APPLICATION LAYER SYSTEM DESIGN FOR REAL-TIME VIDEO COMMUNICATIONS

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY
(Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Eymen Kurdoglu

May 2017

APPLICATION LAYER SYSTEM DESIGN FOR REAL-TIME VIDEO COMMUNICATIONS

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY

TANDON SCHOOL OF ENGINEERING

by

Eymen Kurdoglu

May 2017

Approved:

____

Department Chair Signature

____

Date

University ID:	**N13615561**
Net ID:	**ek1666**

Approved by the Guidance Committee:

Major: Electrical and Computer Engineering

Yao Wang
Professor
in Electrical and Computer Engineering

Date

Yong Liu
Associate Professor
in Electrical and Computer Engineering

Date

Sundeep Rangan
Associate Professor
in Electrical and Computer Engineering

Date

Shivendra Panwar
Professor
in Electrical and Computer Engineering

Date

Vita

Eymen Kurdoglu received B.Sc. degrees in Electrical and Electronics Engineering (2007), and in Physics (2008) from the Middle East Technical University, Ankara, Turkey, and an M.Sc. degree from the School of Computer and Communication Sciences (2010), EPFL, Lausanne, Switzerland. Since 2012 he has been working towards the Ph.D. degree at the Electrical and Computer Engineering Department, New York University Tandon School of Engineering, Brooklyn, NY, USA, under the supervision of Prof. Yao Wang and Prof. Yong Liu. His research interests are P2P networks, video encoding, interactive video communications and multimedia streaming. He is a student member of IEEE and ACM.

Acknowledgments

I would like to thank my advisors, Prof. Yao Wang and Prof. Yong Liu, for all their help, support, and teachings. This thesis would certainly not come to pass without their patience. They taught me quite a few pillars of critical thinking and doing research, such as the courage to ask simple questions regardless of the audience or topic of discussion, appreciating the importance of details in almost everything (for the devil is in the details), and the awareness of how much I know (or don't know). In addition to all this, they are the kindest and the most understanding advisors that a PhD student can ever ask for, helping me at the start, in the middle and the end of my journey here. I'm glad that I have known and befriended them.

Besides my advisors, I would also like to express my deepest gratitude to my guidance committee members, Prof. Sundeep Rangan and Prof. Shivendra Panwar. Their greater knowledge and insightful comments have been very helpful for my research and development of this thesis.

I was also very lucky to have found so many good friends during my time at Poly. I owe my thanks to all, but especially to Mustafa, Nico, Hao, Fanyi, Sourjya, Parisa, Beril, Fraida, Sanjay, Zhili, Amir, Yasin, Anil, Georgios, Oner and Tunc, as well as the Columbia crew. They made my life bearable in my times of need, and straight up fun at other times.

Last but not least, I thank and dedicate this thesis to my parents and sister for their endless love and support.

Dedicated to my family and all my friends

ABSTRACT

APPLICATION LAYER SYSTEM DESIGN FOR REAL-TIME VIDEO COMMUNICATIONS

by

Eymen Kurdoglu

Advisors: Yao Wang, Ph.D. & Yong Liu, Ph.D.

Submitted in Partial Fulfillment of the Requirements for

the Degree of Doctor of Philosophy (Electrical Engineering)

May 2017

Given that the world is becoming ever more connected each day, real-time video communications is an avenue of technology with untapped potential that can bring the human beings ever closer. The main objective of this thesis is to improve the state-of-the-art in the field of real-time video communications with respect to the end-users' quality of experience, through application layer system design, which jointly considers the video and the networking components.

We first focus on designing peer-to-peer multi-party video conferencing systems, where the users with different uplink-downlink capacities send their videos using multicast trees. One way to deal with user bandwidth heterogeneity is employing layered video coding, generating multiple layers with different rates, whereas an alternative is partitioning the receivers of each source and disseminating a different non-layered video version within each group. Here, we aim to maximize the average received video quality for both systems under uplink-downlink capacity constraints, while constraining the number of hops the packets traverse to two. We first show any multicast tree is equivalent to a collection of 1-hop and 2-hop trees, under user uplink-downlink capacity constraints. This reveals that the packet overlay hop count can be limited to two without sacrificing the achievable rate performance. Assuming a fine granularity scalable stream that can be truncated at any rate, we propose an algorithm that solves for the number of video layers, layer rates and distribution trees for the layered system. For the partitioned simulcast system, we develop an algorithm to determine the receiver partitions along with the video rate and the distribution trees for each group. Through numerical comparison, we show that the partitioned simulcast system achieves the same average receiving quality as the ideal layered system without any coding overhead, and better quality than the layered system with a coding overhead of only 10% for the 4-user systems simulated. The two systems perform similarly for the 6-user case if the layered coding overhead is 10%, and partitioned simulcast achieves higher received quality when the layered coding overhead is beyond 10%.

Next, we study interactive video calls between two users, where at least one of the users is connected over a cellular network. It is known that cellular links present highly-varying network bandwidth and packet delays. If the sending rate of the video call exceeds the available bandwidth, the video frames may be excessively delayed, destroying the interactivity of the video call. Here, we present Rebera, a cross-layer design of proactive congestion control, video encoding and rate adaptation, to maximize the video transmission rate while keeping the one-way frame delays sufficiently low. Rebera actively measures the available bandwidth in real time by employing the video frames as packet trains. Using an online linear adaptive filter, Rebera makes a history-based prediction of the future capacity, and determines a bit budget for the video rate adaptation. Rebera uses the hierarchical-P video encoding structure to provide error resilience and to ease rate adaptation, while maintaining low encoding complexity and delay. Furthermore, Rebera decides in real time whether to send or discard an encoded frame, according to the budget, thereby preventing self-congestion and minimizing the packet delays. Our experiments with real cellular link traces show that Rebera can, on average, deliver higher bandwidth utilization and shorter packet delays than Apple's FaceTime.

