_n^\beta) + R_2(t_{n-1}^\beta),\ 0].$$

Therefore, the queuing time can be computed such that it satisfies $R_2(t_n^\beta + b_n) - R_2(t_n^\beta) = B(t_n^\beta)$. Note that, even if the previous development only consider the path a, the extension of the analysis to the packets transmitted over path b is straightforward. The arrival time of packet λ_n, t_n^c is thus fully determined. The minimal playback delay $D(\pi)$ induced by the transmission policy π can finally be expressed as:

$$D(\pi) = \max_{1 \le n \le N}(D_n(\pi)) = \max_{1 \le n \le N}(t_n^c - t_n^d),$$

where $D_n(\pi)$ is the playback delay imposed by the streaming process up to packet λ_n by the transmission policy π. Interestingly, the playback delay is a non-decreasing function of the packet number n. That property expressed in Lemma 6.3.1, will be advantageously used in the scheduling optimization problem.

Lemma 6.3.1. *Given that the streaming server sends the N network packets in parallel on two paths, and that on each path the packets are sent sequentially, the playback delay $D_n(\pi)$ under the given policy vector π is a non-decreasing function of n.*

Sketch. Observe that $D_n(\pi)$ can be expressed as a recursive function of n:

$$D_n(\pi) = \max(D_{n-1}(\pi), t_n^c - t_n^d) \tag{6.2}$$

Hence, $D_i(\pi) \le D_n(\pi)$, $\forall n$, $\forall i$ such that $0 \le i \le n \le N$, with: $D_0(\pi) = 0$ and $D(\pi) = D_N(\pi)$. □

Let us finally define the cumulative quality $\Omega(\pi)$, resulting from the streaming policy π. In a perfect transmission where the set of packets Λ is entirely transmitted, the quality is denoted by $\Omega_0(\pi) = \sum_{n=1}^{N} \omega_n$. Due to delay or bandwidth constraints, the server may decide to drop some packets from Λ. In this case, we iteratively compute the cumulative quality, $\Omega_n(\pi)$, which is decremented each time a packet is dropped. It can be written as :

$$\Omega_n(\pi) = \begin{cases} \Omega_{n-1}(\pi) & \text{if } \varphi_n(\pi) = 1 \\ \Omega_{n-1}(\pi) - \omega_n & \text{otherwise} \end{cases} \tag{6.3}$$

with $\Omega(\pi) = \Omega_N(\pi)$. While Eq. (6.3) does not explicit the influence of other packets that have packet λ_n as their ancestor, the status $\varphi_n(\pi)$ of packet λ_n, directly affects the status of all packets dependent on λ_n.

Lemma 6.3.2. *Ω_n is a non-increasing function of the packet number n.*

Sketch. Observe that ω_n is by definition a non negative value. Hence, $\Omega_n \le \Omega_i$, $\forall n \le N$, $\forall i \le n$. □

The two properties expressed in Lemmas 6.3.1 and 6.3.2 are used later in the derivation of efficient search algorithms for the optimal scheduling policy.

6.3.2 Constrained Buffer Nodes

A similar timing analysis can be performed in the case where the buffering space in the intermediate nodes on each path is limited to B_a and B_b respectively. The buffer capacities in the intermediate nodes may significantly influence the optimal packet scheduling strategy in multipath streaming scenarios. In contrary to single path scenario, the overall packet scheduling is not necessarily sequential any more, which allows to use buffers as a form of staging step. Buffers allows for smoothing bandwidth fluctuations between successive path segments, when delay constraints permit it.

We reasonably assume that the buffering space is larger than any video packet in Λ. B_a and B_b represent the buffer sizes allocated by the intermediate nodes to the streaming process and they are known by the server. The server estimates the buffer fullness based on its knowledge about the network bandwidth, or with help of feedbacks from intermediate overly nodes. It tries to avoid buffer overflows by adapting the sending time of each packet to the buffer fullness. Note that it may no longer use the full available bandwidth, without risking to lose packets.

The streaming policy has to take into account these new constraints. In particular, if packet λ_n has to be transmitted on path a under policy π, its sending time t_n^s is such that there is enough buffer space available when it reaches the intermediate node. Additionally, the packet λ_n can only be sent when all the previous packets on the same path have been transmitted. Using the same notation as defined hereabove, t_n^s becomes the smallest value that simultaneously verifies the following conditions :

$$\begin{cases} R_1(t_n^s) \geq S_n^a(\pi) \\ t_n^s + t_n^1 + d_1(t_n^s) \geq \tau_n \end{cases} \tag{6.4}$$

where τ_n represents the earliest time at which there is enough space in the intermediate buffer to receive packet λ_n, when the buffer is drained at a rate $r_2(t)$. Equivalently, τ_n can be computed recursively, since it verifies the inequality

$$B_a - (B(t_{n-1}^\beta) + s_{n-1} - R_2(\tau_n) + R_2(t_{n-1}^\beta)) \geq s_n.$$

We can also define the maximum buffer occupancy during the whole streaming process as

$$B_a^{max}(\pi) = \max_{1 \leq i \leq N} (B(t_i^\beta)) \leq B_a.$$

The timing analysis on path b follows immediately. The strategy π is thus completely defined, and we can compute $D(\pi)$ and $\Omega(\pi)$ similarly to the case with unlimited buffers. A similar reasoning can be applied in order to prevent buffer overflow at the client, in the case where the client also has a limited storage space.

6.4 Distortion Optimized Streaming

6.4.1 Optimal Solution: Depth-First Branch & Bound (B&B)

Since the sending and arrival times for each packet λ_n can be computed for a given transmission policy π (see Section 6.3), we can now search for the optimal packet scheduling π^* that maximizes the client video quality given an imposed playback delay. We first present an efficient algorithm that finds the optimal transmission policy vector π^* for a given encoded video sequence, network topology and playback delay. While being too complex to implement in practice, the algorithm is used as a performance benchmark for the development of sub-optimal, faster scheduling methods. The novelty of the algorithm resides in the use of branch and bound (B&B) methods [197] in a multipath video-streaming framework[1], and on adapting pruning rules to the specific characteristics of this scenario. The pruning rules make the algorithm much faster than a brute search but

[1]While B&B techniques have been used for years by the optimization community, they have only recently been employed in a streaming scenario [115, 198].

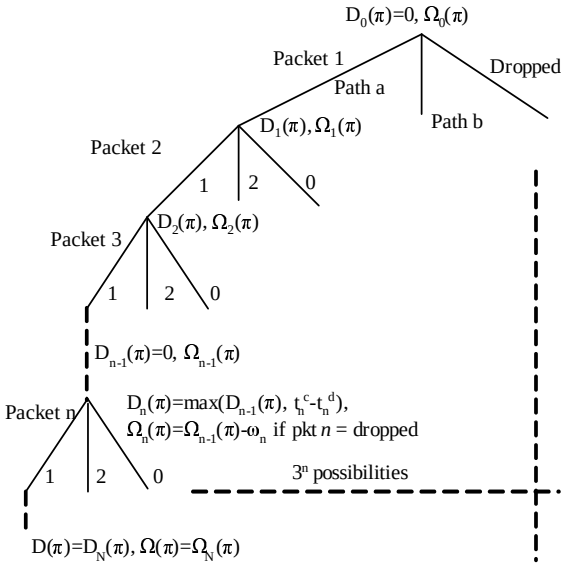


FIGURE 6.4: *Depth First Branch & Bound Algorithm*

still do not guarantee polynomial execution times on all streaming scenarios. The optimization problem still has a combinatorial complexity.

The scheduling of N packets on two available paths can be organized as a decision tree of depth N (Figure 6.4). At each stage n in the tree, packet λ_n can be sent on path a, on path b, or can be dropped. Hence, at depth N, the decision tree will contain 3^N leaves, according to the number of scheduling possibilities of the N packets on the 2 paths. At each stage n in the tree we can compute $D_n(\pi)$, the minimum playback delay and $\Omega_n(\pi)$, the cumulative video quality measure, for a partial scheduling up to packet λ_n, according to the recursive Eq. (6.2) and Eq. (6.3), presented in Section 6.3. This computation can be done for each one of the valid scheduling policies, for the first n packets. As mentioned in Section 6.3.1, $D_n(\pi)$ and $\Omega_n(\pi)$ are non-decreasing, and respectively non-increasing functions in n. These two functions are used to establish a fast search on the decision tree for the optimal transmission policy vector π^*. A depth-first search is performed on the decision tree, starting with an initial policy vector π that satisfies the delay constraint $D(\pi) \leq \Delta$, where Δ is the playback delay imposed by the client. The policy π becomes our initial optimal policy π^* with $\Omega^* = \Omega(\pi^*)$. The initial policy is computed using a simple Earliest Delivery Path First algorithm with a complexity of $O(N)$, similar to [135]. The EDPF algorithm schedules frames in a FIFO order. Packets belonging to a given frame are scheduled according to their importance ω_n, on the path that guarantees the earliest arrival time at the client. If a packet cannot be successfully scheduled, it is dropped without transmission, along with all his children packets, to avoid waste of network resources.

Since an EDPF strategy is often sub-optimal in a multipath scenario, we start searching the decision tree for better transmission policies, with $\Omega > \Omega^*$. We start with the leftmost transmission policy represented on the tree (equivalent to sending all packets on path a) and move through the decision tree towards right. For each new policy π', we compute $D_n(\pi')$ and $\Omega_n(\pi')$ successively for $n = 1..N$. At any packet λ_n for which $D_n(\pi') > \Delta$ or $\Omega_n(\pi') \leq \Omega^*$, the computation of $D_n(\pi')$ is stopped, and the decision tree is pruned for all policies that have the same scheduling up to packet λ_n (i.e., $\{\pi\}$ s.t. $\pi_i = \pi'_i$, $\forall i$, $1 \leq i \leq n$). If $D_N(\pi') \leq \Delta$ and $\Omega(\pi') \geq \Omega^*$, the policy π' becomes the new optimal policy π^* and $\Omega^* = \Omega(\pi')$. The operation is repeated until the set of all feasible policies Π represented on the decision tree has been covered. When the search is complete, the optimal policy π^* maximizes the video quality at the receiver and respects the playback delay constraints.

The B&B method provides an efficient way of computing the optimal transmission policy vector

π^*. The speed of the method depends on the pruning efficiency, which in turn, depends on the quality of the initial policy. However, the method is not scalable with N, since it cannot compute the optimal solution in polynomial time. The worst case complexity of the method remains $O(3^N)$. The extension of the algorithm to more paths follows easily. In the general case of K independent network paths between the streaming server and the client, the complexity grows to $O((K+1)^N)$.

6.4.2 Heuristic Solution: Load Balancing Algorithm (LBA))

Since the B&B algorithm may be too complex in practice, this subsection now presents a heuristic approach, which finds a close-to-optimal solution in polynomial time. The algorithm is inspired from load balancing techniques, which proved to be very effective in solving problems of task scheduling in multiprocessor systems [199]. In short, the algorithm performs a greedy scheduling of the most valuable packets first. Less valuable packets are scheduled only if the network capacity permits, and only if they do not lead to the loss of a more valuable packet already scheduled (due to subsequent late arrivals at the client).

First, the N network packets are arranged in descending order of their value. Hence, we obtain a new representation of the encoded bitstream, $\Lambda' = \{\lambda'_1, \lambda'_2, ..., \lambda'_N\}$, such that: $\omega_1(\lambda'_1) \geq \omega_2(\lambda'_2) \geq ... \geq \omega_N(\lambda'_N)$. Then, similarly to the EDPF algorithm, a greedy algorithm (see `Algorithm 5`), schedules the N ordered packets on the two network paths, while additionally taking care of the packet interdependencies. `Algorithm 5` presents the sketch of the complete algorithm, where, for the sake of clarity, we redefine the action imposed on packet λ'_n, q'_n as:

$$q'_n = \begin{cases} a & \text{if packet } \lambda'_n \text{ is sent on } \textit{path } a; \\ b & \text{if packet } \lambda'_n \text{ is sent on } \textit{path } b; \\ 0 & \text{if packet } \lambda'_n \text{ is dropped without sending;} \\ \infty & \text{if packet } \lambda'_n \text{ is not scheduled yet.} \end{cases}$$

To decide which action to take on each packet λ'_n, the algorithm first attempts to schedule all ancestors that have not been scheduled yet. If one of them cannot be scheduled, then the algorithm automatically drops the packet λ'_n. This ensures that our algorithm does not waste network resources on transmitting network packets that cannot be correctly decoded at the receiver.

All packets marked to be scheduled on a given path, are reordered according to their decoding deadlines before transmission. When a new packet is inserted, it triggers a new packet ordering. If a packet λ'_n can be scheduled on both network paths without interfering with the packets already scheduled, the algorithm will chose the path that offers the shortest arrival time for packet λ'_n. If packet λ'_n can only be scheduled on one path, the algorithm will insert the packet on that path. Otherwise packet λ'_n cannot be scheduled on any of the two paths, without interfering with the already scheduled packets, and the algorithm will drop packet λ'_n without transmitting it. Hence, the algorithm prevents that the transmission of one packet forces the loss of a more important packet previously scheduled, because of late arrival at the client. Note that in the case where the value of each network packet is directly proportional to the size of the packet, the algorithm offers a real load balancing solution for the two network paths.

`Algorithm 5` performs an initial ordering of the N packets in the new set Λ'. Any common sorting algorithm that works with complexity $O(N \log N)$ can be employed. Afterwards, for each packet λ'_n that must be scheduled, the algorithm requires a search among the packets already scheduled on each of the paths, in order to insert the new packet according to its decoding deadline. The operation requires $O(N)$ computations and is repeated N times, for each packet in Λ'. The complexity of the proposed algorithm is thus $O(N^2)$. For the more general case of K disjoint paths between the server and the client, the algorithm requires the computation of arrival times on all the paths, for all scheduled packets. The insertion of one packet therefore requires $O(KN)$ operations, and is performed for all N packets. The total complexity of `Algorithm 5` grows linearly with the number of network paths, being of $O(KN^2)$. In conclusion, the proposed heuristic algorithm has a complexity that grows linearly with the number of network paths K, and quadratic with the number of video packets N. However, it generally leads to suboptimal strategies due to the greedy optimization strategy. The extensive simulations presented in the next section show

Algorithm 5 Load Balancing Algorithm (LBA) for finding π

Input: Λ, ω_n, s_n, $1 \leq n \leq N$
Output: Suboptimal transmission policy vector π;

Initialization: Create Λ': arrange packets in order of importance ω_n;
$n := 1$;
while $n \leq N$ **do**
  if Packet λ'_n s.t. $q'_n = \infty$ **then**
    invoke Schedule_Packet(n);
  end if
  $n := n + 1$;
end while
Procedure: Schedule_Packet(n)
for all packets λ'_k in A_n s.t. $q'_k = \infty$ **do**
  invoke Schedule_Packet(k);
end for
invoke do_Schedule(n);
Procedure: do_Schedule(n)
if $\exists$ packet $\lambda'_k \in A_n$ s.t. $q'_k = 0$ **then**
  $q'_n = 0$;
  return;
else
  attempt the insertion of packet λ'_n on path a and on path b, ordered according to the decoding deadlines, without compromising the decoding of any other scheduled packet;
  if t_n^c(path a), t_n^c(path b) $\leq t_n^d + \Delta$ **then**
    choose the path with shorter t_n^c;
    set q'_n accordingly;
  else
    if t_n^c(path a), t_n^c(path b) $> t_n^d + \Delta$ **then**
      $q'_n = 0$;
    else
      schedule packet λ'_n on the path with $t_n^c \leq t_n^d + \Delta$;
      set q'_n accordingly;
    end if
  end if
end if

that the performance are nevertheless very close to optimal. The combination of efficiency and low complexity makes `Algorithm 5` a suitable solution for fast multipath packet scheduling, especially beneficial in real-time video streaming.

6.4.3 Real-time streaming: Sliding Window Approach

We now relax the assumptions of full knowledge of media packets and channel bandwidths, and we present the adaptation of the above algorithms to the case of live streaming. In this case, the server does not anymore have the knowledge of the complete video sequence. Instead it receives the network packets directly from an encoder. The server may buffer live streams for δ seconds, in order to increase the scheduling efficiency. It has therefore a limited horizon, which we call the prefetch time δ. In other words, the prefetch time, or prefetch window, refers to the look-ahead window employed by the server. At any given time t, the server is therefore aware only of the network packets $\{\lambda_n\}$ with decoding time-stamps $t_n^d \leq t + \delta$.

We assume that $N(t)$ is the number of packets that are available at the server at time t, and that $\Lambda(t) = \{\lambda_1, \lambda_2, ..\lambda_{N(t)}\}$ now represents the set of these packets ordered according to their decoding deadlines. $N(t)$ is equal to the number of packets containing data from the video sequence up to time $t + \delta$, minus the packets that were already transmitted to the client in the time interval $[0, t]$. Note that we use the terms of prefetch and sliding window interchangeably, as referring to the same concept.

The previously defined B&B and LBA methods are now applied on the set $\Lambda(t)$ in order to compute a transmission policy vector π for the $N(t)$ packets under consideration at time t. Neglecting the computation time, even for the B&B method, we can start transmitting the packets on the two paths according to the policy π, at time t. Let T be the time interval between two successive video frames, and without loss of generality, let t and δ be multiples of T. Hence, $t + \delta = kT$. At time t, the server can send packets that contain data from the encoded video sequence up to frame k. At time $t + T$, the packets containing data from frame $k + 1$ will be available at the server. At this time, the server will stop the transmission process of all packets from the previous sliding window that have not been sent yet, and add them to the new sliding window, along with the new packets from frame $k + 1$. B&B and the LBA methods are then applied on the new sliding window. The implementation of our algorithms on top of a sliding window mechanism adapts the scheduling to new packets, as soon as they are available at the server.

It is worth mentioning, that in the case of real-time video streaming, `Algorithm 5` is equivalent to a sequential greedy packet scheduling algorithm that considers first the most important packets in the sliding window, while for a sliding window of just one frame, our LBA method in essence reduces to the EDPF algorithm, enhanced with a packet discard strategy [137].

Interestingly, the LBA algorithm has the same behavior even in the case when the exact weights of each packet, w_n, are not known. It suffices to know only the relative ordering of the video packets according to their weight, along with the packet dependencies. While computing online the exact weight of each packet might be difficult (esp. in realtime streaming scenarios), the relative ordering of the packets can be easily performed, since it is generally accepted that an I frame packet is more important than a P or a B frame packet, and a base layer packet is more important than an enhancement layer packet. In the same time, the packet dependencies are known from the encoding and packetization processes.