Finally, we consider video calls affected by bursty packet losses, where forward error correction (FEC) is applied on a per-frame basis due to tight delay constraints. In this scenario, both the encoding (eFR) and the decoded (dFR) frame rates are crucial;

a high eFR at low bitrates leads to larger quantization stepsizes (QS), smaller frames, hence suboptimal FEC, while a low eFR at high bitrates diminishes the perceptual quality. Coincidently, damaged frames and others predicted from them are typically discarded at receiver, reducing dFR. To mitigate frame losses, hierarchical-P coding (hPP) can be used, but at the cost of lower coding efficiency than IPP..I coding (IPP), which itself is prone to abrupt freezing in case of loss. Here, we study the received video call quality maximization for both hPP and IPP by jointly optimizing eFR, QS and the FEC redundancy rates, under the sending bitrate constraint. Employing Q-STAR, a perceptual quality model that depends on QS and average dFR, along with R-STAR, a bitrate model that depends on eFR and QS, we cast the problem as a combinatorial optimization problem, and employ exhaustive search and hill climbing methods to solve explicitly for the eFR and the video bitrate. We also use a greedy FEC packet distribution algorithm to determine the FEC redundancy rate for each frame. We then show that, for iid losses, (i) the FEC bitrate ratio is an affine function of the packet loss rate, (ii) the bitrate range where low eFR is preferred gets wider for higher packet loss rates, (iii) layers are protected more evenly at higher bitrates, and (iv) IPP, while achieving higher Q-STAR scores, is prone to abrupt freezing that is not considered by the Q-STAR model. For bursty losses, we show that (i) layer redundancies are much higher, rising with the mean burst length and reaching up to 80%, (ii) hPP achieves higher Q-STAR scores than IPP in case of longer bursts, and (iii) the mean and the variance of decoded frame distances are significantly smaller with hPP.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It should be no surprise to us that the example given for the meaning of “real time” in the New Oxford English dictionary, is about *“watching events unfold in real time on TV”*. This phrase, being synonymous with “experiencing without any observable delay”, is almost entirely used in the context of telecommunications. Indeed, delay-free visual and acoustic telecommunication is arguably the most fundamental form of direct interactive communication for human beings, and hence, creating real-time telecommunication machines that allow two or more people, who are spatially apart, to see and hear each other has been the goal of many inventors and engineers since the telephone was patented in 1876. Today, we all reap the combined benefits of the efforts that have been made for over a century; communication in real time through audio and video has become a reality, and has gone on to become an integral part of our daily lives in the last decade.

Be that as it may, there is very likely room for improvement for such systems. The means of real-time communication that are used today are far from perfect. It is a common experience for many people to have started a video call with an uneasy “can you hear/see me?” question. The visual signal on our displays are sometimes choppy, full of artifacts, or the latency so high that it becomes meaningless to continue the video call. In this dissertation, the main goal is to investigate the means and methods to improve the quality of experience for the users of such real-time video communication systems, from an application layer point of view. In other words, we will examine particular methods given the type or speed of the communication media on which the connection is established, or given the capabilities of the visual sensors that enable us to obtain the video in the first place. In the upcoming sections, we first describe the state-of-the-art, along with the preliminaries. Then, we address the challenges of such systems, and briefly mention our contributions as to how these challenges can be met.

1.1 Background

Modern multimedia communication systems operate over packet-switched networks. Therefore, each component of such a system manipulates digital data, while sitting on a particular layer of the network stack [1]. While the components in any given layer of the network stack, be it the physical layer, the link layer, the network layer, the transport layer, or the application layer, can be tailored to better facilitate the multimedia communication, we are concerned only with those that are either on the application layer directly, or on those that can be considered on both the transport and application layers.

The functionality and therefore the design of multimedia communication systems is determined by the set of end-users. Depending on whether they create and/or consume the multimedia data, the end-users can be *sources*, *sinks*, or *both*. A typical source (sink) consists of video and audio encoders (decoders) at the application layer, packet transmitters (receivers) at the transport layer, and data queues in-between the layers. Among these, *video coding* and *video transmission* are considerably more challenging to design, compared to their audio counterparts, due to their intense computation and bitrate requirements. Consequently, almost all of the research efforts on multimedia communications in the last two decades have been concentrated on the more interesting video aspects. Following a similar approach, only video communications is considered in this dissertation.

The research that has been done so far in this field can be categorized in terms of the latency requirement of the underlying application. On the one hand, video-on-demand (VoD) and live video streaming systems have relatively relaxed latency requirements, on the order of tens of seconds and a few seconds, respectively. On the other hand, real-time video delivery requires delays of at most a few hundred milliseconds [2]. This is especially challenging in the case of interactive video applications, such as video calling and video conferencing, since the latency that a particular end-user experiences with another is the round trip time (RTT) of the connection between the corresponding parties. In this dissertation, by video calling, we will refer to an interactive, real-time video communication between two end-users on the Internet, while video conferencing or multi-party video conferencing will refer to a collection of video calling sessions between more than two end-users.

1.2 Challenges and Contributions

In this section, we go through the main challenges of real-time video delivery for video calling and video conferencing, and state the contributions of this dissertation for each challenge described. We start with video conferencing in the first part, that is, we consider a high-level design approach for peer-to-peer video conferencing. In doing so, our aim is to study the inter-play between the choice of the video coder, i.e., layered encoding or non-layered encoding, and the distribution method for the particular type of video, which amounts to determining the overlay network connections built at the application layer. Then, in the second part, we focus on video calling between two users, and identify two main design challenges as (i) how to adapt the sending bitrate of the source dynamically by predicting the amount of available network bandwidth in the immediate future, and (ii) given the total sending bitrate, how to maximize the video quality in the presence of packet losses, by jointly optimizing video encoding and FEC parameters. We address the first problem within the challenging context of cellular networks, and present a simple and effective method to estimate the available network bandwidth and adapt the sending bitrate. Next, we turn our attention to the second problem, and show how the problem of providing packet loss resiliency is coupled with the maximization of video quality. After solving the combinatorial optimization problem that follows, we compare the solutions regarding two mainstream video coding structures and present a discussion of our results.

1.2.1 High Level Design of P2P-MPVC Systems

Advances in broadband and video encoding technologies have enabled multi-party video conferencing (MPVC) applications, such as [3], [4], and [5], to flourish on the Internet. Most existing MPVC solutions are server-centric [6], such as [7] [8] [9]. These products may prove costly to the end-user and/or ignore the geographic location of users in the same conference; video streams of users located far from the servers may traverse long-delay paths, even if there is no transcoding delay involved or low-delay selective forwarding units (SFUs) are employed [10]. Another delivery solution for MPVC is *peer-to-peer (P2P)*, where users send their data to each other directly. P2P-MPVC, where multiple users multicast voice and video with intense bandwidth requirements in real-time, has to deal with the inherent heterogeneity of users in the same conference, in terms of uplink/downlink capacity, computation and energy supply. The prior P2P-MPVC solutions proposed in [11] [12] [13] [14] [15] have been shown to achieve low delay by exploiting user locality and high rate in face of user uplink capacity heterogeneity. However, in a typical MPVC scenario, each user downloads multiple streams, so the downlink may potentially become a bottleneck for some users, even if their downlink capacity is higher than their uplink capacity. To deal with the user downlink capacity heterogeneity, one solution is *layered coding*, where each source encodes multiple video layers using layered video coding techniques, e.g. SVC [16], and receivers with higher downlink capacity can download more layers to receive better video quality. An alternative solution is *partitioned simulcasting*, where receivers are partitioned into groups, with receivers in each group having similar downlink capacities, and receiving the same video that is encoded using single-layer video coding techniques, e.g. AVC [17]. In this solution, each source sends multiple single-layer videos at different rates to different groups. For P2P-MPVC, layered coding and partitioned simulcasting enable different P2P sharing opportunities. With partitioned simulcast, only receivers watching the same version can share video with each other; while with layered coding, users receiving different sets of video layers can still share their common layers, leading to a higher P2P sharing efficiency. However, the flexibility of layered coding comes at the price of non-negligible rate overhead, that is, to achieve the same perceptual quality, layered coding has to use higher bitrate[1] than non-layered coding [16]. Because of this rate overhead, the higher rates that can be delivered to the receivers by the layered system do not necessarily lead to higher perceptual quality.