These observations emphasize the low complexity of our proposal. We argue that, due to its low complexity, the LBA algorithm can be implemented at a real-time streaming server. The LBA algorithm presents a complexity that depends on the number of frames scheduled (N) and the size of the sliding window. Its complexity, C, varies according to: $C = 2(\frac{\delta}{frame\_rate})^2(N - \frac{\delta}{frame\_rate})$. Along with any simple bandwidth prediction mechanism able to estimate the bandwidth for the duration of the sliding window, it provides a valuable algorithm for any practical multipath streaming scenario. We demonstrate the good performance of the live streaming algorithm in Section 6.5, where it is compared to long horizon scheduling mechanisms.

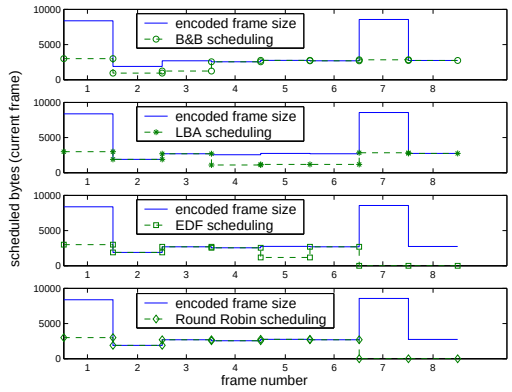


FIGURE 6.5: *Packet scheduling obtained by the B&B, LBA, EDPF, and simple round robin algorithms for an IBPBPBIB frame sequence (foreman_cif sequence).*

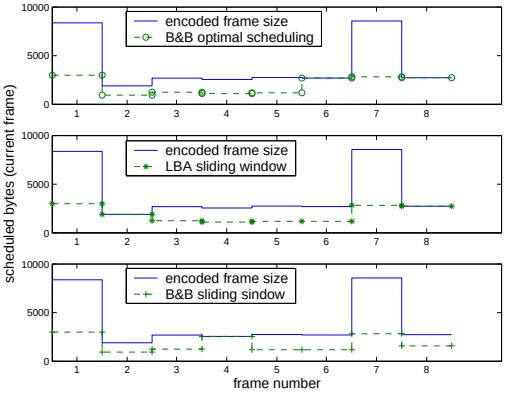


FIGURE 6.6: *Packet scheduling obtained by the B&B and LBA methods with sliding window, compared to the optimal scheduling for an IBPBPBIB frame sequence (foreman_cif sequence).*

r_1	r_2	r_3	r_4	B&B	LBA	EDPF
250	300	100	200	51.8%	47%	39.7%
300	300	100	200	58.9%	51.5%	43.4%
250	250	200	250	66.6%	60.6%	48.2%
250	250	250	250	68.2%	60.6%	48.2%
300	300	300	400	88%	82.2%	82.2%

TABLE 6.1: *Heuristic algorithms performance comparison*

6.5 Simulation Results

6.5.1 Simulation Setup

This section now presents and discusses the performance of the proposed scheduling algorithms, and compares the heuristic-based solution to the optimal performance bound, in both stored video scenarios and live streaming services. Video sequences are compressed with an MPEG4-FGS [7] encoder, at 30 fps with various GOP structures. We use two different CIF sequences, *foreman* and *news*, encoded in one base layer BL, and one or two enhancement layers (EL1 and EL2). Each encoded frame is split into network packets, one for each encoded layer. We set the weights ω_n of the packets as a function of their relative importance to the encoded bitstream (depending on the type of encoded frame, I, P or B, and on the encoded layer they represent, BL, EL1 or EL2), as illustrated in Figure 6.2.

We simulate network scenarios containing two and three disjoint paths between the server and the client. We conduct experiments for segment bandwidths which vary in time, for the theoretical case when the server knows them in advance, or when it predicts them based on past values. We experiment stored or live streaming scenarios, with limited prefetch window. Finally, we consider unlimited client buffers, and negligible network latencies (i.e., $d_i(t) = 0, \forall i, \forall t$). We compare the performance of the proposed algorithms to the one of EDPF [135]. We also compare to a simple RoundRobin algorithm, which greedily schedules video packets in a FIFO order, according to the available bandwidth on each of the paths. Finally, we also test our algorithm in scenarios with packet loss, in order to evaluate its behavior in very adverse conditions.

r_1	r_2	r_3	r_4	B&B	LBA	B&B *SW*	LBA *SW*
200	300	400	400	75.8%	65.5%	70.4%	65.5%
300	300	100	200	50.6%	47%	44.9%	47%
300	300	200	200	64%	60.8%	60.6%	60.8%
250	300	200	300	57.6%	51.4%	5.1%	51.5%
300	300	250	300	71%	60.8%	69.7%	60.8%

TABLE 6.2: *Algorithm comparison with Sliding Window*

6.5.2 Stored Streaming Scenarios

The proposed algorithms are first compared in the case of stored video scenarios, where the whole sequence is available at the streaming server, before running the scheduling algorithms. The two sequences are encoded into a BL of $300kbps$ and $450kbps$ respectively, and one EL of $550kbps$. Due to the high complexity of the B&B algorithm, which computes the performance upper-bound, we use a GOP of 6 frames, with one B frame between P frames. In a first approximation, we choose the following packets weights: $\omega_i = 5$, for I frame base layer packet, $\omega_i = 4$, for the base layer of the first P frame, $\omega_i = 3$, for the base layer of the second P frame, $\omega_i = 2$, for the base layer of B frames, and $\omega_i = 1$, for enhancement layer packets.

Figure 6.5 presents the video rate trace at the decoder, when the server schedules the network packets according to the optimal B&B method, the LBA algorithm, the EDPF algorithm [135], and RoundRobin. The segment bandwidths are set to $r_1 = 300kbps$, $r_2 = 500kbps$, $r_3 = 400kbps$ and $r_4 = 100kbps$, the intermediate buffers are unlimited and the maximum playback delay imposed by the client is set to $\Delta = 150ms$.

It can be observed that, while the proposed LBA algorithm manages to successfully schedule almost the same number of packets as the optimal B&B solution, the simple EDPF algorithm and the RoundRobin method have clearly worse performance since they mostly drop the end of the sequence. This is due to the fact that the proposed LBA algorithm makes sure that the most important packets (the packets from the base layer starting with the I frames, then P and B frames) can be scheduled, and only afterwards adds the enhancement layer packets, if the network rate permits it. On the contrary, the EDPF or RoundRobin algorithms schedule as much as possible from any frame, without taking into account future frames. In this way, entire GOPs could be lost, because packets of the I frame cannot meet the decoding deadline at the client.

A different representation is provided in Table 6.1. It presents the performance of the LBA and EDPF algorithms compared to the optimal solution for the $foreman\_cif$ sequence, as a function of the available channel bandwidth. The performance here is measured in terms of the percentage of successfully scheduled data bytes out of the total encoded stream. We observe that for a large variety of rates, the proposed LBA algorithm performs much closer to the optimal than the EDPF approach. In the same time, for some rates, the LBA algorithm suffers a loss in performance compared to the optimal B&B method, mainly due to the greediness of its scheduling strategy.

6.5.3 Streaming with Limited Look-ahead

The proposed solutions are now compared in the case of live video streaming, where the server knowledge is limited to the packets within the prefetch window. The prefetch window is set to 3 frames (i.e., $\delta = 100ms$), the maximal playback delay is $\Delta = 100ms$ and the bandwidths of the 4 network segments are constant in time. Figure 6.6 compares the real time B&B and LBA methods, where the original algorithms are applied on top of a sliding window mechanism (as explained in Section 6.4.3). The performance of the optimal B&B method applied to the whole sequence is also provided for the sake of comparison. It can be seen that the B&B method is no longer optimal when combined with a sliding window, as expected. The proposed LBA algorithm can even provide better performance in the live scenario.

The algorithms are also compared in terms of the proportion of transmitted information, for

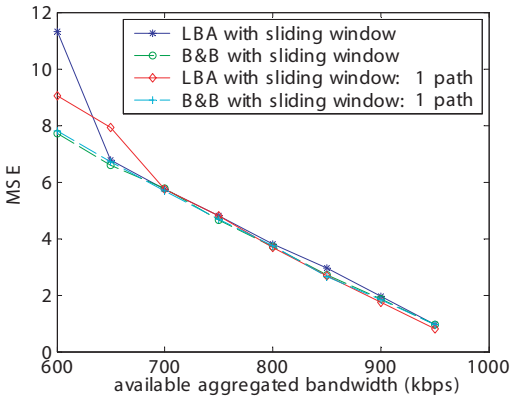


FIGURE 6.7: *MSE values between the original encoded sequence and the scheduled one (100 frames).*

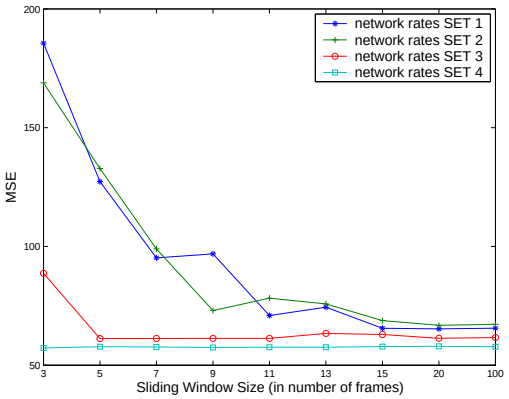


FIGURE 6.8: *MSE values for different network rate sets as a function of Sliding Window size.*

different network conditions, in Table 6.2. The values represent the percentage of successfully decoded data at the client, out of the full stream. Interestingly enough, the real time LBA algorithm has a similar performance to the case of stored video scenario. The sliding window, even with low prefetch time, does not significantly affect the behavior of the scheduling algorithm. This property, along with the low complexity of the algorithm, shows that LBA represents a valid solution to the multipath packet scheduling problem, in the case of live streaming.

The algorithms are also compared in terms of the MSE perceived at the receiver. Figure 6.7 presents the distortion due to the network bandwidth constraints, computed between the original encoded video sequence and the sequence available to the client. The MSE values obtained by the real time B&B and LBA scheduling algorithms on two paths (with equal rates) are compared to the ones obtained by using a single network path with equivalent aggregated bandwidth. The decoder in this case implements a simple error concealment strategy based on previous frame repetition. Both schemes perform quite similarly when the aggregate bandwidth becomes large. We observe that, while the multipath scenario does not require a large bandwidth network path, there is virtually no loss in video quality when using two parallel network paths, instead of a single high bandwidth channel. This proves the efficiency of the proposed algorithms, relatively to the distortion lower-bound provided by the single channel scenario. Obviously, multipath streaming is useful when there is no single high bandwidth channel available, which is used here only to assess the scheduling policy performance. Note that the EDPF algorithm is voluntarily omitted here due to the high MSE values reached when it fails to schedule entire frames or GOPs.

We now analyze the influence of the Sliding Window size on the LBA packet scheduling process. As seen before, in the case of constant link rates, the packet scheduling process is barely influenced by the size of the sliding window. However, it is not the case if we allow the link rates to vary in time. We tested the performance of the LBA algorithm with various sizes for the slinding window. We use the $foreman\_cif$ sequence (the first 100 frames) and variable network rates on small time scales (hundreds of milliseconds). We omit the results of the B&B algorithm due to the intractability of the computations for larger window sizes, and those of the EDPF scheduling, since it does not take into account the sliding window size.

We present the MSE results in function of the size of the sliding window, for various network rate sets of different aggregated average bandwidths (Figure 6.8). We can observe that, for small sliding windows, the LBA algorithm behavior is close to the one of the EDPF algorithm, which may lose entire GOPs. Results are improving once the sliding window increases, since the LBA algorithm has more flexibility in scheduling the video packets. Finally, given a reasonable sized window ($\delta = 0.5s$), the results of the LBA are comparable to the case of entire sequence knowledge before scheduling. This depends on the ergodicity of the sequence source rate.

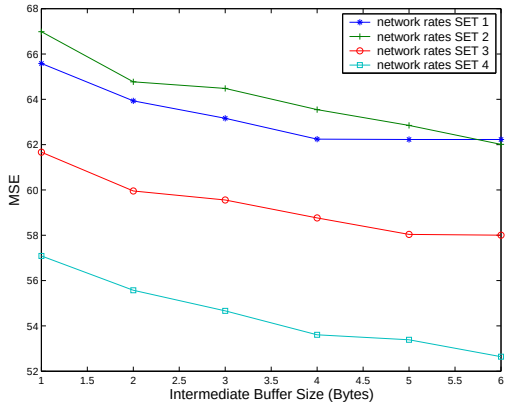


FIGURE 6.9: *MSE values for different network rate sets as a function of Intermediate Buffer size.*

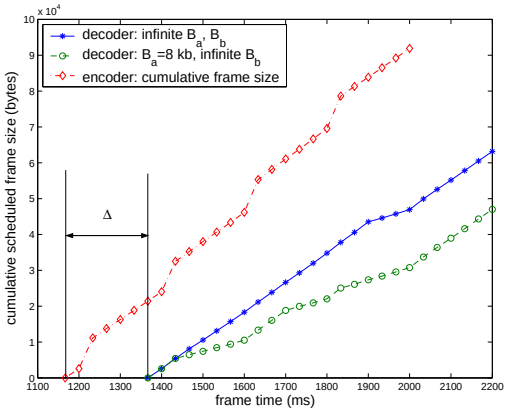


FIGURE 6.10: *Encoded video frame rate (cumulative) and decoded video frame rates (cumulative) in the case of infinite and constrained intermediate buffers.*

We now further investigate the effect of the size of intermediate buffers on scheduling performance. For the same network rate sets as before we vary the size of intermediate node buffers (B_a and B_b). We observe that, for the same network rates, bigger intermediate buffers allow for the scheduling of more video packets, with improved smoothing of the rate variations; the difference being noticeable in terms of MSE (Figure 6.9).

Finally, we study the effect of the intermediate buffer size on the packet load balancing on the two network paths. We compare the scheduling process on the two network paths in the case where the intermediate nodes have infinite or limited buffer space. Figure 6.10 presents the cumulative encoded frame rate of the total bitstream and the successfully scheduled bitstream rate in the case of infinite intermediate buffers, compared to the case when the buffer of node a is limited to $8kB$. Similarly, Figure 6.11 presents the same scheduling process in the same cases, separately for each of the two network paths. We observe major differences in the packet scheduling on the two paths between the two scenarios. A small buffer size on the first network path will render it unusable for a considerable period of time. This shortage is partially compensated by sending the base layer packets on the second link during the specific period. However, the effects on the received bitstream are noticeable. The scheduling of the bitstream in the case of unlimited intermediate buffers is therefore smoother. Finally, it is interesting to observe, that, due to the finer granularity of the base layer packets (in our setup the size of a base layer packet is in general smaller than the size of an enhancement layer packet), we can schedule on path b more data than in the unlimited buffer case.

6.5.4 Streaming with Link Rate Estimation and Channel Losses

Next, we release the assumption of a perfect channel knowledge, and we test our proposed scheduling algorithm in the case where the server estimates the channel availability, and the transmission process suffers losses on the network links. We programm our simulation scenario in ns-2 [194], where we simulate 10 background flows for each link. These flows are generated according to the On/Off Exponential distribution, with average rates between 100 and $300kbps$. The available instantaneous rate for our streaming application is considered to be the difference between the total link bandwidth and the aggregated instantaneous rate of the background traffic. While the exact shape of the background traffic is not important for our work, the On/Off exponential distribution of background traffic leaves a constant average available rate for our application, with instantaneous rate variations that can be larger than 100% (please refer to [200] for other types of traffic). In the same time, we generate packet losses on each of the network paths, according to

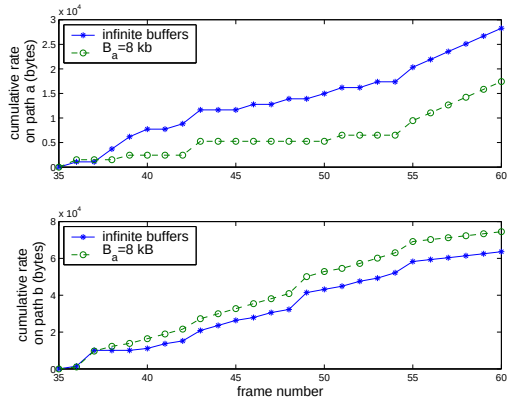


FIGURE 6.11: *Video scheduling on the two paths with infinite intermediate buffers vs. constrained buffer on path a ($B_a = 8kB$).*

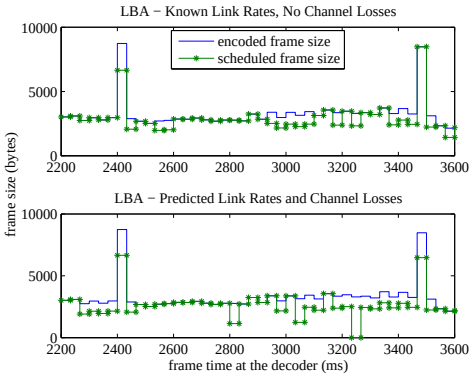


FIGURE 6.12: *LBA performance on 3 network paths with predicted parameters and channel losses.*

an iid process with probabilities equivalent to packet loss rates between 1 and 3%.

Next, the server implements a simple bandwidth estimation algorithm, based on an auto-regressive model. It estimates the bandwidth for each time window of size T_p, as follows. The available rate r_{k+1} of a segment in the next time interval $k+1$, is given by: $r_{k+1} = \gamma \frac{\sum_{j=1}^{k-1} r_j}{k-1} + (1-\gamma) r_k$, where γ is the prediction coefficient. While the instantaneous rate variations of the channel can happen on very small time scales (of tens to hundreds of milliseconds), the fastest estimation mechanisms [16] provide accurate results on time intervals of the size of a few round-trip times (e.g., at least one second or more). In simulations we therefore set $T_p = 1s$. Note finally that exact rate prediction is not crucial for the proposed algorithms, even if accurate prediction can only improve the performance.

We test the LBA protocol in the case when the server disposes of three disjoint paths for transmission, and the video is scalably encoded into one BL and two ELs. We use a GOP of 31 frames, with 15 P frames between I frames and one B frame between P frames. The two enhancement layers are created by splitting the FGS enhancement layer created by the MPEG-4 FGS encoder. We split the bitplanes such that the two layers have similar average rate, similar to [201]. We set the rates to $300kbps$ for the BL, and $260kbps$ respectively for the two ELs. The packet weights are set in a similar manner as in the previous experiments.

We schedule the first 100 frames of $foreman\_cif$, and we compare the results obtained by our algorithm and the EDPF algorithm [135, 137], in the case the server knows the rates in advance and there is no channel loss, with the case when it predicts the rates based on the auto regressive model presented above, and the transmission process suffers from path losses. We set the average rates on the three network paths to 280, 200 and $170kbps$, and the packet loss probabilities to 1, 3 and 2% respectively.

The maximum playback imposed by the client is $D = 200ms$. For the computation of the scheduling policy based on predicted rates, we however use a more conservative delay of $D_1 = 150ms$, in order to cope with big shifts in link rates and avoid the drop of important packets. The scheduling results are presented in Figure 6.12 and Figure 6.13. We observe that in the case of LBA, the performance degradation compared with the optimal case, when all rates are known, is negligible. While, in the optimal case, the algorithm correctly schedules 201 packets, out of 300, representing 67% of the total stream, in the case of prediction, it manages to schedule 186 packets, representing 62%. While no frame is lost due to frame dropping or late packet arrivals at the client, we observe a limited number of lost frames due to the loss of BL packets on the transmission process. Simple rate prediction, combined with conservative playback delay settings, offers performance in terms of client video quality that is comparable to the case where rates are

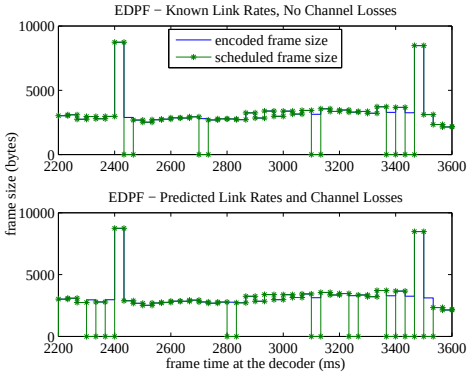


FIGURE 6.13: *EDPF performance on 3 network paths with predicted parameters and channel losses.*

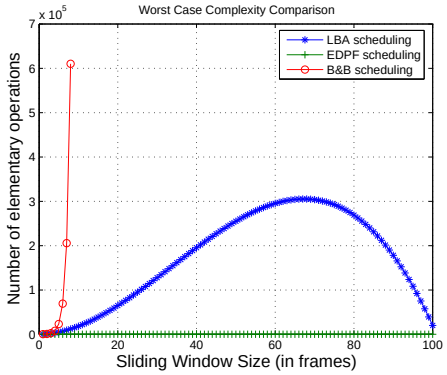


FIGURE 6.14: *Complexity comparison between B&B, LBA and EDPF*

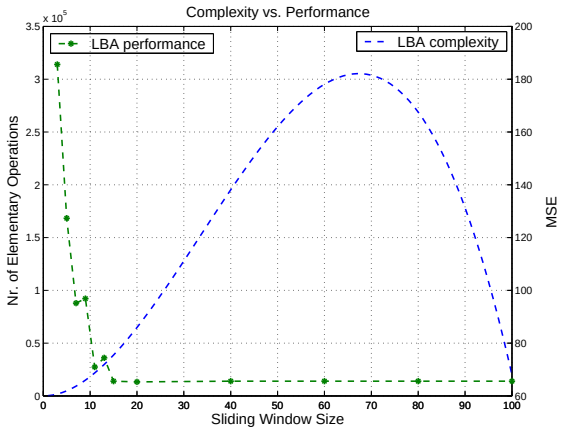


FIGURE 6.15: *LBA performance vs. complexity. (100 frames, average aggregated bandwidth of 450 kbps)*

perfectly known at the server[2].

We observe that EDPF tends to schedule entire frames and drop less important frames in favor of more important ones. On the other hand, the LBA algorithm prefers to schedule the most important video layers first, and only then schedule packets belonging to the enhancement layers, in the network bandwidth permits it. Due to the fact that LBA can handle scalable video streams, we also observe that it is more robust to channel losses than EDPF. LBA loses an entire frame only if a BL packet is lost due to channel errors.

In the context of simple error concealment methods at the client (e.g., frame replacement), the LBA scheduling will provide a smoother quality of the received video (7.2 MSE points compared to 22.4 MSE points in the case of EDPF). In the same time, due to the variable size of the frames, EDPF is more vulnerable to network rate variations and prediction errors than LBA.

Note that packet loss can be mitigated by use of error resilient mechanisms (e.g. FEC or packet retransmissions [106]), and we present results in lossy scenarios to evaluate the performance of the schemes in limit conditions. The design of a scheduling strategy adapted to lossy environments is however outside of the scope of the present work.

[2]For a more detailed analysis of media streaming with conservative delay on variable rate channels, please see [202].

6.5.5 Complexity Considerations

Finally, we analyze the complexity of the proposed algorithms and we try to derive a good trade-off for our LBA method, between complexity and performance, as a function of the size of the sliding window. While the B&B algorithm has a prohibitive exponential complexity as a function of the size of the sliding window, the EDPF and the Round Robin algorithms are very simple, their complexity being linear in terms of the number of total scheduled frames, and independent of the size of the sliding window. The complexity of our algorithm lies between the two bounds (Figure 6.14). It takes more operations than the simple EDPF scheduling, but it is still polynomial in complexity and can be performed in real time. In the same time, it is similar in complexity to the EDPF algorithm with the selective frame discard enhancement [135].

Figure 6.15 presents the performance of the LBA algorithm for different sizes of the sliding window. We superimpose the complexity curve with the performance curve in order to find the operational sliding window size as a function of the two values. We observe that for low values of the sliding window size, the performance of the LBA algorithm matches the one of the scenario when all frames are known in advance. In the same time, the complexity of the algorithm remains low. Low complexity and good performance, even for small sliding window sizes that allow to maintain low end-to-end delays, make the LBA a suitable candidate for real time packet scheduling in multimedia streaming.

6.6 Discussion and Conclusions

This work addresses the problem of the joint selection and scheduling of video packets on a network topology that offers multiple paths between the streaming server and the media client. We use an encoded video abstraction model that factors in the variable importance of video packets, as well as their interdependencies. An optimization problem is then formulated, which aims at maximizing the video quality at the client under a given playback delay. A formal analysis of packet transmission timing leads to the derivation of efficient algorithms to find the transmission policy that maximizes the video quality at the client. Because of the complexity of the optimal method, we propose fast, polynomial time algorithms that still offer close-to-optimal solutions. Both methods have been implemented in the case of stored videos, and real-time streaming with the help of a sliding window mechanism. Simulation results in both scenarios prove that our proposed heuristic-based solution performs well in terms of final video quality, and is moreover suitable for the case of real-time streaming under strict delay constraints. They also show that our methods outperform other common scheduling algorithms from the literature.

We identify a generic practical scenario in which our algorithm can be applied, as a streaming system in which one video server sends an encoded video to one or more clients in real time. Such a scenario can be easily imagined in the context of Content Distribution Networks, wireless video transmissions via several interfaces, or peer-to-peer applications. The video is encoded into multiple layers adding up to a very good quality, and the available aggregated rate between the server and any client represents the share of the total link bandwidth allocated to, or reserved by the streaming application. In such a scenario, for each of the clients, our algorithm will adaptively chose the right set of video packets to send on the network, in order to maximize the received video quality, given the available rates and the imposed playback delay. Because of its low complexity, the algorithm is scalable within large streaming scenarios.

Our method can be easily adapted to network scenarios characterized by weaker assumptions in terms of server knowledge about link rates and loss processes. We show how the algorithms perform in the case of predicted network rates, when the server uses a simple auto-regressive prediction mechanism. By using conservative scheduling parameters, our scheduling methods cope with large variations in instantaneous network rates, with a negligible increase in the distortion perceived at the client, as detailed in the next chapter. Furthermore, packet loss can be effectively addressed by implementing FEC schemes on top of our scheduling mechanisms.

Chapter 7

Packet Media Streaming with Imprecise Rate Estimation

7.1 Introduction

Our streaming solutions from the previous chapters generally rely on the knowledge of the channel bandwidth, in order to select the media packets to be transmitted, according to their sending time. However, the streaming server usually cannot have a perfect knowledge of the channel bandwidth, and important packets may be lost due to late arrival, if the scheduling is based on an over-estimated bandwidth. Robust media streaming techniques should take into account the mismatch between the values of the actual channel bandwidth and its estimation at the server.

Even the best rate estimation algorithms are not able to follow the rate variations of the channel, and often work on a coarser timescale [16]. Since channel prediction errors are inevitable and can lead to late arrivals of important media packets, the streaming server has to adjust the packet selection and scheduling strategies in order to cope with estimation mismatches. Our proposed method relies on a simple FIFO scheduling mechanism; however, we increase the algorithm's robustness by using a conservative virtual playback delay, smaller than the playback delay imposed by the client. The scheduling process considers the conservative playback delay as the hard deadline for packet arrival at the client, hence it is more aggressive in the packet selection process. On the other side, the difference between the conservative scheduling delay and the effective playback delay after which the client starts playing the video, transparently compensates for the eventual late packet arrivals due to the overestimation of the end-to-end bandwidth.

Overall, we observe that a very conservative scheduling delay tends to limit the selection of transmitted media data to only a few packets, which penalizes the quality at the receiver. Alternatively, a scheduling delay that is too close to the effective playback delay may result in late arrival of packets, which also penalizes the quality. Hence, the purpose of this chapter is to analyze the trade-off between robustness against channel prediction errors and packet selection limitations, observed as a result of tighter scheduling constraints.

The rest of this chapter is organized as follows: We formulate in Section 7.2 an optimization problem whose goal is to find the optimal conservative delay used in the scheduling process, which maximizes the quality of the received video for a given channel rate model, and a given playback delay at the client. We discuss the complexity of the exact solution for the optimization problem and we present a fast solution in Section 7.3. Section 7.4 presents our simulation results and Section 7.5 concludes this chapter.

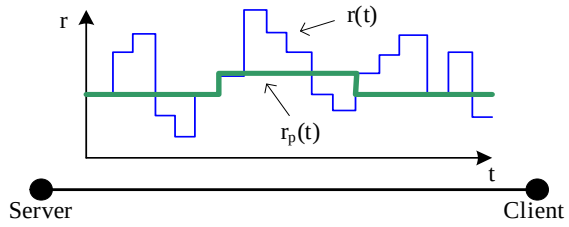


FIGURE 7.1: *Network end-to-end model with rate variations $r(t)$ and estimated rate $r_p(t)$.*

7.2 Streaming with Conservative Delay

7.2.1 System Overview

As in the previous chapter, we consider a single path streaming scenario between a server S and a client C. The media stream can either be pre-stored at the server (VoD), or can be obtained in real time (real-time streaming). The video content is encoded into one or more layers and fragmented into network packets such that one packet contains information related to one frame and one video layer. Let $\Lambda = \{\lambda_1, ..., \lambda_n\}$ be the set of available packets at the server, with n representing the total number of packets. Similarly to [198], each packet λ_i is completely characterized by its size s_i, its decoding deadline t_i, its importance ω_i and its list of dependency packets A_i, which are necessary for a correct decoding.

The intermediate network between S and C is modelled as an end-to-end channel characterized by the variable rate $r(t)$. While we consider no link error in our model, packets can still be lost from a media application perspective, due to late arrivals. The server S estimates on a periodic interval, the available channel rate $r_p(t)$, using any estimation mechanism Γ (Figure 7.1). Based on that estimation, the streaming application employs a generic scheduling algorithm Ψ that decides the subset of packets $\pi \subseteq \Lambda$ that are sent in a FIFO order to the client, so that the reconstructed video quality is maximized, given the playback delay Δ imposed by the client. The video quality measure Ω, can be computed at the client as:

$$\Omega = \Omega_S(\pi) - \Omega_L(\pi), \tag{7.1}$$

where $\Omega_S(\pi) = \sum_i \omega_i$, $\forall \lambda_i \in \pi$ represents the quality of the video packets selected for transmission, and $\Omega_L(\pi) = \sum_i (\omega_i \cdot \epsilon_i)$ represents the video quality degradation due to packets that cannot be decoded because of late arrivals at the client. ϵ_i represents the probability that packet λ_i arrives past its decoding deadline at the client. These late arrivals are caused by channel bandwidth variations, and inaccuracy in the rate estimation used by the server. Indeed, the estimation of the available rate in the future time instants is generally not perfect, and often not able to exactly follow the frequent variations of the bandwidth.

We propose to modify the scheduling strategy, in order to be robust to over-estimations of the channel rate. We define a virtual playback delay, or scheduling delay δ, which is used by the server to compute the subset of packets to be sent. As δ is smaller than the actual playback delay Δ, the server will select a reduced number of packets for transmission (Ω_S decreases), but the selected packets have a lower probability to be lost (Ω_L increases). In other words, π now contains only packets that are likely to reach the client before their decoding deadline ($t_i + \delta$) with a streaming rate r_p, and each packet λ_i is scheduled and transmitted only once. The choice of the virtual playback delay becomes obviously a trade-off between source quality, and robustness to rate variations, and its optimization is proposed in the next sections.

7.2.2 Illustrative Example

We demonstrate the rationale behind our proposed mechanism by a concrete example. Imagine that server S needs to decide at time t whether to send packet λ_i to the client C or not. The

Table 7.1: *Example Parameter Values for Conservative Delay Scheduling.*

Instantaneous Rate (*kbps*)	420
Predicted Rate (*kbps*)	450
Packet Size s_i (*bits*)	8000
Packet Weight ω_i	1000
Decoding Deadline t_i	0
Playback Delay Δ (ms)	200
Conservative Playback Delay δ (ms)	180
Time t (ms)	0

scheduling decision is based on the predicted network rate at moment t, $r_p(t)$, the size s_i, weight ω_i, dependency list A_i and decoding deadline t_i of packet λ_i, and on the conservative playback delay δ. In the same time, C expects packet λ_i before time $t_i + \Delta$, so that it can successfully decode it.

For the sake of clarity, assume that the list $A_i = \emptyset$, e.g., packet λ_i can be independently decoded at C, and that the server's buffer does not contain any other media packets except λ_i. The rest of the parameters are set according to Table 7.1.

Observe that S takes the decision to send the packet on the network after computing the expected arrival time at the client: $T_p = t + \frac{s_i}{r_p(t)} \approx 177ms \leq 180ms = t_i + \delta$. Even if the channel rate is overestimated and packet λ_i arrives at the client at the real arrival time $T_a = t + \frac{s_i}{r(t)} \approx 190ms > t_i + \delta$, packet λ_i still arrives on time for successful decoding at the client, as $t_i + \Delta = 200ms$.

On the contrary, imagine the same procedure is applied to packet p_j, under the same conditions, except $s_j = 9.000$ bits and the scheduler does not use the conservative delay δ, but rather directly the playback delay Δ. S decides to send the packet, as $T_p = t + \frac{s_j}{r_p(t)} = 200ms \leq 200ms = t_i + \Delta$. However, packet p_j is useless for the client as it arrives past its decoding deadline: $T_a = t + \frac{s_j}{r(t)} \approx 220ms > t_i + \Delta$. In such a case packet p_j consumes network resources that could be used more effectively.

Finally, please observe that in the case where S uses the conservative delay δ in scheduling packet p_j, the decision would be to drop the packet, as it is likely to arrive late according to the predicted bandwidth. This insight lies the ground for the trade-off between robustness against channel prediction errors, and packet selection limitations, observed as a result of tighter scheduling constraints.

7.2.3 Optimization Problem

The virtual playback delay δ used by the scheduler represents a compromise between a conservative selection of packets that minimizes the probability of late arrivals, and the selection of a sufficient number of packets for an effective quality. Given the video sequence, the quality metric Ω, the scheduling strategy Ψ, the rate estimation algorithm Γ, and the playback delay Δ, the optimization problem translates into finding the optimal conservative delay $\delta \leq \Delta$ to be used by the scheduler, in order to maximize the received video quality Ω, for a given channel model:

$$\delta^* = \arg \max_{\forall \delta \leq \Delta} \Omega(\delta) \tag{7.2}$$

In general, this optimization problem does unfortunately not provide any simple solution. Even for fixed Ψ, Γ and Δ, the scheduling policy π can greatly vary with the choice of δ, hence finding the optimal solution for the problem has combinatorial complexity. However, for small values of Δ (as in practical real-time streaming scenarios), δ^* can be accurately approximated in real-time. In the next section we present our approach towards finding an appropriate solution, based on heuristics from real-time video streaming.

7.3 Finding the Conservative Delay

7.3.1 General Solution

On the one hand, the quality metric $\Omega_L(\pi)$ depends only on the difference $\Delta - \delta$, for a given transmission policy π and the channel model. Very conservative values for δ will ensure a big difference $\Delta - \delta$, hence more flexibility in dealing with rate prediction errors, and consequently a smaller value for Ω_L (see Figure 7.2).

On the other hand, the quality measure $\Omega_S(\pi)$ depends only on the packets scheduled for transmission, according to the predicted rate $r_p(t)$ and δ. Interestingly, our experiments show that, for a given channel model, Ω_S does not vary much with δ, as long as δ is large enough to accommodate the transmission of the largest video packets of the sequence.

Let $R^i(\Delta)$ be the cumulative rate of the channel up to time $t_i + \Delta$: $R^i(\Delta) = \int_0^{t_i+\Delta} r(t)dt$, and $R_p^i(\delta)$ be the cumulative estimated rate up to time $t_i + \delta$: $R_p^i(\delta) = \int_0^{t_i+\delta} r_p(t)dt$. For given δ and Δ, we define the effective data transfer $\mathcal{C}_\Delta^\delta(i)$ on the time interval $[0, t_i + \Delta]$, as the amount of data scheduled according to the predicted rare r_p before $t_i + \delta$, and received before $t_i + \Delta$ according to the actual bandwidth r:

$$\mathcal{C}_\Delta^\delta(i) = R_p^i(\delta) \cdot Pr\{R_p^i(\delta) \leq R^i(\Delta)\}. \tag{7.3}$$

An illustration of the effective data rate transfer is given in Figure 7.3.

Given this measure, we transform the original optimization problem into a new problem that chooses δ in order to maximize $\mathcal{C}$. The optimal value of δ becomes:

$$\delta^* = \arg \max_{0 \leq \delta \leq \Delta} \mathcal{C}_\Delta^\delta(i). \tag{7.4}$$

$\mathcal{C}_\Delta^\delta(i)$ is invariant in time, as long as the channel model does not change, hence it can be computed at any t_i. The previous optimization problem translates into maximizing the chances of every packet λ_i, scheduled for transmission at time t, to reach its destination by time $t + \Delta$. Unlike the original optimization problem of Eq. (7.2), Eq. (7.4) depends only on the channel model, hence it is easy to solve, once this model is known. It can be noted that both optimization problems are equivalent in the case of a smooth video model (the video packets have the same size and importance, and there are no dependency among them). We later show in Section 7.4 that even in realistic video streaming scenarios the solution obtained for this problem is a very good approximation of the optimal solution, as long as the playback delay is long enough.