In this dissertation, we study the achievable average video quality in P2P-MPVC through layered coding and partitioned simulcast systems. We consider only the optimal rate allocation problem under node capacity constraints and assume rate-quality relations of the layered and single-layer coders. Design of a real system that must explicitly consider packet loss and delay, is beyond our scope in this particular

[1]However, layered coding offers additional error resilience against packet losses, which, when factored in, generally leads to bitrate gains.

study. Our main contributions are summarized as the following:

(1) We cast the problem of finding the optimal video flow configuration in P2P-MPVC as a tree packing problem in a complete graph *where the only bottlenecks are incoming and outgoing capacity limitations* on the nodes. We then show that, in this setting, any multicast distribution tree can be replaced by a collection of 1-hop and 2-hop trees, which substantially reduces the number of trees we have to consider in the tree construction for either layered or partitioned simulcast system. This result also reveals that, even when we use only 1-hop and 2-hop trees to limit the transmission delay, we can achieve the same video rates as when we can use any distribution trees.

(2) For layered coding, we develop a layer assignment heuristic, which determines, for each receiver, the layers received from each source. After describing the trees used to deliver each assigned layer, we solve for the optimal tree rates and consequently layer rates that maximize the average video quality among all users, and show that although the tree rates are not unique, the optimal layer rates are unique. Finally we develop an algorithm to refine the tree rates, to favor 1-hop trees to decrease potential delay and jitter.

(3) We study the optimal receiver partitioning problem, which determines the receiver partitions for all sources, and the rates of the single-layer videos distributed in each receiver group, to maximize the average video quality. Instead of performing an exhaustive search over all partitions, we propose a fast heuristic algorithm that finds the partitions for each source and determines the group rates for each source.

(4) We compare the performances of both systems through numerical simulations. In our simulations, the proposed layered system achieves the best video rates, whereas our partitioned simulcast heuristic can also achieve close-to-optimal video rates when the number of users is small (3 or 4). Finally, due to the bitrate overhead of the SVC encoder, the partitioned simulcast system outperforms the layered system in terms of the achieved video quality in both the 4-user and 6-user cases, even when the layered coding overhead is as low as 10%. Partitioned simulcast system performs similarly as the layered system at 10% rate overhead in the 6-user case. We note that, the proposed layered system formulation requires a scalable coder that can generate a successively refinable bitstream that can be divided into any number of layers at any rates.

1.2.2 Bandwidth Estimation & Video Adaptation for Video Calls over Cellular Networks

Despite their popularity in wired and Wi-Fi networks, video calling applications, such as [3], [4], and [18], have not found much use over cellular networks. The fundamental challenge of delivering real-time video over cellular networks is to simultaneously achieve high-bitrate and low-delay video transmission on highly volatile cellular links with rapidly-changing available bandwidth (ABW). On a cellular link, increasing the sending bitrate beyond what is made available by the PHY and MAC layers leads to self-congestion, and intolerable packet delays, hence frame delays. Excessively delayed frames will have to be treated as lost. On the other hand, a conservative sending bitrate clearly leads to the under-utilization of the cellular channel and consequently a lower video quality than what is possible.

The tight design space calls for a joint application and transport cross-layer design, involving real-time video encoding bitrate control, sending bitrate adjustment, and error control. Ideally, the transmitted video bitrate should closely track the ABW on the cellular links. However, the traditional *reactive congestion control algorithms* [19, 20], which adjust the sending bitrate based on packet loss and/or packet delay information, are too slow to adapt to the changes in the ABW, leading to either bandwidth under-utilization or significant packet delays [21]. It is more preferable to design *proactive congestion control algorithms* that calculate the sending bitrate based on cellular link ABW forecasts. Meanwhile, for video adaptation, video encoder can adjust various video encoding parameters, so that the resulting video bitrate matches the target sending bitrate determined by the congestion control algorithm. However, accurate rate control is very challenging for low-delay encoding, and significant rate mismatch is often present with the state-of-the-art video encoders. In addition, what makes the problem even more challenging is that the lost and late packets can render not only their corresponding frames, but also other frames that are predicted from their frames non-decodable at the receiver. The encoder and the transport layer should be designed to be error resilient so that lost and late packets have minimal impact on the decoded video.

In this dissertation, we propose a new real-time video delivery system, Rebera, designed for cellular networks, where we aim to maximize the sending rate of the video source and error resilience, while keeping the one-way frame delays sufficiently small. Our system consists of a proactive congestion controller, a temporal layered encoder, and a dynamic frame selection module. Our proactive congestion controller uses the video frames themselves to actively measure the current ABW in real-time, and then employs the well-known linear adaptive filtering methods [22] to predict the future capacity, based on the past and present capacity measurements. For basic error resilience, we resort to layered encoding, which enables unequal error protection (UEP). However spatial and quality layering incurs significant encoding complexity and coding efficiency loss, making them unattractive for practical deployment. Thus we consider only temporal layering, which provides a certain level of error resilience even without using explicit UEP. To minimize the delays for real-time delivery, we use hierarchical-P coding (hPP) structure [23] for temporal layering. To address the rate control inaccuracy of the encoder, we propose a dynamic frame selection algorithm for hierarchical-P, where the goal is to select in real-time which encoded frames to send, subject to a budget determined by the predicted capacity value. Our frame selection algorithm takes into account quality implications of the layers, decoding dependencies between the frames, and the smoothness of frame inter-arrivals to maximize the delivered video quality under the said budget. We have implemented the complete system, called "Rebera" for real-time bandwidth estimation and rate adaptation, to evaluate its performance and compare it with Apple's FaceTime video call application. Our implementation relies on the x264 real-time H.264 video encoder [24], which we have modified to produce a hierarchical-P stream. As a result, we can directly control the encoded video rate according to the measured capacity. Thanks to the combination of all these components we have mentioned, Rebera is able to achieve higher bandwidth utilization and lower frame delays than FaceTime. In this study, we do not consider UEP among the temporal layers and the error resilience aspect of the system.

1.2.3 Perceptual Quality Maximization and Packet Loss Resiliency through Joint Optimization of Video Encoding and FEC Parameters

Another fundamental challenge in real-time video delivery, aside from simultaneously achieving high-bitrate and low-delay transmission, is to provide resiliency against packet losses that degrade the perceptual quality of the video displayed at the receiver side. Traditionally, packet loss resiliency is provided through a combination of layered video coding, forward error correction (FEC), and automatic repeat request (ARQ) at the sender side, along with error concealment techniques at the receiver side. ARQ introduces an additional round-trip delay in case of lost packets, and is therefore unsuitable due to the stringent delay requirement of real-time video delivery. This requirement also restricts the video data, i.e., frame, group of pictures (GoP), or intra-period, on which block-code based FEC can be applied, since the decoding cannot be completed without receiving sufficient number of source packets. For video calls, applying FEC with each frame removes the additional FEC decoding delays, since each frame can be encoded and decoded individually. However, the efficiency of block FEC codes is reduced when the block length is small. This is typically the case if the encoder does not reduce the frame rate at low target bitrates, resulting in frames that have only limited number of packets.

As indicated, adjusting the encoding frame rate (eFR) based on the sending bitrate (SBR) enables efficient frame-level FEC. More broadly, choosing the temporal resolution (eFR), amplitude resolution (quantization stepsize), and the spatial resolution (picture size), denoted jointly by STAR in [25], under an SBR constraint, is another challenge closely coupled with FEC. Different STAR combinations, each of which achieves the same video bitrate, leads to different video qualities for different video contents. In [26], it was shown that significant quality gains can be achieved by picking the STAR that maximizes the perceptual quality at a given SBR. In the presence of packet losses, however, FEC provides loss resiliency at the cost of additional bits, which limits the spatial and amplitude resolutions that can be achieved.

For error concealment, a simple technique, called frame-copying [27], is prevalent in industry applications [28]. This technique freezes the last correctly decoded frame on screen in case of a damaged frame due to missing packets, until another frame that can be decoded without errors. This is because missing packets typically affect a large region on the frame, and any error propagates to the following frames over extended regions due to the use of motion-compensated temporal prediction. Due to frame-copying error concealment, the decoded frame rate (dFR) is generally lower than eFR, ultimately impacting the received video quality. Frame-copying may also cause severe video freezes if the underlying coding structure is IPP...I (IPP). The use of temporal layering achieved through hierarchical-P coding (hPP) [23], mitigates the effect of packet losses and frame freezes, with minimal additional complexity.

However, layered coding presents additional coding overhead compared to non-layered coding achieved through the more typical IPP...I coding (IPP).

Consequently, it is crucial to study the delicate interplay between the FEC, STAR, and different encoding methods for real-time video delivery. In this dissertation, we study the perceptual video quality maximization for video calls that are subject to varying bandwidths and packet losses, by jointly optimizing FEC, eFR and the quantization stepsize (QS), using either hPP or IPP coding. We consider a constant picture size to reduce the problem complexity. We assume that the congestion control module periodically predicts the end-to-end available bandwidth in the network, and that the video frame delays are minimized as long as the SBR does not exceed the available bandwidth [29] [21] [30]. Similar to the previous studies, the sender allocates a fraction of the SBR for encoding the video, while the remaining bitrate is used for FEC that protects the compressed video stream through redundancy. Block codes are applied on each individual frame, with potentially different code rates. At the receiver side, we assume that the frame-copying error concealment is used. We utilize the Q-STAR model in [25] to evaluate the decoded video quality, which represents the perceptual quality as a function of dFR and QS. When certain encoded frames are not decodable due to packet losses that cannot be recovered by FEC decoding, we simply approximate the decodable frame rate by the number of decodable frames per second. This is a reasonable assumption under the hPP structure and our unequal error protection strategy, to be detailed in Sections 4.2 and 4.3, which generally yields evenly spaced decoded frames. We also leverage the R-STAR model proposed in [31], which relates the video rate with the eFR and QS, in our formulation. With the rate model, the QS can be written as a function of the video rate and eFR. Therefore, we can parameterize the encoding parameters by eFR and video rate. We assume the total sending rate for each intra-period is given. For each intra period, we cast the problem of optimizing eFR, the video bitrate, and the FEC redundancy rate for each frame as a combinatorial optimization problem, and solve it through a combination of exhaustive search and hill-climbing methods. We also propose a greedy algorithm that solves for the suboptimal redundancy rate for each frame under a given total FEC bitrate constraint. Our evaluations show that, for independent and identically distributed (iid) packet losses with up to 20% loss, the optimal FEC redundancy ratio can be approximated with an affine function of the packet loss rate, and the quality drop compared with the lossless case is at most 35% even when the packet loss rate is 20%. Meanwhile, the bitrate range, in which low eFR is preferred, gets wider for higher packet loss rates, where the encoding frame rate of 15 Hz is selected over 30 Hz for SBR values up to 1.3 Mbps when the packet loss rate is 20%. We note that, IPP achieves higher Q-STAR scores, but is prone to abrupt freezing that is not considered by the Q-STAR model. Through simulations, we show that hPP provides significantly more regular frame intervals than IPP at small SBRs. For bursty packet losses, our evaluations indicate that the FEC redundancy ratios increase up to 80% for I-frames, when the average burst length is close to the average block length. In this case, hPP is able to deliver higher Q-STAR scores than IPP in a large SBR range, along with smaller mean and variance of decoded frame distances.

1.3 Dissertation Outline

The rest of the dissertation is organized as follows. In Chapter 2, we study the high-level design of P2P-MPVC systems, by presenting first a general formulation for the multicast flow configuration problem, and then narrowing down the formulations for P2P-MPVC, followed by the solutions for the layered distribution and the partitioned simulcast, and the numerical evaluations. In Chapter 3, we turn our attention to two-party video calling, where we study the congestion control problem in the context of cellular networks. We present a novel method to measure the current available bandwidth, which we use to estimate the future available bandwidth and adapt the sending bitrate at the sender side. We also introduce, for hierarchical-P coding structures, a method to decide in real time whether to send or discard an encoded frame, according to the budget, thereby preventing self-congestion and minimizing the packet delays. The chapter is concluded with extensive emulation evaluations comparing the proposed system with Apple's FaceTime, along with experiments performed in the wild. In Chapter 4, we optimize the video encoding and FEC parameters given the sending bitrate, so as to maximize the perceptual quality at the receiver side, in the presence of packet losses. We show that achieving packet loss resiliency and choosing video encoding parameters given the sending bitrate and the packet loss model parameters are closely connected, and present an optimization problem formulation that aims to maximize the Q-STAR metric introduced earlier. Using this formulation, we compare the performance of two prevalent video coding structures used throughout the industry. The dissertation is concluded in Chapter 5.