7.3.2 Example Channel Model

We now develop all necessary relations for a typical channel modelled as a discrete-time system, with a sampling interval of T_s seconds. The network can communicate a maximum of $r_i T_s$ bits of data in the time interval $[iT_s, (i+1)T_s]$, where r_i is the available bandwidth of the channel in the i^{th} time interval. The channel rate r_i is given as a Gaussian autoregressive process of the form:

$$r_i = \mu + (1 - \alpha) \sum_{j=0}^{\infty} \alpha^j n_{i-j}, \;\; j \in Z, \;\; n_k = 0, \;\; \forall k < 0. \tag{7.5}$$

Each n_j is an independent zero mean Gaussian random variable with variance σ^2, α is a modelling parameter, and μ denotes the average available bandwidth. The validity of that model for internet traffic traces on time scales of milliseconds up to a few seconds has been verified in [18].

A simple auto-regressive prediction model is used for bandwidth estimation at the server, where the available rate of the network in the next time interval, $k + 1$, is given by:

$$r_{k+1} = \gamma \frac{\sum_{j=1}^{k-1} r_j}{k-1} + (1 - \gamma) r_k, \tag{7.6}$$

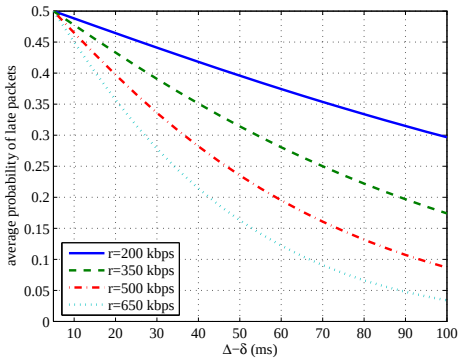


FIGURE 7.2: *Average probability of late packets when δ varies between 0 and Δ ($\Delta = 300ms$).*

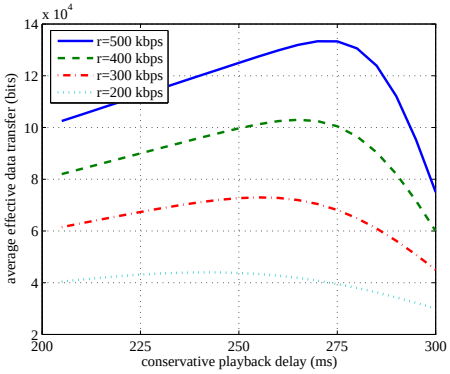


FIGURE 7.3: *Effective average data transfer when δ varies between 0 and Δ ($\Delta = 300ms$).*

where γ is the prediction coefficient. The estimation is run periodically, on time windows of size T_p. While instantaneous rate variations of the channel can happen on very small time scales (of tens to hundreds of milliseconds), the fastest estimation mechanisms provide accurate results on time intervals of the size of a few round-trip times (e.g., one second or more), and prediction inaccuracies cannot be avoided.

Assuming that $t_i + \Delta = k \cdot T_s \leq T_p$, with k an integer[1], we can compute:

$$R^i(\Delta) = k \cdot \mu + \sum_{j=0}^{k}(1-\gamma) \cdot \gamma^{j-1} \cdot \sum_{l=1}^{k-j} n_l. \tag{7.7}$$

Finally, $\mathcal{S}_i$ denotes the cumulative size of the transmitted packets up to packet λ_i: $\mathcal{S}_i = \sum_{j=1}^{i} s_j$, $\forall p_j \in \pi$. The probability that a packet arrives too late at the receiver, ϵ_i, can be computed as:

$$\epsilon_i = Pr\{\mathcal{S}_i > R^i / \mathcal{S}_i \leq R_p^i\}. \tag{7.8}$$

Since R^i is a normal random variable and R_p^i is a known constant, given any δ and Δ, the error probabilities ϵ_i can be easily computed with the help of the $erfc$ function.

7.3.3 Scheduling Algorithm

While the presented robustness mechanism is generic, and can be applied to any packet scheduling algorithm, in this section we describe the specific algorithm employed in the experimental phase of this work.

The algorithm is an adaptation of the LBA scheduling algorithm introduced in the previous chapter, to the single path network scenario presented above. In short, the algorithm performs a greedy scheduling of the most valuable packets first. Less valuable packets are scheduled only if the network capacity permits, and only if they do not lead to the loss of a more valuable packet already scheduled (due to subsequent late arrivals at the client).

First, the n network packets are arranged in descending order of their weight, obtaining a new representation of the encoded bitstream, $\Lambda' = \{\lambda'_1, \lambda'_2, ..., \lambda'_n\}$. Then, the algorithm attempts a greedy scheduling of the packets on the network link, starting with the most important one. To decide which action to take on each packet λ'_i, the algorithm first attempts to schedule all ancestors that have not been scheduled yet. If one of them cannot be scheduled, then the algorithm

[1] The extension of the computation for the general case, on multiple prediction intervals, and when k is not an integer can be computed in a straightforward manner, based on the analysis presented here.

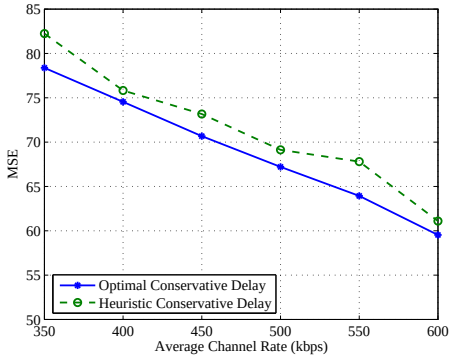


FIGURE 7.4: *Quality Evaluation for Scheduling with Heuristic and Optimal Conservative Playback Delay.*

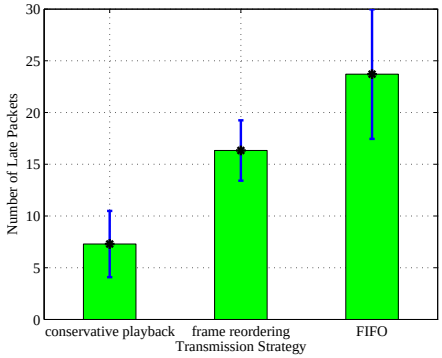


FIGURE 7.5: *Late Packets: Conservative δ; Frame Reordering; FIFO Scheduling.*

automatically drops the packet λ_i'. This ensures that our algorithm does not waste network resources on transmitting network packets that cannot be correctly decoded at the receiver.

Finally, all packets marked to be transmitted, are reordered according to their decoding deadlines before transmission. When a new packet is inserted for transmission, it triggers a new packet ordering. If packet λ_i' can be inserted, without compromising the arrival time of any other already scheduled packet, then it is scheduled for transmission. Otherwise, packet λ_i' is dropped. Please observe that the scheduling algorithm can be run on the total video sequence to be streamed, in the case of VoD streaming, or on a limited window of video packets in the case of real-time streaming.

The total complexity of the scheduling algorithm is driven mainly by the sorting and insertion operations. While the sorting can be performed by any algorithm in time $O(n \log n)$, the insertion of each packet λ_i' requires a complete parse through all previously scheduled packets. Hence the total complexity of the algorithm is $O(n^2)$.

7.4 Simulations

We discuss the performance of the streaming application with conservative delay and we compare the results obtained by our heuristic solution for δ with the optimal solution, obtained through a full search, and with other frame reordering techniques. We scalably encode the $foreman\_cif$ sequence (130 frames) using MPEG4-FGS, at 30 frames per second, with a GOP structure of 31 frames ($IPBPBPB...$). By splitting the bitplanes, we encode one BL and 2 ELs of average rates of $260kbps$. In all our experiments we use the simple packet scheduling algorithm as presented above. We set the weights ω_i of the packets as a function of their relative importance to the encoded bitstream (depending on the type of encoded frame, I, P or B, and on the encoded layer they represent, BL, EL1 or EL2), as illustrated in Figure 6.2. In a first approximation, we choose the following packets weights: 5 for I frame BL packets, 4 for the P frame BL packets, 3 for the B frame BL packets, 2 for the EL1 packets, and 1 for the EL2 packets [198].

For the channel model and estimation mechanism, we set the required parameters to $\alpha = \gamma = 0.8$, $T_s = 20ms$, $T_p = 1s$, and we vary $\sigma^2 \in [100, 250]$, according to the channel average rate. These values ensure realistic channel variations on small time scales around the average bandwidth value. Finally, we set $\Delta = 200ms$.

We compare the results obtained by streaming with the heuristic δ, computed according to Eq. (7.4), and the optimal δ^*, obtained after a full search through all possible values for $\delta \in [0, \Delta]$. We use different channel average rates and we average over 10 simulations for each case. The results are presented in Figure 7.4. We observe that for all simulated rates, our results in terms of MSE

TABLE 7.2: *δ^* and δ for Various Average Channel Rates.*

Rate ($kbps$)	350	400	450	500	550	600
Optimal δ^* (ms)	163	156	172.5	161	154	155.5
Heuristic δ (ms)	172	170	168	167	166	165
$\frac{\Omega(\delta^*)-\Omega(\delta)}{\Omega(\delta^*)}$ (%)	4.94	1.71	3.53	2.86	6.04	2.63

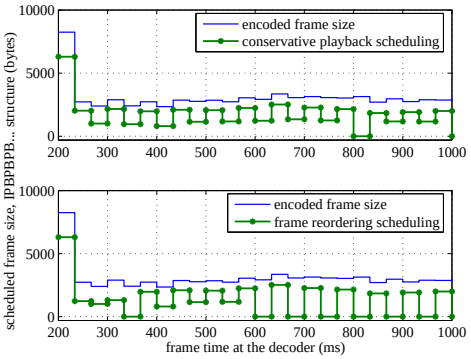


FIGURE 7.6: *Example of Conservative Playback Delay δ and Frame Reordering Scheduling.*

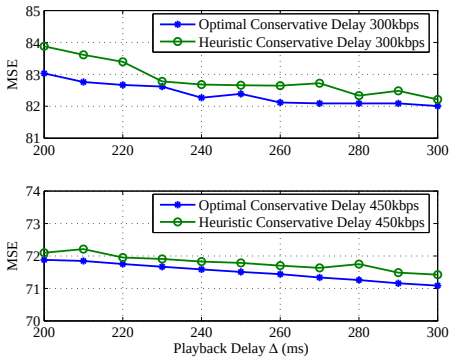


FIGURE 7.7: *Quality Evaluation for Scheduling with Heuristic and Optimal Conservative Playback Delay δ for ns-2 Network Rate Traces.*

are very close to the optimal ones. This validates our simplification to the original optimization problem, presented in Section 7.3. At the same time, Table 7.2 presents the obtained values for the heuristic and optimal δ for the same channel conditions as above, along with the relative error between the streaming performances. We observe that the values are very close and that δ^* is in general more conservative than δ. An explanation of this phenomenon resides in the fact that the sequence under consideration does not present any scene changes and the packet sizes remain constant in time.

Next, we compare the proposed conservative δ streaming with other frame reordering streaming techniques. We use a simple technique similar to the one presented in [138], which brings forward all I and P frames by two positions in the original bitstream before scheduling. Both techniques are compared in terms of number of late packet arrivals with a simple FIFO scheduling scheme that is unaware of channel rate variations. Simulation results are averaged over 100 channel realizations for an average rate of $500kbps$. Figure 7.5 presents the number of late packets for each of the 3 schemes with the 95% confidence intervals. We observe that the conservative δ scheme performs the best in terms of average number of late arrivals, due to the fact that the application can transparently use the difference $\Delta - \delta$ to compensate for unpredicted channel rate variations. Figure 7.6 presents one scheduling example for the conservative δ and frame reordering techniques. We observe that in the case of frame reordering, the strategy trades off a higher confidence in receiving I and P frames on time, at the expense of less important B frames. Hence, some B frames are lost due to late arrivals. On the contrary, the conservative δ strategy manages to schedule a similar amount of packets, and uses the extra time $\Delta - \delta$ to minimize the impact of rate variations on late arrivals. Hence, less packets are late at the receiving end of the application.

Finally, we test the proposed conservative delay scheduling method on network rate traces generated with the help of the ns-2 simulator in the presence of background traffic. We simulate 10 background flows that use the same bottleneck link as our media stream. These flows are generated according to the On/Off Exponential distribution, with average rates between 100 and $300kbps$. The available instantaneous rate for our streaming application is considered to be the difference

between the total link bandwidth and the aggregated instantaneous rate of the background traffic. Even if the average available rate stays constant, instantaneous rate variations can be larger than 100%. We compare the performance of the scheduling obtained by using the heuristic and the optimal conservative delays, respectively, by averaging the obtained results over 10 randomly generated network rate traces. Results are presented in Figure 7.7 for average network rates of 300 and 450*kbps*. We observe that the results are very close, even if the exact channel model is not known when the conservative delay is computed, and the channel estimation method is imperfect[2]. Results show that being conservative in terms of scheduling delay and initial channel rate estimate, increases the robustness of the streaming application, without significantly penalizing the received video quality. It indicates that our method is robust even in extreme cases when exact information related to the channel model is not available.

7.5 Conclusions

We present a new mechanism to improve the robustness of adaptive media stream scheduling algorithms against network channel variability and estimation inaccuracies. By using a conservative virtual playback delay in the scheduling process we compensate for possible prediction errors. The difference between the conservative and actual playback delay imposed by the client transparently absorbs the negative effects of inexact rate estimation (e.g., increased packet delay at the client due to channel variations). We propose a method to determine the value of the conservative delay, as a trade-off between source quality, and robustness to bandwidth variations. The proposed solution is generic and can be employed with any given streaming mechanism. Results show that being conservative in choosing the scheduling delay pays off, even if the exact channel model is unknown (e.g., on simulated network rate traces with competing background traffic) and the rate estimation mechanism only approximates the channel rate variations over time. The simplicity and effectiveness of our solution make it appropriate for any real-time streaming mechanism over best-effort networks.

[2]For more details on efficient bandwidth estimation mechanisms we refer the reader to [16].

Chapter 8

Media Streaming over Multiple Wireless Networks

8.1 Introduction

In this chapter we rely on the theoretical work presented until now, and we discuss a possible practical streaming scenario. We envision a setup where users can access various applications with different Quality-of-Service (QoS) requirements over possibly multiple access networks (Figure 8.1). We solve a global optimization problem that periodically computes the optimal rate allocation and network selection for each user/application, given a universal quality metric. To this end, we take into account the parameters of the networks available to each user, and the specific characteristics of wireless applications. One by one, the behavior of each considered application is designed as a function of the user's network access parameters. Specifically, we derive a distortion model for streaming applications, which depends on the available data rate, transmission loss process at each client, and specific video sequence characteristics. Similarly, voice and data transfer applications are analyzed. Then, we define a universal quality metric that maps the QoS behavior of all applications as a function of the network parameters. Our final goal is to maximize the overall QoS of the system, under the given network resource constraints.

Real systems will often offer a limited choice in the mode of operation of the accessed applications; e.g., different voice transcoders operating at different rates in the case of voice conversations, a limited number of scalable encoded video layers for streaming applications, or a set of standard download rates for data transfer applications. Our final solution consists of an optimal decision on the mode of operation (total required rate) and network resource allocation for each client accessing a specific application. Such a global solution requires the computation over the whole set of application modes, for every user. Given the time varying nature of the wireless connections and the dynamics of users leaving/joining the system, the optimality of our solution is insured by iterative computations that take into account the actualized system status. To this end, we provide fast heuristic algorithms that can be used in real time system optimizations, based on the utility trade-off between system performance improvement and required resources [33]. We show that our QoS metric behaves well in a large set of system setups, and outperforms other traditional QoS metrics based on throughput, in terms of overall achieved quality, user fairness and adaptability to dynamic system setups. Finally, we show that our proposed heuristic algorithms obtain a close to optimum system performance with a low computational effort.

Our contributions in this chapter are two-fold:

- In the context of multiple parallel applications over wireless networks, we discuss the opportunity of a single unifying quality metric that maps the specific requirements of each considered application to a single value. Later, this quality metric is used in our optimization framework for improving the overall system performance;

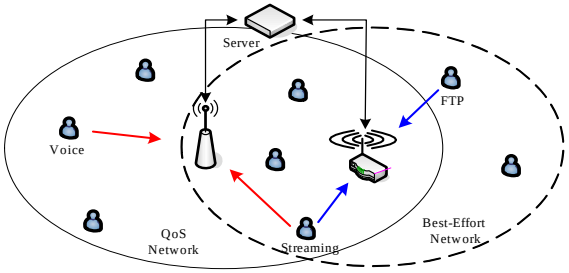


FIGURE 8.1: *Multiple wireless networks framework: more clients have access to multiple applications via more wireless networks.*

- Finally, we propose a fast heuristic algorithm that computes a close to optimum resource allocation solution in an iterative process, by taking into account the network access characteristics at each active client, along with the specific requirements of its desired application.

The rest of this chapter is organized as follows. Section 8.2 presents the considered applications and available access networks. We present our joint optimization problem in Section 8.3 and explain our heuristic approach to solving it in Section 8.4. We offer a concrete modelling example in Section 8.5. Extensive simulation results are presented in Section 8.6, while Section 8.7 concludes this chapter.

8.2 System Model

8.2.1 Multiple Applications

Assume N active users that simultaneously access via a server S any one of three different types of applications, namely voice conversation (V), real-time media streaming (M) and FTP download (F). Let user i, $1 \leq i \leq N$ access one of the available applications k, $k \in \{V, M, F\}$, and let $\mathcal{M}_i = r_i$ be the mode of operation of user i, decided by S. It describes the average rate allocated to user i that has chosen application k. We assume that S can scalably adapt the transmission process to the channel conditions of user i. To this end, for each application k, the server can choose the right transmission parameter, from a predefined set of available parameters $\mathcal{P}_k$.

First, we consider a multimedia streaming application that transmits a scalable encoded stream to the end user. Let L be the number of available encoded media layers available at the server S, where the layer $l \leq L$ is characterized by its average encoding rate ρ_l. Additionally, we assume that the server S can protect each media layer against transmission errors, with one of E forward error correction schemes $FEC(n_e, k_e)$, $e = 1, \ldots, E$. We define $\mathcal{P}_M = \{\rho_m : 1 \leq m \leq O\}$ as the set of available streaming modes, where $O = L \cdot E$ represents the total number of feasible combinations between the media encoded layers and FEC schemes, and ρ_m is the total rate imposed by mode m. The final perceived quality at the end user depends on the number of media layers transmitted, and the loss process that affects the media packets after FEC decoding, and can be computed as shown in Chapter 5.