Chapter 2

P2P-MPVC: Layered Distribution vs. Partitioned Simulcast

This chapter is organized as follows. We briefly discuss the related work on P2P-MPVC in Section 2.1. General formulation for P2P-MPVC and our two-hop optimality result are presented in Section 2.2. The layered system design is presented in Section 2.3, where we discuss proposed solutions for layer assignment, optimal tree rate and layer rate allocation, and an algorithm to assign tree rates to favor 1-hop trees. In Section 2.4, we first study the optimal partitioning problem with non-layered video coding. We then present a fast heuristic partition algorithm that determines both the partition for each source and the video rate for each receiver group. Numerical simulation results comparing layered coding with partitioned simulcasting are reported in Section 2.5.

2.1 Prior Work

Video distribution in P2P networks have been extensively studied in the past, especially in the context of streaming in [32] [33] [34], among many others. For P2P-MPVC systems, the problem of optimal flow configuration has largely been explored under the assumption that only the uplink capacity of the peers present a bottleneck. The paper in [11] presents the optimality result for a special set of trees consisting of at most two hops, often called the Mutualcast trees in the literature, in a single-rate and single-source setting. In [12], this work is extended to include the multi-source case and distributed algorithms are presented to find optimal trees and video rates. The same authors then make a further extension in [13], where a multi-rate solution is considered with only layered video in uplink-constrained P2P networks without any downlink capacity constraints. The work in [14] investigates a resource sharing solution among different MPVC swarms leading to a higher multiplexing gain. In [15], the solution is once again extended to consider bandwidth and delay constraints on underlay links inside the network, where the distribution trees with excessive delays are not used, incurring some rate loss. Almost all of the prior studies assume P2P-MPVC systems are constrained by peer uplink only, and focus on video distribution design without considering the impact of scalable video coding overhead.

In contrast, we consider the problem of optimal flow configuration in a multi-source, multi-rate video conferencing scenario in a P2P network *with both uplink and downlink capacity constraints*, which is a more realistic characterization of today's networks. As a result, previous application layer flow design solutions are rendered either inapplicable or inefficient. We obtain new formulations regarding the dissemination of source videos with layered and single-layer coding, in order to consider the *impact of achievable rate-quality relation by practical video coders.*

2.2 Problem Statement and Multicast Trees

2.2.1 General Tree Packing Formulation for P2P-MPVC

We examine a P2P-MPVC scenario, where each user sends its own video to all other users through an overlay P2P network. The overlay is represented by a complete, directed graph $G = (N, E)$, where N is the set of users, E is the set of directed links, and $n \triangleq |N|$ is the number of users in the conference (Fig. 2.1). We assume that n is small, and as a result, each user can maintain $n - 1$ video connections with the rest of the peers[1] at any given time. We further assume that only the user uplink and downlink capacities create bottlenecks in the network, bounding the incoming and outgoing flows of a node. For user $i \in N$, uplink and downlink capacities are denoted by U_i and D_i, respectively. We also assume $D_1 \leq \cdots \leq D_n$ without loss of generality. Each user concurrently hosts an application layer multicast session to distribute its own video to all others. In other words, the receiver set R_i for node i is $N \setminus \{i\}$, where $\setminus$ is the set difference operator.

[1]We use the terms *users*, *nodes*, *peers* and *participants* interchangeably.

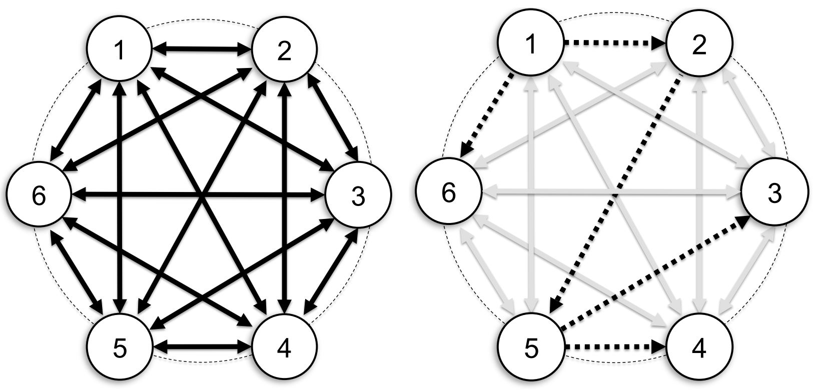

Figure 2.1: A P2P-MPVC overlay example with $n = 6$ (left), an example multicast tree T (dashed bold) with $s_T = 1$, $V_T = \{1, 2, 3, 4, 5, 6\}$ (right)

In P2P-MPVC, portions of user i's video stream are distributed via a number of directed multicast trees, which are rooted at user i and include, in general, a subset of its receiver set R_i. Different video packets from user i may be routed along different trees, where nodes on the trees replicate the packets and send them to their downstream nodes. Let us now denote the source node of a multicast tree t by s_t, its receiver set by V_t, and the packet flow rate along t by x_t. Then, the total communication rate r_{ij} from user i to user j equals the sum of the rates of trees that are rooted at node i and include node j,

$$r_{ij} = \sum_{t:\ \substack{s_t = i \\ j \in V_t}} x_t. \tag{2.1}$$

Therefore, for the set T of *all* multicast trees in G and an arbitrary concave utility function f that measures the video quality at rate r, a general application layer flow configuration problem can be formulated as a tree-packing problem

$$\max_{\substack{x_t \geq 0 \\ \forall t \in T}} \quad \sum_{i \in N} \sum_{j \in R_i} f\left(r_{ij}\right) \quad \text{subject to} \tag{2.2}$$

$$\sum_{t \in T \ :\ i \in V_t} k_{i,t}\ x_t \leq U_i, \quad \forall i \in N \tag{2.3a}$$

$$\sum_{\forall i \ :\ j \in R_i} r_{ij} \leq D_j, \quad \forall j \in N, \tag{2.3b}$$

where r_{ij}, given in Eq. (2.1), is the rate of node i's video that node j receives, and $k_{i,t}$ is the number of outgoing branches at node i on tree t. The constraints in Eqs. (2.3a) and (2.3b) are uplink and downlink capacity constraints for each node, respectively. For simplicity, we assume that the videos from all participants in a conference have similar characteristics and use $f(r)$ to represent the video quality-bitrate relation for each user. The downside of this tree-packing formulation is that, $|T|$, the number of trees that we need to consider, is very large.