Then, we model the voice application. We consider N_V available voice transcoders at the server S. Each transcoder v is characterized by its encoding rate ρ_v. We define $\mathcal{P}_V = \{\rho_v : 1 \leq v \leq N_V\}$ as the available parameter set for the voice application. The perceived quality of the voice application at the end client depends on the complexity of the transcoder v, and hence the allocated rate ρ_v, and the error process p that affects the data transmission.

Finally, we assume $\mathcal{P}_F = \{\rho_f : 1 \leq f \leq N_F\}$ as the available parameter set for the FTP application. ρ_f represents the download rate of the FTP session. The perceived quality of the application will depend on the total download time, hence on the allocated download rate and error process that affects the data transmission.

We define the QoS metric $\Gamma(\mathcal{M}_i) = f(r_i, p_i)$ as a function of the allocated rate r_i and the average loss probability p_i affecting the data transmission of application k, towards user i. A concrete example of such a QoS metric, along with the appropriate mappings between this metric and the perceived quality of the applications presented above is given in Section 8.5. Finally, we define $\mathcal{M} = \{\mathcal{M}_i : 1 \leq i \leq N\}$ as the global operation mode of the system, when the server S allocates the rate $r_i = \rho_k \in \mathcal{P}_k$ to each active user i, accessing application k.

8.2.2 Multiple Networks

Even if the problem formulation proposed here is generic, we constrain ourselves to a scenario with two active networks that relay application data between the server S and user i. Q_Net is a QoS modelled network, characterized by a guaranteed service to all active users when network loads are inferior to the congestion point (e.g., through spreading codes and transmission time intervals assignment in the case of an HSDPA system), and high blocking probability in saturated regime. Its total resources are characterized by the instantaneous total throughput R^Q, which takes into account the channel conditions of all active users in the network. R^Q is preferentially distributed among active users according to the importance of their accessed application (e.g., HSDPA systems prioritize voice conversations over streaming applications and FTP downloads). R^Q is periodically estimated on time intervals T, possibly with a certain prediction error, which translates into a generally small packet error probability p_i^Q that equally affects all active users.

The second network, BE_Net, is modelled as a Best Effort network that provides services to clients on a first-come-first-serve basis (e.g., a WiFi hotspot). Each active client i in this network can access resources at a maximum data rate R_i^B and is affected by an average loss process p_i^B, over time intervals T. While channel conditions in wireless environments change on very short time scales (e.g., up to a few tens of ms), we assume that R_i^B and p_i^B represent average values computed on larger time scales T (e.g., one to a few seconds), and represent the average channel conditions for user i on the given period T.

Let $[r_i^Q, r_i^B]$ be the rate allocation of user i over the two networks, with $r_i = r_i^Q + r_i^B$. Please observe that application rates $r_i^Q = 0$ or $r_i^B = 0$ imply that user i is inactive in the given network. Finally, let the tuple $\tau_i = [r_i^Q, p_i^Q, r_i^B, p_i^B]$ characterize the application rates and channel conditions for each user i in the two networks. The following resource constraints apply:

$$\sum_{i=1}^{N} r_i^Q \leq R^Q, \qquad \sum_{i=1}^{N} \frac{r_i^B}{R_i^B} \leq 1. \tag{8.1}$$

for Q_Net and BE_Net respectively. While the first constraint refers to the total available throughput on the Q_Net, the second one refers to the maximum available time for transmission on the downlink at the access point of the BE_Net. Finally, under these conditions, the total error probability that affects the transmission to user i, reads : $p_i = \dfrac{r_i^Q \cdot p_i^Q + r_i^B \cdot p_i^B}{r_i^Q + r_i^B}$.

8.3 Network Selection and Rate Allocation Problem

We assume that the server S periodically solves the optimization problem, in full knowledge of the connection parameter tuple τ_i, $\forall i : 1 \leq i \leq N$, and of the application parameter sets $\mathcal{P}_k$, $\forall k \in \{V, M, F\}$. Within each time interval T, we optimize the allocation of network resources among the N clients, with the final goal of maximizing the overall quality of the system. In other words, we are looking for the optimal global operation mode $\mathcal{M}^* = \{\mathcal{M}_i^* : 1 \leq i \leq N\}$ containing the optimal application mode for each client i, where $\mathcal{M}_i^* = r_i^* \in \mathcal{P}_k$, k being the application accessed by client i:

$$\mathcal{M}^* = \arg\max_{\mathcal{M}} \sum_{i=1}^{N} \Gamma(\mathcal{M}_i) \tag{8.2}$$

under the constraints provided by Eq. (8.1). A discrete search through all operation modes leads to the solution $\mathcal{M}^*$ with optimal overall QoS. Alternatively, in the next section, we offer a heuristic algorithm that achieves close-to-optimal results with a faster convergence time.

8.4 Utility Based Rate Allocation Algorithm

Algorithm 6 Utility based rate allocation algorithm

Input:
R_Q, p_i^Q, R_i^B, p_i^B, $\forall$ user i;
$\mathcal{P}_k$, $\forall k \in \{V, M, F\}$, ordered in ascending order of ρ_k;
$\mathcal{M}_i = 0$, $\forall$ user i;
Output:
Global Rate Allocation Mode $\mathcal{M}$;
Procedure RateAllocation
While (1)
for $i = 1$ to N **do**
  Compute the utility of $i \rightarrow \mathcal{M}_i^{'}$:
  $U_i = \frac{\Gamma(\mathcal{M}_i^{'}) - \Gamma(\mathcal{M}_i)}{r_i^{'} - r_i}$;
end for
find $i^* = arg\max_i U_i$;
Push($i^*, \mathcal{M}_{i^*}^{'}, Q\_Net$);
Procedure Push($i, \mathcal{M}_i^{'}, Q\_Net$)
if Q_Net has enough free resources **then**
  $i \rightarrow \mathcal{M}_i^{'}$;
  update free resources on Q_Net;
else
  Switch($i, \mathcal{M}_i^{'}, Q\_Net$);
end if
Procedure Switch($i, \mathcal{M}_i^{'}, Q\_Net$)
find user j that can transfer part of his allocated rate r_j to BE_Net with minimum H_j;
if $U_i - H_j > 0$ **then**
  perform the switch of user j rate: $G(j, r)$;
  $i \rightarrow \mathcal{M}_i^{'}$;
  update free resources on Q_Net and BE_Net;
else
  Break;
end if

In this section we introduce our heuristic approach for solving the rate allocation optimization problem. We build on the utility framework introduced in [33], and present an algorithm that iteratively takes a locally optimal decision on each user's application mode.

Let $\mathcal{P}_k$, $k \in \{V, M, F\}$ be the sets of application modes ordered in increasing order of their required rates, and let $\mathcal{M}_i$ be the allocated mode of user i at a given iteration of our algorithm. We define $i \rightarrow \mathcal{M}_i^{'}$ as the transition of user i to the next application mode $\mathcal{M}_i^{'}$ requiring the next higher application rate $r_i^{'}$. The utility of this transition can be computed as:

$$U_i = \frac{\Gamma(\mathcal{M}_i^{'}) - \Gamma(\mathcal{M}_i)}{r_i^{'} - r_i},$$

and represents the trade-off between the system quality improvement and the extra resources required by user i's transition. Our algorithm starts from the initial setup when the clients have no allocated network resources. During each iteration, the proposed algorithm finds the user i^*

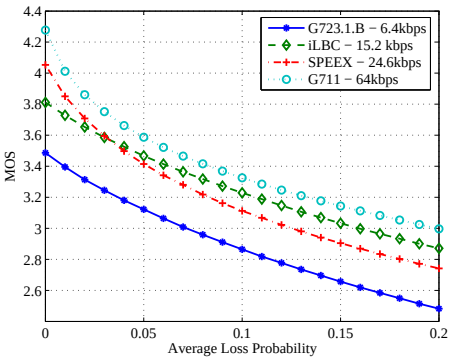


FIGURE 8.2: *Voice Application MOS: mapping between MOs and increasing loss probabiltiy for every considered voice codec.*

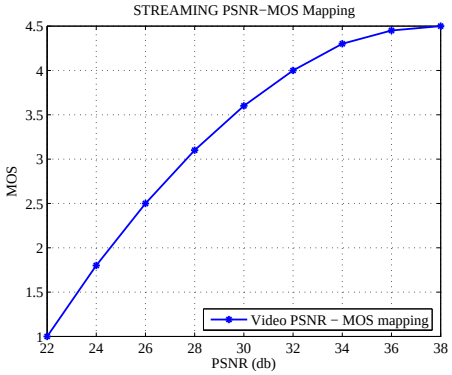


FIGURE 8.3: *Streaming Application MOS: mapping between MOS and PSNR for the foreman sequence.*

that brings the highest utility to the overall system by its transition to the next (higher quality) application mode:

$$i^* = \arg \max_i U_i,$$

The extra resources will be allocated to user i^* starting with the resources of Q_Net. Once the resources of Q_Net are depleted, the algorithm finds a different user j that can free the required resources for user i^*, by reallocating part of its rate $r \leq r_j$ on the other network BE_Net. Let $G(j, r)$ be the operation by which rate $r \leq r_j$ of user j is redirected through BE_Net, and let H_j be the loss in system utility caused by the switch. This operation is performed as long as the overall utility of the system is still improved ($U_i - H_j > 0$), and as long as free network resources still exist in the overall system. The algorithm stops when there are no more free resources in the network system, or when no other possible user transition can bring any improvement in the overall system utility.

`Algorithm 6` represents a sketch of the proposed algorithm. The **Push** procedure always attempts to increase the system's utility by allocating the free Q_Net resources to the best user. If the free resources are not enough, the **Switch** procedure tries to find a new user that can free up enough resources by reallocating parts of its allocated rate through the BE_Net. As long as the network resources allow it, the procedures repeat until no higher modes are available at any client, or no extra utility improvement can be brought to the overall system.

The complexity involved in the search for i^* is $O(N)$, the same being valid for the **Switch** procedure. In the worst case, the algorithm requires $O(N \cdot |\mathcal{P}_k|)$ iterations to pass through every application mode of every user. Hence the total complexity of the algorithm is $O(N^2 \cdot |\mathcal{P}_k|)$. For a reasonable number of wireless users, and a finite set of available application modes, the algorithm will converge rapidly to a global rate allocation vector $\mathcal{M}$. Its performance is further studied in Section 8.6.

8.5 MOS Quality Metric

In this section we exemplify on a concrete quality metric Γ based on the *MOS* (Mean Opinion Score) value [203].

MOS reflects the average user satisfaction on a scale of 1 to 4.5. The minimum value reflects an unacceptable application quality, and the maximum value refers to an excellent QoS. The perceived quality of each of the three applications is converted into an equivalent *MOS* value, which is later used in the optimization problem.

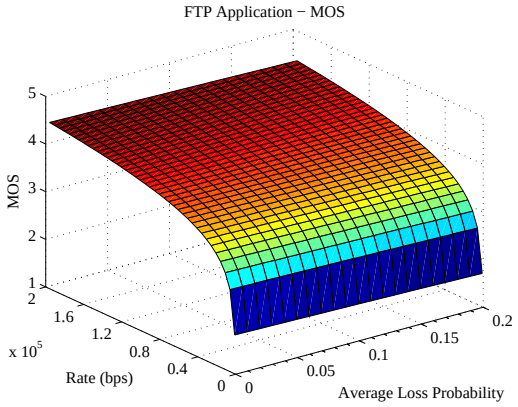


FIGURE 8.4: *FTP Application MOS: mapping between MOS and throughput.*

The performance of different voice transcoders as a function of network losses is mapped to MOS values using the $PESQ$ algorithm on a representative set of voice samples [157] in Figure 8.2. We observe that, while good network conditions lead to increased user experience, high packet error rates degrade the perceived quality of the voice communication.

The perceived media streaming quality is initially mapped into an MSE (mean square error) distortion measure, as presented in Section 8.2.1. Later on, a nonlinear mapping between MSE and MOS values is used, as illustrated in Figure 8.3.

Finally, the perceived quality of the FTP application is mapped to MOS values according to a logarithmic function of the achieved throughput: $MOS = a \cdot \log(b \cdot r(1-p))$. The variables a and b are system dependent parameters, and can be set by the network operator (Figure 8.4).

8.6 Simulation Results

8.6.1 Simulation Setup

We test the performance of our proposed rate allocation and path selection method, and we compare its performance against a classic optimization solution that uses application throughput as a quality metric.

We use 4 voice transcoders, namely G.723.1B, iLBC, SPEEX and G.711 with average encoding rates of 6.4, 15.2, 24.6 and $64kbps$ respectively. To simulate the media streaming application, we encode the $foreman\_qcif$ sequence (300 frames) with the H.264/SVC codec. We encode one base layer and one enhancement layer, each of $70kbps$. Additionally, we use one forward error correction mode $FEC(20, 17)$ which can correct up to 3 packet errors in a block of 20 packets. For FTP downloads, we set 4 available download rates of 50, 100, 150 and $200kbps$ respectively.

Due to the high complexity of the full search algorithm for finding the overall optimal rate allocation solution, we use small network scenarios (5 or 6 users) in order to validate the MOS quality metric, and the proposed heuristic algorithm. Later we compare our proposed heuristic algorithm with other heuristics in larger network setups. For comparison purposes we define as $OptimMOS$ and $OptimTh$ the full search algorithms which optimize the network resource allocation based on the MOS, and respectively $Throughput$ QoS metrics. In the same time we define `Algorithm 6` as $Heuristic$, while $Switch$ represents the same heuristic algorithm, with the constraint that no user can be allocated resources from both networks in the same time (e.g., when the algorithm decides to switch one client from one network to another, its whole allocated rate is rerouted through the new network). $SwitchTh$ is similar to $Switch$, but acts according to the $Throughput$ QoS metric.

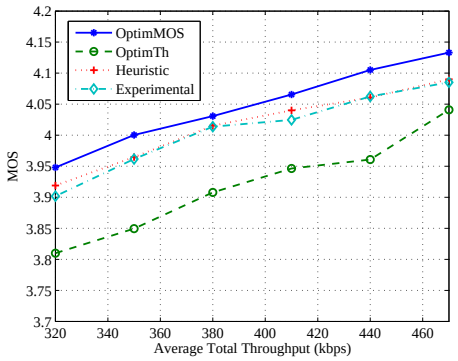


FIGURE 8.5: *Average system MOS values as a function of overall system rate: MOS vs. Throughput Optimization*

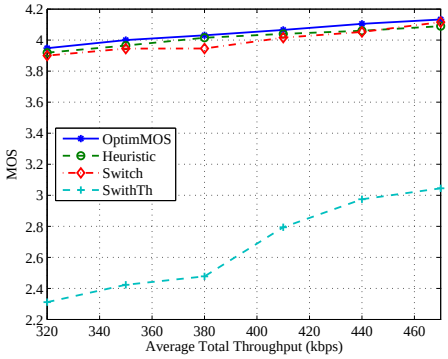


FIGURE 8.6: *Average system MOS values: Heuristic algorithms.*

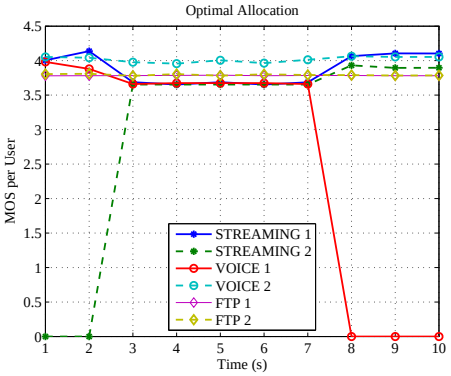


FIGURE 8.7: *Client performance when users are added/removed to/from the system: OptimMOS algorithm.*

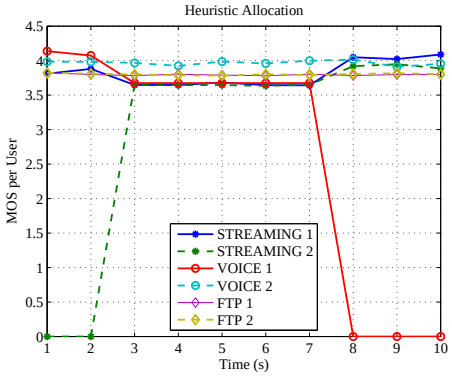


FIGURE 8.8: *Client performance when users are added/removed to/from the system: Heuristic algorithm.*

8.6.2 Small Network Scenarios

A total of 6 clients are placed in the coverage area of both networks (3 voice, 2 FTP, and one streaming user). Server S performs the optimization of the rate allocation periodically, every $T = 1s$. The average throughput R^Q of Q_Net varies in the interval $[100, 150]kbps$ and the prediction error p_i^Q is kept around 1%. The connection data rate R_i^B of the users in the BE_Net is set in the interval $[220, 310]kbps$, and the individual average loss probabilities p_i^B are randomly chosen in the interval $[1, 15]\%$. We average our results over 100 simulation runs of 10 seconds each.

We first compare the average performance of the overall system, when the optimization is performed according to the *MOS* and throughput quality metrics. We start by identifying the traffic distribution obtained by each optimization metric over the two networks. Table 8.1 presents the fraction of traffic that passes through both networks, for each application. We observe that the *MOS* optimization rightfully uses the Q_Net resources for the voice and streaming applications, while the FTP traffic is forwarded through BE_Net. On the other hand, the throughput optimization favors the FTP application, as it forwards part of its traffic over Q_Net (hence increasing the offered rate for the application), at the expense of lower available resources for the voice and streaming applications that share the same network. This explains the lower overall system performance obtained for the throughput metric, compared to *MOS* (Figure 8.5). For a total average system throughput varying from 320 to $460kbps$, the *MOS* optimization outperforms the

TABLE 8.1: *Traffic distribution over the two networks (in %).*

Application	*MOS* Optimization Q_Net	BE_Net	Throughput Optimization Q_Net	BE_Net
Voice	100	0	100	0
Streaming	88.5	11.5	94	6
FTP	1	99	12	88

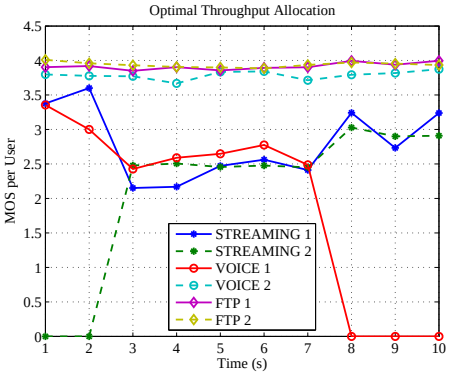


FIGURE 8.9: *Client performance when users are added/removed to/from the system: OptimTh algorithm.*

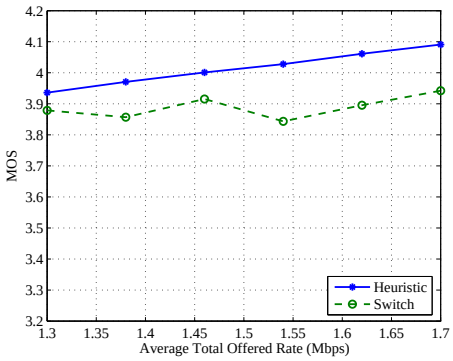


FIGURE 8.10: *Average system MOS values: Heuristic vs. Switch, 20 users.*

throughput optimization in most cases by as much as 0.15 *MOS* points. We also observe that the *Heuristic* algorithm closely matches the optimal behavior, and the experimental results obtained after performing experiments with real video sequences. In the same time, Figure 8.6 presents the quality performance among the proposed heuristic algorithms. While *Switch* and *Heuristic* are quite close to optimum, *SwitchTh* fails to allocate enough resources to some of the users, hence the important degradation in overall system performance.

Finally, we test the two optimization metrics in dynamic systems where users are allowed to join/leave the networks. We start with 5 clients (2 voice, 1 streaming and 2 FTP users). At time $t = 3s$ we add a streaming user, and at time $t = 8s$ we remove one voice user. Figure 8.7, Figure 8.8 and Figure 8.9 present the average application performance for each user. We observe that in the case of *MOS* optimization, the system is able to cope with the extra user at the expense of a small quality degradation for the existing users, for both *OptimalMOS* and *Heuristic* algorithms. On the other hand, the throughput optimization is unfair, as some of the clients are penalized more than the others, and the overall performance is worse.

8.6.3 Large Network Scenarios

In this case we are using a total of 20 clients placed in the coverage area of both networks (7 voice, 6 streaming and 7 FTP clients). The total rate of the system is varied in the interval $[1.3, 1.7]Mbps$ with $R^Q \in [300, 600]kbps$. The loss probabilities for the two networks and the simulation setup are similar as in the previous example.

We are looking at the overall average performance of the *Heuristic* and *Switch* algorithms when more active users are present in the system (Figure 8.10). Intentionally, we omit the performance of the *SwitchTh* algorithm, due to its very poor results. We observe that while *Switch* performs quite good, *Heuristic* still provides a significant improvement in total system quality. This is mainly due to the extra system granularity in allocating the resources of the two networks among the clients, if clients are allowed to connect in parallel to both networks.

Next, we present the average traffic distribution on the two networks, for each type of applica-

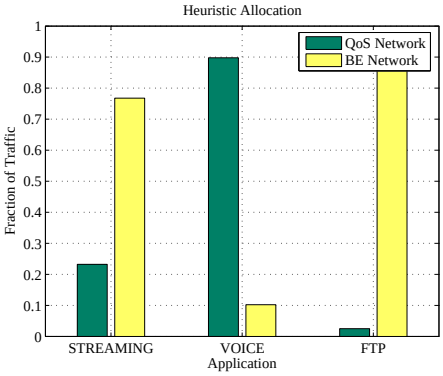


FIGURE 8.11: *Average traffic distribution per application type, per network: Heuristic algorithm, 20 users.*

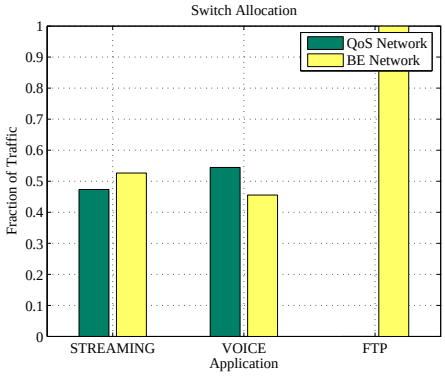


FIGURE 8.12: *Average traffic distribution per application type, per network: Switch algorithm, 20 users.*

tion, when each of the two algorithms is used to compute the overall rate allocation. Figure 8.11 and Figure 8.12 present the distributions obtained by the *Heuristic* and respectively *Switch* algorithms. We observe that *Heuristic* manages to allocate the Q_Net resources mostly to the voice application and as much as possible to the streaming application. The FTP clients are mostly scheduled on BE_Net, which represents an intuitive result. On the other hand, *Switch* schedules almost half of the voice applications on the BE_Net, at the advantage of streaming applications. While surprising, this result is explained by the fact that voice applications, usually requiring less network resources, are easier to switch on the best-effort network, when the QoS network becomes congested. Such a behavior can however be corrected by applying different weights to the clients, depending on the importance of the accessed application.

Finally, we test our algorithms in dynamic systems. We allow 4 new users to join the system at time $t = 3s$ (2 voice, 1 streaming and 1 FTP clients), while at time $t = 8s$, other 4 users area leaving. Figure 8.13 and Figure 8.14 present the results obtained by *Heuristic* and *Switch* respectively. In the first case, we observe that the algorithm manages to keep a rather constant application quality for all active clients, by redistributing parts of the network resources to the new users. This way, *Heuristic* achieves fairness among all users, even if they access different types of applications. On the other hand, *Switch* copes worse with the system dynamics; we observe that the voice and streaming users are penalized, compared to the FTP users. Again, this is due to the lack of granularity in reallocating network resources, when new users enter the system. This highlights the benefit of resource allocation flexibility given by the multipath network scenario assumed by the proposed algorithm.

8.7 Conclusions

We introduce a new optimization framework for the rate allocation and network selection for clients accessing multiple applications over parallel networks. In the optimization process we take into account the available network resources and the connection parameters of each client, along with the specific quality requirements of each application. We unify the performance of all applications under a single *MOS* quality metric, which is later used in the optimization process. Compared to traditional optimization metrics based on throughput, the *MOS* approach achieves a more fair resource allocation among active clients, and proves to be more scalable in dynamic systems. We finally provide a heuristic algorithm based on utility functions, which achieves a close to optimal resource allocation with low computational resources. Comparing to other heuristic approaches, our algorithm is more stable and adaptable in dynamic situations, emphasizing the flexibility given by the resource aggregation paradigm in multipath network scenarios. The obtained results

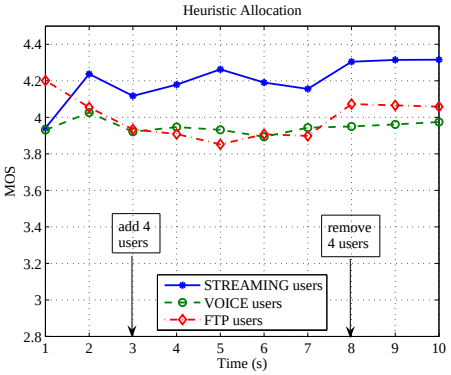


FIGURE 8.13: *Average performance per application in case users join/leave the network: Heuristic algorithm.*

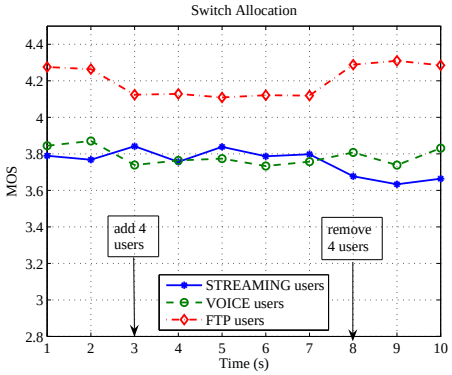


FIGURE 8.14: *Average performance per application in case users join/leave the network: Switch algorithm.*

encourage us to further investigate the possibility of multiple wireless networks interconnecting towards the final benefit of the end users.

Chapter 9

Conclusions

9.1 Thesis Achievements

This thesis addresses the problem of internet media streaming from the end-user perspective. We take a combined approach by looking at the same time at the characteristics of the transport medium, and of the particular application under consideration, in order to develop efficient streaming algorithms and protocols. We take advantage of the path diversity offered by the latest network architectures and present a complete framework for video streaming over multipath networks. Within this context, we separately discuss the most important issues concerning an efficient streaming process, and we present our analysis, results and conclusions. Finally, we integrate the proposed mechanisms and algorithms into a possible system for video streaming.

First we discuss the issue of path selection and rate allocation for multipath streaming systems. Our main objective is to jointly find (i) the optimal streaming rate for a given, pre-encoded video packet stream so that the quality at receiver is maximized, and (ii) which network paths should be used for relaying the video stream to the client. Our analysis leads to the theoretical foundations for an efficient algorithm that computes the optimal path selection and rate allocation solution for our scenario. We learn that the network paths should be used in a greedy manner, starting with the ones affected by the lowest loss probabilities, and that once used, a path should be utilized at its full resources. Interestingly enough, our simulation results emphasize the trade-off between allocating more network resources to the streaming process, and hence, allowing for an increased encoded media quality, and the increased risk of erroneous decoding due to extra transmission errors, induced by added transmission paths. This insight motivates the use of a limited number of streaming paths for the media transmission, and explains why a simple network flooding with media packets is not necessarily optimal. Furthermore, we propose distributed methods for implementing our findings in large network scenarios, where the available end-to-end network paths are not known a-priory. We show that fast heuristic rate allocation rules implemented at intermediate nodes lead to the construction of good transmission paths, later utilized by the streaming application.

Next, we offer an insight study of various forward error correction and scheduling techniques in multipath scenarios. We emphasize the streaming quality improvement offered by priority scheduling strategies, combined with unequal error protection, based on the different importance of media packets. Furthermore, we discuss practical systems, with limited flexibility in choosing the forward error correction parameters, and we show that efficient systems will generally insure the strong protection of the most important packets of the media application in a joint source channel coding setup. Finally we explore the possibility of in-network processing, and we identify network scenarios where intermediate node error correction is beneficial for the application. Our analysis offers valuable solutions for the design of practical streaming systems, and emphasizes the relevant trade-offs.

Our packet selection and scheduling analysis is presented next. Based on the knowledge of the media bitstream structure, and on a careful timing analysis of the packet transmission process, we identify optimal and heuristic scheduling algorithms for multipath streaming applications. Based on load balancing and prefetch window techniques we improve the streaming process in terms of application smoothness and number of late packet arrivals. Our methods is efficient in terms of network resources consumption and insures graceful quality degradation at the client when the network becomes unaccommodating. We also offer simple robustness mechanisms that protect the performance of the streaming process in the wake of undetected network variations or estimation errors. Our results show that the proposed algorithm along with the implemented robustness methods offer a fast scheduling solution that outperforms existing proposals.

Finally, we describe a possible practical system for multimedia services integrated in a general network scenario with clients accessing different types of applications. We discuss a possible multipath network scenario obtained by the inter-operability of parallel wireless services, where multiple clients can access various applications by connecting to one or more wireless networks. We address the path selection and rate allocation problem for each client, along with forward error correction decisions, in order to maximize the overall system performance under a unifying quality metric. Our analysis and algorithms take into account the connection parameters of each client in each of the accessed networks, and periodically compute an optimal system resource allocation, in order to cope with client dynamics and network variability. Our heuristic algorithm outperforms other methods, while the proposed unifying optimization metric achieves a more fair resource allocation than classical optimization metrics.

9.2 Future Directions

Recent developments in coding theory and applications open new research issues in the domain of real time applications over the internet. In particular, many-to-one streaming setups based on rateless codes appear promising, as this class of codes offers the decoding flexibility required by highly dynamic network systems. We identify peer-to-peer streaming systems as a suitable application that could benefit for the implementation of error correction strategies based on rateless codes. The simple implementation of such codes in distributed scenarios represents a great motivation for such systems. However, the real-time nature of such systems also poses several major problems in terms of content synchrony, application delays, and coding decisions. Future investigation of these aspects could provide solutions that bring the implementation of efficient peer-to-peer streaming systems closer to reality.

While part of the existing internet paradigm pushes the application processing and decision computation at the edge nodes, increased capabilities at intermediate nodes allow for in-network processing of traversing data flows. Network coding emerges as a powerful tool for throughput maximization in packet networks, based on simple linear operations performed on incoming packets at each router. Further extending the range of network coding applications for future streaming systems seems a natural step. Large scale streaming systems which require distributed implementations with no central authority could greatly benefit of such processing paradigms. However, efficient streaming systems based on network coding should address the inherent problems, e.g., real-time in-network shaping and adaptation of the incoming streams to variable network conditions, or minimizing in-network processing delays.

Finally, the streaming process could be analyzed from a cross layer design perspective. With a final goal of optimizing the application quality as perceived by the end user, decisions at the application layer should be based on the knowledge exchanged by different layers of the network stack. Highly variable network setups like wireless systems could greatly benefit from such strategies. The inter-operability among network layers could improve the overall system performance and ensure smoother transitions in the case of drastic network variations. In such a context we emphasize the importance of the trade-off between the increased application performance and the additional computation cost and execution time.

Bibliography

[1] L. Golubchik, J. Lui, T. Tung, A. Chow, and W. Lee, "Multi-Path Continuous Media Streaming: What Are the Benefits?" *ACM Journal of Performance Evaluation*, vol. 49, no. 1-4, pp. 429–449, Sept 2002.

[2] Y. Li, S. Mao, and S. S. Panwar, "The Case for Multimedia Transport over Wireless Ad Hoc Networks," in *Proceedings of IEEE/ACM BroadNets*, October 2004.

[3] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On Multiple Description Streaming with Content Delivery Networks," in *Proceedings of IEEE INFOCOM*, vol. 3, 23-27 June 2002, pp. 1736–1745.

[4] T. Nguyen and A. Zakhor, "Distributed Video Streaming over the Internet," in *Proceedings of Multimedia Computing and Networking (MMCN)*, January 2002.

[5] J. G. Apostolopoulos and M. D. Trott, "Path Diversity for Enhanced Media Streaming," *IEEE Communications Magazine*, vol. 42, no. 8, pp. 80–87, August 2004.

[6] ITU, *Recommendation H.264*, March 2005.

[7] H.M. Radha, M. van der Schaar and Y. Chen, "The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53–68, March 2001.

[8] S. Savage, A. Collins, and E. Hoffman, "The End-to-End Effects of Internet Path Selection," in *Proceedings of ACM SIGCOMM*, 1999.

[9] B. Abdouni, B. Cheng, A. L. H. Chow, and L. Golubchik, "Picture-Perfect Streaming over the Internet: Is There Hope?" *IEEE Communications Magazine*, pp. 72–79, August 2004.

[10] J. Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer-Verlag, 2001.

[11] R. L. Cruz, "A Calculus for Network Delays, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory, Vol. 37, No. 1*, 1991.

[12] W. Jiang and H. Schulzrinne, "Modeling of Packet Loss and Delay and Their Effect on Real-Time Multimedia Service Quality," in *Proceedings of the ACM International Workshop on Network and Operations Systems Support for Digital Audio and Video, NOSSDAV*, 2002.

[13] H. Sanneck and G. Carle, "A Framework for Packet Loss Metrics Based on Loss Runlength," in *Proceedings of the SPIE/ACM SIGMM*, 2000.

[14] F. Kelly, "Notes on Effective Bandwidths, in: F.P. Kelly, S. Zachary, I. Zeidins (Eds.), Stochastic Networks: Theory and Applications," pp. 141–168, 1996, Oxford Univerisity Press.

[15] Y. Bejerano and R. Rastogi, "Robust Monitoring of Link Delays and Faults in IP Networks," in *Proceedings of the Annual Computer Science Conference*, 2003.

[16] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of Passive and Active Measurement Workshop* , April 2003.

[17] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting Shared Congestion of Flows Via End-to-End Measurement," *IEEE Transactions on Networking, Vol. 10, No. 2*, vol. 10, no. 3, pp. 381–395, June 2002.

[18] A. Sang and S.-Q. Li, "A Predictability Analysis of Network Traffic," in *Proceedings of IEEE INFOCOM*, vol. 1, 2000, pp. 342–351.

[19] S. Tao and R. Guerin, "On-Line Estimation of Internet Path Performance: An Application Perspective," in *Proceedings of IEEE INFOCOM*, March 2004.

[20] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet," *Journal of IEEE/ACM Transactions on Netowrking (TON)*, vol. 10, no. 4, pp. 541–550, August 2002.

[21] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[22] Y. Cui, K. Xu, and J. Wu, "Precomputation for Multi-constrained QoS Routing in High-speed Networks," in *Proceedings of IEEE INFOCOM*, vol. 2, 2003, pp. 1414–1424.

[23] T. Korkmaz and M. M. Krunz, "Routing Multimedia Traffic with QoS Guarantees," *IEEE Transactions on Multimedia*, vol. 5, no. 3, pp. 429–443, September 2003.

[24] C. Brendan, S. Traw, and J. M. Smith, "Striping Within the Network Subsystem," *IEEE Network*, pp. 22–32, July/August 1995.

[25] E. Gustafsson and G. Karlsson, "A Literature Survey on Traffic Dispersion," *IEEE Network*, vol. 11, no. 2, pp. 28–36, March/April 1997.

[26] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: A Distance-Vector Multipath Routing Protocol," in *Proceedings of IEEE INFOCOM*, vol. 1, 2001, pp. 557–564.

[27] W. Wei and A. Zakhor, "Multipath Unicast and Multicast Video communication over Wireless Ad Hoc Networks," in *Proceedings of IEEE/ACM BroadNets*, October 2004.

[28] S. Bak, A. M. K. Cheng, J. A. Cobb, and E. L. Leiss, "Load-Balanced Routing and Scheduling for Real-Time Traffic in Packet-Switched Networks," in *Proceedings of IEEE Conference on Local Computer Networks*, 2000.

[29] M. Kurant and P. Thiran, " Survivable Routing of Mesh Topologies in IP-over-WDM Networks by Recursive Graph Contraction," *To appear in JSAC special issue on Traffic Engineering for Multi-Layer Networks*, 2007.

[30] Z. Ma, H.-R. Shao, and C. Shen, "A New Multi-Path Selection Scheme for Video Streaming on Overlay Networks," in *Proceedings of IEEE ICC*, 2004.

[31] W.-L. Yang, "Optimal and Heuristic Algorithms for Quality-of-Service Routing with Multiple Constraints," *ACM Performance Evaluation*, vol. 57, no. 3, pp. 261–278, July 2004.

[32] D. Xu, Y. Chang, Y. Xiong, C. Qiao, and X. He, "On Finding Disjoint Paths in Single and Dual Link Cost Networks," in *Proceedings of IEEE INFOCOM*, March 2004.

[33] F. Kelly and T. Voice, "Stability of End-to-End Algorithms for Joint Routing and Rate Control," *ACM SIGCOMM Computer Communcation Review*, vol. 35, no. 2, pp. 5–12, April 2005.

[34] K. Ma and K. M. Sim, "Loose-strain loop free conditions for multiple path IP routing," *IEE Proc.-Commun.*, vol. 151, no. 3, pp. 243–250, June 2004.