Before proceeding to the next subsection, note that the following constraints always hold for any multicast tree set chosen in any P2P-MPVC system.

$$\sum_{(i,j)} r_{ij} \leq \sum_{m} U_m \tag{2.4a}$$

$$\sum_{i} r_{ij} \leq D_j \text{ and } \max_{j}(r_{ij}) \leq U_i, \quad \forall j \in N \tag{2.4b}$$

(2.4a) simply states that the total data delivery rate in the network cannot be larger than the total upload bandwidth in the network. On the other hand, (2.4b) indicates that, the total receiving rate of a node, or the highest data rate it can transmit, cannot be larger than its downlink and uplink capacities, respectively. The region defined by these constraints presents, in general, a loose upper bound on the achievable video rates.

2.2.2 Optimal Multicast Trees

A crucial design problem for P2P-MPVC is to determine which trees should be considered in Problem (2.2). It was shown in [12] that employing the two types of 1-hop and 2-hop trees introduced in [11] is sufficient to maximize the throughput and the utility in a P2P-MPVC scenario without helper nodes, under the assumption that the network is *only uplink-limited*. Hereafter, we call such trees *Mutualcast (MC) trees*. MC trees employed by user i consist of a single 1-hop tree that reaches all $j \in R_i$ and $|R_i|$ 2-hop trees, each passing through a particular $j \in R_i$ and then branching to the rest (Fig. 2.2). In such an uplink-limited setting, all receivers of a source receive the source video at the same rate.

However, to the best of our knowledge, there is no optimality result for any trees in an uplink- *and* downlink-limited network. At this point, we present the following

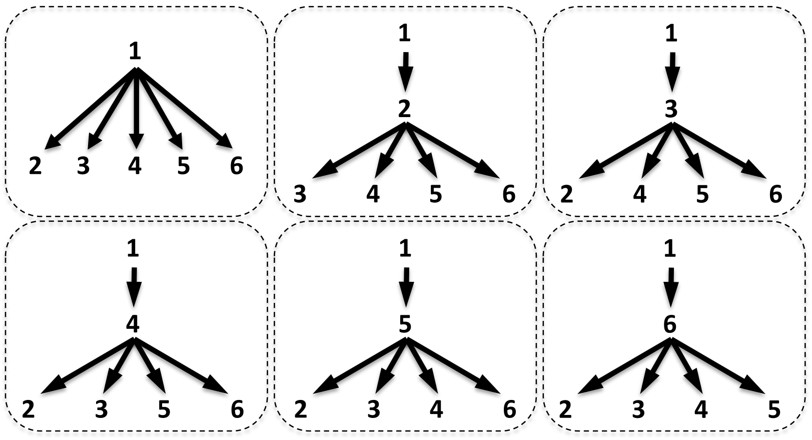

Figure 2.2: MC trees of user 1 to distribute its video to $R_1 = \{2,3,4,5,6\}$: 1 1-hop tree (upper left) and 5(=$|R_1|$) 2-hop trees.

theorem, which shows that any given tree with flow f can be replaced by MC trees covering the same node set as before.

Theorem 1 *In a directed, complete graph where the nodes have incoming and outgoing capacity constraints, a multicast tree t with root s, receiver set V with $|V| > 1$, and edge set E can be replaced by 1-hop and 2-hop MC trees that are rooted at s and include V, such that the aggregate download and upload rate of each node in all the MC trees are exactly the same as in the original tree t.*

Proof 1 *Let the flow along t be equal to x. We denote the number of outgoing branches at node j by k_j, therefore node j is a leaf iff $k_j = 0$. Clearly, the total outgoing flow of j is $k_j x$. Then, for each node $j \neq s$ with $k_j > 0$, build a 2-hop MC tree rooted at s and going through j with a flow of $k_j x/(|V|-1)$ and a 1-hop MC tree with a flow of $(k_s-1)x/(|V|-1)$. We are done if we can show that the total incoming and outgoing flows for each node is the same as before. Each node is now receiving*

$$r = \frac{x}{|V|-1}\left(k_s - 1 + \sum_{j\in V} k_j\right) \tag{2.5}$$

$$= \frac{x}{|V|-1}\left(k_s - 1 + |V| - k_s\right) \tag{2.6}$$

$$= x \tag{2.7}$$

(2.6) *follows because for a tree, we have $k_s + \sum_{j\in V} k_j = |E| = |V|$. In this new configuration, it is easy to see that the total outgoing and incoming flows of node j are still* *equal to $k_j x$ and x, respectively. Therefore, the same amounts of upload and download capacities are consumed and hence the equivalency with the old configuration.*

An illustration of the tree construction is given in Figure 2.3. As we can replace any tree with a combination of MC trees covering the same node set, we need merely consider MC trees covering different node sets. We emphasize this with a remark.

Remark 1 *In a directed, complete graph where the nodes have incoming and outgoing capacity constraints, any feasible flow configuration achieved by a tree can also be achieved by a combination of 1-hop and 2-hop MC trees that cover the same nodes as the original tree. Thus, to find an optimal set of multicast trees, it is sufficient to consider only MC trees. This means that we can bound the one-way overlay hop count by 2 without losing rate optimality.*