[35] S. Nelakuditi and Z.-L. Zhang, "On Selection of Paths for Multipath Routing," in *Proceedings of International Workshop on QoS*, June 2001.

[36] C. Cetinkaya and E. W. Knightly, "Opportunistic Traffic Scheduling Over Multiple Network Paths," in *Proceedings of IEEE INFOCOM*, 2004.

[37] Y. Lee, Y. Seok, and Y. Choi, "Traffic Engineering with Constrained Multipath Routing in MPLS Networks," *IEICE Transactions on Communications*, vol. E85-A, no. 1, January 2002.

[38] W. T. Zaumen and J. J. Garcia-Luna-Aceves, "Loop-Free Multipath Routing Using Generalized Diffusing Computations," in *INFOCOM (3)*, 1998, pp. 1408–1417.

[39] K. C. Leung and V. O. K. Li, "Flow Assignement and Packet Scheduling for Multipath Routing," *Journal of Communications and Networks*, vol. 5, no. 3, September 2003.

[40] J. Chen, S.-H. G. Chan, and V. O. K. Li, "Multipath Routing for Video Delivery Over Bandwidth-Limited Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 10, pp. 1920–1932, December 2004.

[41] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal Resource Allocation in Overlay Multicast," *Journal of IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 8, pp. 808–823, August 2006.

[42] S. Sarkar and L. Tassiulas, "A Framework for Routing and Congestion Control for Multicast Information Flows," *IEEE Transactions on Information Theory*, vol. 48, no. 10, pp. 2690–2708, October 2002.

[43] K. Foltz and J. Bruck, "Splitting Schedules for Internet Broadcast Communication," *IEEE Transactions on Inforamtion Theory, Vol. 42, N0. 2*, 2002.

[44] C. Boutremans and J.-Y. L. Boudec, "Adaptive Joint Playout Buffer and FEC Adjustement for Internet Telephony," in *Proceedings of IEEE INFOCOM*, 2003.

[45] S. Choi and Y. Kim, "Temporally Enhanced Erasure Codes for Real-Time Communication Protocols," *Journal of Computer Networks, vol 38*, 2002.

[46] C. Gkantsidis and P. R. Rodriguez, "Network Coding for Large Scale Content Distribution," in *Proceedings of IEEE INFOCOM*, March 2005.

[47] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[48] L. Libman and A. Orda, "Optimal Sliding-Window Strategies in Networks with Long Round-Trip Delays," in *Proceedings of IEEE INFOCOM*, 2003.

[49] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A Survey of Active Network Research," *IEEE Communication Magazine, Vol. 35, No.1*, 1997.

[50] T. Rappaport, A. annamalai, and W. Tranter, "Wireless Communications: Past Events and a Future Perspective," *IEEE Communications Magazine - 50th anniversary commemorative issue*, pp. 148–161, May.

[51] N. T. MODUS, "Selection of MCS levels in HSDPA," Tech. Rep. R1-01-0589, 2001.

[52] J. Yang, A. Khandani, and N. Tin, "Adaptive Modulation and Coding in 3G Wireless Systems," Tech. Rep. TR UW-EandCE-no.2002-15, 2002.

[53] G. 3rd Generation Partnership Project, "UMTS - UE Radio Access Capabilities (Release 1999)," Tech. Rep. STD-T63-25.306, 1999.

[54] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *Proceedings of ACM Mobicom*, October 2004, pp. 216–230.

[55] D. Qiao and S. Choi, "Goodput Enhancement of IEEE 802.11a Wireless LAN via Link Adaptation," in *Proceedings of IEEE ICC*, 2001.

[56] V. Srinivasan, C.-F. Chiasserini, P. S. Nugehalli, and R. R. Rao, "Optimal Rate Allocation for Energy-Efficient Multipath Routing in Wireless Ad Hoc Networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 3, pp. 891–899, May 2004.

[57] Y. Y. Kim and S. Q. Li, "Capturing Important Statistics of a Fading/Shadowing Channel for Network Performance Analysis," *IEEE Journal of Selected Areas in Communications*, vol. 17, no. 5, May 1999.

[58] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunisitc Media Access for Multirate Ad-Hoc Networks," in *Proceedings of ACM MOBICOM*, 2002.

[59] A. Valera, W. K. G. Seah, and S. V. Rao, "Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks," in *Proceedings of IEEE INFOCOM*, July 2003.

[60] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," in *Proceedings of IEEE INFOCOM*, vol. 2, 2005, pp. 882–893.

[61] Swisscom Mobile Unlimited UMTS/GPRS/WLAN, http://www.swisscom-mobile.ch/scm/gek_mobile-unlimited-en.aspx.

[62] G. 3rd Generation Partnership Project, "Feasability study on 3GPP system to Wireless Local Area Network WLAN interworking - Release 6," Tech. Rep. TR 22.934, 2003.

[63] K. Ahmavaara and H. Haverinen and R. Pichna, "Interworking Architecture between 3GPP and WLAN Systems," *IEEE Communications Magazine*, pp. 74–81, November 2003.

[64] X. G. Wang, J. Mellor, and K. Al-Begain, "Towards Providing QoS for Integrated Cellular and WLAN Networks," in *Proceedings of PGNET*, 2003.

[65] I. Akyildiz, Y. Altunbasak, F. Fekri, and R. Sivakumar, "AdaptNet: An Adaptive Protocol Suite for the Next-Generation Wireless Internet," *IEEE Communications Magazine*, 2004.

[66] P. Mahonen, J. Riihijarvi, M. Petrona, and Z. Shelby, "Hop-by-Hop Toward Future Mobile Broadband IP," *IEEE Communications Magazine*, 2004.

[67] G. J. Conklin, G. S. Greenbaum, and K. O. Lillevold, "Video Coding for Streaming Media Delivery on the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, 2001.

[68] J. Lu, "Signal Processing for Internet Video Streaming: A Review," in *Proceedings of SPIE Image and Video Communications and Processing*, 2000.

[69] B. Girod and N. Farber, "Feedback-Based Error Control for Mobile Video Transmissions," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1707–23, October 1999.

[70] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error Resilient Video Coding Techniques, Real-Time Video Communications over Unreliable Networks," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, July 2000.

[71] Y.-K. Wang, M. M. Hannuksela, and V. Varsa, "The Error Concelament Feature in the H.26L Test Model," in *Proceedings of IEEE ICIP*, 2002.

[72] Y. Wang and Q. Zhu, "Error control and concealment for video communications: A review," vol. 86, no. 5, pp. 974–997, May 1998, special issue on Multimedia Signal Processing.

[73] S. Wenger, "Video redundancy coding in H.263+," in *Proc. Workshop on Audio-Visual Services for Packet Networks*, Aberdeen, Scotland, Sep. 1997.

[74] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," vol. 18, no. 6, pp. 966–976, Jun. 2000.

[75] M. Zink, J. Schmitt, and C. Griwodz, "Layer-Encoded Video Streaming: A Proxy's Perspective," *IEEE Communications Magazine*, pp. 96–103, August 2004.

[76] Y. Wang, A. Reibman, and S. Lin, "Multiple Description Coding for Video Delivery," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 57–70, January 2005.

[77] R. Puri and K. Ramchandran, "Multiple Description Source Coding Through Forward Error Correction Codes," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, vol. 1. Asilomar, CA: IEEE, Oct. 1999, pp. 342–346.

[78] D. Saparilla and W. W. Ross, "Optimal Streaming of Layered Video," in *Proceedings of IEEE INFOCOM*, vol. 2, 26-31 March 2000, pp. 737–746.

[79] M. Dai, D. Loguinov, and H. M. Radha, "Rate-Distortion Analysis and Quality Control in Scalable Internet Streaming," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1135–1146, December 2006.

[80] F. Wu, H. Sun, G. Shen, S. Li, Y.-Q. Zhang, B. Lin, and M. C. Li, "SMART: An Efficient, Scalable and Robust Streaming Video System," *EURASIP Journal on Applied Signal Processing*, vol. 2, pp. 192–206, 2004.

[81] D. G. Sachs, I. Kozinetsev, M. Yeung, and D. L. Jones, "Hybrid ARQ for Robust Video Streaming over Wireless LANs," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, 2001, pp. 317–311.

[82] G. B. Horn, P. Knudsgaard, S. B. Lassen, M. Luby, and J. E. Rasmussen, "A Scalable and Reliable Paradigm for Media on Demand," *IEEE Computer Magazine, Vol. 34, No. 9*, 2001.

[83] Liang Y.J., Apostolopoulos J.G. and Girod, B., "Analysis of packet loss for compressed video: does burst-length matter?" in *Proceedings of IEEE ICASSP*, vol. 5, April 2003, pp. 884–687.

[84] P. Frossard, "FEC Performance in Multimedia Streaming," *IEEE Communication Letters*, 2001.

[85] P. Frossard and O. Verscheure, "Joint Source/FEC Rate Selection for Quality-Optimal MPEG-2 Video Delivery," *IEEE Transactions on Image Processing, vol.10, no.12*, 2001.

[86] Y.-C. Su, C.-S. Yang, and C.-W. Lee, "Optimal FEC Assignment for Scalable Video Transmission over Burst Error Channel with Loss Rate Uplink," in *Proceedings of the Packet Video, Nantes, France*, 2003.

[87] T.-W. A. Lee, S.-H. G. Chan, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Optimal Allocation of Packet-Level and Byte-Level FEC in Video Multicast over Wired and Wireless Networks," in *Proceedings of the IEEE Globecom*, 2001.

[88] H. Radha and M. Wu, "Overlay and Peer-to-Peer Multicast with Network-Embedded FEC," in *Proceedings of the IEEE ICIP*, 2004.

[89] Y. S. et al., "Overlay Multi-Hop FEC Scheme for Video Streaming over Peer-to-Peer Networks," in *Proceedings of the IEEE ICIP*, 2004.

[90] W.-T. Tan and A. Zakhor, "Error Control for Video Multicast using Hierarchical FEC," in *Proceedings of the 6th IEEE International Conference on Image Processing (ICIP)*, 1999.

[91] Q. Zhang, G. Wang, W. Zhu, and Y.-Q. Zhang, "Robust Scalable Video Streaming over Internet with Network-Adaptive Congestion Control and Unequal Loss Protection," in *Proceedings of the Packet Video Workshop EURASIP/IEEE*, 2001.

[92] B. Girod, J. Chakareski, M. Kalman, Y. J. Liang, E. Setton, and R. Zhang, "Advances in Network-adaptive Video Streaming," in *Proceedings of the 2002 Tyrrhenian International Workshop on Digital Communications (IWDC)*, 2002.

[93] J. Rexford and D. Towsley, "Smoothing Varaible-Bit-Rate Video in an Internetwork," *IEEE/ACM Transactions on Networking, Vol. 7*, 1999.

[94] P. Thiran, J. Y. L. Boudec, and F. Worm, "Network Calculus applied to optimal multimedia streaming," in *Proceedings of IEEE INFOCOM*, 2001.

[95] J. Y. B. Lee, "Buffer Management and Dimensioning for a Pull-Based Parallel Video Server," *IEEE Transactions on Ciruits and Systems for Video Technology, Vol. 11, No. 4*, 2001.

[96] J. Liu and J. Xu, "Proxy Caching for Media Streaming Over the Internet," *IEEE Communications Magazine*, pp. 88–94, August 2004.

[97] J. Brasill and T. Kim, "Large-Scale Personalized Video Streaming with Program Insertion Proxies," *IEEE Communications Magazine*, pp. 104–110, August 2004.

[98] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution," in *Proceedings of IEEE INFOCOM*, 2002.

[99] Z. Miao and A. Ortega, "Scalable Proxy Caching of Video under Storage Constraints," *IEEE Journal on Selected Areas in Communications, Special Issue on Internet Proxy Services*, 2002.

[100] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-Path vs. Multi-Path Overlay Routing," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurements*, October 2003, pp. 91–100.

[101] J. Chakareski, E. Setton, and B. Girod, "Video Streaming With Diversity," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2003.

[102] B. A. et al., "Multi-path Streaming: Optimization and Evaluation," in *Proceedings of the ACM MM*, 2004.

[103] R. Karrer and T. Gross, "Multipath Streaming in Best-Effort Netoworks," in *Proceedings of IEEE ICC*, vol. 2, 11-15 May 2003, pp. 901–907.

[104] A. C. Begen, Y. Altunbasak, O. Ergun, and M. H. Ammar, "Multi-path selection for multiple description video streaming over overlay networks," *Signal Processing: Image Communication*, vol. 20, pp. 39–60, 2005.

[105] H. Man and Y. Li, "A Multipath Video Delivery Scheme over Diffserv Wireless Lans," in *Proceedings of SPIE-IS&T Electronic Imaging*, 2004.

[106] D. Tian, Y.-C. Lee, G. AlRegib, and Y. Altunbasak, "Packetized Media Streaming over Multiple Wireless Channels," in *Proceedings of the IEEE Int. Conf. Communications (ICC)*, 2004.

[107] E. Setton, X. Zhu, and B. Girod, "Congestion-Optimized Multipath Streaming of Video over Ad Hoc Wireless Networks," in *Proceedings of IEEE ICME*, 2004.

[108] D. Tian, X. Li, G. Al-Regib, Y. Altunbasak, and J. Jackson, "Optimal Packet Scheduling for Wireless Streaming with Error-Prone Feedback," in *Proceedings of IEEE WCNC*, vol. 2, 21-25 March 2004, pp. 1287–1292.

[109] J. Chestefield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez, "Exploiting Diversity to Enhance Multimedia Streaming over Cellular Links," in *Proceedings of IEEE INFOCOM*, March 2005.

[110] S. A. Malik and D. Zeghlache, "Resource Allocation for Multimedia Services on the UMTS Downlink," in *Proceedings of IEEE International Conference on Communication*, vol. 5, 2002, pp. 3076–3080.

[111] T. Nguyen, P. Mehra, and A. Zakhor, "Path Diversity and Bandwidth Allocation for Multimedia Streaming," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, 6-9 July 2003, pp. 1–4.

[112] T. Nguyen and A. Zakhor, "Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks," in *Proceedings of IEEE INFOCOM*, vol. 3, 2003, pp. 663–672.

[113] Z.-L. Zhang, S. Nelakuditi, R. Aggrawal, and R. P. Tsang, "Efficient Selective Frame Discard Algorithms for Stored Video Delivery Across Resource Constrained Networks," *Real Time Imaging*, vol. 7, pp. 255–273, 2001.

[114] J. Huang, C. Krasic, and J. Walpole, "Adaptive Live Video Streaming by Priority Drop," in *Proceedings of Packet Video Workshop*, 2003.

[115] M. Roder, J. Cardinal, and R. Hamzaoui, "Branch and Bound Algorithms for Rate-Distortion Optimized Media Streaming," *IEEE Journal on Transactions on Multimedia*, vol. 8, no. 1, pp. 170–178, February 2006.

[116] Z. Miao and A. Ortega, "Expected Run-time Distortion Based Scheduling for Delivery of Scalable Media," in *Proceedings of the International Conference of Packet Video*, 2002.

[117] P. A. Chou and Z. Miao, "Rate-Distortion Optimized Streaming of Packetized Media," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 390–404, April 2006.

[118] Z. Jiang and L. Kleinrock, "A Packet Selection Algorithm for Adaptive Transmission of Smoothed Video Over a Wireless Channel," *IEEE Journal oj Parallel and Distributed Computing*, vol. 40, no. 4, pp. 494–509, April 2005.

[119] A. K. et al., "Advances in Efficient Resource Allocation for Packet-Based Real-Time Video Transmission," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 135–147, January 2005.

[120] E. Vergetis, R. Guerin, and S. Sarkar, "Realizing the Benefits of User-Level Channel Diversity," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 15 – 28, October 2005.

[121] J. Chen and S.-H. Chan, "Multipath Routing for Video Unicast over Bandwidth-Limited Networks," in *Proceedings of IEEE Globecom*, vol. 3, 25-29 November 2001, pp. 1963–1967.

[122] S. Tao and R. Guerin, "Application-Specific Path Switching: A Case Study for Streaming Video," in *Proceedings of ACM Multimedia*, October 2004, pp. 136–143.

[123] E. G. Steinbach, Y. J. Liang, and B. Girod, "A Simulation Study of Packet Path Divsersity for TCP File Transfer and Media Transport on the Internet," in *Proceedings of the 2002 Tyrrhenian International Workshop on Digital Communications (IWDC)*, 2002.

[124] Y. C. Lee, Y. Altunbasak, and R. M. Mesereau, "Optimal Packet Scheduling for Multiple Description Coded Video Transmissions over Lossy Networks," in *Proceedings of IEEE Globecom*, vol. 6, 1-5 December 2003, pp. 3569–3573.

[125] R. Zhang, S. L. Regunathan, and K. Rose, "End-to-End Distortion Estimation for RD-based Robust Delivery for Pre-compressed Video," in *Proceedings of Asylomar Conference on Signals, Systems and Computers*, vol. 1, 2001, pp. 210–214.

[126] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate Control for Robust Video Transmission over Burst-Error Wireless Channels," *IEEE Journal of Selected Areas in Communications*, vol. 17, no. 5, pp. 756–773, May 1999.

[127] J. Chakareski and P. Frossard, "Rate-Distortion Optimized Distributed Packet Scheduling of Multiple Video Streams over Shared Communication Resources," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 207–218, April 2006.

[128] Y. J. Liang, E. Setton, and B. Girod, "Channel-Addaptive Video Streaming using Packet Path Diversity and Rate-Distortion Optimized Reference Picture Selection," 2002.

[129] E. Setton and B. Girod, "Congestion-Distortion Optimized Scheduling of Video over a Bottleneck Link," in *Proceedings of the IEEE MMSP*, 2004.

[130] X. Zhu, S. Han, and B. Girod, "Congestion-Aware Allocation for Multipath Video Streaming over Ad Hoc Wireless Networks," in *Proceedings of IEEE ICIP*, 2004.

[131] J. Chakareski and B. Girod, "Rate-distortion Optimized Packet Scheduling and Routing for Media Streaming with Path Diversity," in *Proceedings of the IEEE Data Compression Conference (DCC)*, 25-27 March 2003, pp. 203–212.

[132] ——, "Server Diversity in Rate-Distortion Optimized Media Streaming," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 3, 14-17 September 2003, pp. III – 645–8 vol.2.

[133] M. Kalman, E. Steinbach, and B. Girod, "Rate-Distortion Optimized Video Streaming with Adaptive Playout," in *Proceedings of IEEE ICIP*, 2002.