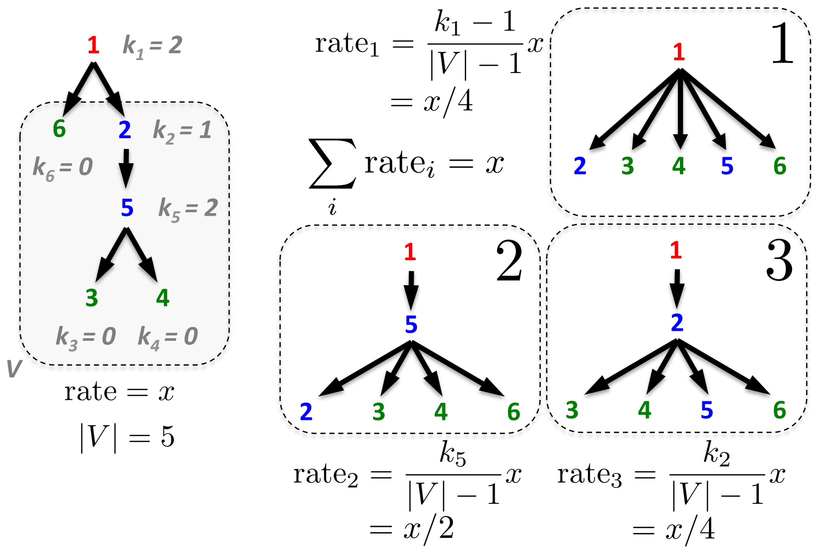
1
$k_1 = 2$
6
2
$k_2 = 1$
$k_6 = 0$
5
$k_5 = 2$
3
4
$k_3 = 0$
$k_4 = 0$
V
rate $= x$
$|V| = 5$
$\text{rate}_1 = \frac{k_1 - 1}{|V| - 1}x = x/4$
$\sum_i \text{rate}_i = x$
1
2 3 4 5 6
2
1
5
2 3 4 6
3
1
2
3 4 5 6
$\text{rate}_2 = \frac{k_5}{|V| - 1}x = x/2$
$\text{rate}_3 = \frac{k_2}{|V| - 1}x = x/4$

Figure 2.3: Any node's uplink/downlink capacity used in the multicast tree shown in the left diagram of Fig. 2.1 (shown on left here) is the same as the summation of the uplink/downlink capacities used in 1-hop (upper right) and 2-hop (lower right) MC trees. In both cases, each node in N receives a packet flow with rate x in total. Green nodes are leaves, blue nodes have at least one outgoing link, red node is the source.

2.3 Design of The Layered System

Layered video encoding [16] produces a bitstream that comprises a hierarchy of substreams, or layers, that allow the bitstream to be decoded partially, starting from the base layer and followed by the enhancement layers. Decoding more layers results in improved fidelity. Therefore, the users may receive the same video sequence at different rates and quality levels, depending on the layers received and decoded. With respect to single-layer coding, the cost of this flexibility of layered encoders is the *coding overhead*[2], which is the additional video rate needed to achieve the same quality as a non-layered coder.

In this section, we look into the problem of layered video distribution in P2P-MPVC with upload and download constraints. We first describe the multicast tree sets to be employed. Afterwards, we formulate the optimal flow configuration problem as a tree-packing problem with continuous rates, replacing $f(r)$ in (2.2) with the achievable video quality curve $Q_{LV}(r)$ of the layered coder. Note that, in practice, the $Q_{LV}(r)$ curve of the SVC encoder depends on the base layer rate, the number of layers and the desirable rates of each enhancement layer, which are all supposed to be found through the design of the layered system. To circumvent this problem, instead of generating a layered stream with a finite number of layers, we assume that a coder can generate a fine granularity scalable stream that can be truncated at any rate. We recognize that this is not feasible in practice, but analysis based on this idealistic assumption obtains performance upper bound for layered coding and provides important insight for our comparison study.

2.3.1 Layer Assignment Heuristic

According to Remark 1, we do not have to consider every multicast tree in G in our formulation. Instead, we can classify them according to their roots and the nodes they cover, since all trees with the same root and covering the same node set, can be replaced with the same collection of MC trees. Further, all nodes in a given tree receive the same video data at the same rate. Thus, for a layered video stream, the different node sets that shall be covered by the multicast trees, correspond to the receiver sets of different video layers. Denote the set of users that receive layer l of user i's video by $S_l^{(i)}$, usually referred to as *subscribers of the l^{th} layer* in the literature. Due to layered coding hierarchy, $S_l^{(i)}$ have to be nested, since the users must receive all the layers up to $l-1$ in order to decode the l^{th} layer. As a result, all receivers subscribe to the first (base) layer. Thus, for user i's video stream, we

[2]However, in terms of the total number of bits generated to obtain a multi-rate stream, layered encoding has a *coding gain*.

have $S_{L_i}^{(i)} \subset S_{L_i-1}^{(i)} \subset \cdots \subset S_1^{(i)} = R_i$, where L_i is the number of video layers that user i generates. We do not assume that L_i is given, nor that it is bounded by source capabilities or user preferences. Rather, L_i is only bounded by the number of receivers R_i and will be determined with the following subscriber determination heuristic. First, we have $S_1^{(i)} = R_i$. Then, we remove the node(s) with the smallest total download capacity. The remaining nodes make up $S_2^{(i)}$, i.e., they are receivers of layer 2. We proceed in this fashion until every node is removed. With this heuristic, the number of layers for source i equals $|R_i|$, iff all receivers have different downlink capacities.

2.3.2 Tree Construction for Layer Distribution

Next, for every video layer l in every user i's video stream, we build a set of MC trees to distribute that layer to its subscribers. Specifically, we construct a single 1-hop tree and $|S_l^{(i)}|$ 2-hop trees, all rooted at i. Each 2-hop tree passes through a particular subscriber $j \in S_l^{(i)}$ and branches to the rest of the subscribers, with a rate of x_{ijl}. Similarly, the rate of the 1-hop tree is denoted by x_{iil}. So, if z_{il} is the rate of layer l of user i's video, we have

$$z_{il} = x_{iil} + \sum\nolimits_{j \in S_l^{(i)}} x_{ijl}. \tag{2.8}$$

Eq. (2.8) indicates that, layer l of user i's video is distributed to the subscribers not via a single tree, but $|S_l^{(i)}|+1$ trees in general, allowing the users to share their upload bandwidth with others, thereby maximizing the network utility in the optimal flow configuration. Finally, the upload bandwidth b_{il} that user i requires to drive its own layer l distribution trees is

$$\begin{aligned} b_{il} &= |S_l^{(i)}| x_{iil} + \sum\nolimits_{j \in S_l^{(i)}} x_{ijl} && (2.9) \\ &= \left(|S_l^{(i)}| - 1\right) x_{iil} + z_{il}. && (2.10) \end{aligned}$$

The terms on the right hand side of Eq. (2.9) is the upload capacity consumed by driving the 1-hop and 2-hop trees. Eq. (2.10) follows simply from Eq. (2.8) and Eq. (2.9). It is noteworthy that the first term on the right hand side of (2.10) is the allocated bandwidth for the 1-hop MC tree for this layer, which would be zero in a server-centric MPVC system.