[134] A. C. Begen, Y. Altunbasak, and M. A. Begen, "Rate-Distortion Optimized On-Demand Media Streaming with Server Diversity," in *Proceedings of IEEE ICIP*, 2003.

[135] K. Chebrolu and R. Rao, "Bandwidth Aggregation for Real-Time Applications in Heterogeneous Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 388–403, April 2006.

[136] S. H. Kang and A. Zakhor, "Effective Bandwidth Based Scheduling for Streaming Multimedia," in *Proceedings of IEEE ICIP*, 2003, pp. 633–636.

[137] K. Chebrolu and R. R. Rao, "Selective Frame Discard for Interactive Video," in *Proceedings of IEEE ICC*, 2004, pp. 4097–4102.

[138] S. Wee, W.-T. Tan, J. Apostolopoulos, and M. Etoh, "Optimized Video Streaming for Networks with Varying Delay," in *Proceedings of IEEE Conference on Multimedia and Expo (ICME'02)*, July 2002.

[139] J.-W. Ding, Y.-M. Huang, and C.-C. Chu, "An End-to-End Delivery Scheme for Robust Video Streaming," in *Proceedings of Second IEEE Pacific Rim Conference on Multimedia (PCM)*, 2001.

[140] W. Xu and S. Hemami, "Delay-Optimized Robust Transmission of Images over Multiple Channels," in *Proceedings of the IEEE ICME*, 2003.

[141] T. Stockhammer, M. Hannuksela, and T. Wiegand, "H.264/AVC in Wireless Environments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 657–673, July 2003.

[142] M. Etoh and T. Yoshimura, "Advances in Wireless Video Delivery," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 111–122, January 2005.

[143] Y. Wang, S. Panwar, S. Lin, and S. Mao, "Wireless Video Transport Using Path Diversity: Multiple Description vs. Layered Coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 1, 22-25 September 2002, pp. 21–24.

[144] H. Holma and A. Toskala, *WCDMA for UMTS*. Wiley and Sons, 2001.

[145] M. Rossi, F. H. P. Fitzek, and M. Zorzi, "Error Control Techniques for Efficient Multicast Streaming in UMTS Networks: Proposals and Performance Evaluation," in *Proceedings of SCI*, 2003.

[146] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, October 2003.

[147] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer, "Application-Driven Cross Layer Optimization for Video Streaming over Wireless Networks," *IEEE Communications Magazine*, pp. 122–130, January 2006.

[148] M. van der Schaar and S. Shankar, "Cross-Layer Wireless Multimedia Transmission: Challanges, Principles, and New Paradigms," *IEEE Wireless Communications*, vol. 12, no. 4, pp. 50–58, August 2005.

[149] J. Gross, J. Klaue, H. Karl, and A. Wolisz, "Cross-Layer Optimization of OFDM Transmission Systems for MPEG-4 Video Streaming," *Computer Communications*, vol. 27, pp. 1044–1055, 2004.

[150] M. van der Schaar, S. Krishnamachari, S. Choi, and X. Xu, "Adaptive Cross-Layer Protection Strategies for Robust Scalable Video Transmission over 802.11 WLANs," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1752–1763, December 2003.

[151] Y. Huh, M. Hu, M. Reisslein, and J. Zhang, "MAI-JSQ: A Cross-Layer Design for Real-Time Video Streaming in Wireless Networks," in *Technical Report, Telecommunications Research Center, Dept. of Electrical Engineering, Arizona State University*, August 2002.

[152] W. Karner, O. Nemethova, P. Svoboda, and M. Rupp, "Link Error Prediction Based Cross-Layer Scheduling for Video Streaming over UMTS," in *Technical Report, Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology*, 2005.

[153] L.-U. Choi, W. Kellerer, and E. Steinbach, "Cross Layer Optimization for Wireless Multi-User Video Streaming," in *Proceedings of IEEE ICIP*, 2004.

[154] ——, "On Cross-Layer Design for Streaming Video Delivery in Multiuser Wireless Environments," *EURASIP Journal on Wireless Communications and Networking*, vol. 2006, pp. Article ID 60 349, 10 pages, 2006.

[155] W. Kellerer, L.-U. Choi, and E. Steinbach, "Cross-Layer Adaptation for Optimized B3G Service Provisioning," in *Proceedings of the 6th Intl. Symposium on Wireless Personal Multimedia Communications WPMC, Japan*, October 2003.

[156] M. Ivrlac and J. Nossek, "Cross Layer Design - An Equivalence Class Approach." in *Proc. IEEE International Symposium on Signals, Systems, and Electronics*, 2004.

[157] S. Khan, S. Duhovnikov, E. Steinbach, M. Sgroi, and W. Kellerer, "Application-driven cross-layer optimization for mobile multimedia communication using a common application layer quality metric." in *Proceedings of Second International Symposium on Multimedia over Wireless, ISMW*, July 2006.

[158] T. Ahmed and A. Mehaouda and R. Boutaba and Y. Iraqi, "Adaptive Packet Video Streaming over IP Networks: A Cross Layer Approach," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 2, pp. 385–401, February 2005.

[159] W. Yuan, K. Nahrstedt, S. V. Adve, D. L. Jones, and R. H. Kravets, "Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems," in *Proceedings of SPIE/ACM Multimedia Computing and Networking Conference MMCN*, January 2003, pp. 1–13, santa Clara, CA.

[160] V. Kawadia and P. R. Kumar, "A Cautionary Perspective on Cross-Layer Design," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 3–11, February 2005.

[161] A. Ganjam and H. Zhang, "Internet Multicast Video Delivery," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 159–170, January 2005.

[162] X. Jia, "A Distributed Algorithm of Delay-Bounded Multicast Routing for Multimedia Applications in Wide Area Networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 828–837, December 1998.

[163] J. N. E. Setton and B. Girod, "Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming," in *Workshop on Advances in Peer-to-Peer Multimedia Streaming at ACM Multimedia*, November 2005, pp. pp 39–48.

[164] X. Su, Y. Shang, T. Wang, and Y. Mai, "Delay-Constrained Transmission of Fine Scalable Coded Content over P2P Networks," *Signal Processing: Image Communication*, vol. 20, pp. 21–37, January 2005.

[165] L. K. et al., "A Transport Layer for Live Streaming in a Content Delivery Network," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1408–1419, September 2004.

[166] K. Hua, M. Tantaoui, and W. Tavanapong, "Video Delivery Technologies for Large-Scale Deployment of Multimedia Applications," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1439–1451, September 2004.

[167] Q. Zhang, W. Zhu, and Y. Zhang, "End-to-End QoS for Video Delivery Over Wireless Internet," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 123–134, January 2005.

[168] R. Rodrigues, B. Liskov, and L. Shrira, "The design of a robust peer-to-peer system," in *Proceedings of SIGOPS European Workshop*, 2002.

[169] H.-Y. Hsieh and R. Sivakumar, "Accelerating Peer-to-Peer Networks for Video Streaming using Multipoint-to-Point Communication," *IEEE Communications Magazine*, pp. 111–119, August 2004.

[170] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The Bittorrent P2P File-sharing System: Measurements and Analysis," in *4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.

[171] S. A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," in *Proceedings of IEEE INFOCOM*, 2006.

[172] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributed Streaming Media Content using Cooperative Networking," in *Proceedings of ACM NOSSDAV*, 2002.

[173] Coolstreaming IPTV, http://www.coolstreaming.us.

[174] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming," in *Proceedings of IEEE INFOCOM*, vol. 3, 13-17 March 2005, pp. 2102– 2111.

[175] N. Magharei and R. Rejaie, "Understanding Mesh-based Peer-to-Peer Streaming," in *Proceedings of ACM NOSSDAV'06*, newport, Rhode Island USA.

[176] V. N. Padmanabhan and K. Sripanidkulchai, "The Case for Cooperative Networking," in *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, USA*, March 2002.

[177] E. Setton, X. Zhu, and B. Girod, "Minimizing Distortion for Multi-Path Video Streaming over Ad Hoc Networks," in *Proceedings of IEEE ICIP*, 2004.

[178] S. Mao, S. Lin, S. S. Panwar, Y. Wang, and E. Celebi, "Video Transport Over Ad Hoc Networks: Multistream Coding With Multipath Transport," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1721–1737, December 2003.

[179] J. Kim, M. Mersereau, and Y. Altunbasak, "Distributed Video Streaming using Unbalanced Multiple Description Coding and Forward Error Correction," in *Proceedings of IEEE Globecom*, 2003.

[180] X. Zhu and B. Girod, "Distributed Rate Allocation for Multi-Stream Video Transmission over Ad-Hoc Networks," in *Proceedings of IEEE ICIP*, 2005.

[181] T. Nguyen and A. Zakhor, "Multiple Sender Distributed Video Streaming," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 315–326, April 2004.

[182] D. Jurca, S. Petrovic, and P. Frossard, "Media Aware Routing in Large Scale Networks with Overlay," in *Proceedings of IEEE ICME*, July 2005.

[183] Y. J. Liang, J. G. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: Does burst-length matter?" in *Proceedings of IEEE ICASSP*, 2003.

[184] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of Video Transmission over Lossy Channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1012–1032, June 2000.

[185] M. Dai and D. Loguinov, "Analysis of rate-distortion functions and congestion control in scalable internet video streaming," in *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video.* New York, NY, USA: ACM Press, 2003, pp. 60–69.

[186] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, 2nd ed. Springer-Verlag, 2000.

[187] P. von Rickenbach, S. Schmid, R. Wattenhofer, and A. Zollinger, "A Robust Interference Model for Wireless Ad-Hoc Networks," in *Proceedings of IEEE WMAN'05*, 2005.

[188] G. Epshtein, "A Better Way to Implement Video over WLAN," in *Metalink - white paper*, 2005.

[189] D. Jurca and P. Frossard, "Video Packet Selection and Scheduling for Multipath Streaming," *IEEE Transactions on Multimedia*, vol. 9, pp. 629–641, April 2007.

[190] N. Lynch, *Distributed Algorithms.* San Mateo, CA: Morgan Kaufmann Publishers Inc., 1996.

[191] D. L. Mills, "Network Time Protocol Implementation, Specification and Analysis," Network Working Report RFC 1305, Tech. Rep., March 1992.

[192] D. Jurca and P. Frossard, "Media-Specific Rate Allocation in Multipath Networks," *IEEE Transactions on Multimedia*, 2006, accepted for publication.

[193] http://www.icir.org/models/tools.html.

[194] http://www.isi.edu/nsnam/ns/.

[195] A. Jovanovic, "Media Aware Rate Allocation and FEC Protection of Streaming Video in Multipath Networks," Master's thesis, EPFL, March 2007.

[196] M. R. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, 23rd ed., V. Klee, Ed. W. H. Freeman and Company, 2002.

[197] J. Jonsson and K. G. Shin, "A Parametrized Branch and Bound Strategy for Scheduling Precedence-Constrained Tasks on a Multiprocessor System," in *Proceedings of IEEE ICPP*, 11-25 August 1997, pp. 158–165.

[198] D. Jurca and P. Frossard, "Distortion Optimized Multipath Video Streaming," in *Proceedings of Packet Video Workshop*, 2004.

[199] F. A. Samadzadeh and G. E. Hedrick, "Near-Optimal Multiprocessor Scheduling," in *Proceedings of ACM Computer Science Conference*, April 1992, pp. 477 – 484.

[200] D. Jurca and P. Frossard, "Packet Selection and Scheduling for Multipath Video Streaming," EPFL, Tech. Rep. TR-ITS-2004.025, November 2004.

[201] P. de Cuetos, M. Reisslein, and K. W. Ross, "Evaluating the Streaming of FGS-Encoded Video with Rate-Distortion Traces," Institut Eurecom, Technical Report RR-03-078, 2003.

[202] D. Jurca and P. Frossard, "Media Streaming with Conservative Delay on Variable Rate Channels," in *Proceedings of the IEEE International Conference on Multimedia and Expo' (ICME)*, July 2006.

[203] S. Khan, S. Duhovnikov, E. Steinbach, and W. Kellerer, "MOS-based multi-user multi-application Cross-layer Optimization for Mobile Multimedia Communication."

Curriculum Vitae

Full name:	**Dan Jurca**
Degrees:	Master of Science in Telecommunications, "Politehnica" University Timisoara, Romania,
Address:	Signal Processing Laboratory (LTS4) Signal Processing Institute (ITS) School of Engineering (STI) Swiss Federal Institute of Technology (EPFL) ELD 241, Batiment ELD Station 11 CH-1015 Lausanne Switzerland
Contact numbers:	Tel. (+41 21) 693 4329 Fax. (+41 21) 693 76 00 E-mail: dan.jurca@epfl.ch
Civil status:	Single
Date and place of birth:	August 05th 1979, Romania
Nationality:	Romanian

Professional Experiences

Since 2003	Research Assistant at the Signal Processing Institute (ITS), Swiss Federal Institute of Technology at Lausanne (EPFL) - leading to a Ph.D. in media streaming. Under the direction of Prof. Pascal Frossard.
	• Research activities: network protocols, distributed algorithms, media streaming, media encoding, optimization problems.
	• Teaching: supervision of several semester, masters or pre-doctoral student projects.
	• Other: webmaster Multimedia Streaming Interest Group, TPC member and reviewer for several international conferences.
2006 ***4 months***	research internship, DoCoMo Euro-Labs, Munich, Germany, System optimization for multiuser access to multiple application over parallel wireless access networks.
2002 ***4 months***	research internship, Fachhochschule Gelsenkirchen, Germany, Implementation of a 3D object tracking platform with a multi-camera system.
2000 ***2 months***	industrial internship, Kathrein Werke GmbH, Rosenheim, Germany, Production planning and antenna assembly.
1999 ***1 year***	Regional Manager, ARDOR Banat, Timisoara, Romania, project management, coordinating the activities, evaluation and reporting, primary accounting.

Studies

Since 2003	Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland Signal Processing Institute (ITS) of the School of Engineering (STI) Working towards a Ph.D. thesis (Docteur ès Sciences).
2002-2003	Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland Pre-Doctoral School, Computer and Communication Sciences.
1997-2002	"Politehnica" University in Timisoara, Romania MSc in Telecommunications.

Languages

Romanian:	mother tongue
English:	fluent
French:	fluent
German:	intermediate

Computer Skills

Programming languages:	Matlab , C/C++, x86 Assembler, TMS320C5x,HTML, PHP.
Operating Systems:	Linux, Windows.

Publications

Journal Papers

- D. Jurca, P. Frossard: "Packet Selection and Scheduling for Multipath Video Streaming", IEEE Transactions on Multimedia, vol. 9, no. 3, pp. 629-641, April 2007.
- D. Jurca, P. Frossard: "Packet Media Streaming with Imprecise Rate Estimation", Advances In Multimedia, Hindawi Publishing, Volume 2007, Article ID 39524 , 2007..
- D. Jurca, J. Chakareski, J.-P. Wagner, P. Frossard: "Enabling Adaptive Video Streaming in P2P Systems", IEEE Communications Magazine, Special Issue on P2P Multimedia Streaming, pp. 108-114, June 2007.
- D. Jurca, P. Frossard: "Media-Specific Rate Allocation in Multipath Overlay Networks", to appear in IEEE Transactions on Multimedia, Sept. 2007.
- D. Jurca, P. Frossard: "Distributed Media Rate Allocation in Multipath Networks", submitted to IEEE Transactions on Multimedia.

Conference Papers

- D. Jurca, W. Kellerer, E. Steinbach, S. Khan, S. Thakolsri, P. Frossard: "Joint Network and Rate Allocation for Video Streaming over Multiple Wireless Networks", submitted to the International Symposium on Multimedia, Taiwan, 2007.
- D. Jurca, W. Kellerer, E. Steinbach, S. Khan, S. Thakolsri, P. Frossard: "Joint Network and Rate Allocation for Simultaneous Wireless Applications", IEEE International Conference on Multimedia and Expo, Beijing, China, 2007.
- D. Jurca, P. Frossard: "Media Streaming with Conservative Delay on Variable Rate Channels", IEEE International Conference on Multimedia and Expo, Toronto, Canada, 2006.
- D. Jurca, P. Frossard: "Distributed Rate Allocation for Media in Overlay Networks", IEEE International Conference on Multimedia and Expo, Toronto, Canada, 2006, (invited paper).
- D. Jurca, P. Frossard: "Media Specific Rate Allocation in Heterogeneous Wireless Networks", International Packet Video Conference, Hangzhou, China, 2006 (invited paper).
- D. Jurca, S. Petrovic, P. Frossard: "Media Aware Routing in Large Scale Networks with Overlay", IEEE International Conference on Multimedia and Expo, Amsterdam, The Netherlands, 2005.
- D. Jurca, P. Frossard: "Distortion Optimized Multipath Video Streaming", IEEE International Packet Video Workshop, California, USA, 2004.
- D. Jurca, J.-P. Hubaux: "Joint Synchronization, routing and Energy Saving in CSMA/CA Hybrid Networks", IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Florida, USA, 2004.
- D. Jurca, P. Frossard: "Optimal FEC Rate for Media Streaming in Active Networks", IEEE International Conference on Multimedia and Expo, Taipei, Taiwan, 2004.

- D. Jurca, L. Toma jr.: "Medicine Tracking: Implementation of a 3D Tracker with a Multi Camera System", Proceeding of the Symposium on Electronics and Telecommunications "ETc 2002", edited by The "POLITEHNICA" University of Timisoara, IEEE and The Association on Electronic Engineers from Timisoara, Romania, 2002.
- L. Toma jr., D. Jurca: "Isolated Word Recognition System" , Proceedings of the Symposium on Electronics and Telecommunications "ETc.2000", edited by The "POLITEHNICA" University of Timisoara, IEEE and The Association on Electronic Engineers from Timisoara, Romania, 2000.

Technical Reports

- D. Jurca, P. Frossard: "Distributed Media Rate Allocation in Multipath Networks", TR-ITS-2006.009, EPFL, Switzerland, September, 2006
- D. Jurca, P. Frossard: "Media-Specific Rate Allocation in Multipath Overlay Networks", TR-ITS-2005.32, EPFL, Switzerland, November 2005.
- D. Jurca, P. Frossard: "Packet Selection and Scheduling for Multipath Video Streaming", TR-ITS-2004.025, EPFL, Switzerland, November, 2004.
- D. Jurca, J-P. Hubaux: "Joint Synchronization, Routing and Energy Saving in CSMA/CA Multi-Hop Hybrid Networks", TR-200435, EPFL, Switzerland, April, 2004.

Master Thesis

- Dan Jurca: "Medicine Tracking: Implementation of a 3D Tracker with a Multi Camera System", Master thesis, "Politehnica" University Timisoara, Romania, June 2002.