2.3.3 Determining the Optimal Layer Rates

Now that the layers are assigned, and the corresponding MC trees that we shall use are determined for each layer and source, we are finally ready to formulate the multi-source, multi-rate flow optimization problem for layered videos, analogous to (2.2)-(2.3), where the optimization is performed over all tree rates $\{x_{ijl}\}$:

$$\max_{x_{ijl} \geq 0} \quad \sum_{i \in N} \sum_{j \in R_i} Q_{LV}\left(r_{ij}\right) \quad \text{subject to} \tag{2.11}$$

$$\sum_{l=1}^{L_i} b_{il} + \sum_{j \neq i} \sum_{l:\ i \in S_l^{(j)}} \left(|S_l^{(j)}| - 1\right) x_{jil} \leq U_i \tag{2.12a}$$

$$\sum_{j \neq i} r_{ji} \leq D_i, \quad \forall i \in N \tag{2.12b}$$

$$r_{ij} = \sum_{l:\ j \in S_l^{(i)}} z_{il}, \quad \forall (i,j), i \neq j \tag{2.12c}$$

The objective function Q_{LV} in (2.11) is a non-decreasing, concave function of r_{ij}. (2.12a) and (2.12b) are uplink and downlink capacity constraints similar to (2.3a) and (2.3b), whereas (2.12c) expresses the video rate r_{ij} as the summation of layer rates received by user j, similar to (2.1). Particularly, (2.12a) follows since a video source can allocate part of its upload bandwidth to relay its own video layers and part of it for helping the other sources for which itself is a receiver. The feasible region defined by inequalities (2.12a)-(2.12c), (2.8) and (2.10) is a convex polytope in tree variables. Note that, one can show that the achievable video rate region is also convex by introducing r_{ij} in the inequalities using Eq. (2.1), and projecting the polytope onto the $\{r_{ij}\}$ space, where the projection must also be convex as the projection operation preserves convexity. As a result, the optimization problem in (2.11)-(2.12) is a non-strictly concave optimization problem in the tree rate variables and the maximizer tree rates are not unique. However, if there exists an interval I such that Q_{LV} is strictly concave in I and the optimal video rates $\{r^*_{ij}\}$ lie in I, then they are unique; hence, the layer rates $\{z_{ik}\}$ are also unique. Centralized solution techniques for such concave optimization problems have been well-understood [35] and thus, any one of these solution methods can be employed to find a solution.

2.3.4 Refining the Multicast Trees

After solving for one set of optimal tree rates in (2.11)-(2.12) that achieves the optimal video rates $\{r^*_{ij}\}$ maximizing the average quality, we determine another set of tree rates $\{x^*_{ijk}\}$ that also achieves $\{r^*_{ij}\}$ and favors 1-hop trees instead of 2-hop trees. This is simply because packets distributed via 1-hop trees suffer less end-to-end delay. Further, the number of trees considered is $O(n^3)$ in the worst case; however, employing a large number of multicast trees could lead to increased jitter in a practical implementation. In Algorithm 1, we present a method to find tree rates that favor 1-hop trees over 2-hop trees, while satisfying the constraints, and achieving the given video rates $\{r_{ij}\}$. The algorithm terminates if the given video rates are not feasible. Note that this algorithm is meant to be used only after the optimal video rates are known by solving (2.11)-(2.12). The main idea behind this algorithm is to maximize the rates of 1-hop trees greedily, starting from the base layer. For each peer i, given the layer rates $\{z_{ik}\}$, we calculate the maximum rate that a 1-hop tree can handle subject to

$$\sum_{k=1}^{L_i} b_{ik} = \sum_{k=1}^{L_i} \left[\left(\left| S_k^{(i)} \right| - 1 \right) x_{iik} + z_{ik} \right] \leq U_i. \tag{2.13}$$

After this step, there exists at least one peer that still has excess upload capacity, otherwise the given video rates are infeasible, as we would have $\sum_{(i,j)} r_{ij} > \sum_{i \in N} U_i$. Then, in order to fill the remaining rate gaps, 2-hop trees are constructed that pass through the peers with excess upload capacities. Rates of 2-hop trees are proportional to the upload capacities of the peers that they pass through.

2.4 Design of The Partitioned Simulcast System

Non-layered encoding [17] produces a bitstream that does not have a layer hierarchy. Although layered coding allows us to generate a flexible stream that offers variable qualities depending on the rate, the cost of such flexibility is an increased bitrate to achieve a certain quality. Coding overhead of layered encoding motivates the use of non-layered video in MPVC systems.

Clearly, multicasting the same non-layered video to all receivers is suboptimal, starving the receivers with higher download capacities. In order to obtain a multi-rate solution, one method is *simulcasting*: user i can simultaneously encode its video

Algorithm 1 Determination of the MC tree rates

```
1:  for all $n \in N$ do                                          ▷ Begin 1-hop tree rates
2:      for $l = 1 \to L_n$ do
3:          if $U_n - |S_l^{(n)}| z_{nl} \geq \sum_{k=l+1}^{L_n} z_{nk}$ then $x_{nnl} = z_{nl}$
4:          else
5:              $x_{nnl} = \frac{U_n - \sum_{k=l}^{L_n} z_{nk}}{|S_l^{(n)}| - 1}$
6:              if $x_{nnl} = 0$ then
7:                  $x_{ink} = 0$ for all $k > l$ and break
8:              end if
9:          end if
10:         $U_n \leftarrow U_n - |S_l^{(n)}| x_{nnl}$
11:     end for
12: end for                                                       ▷ 1-hop tree rates determined
13: for all $n \in N$ do                                          ▷ Begin 2-hop tree rates
14:     for all $l : x_{nnl} \neq z_{nl}$ do
15:         $x_{nml} = \frac{U_m}{\sum_{k \in N} U_k} (z_{nl} - x_{nnl})$
16:     end for
17: end for                                                       ▷ 2-hop tree rates determined
```

at $|R_i|$ different bitrates, and send the generated streams to each receiver, matching their download capacities. The drawback of this method in terms of bandwidth is that the source may not have sufficient upload capacity to send out many different streams in the first place. Instead of generating a single-layer bitstream for each receiver, we propose partitioning the receivers into a smaller number of groups, where the source generates a bitstream for each group, separately. Within each group, the receivers can then share their video.

2.4.1 Optimal Group Rates for Given Partition Collection

Given a user i, let us partition its receiver set R_i into K_i distinct sets, or *groups*, such that the group with index k is denoted by $G_k^{(i)}$. By definition, $G_k^{(i)} \cap G_{k'}^{(i)} = \emptyset$ for $k \neq k'$. Also, we use $P_i = \{G_k^{(i)}, 1 \leq k \leq K_i\}$ to describe the partition itself, where $\bigcup_k G_k^{(i)} = R_i$. The idea is that, each user j in a given group $G_k^{(i)}$ receives user i's video at the same *group rate* $g_k^{(i)} = r_{ij}$, but the users in different groups may have different rates. Hence, the users with higher download capacities can receive more, resulting in a higher average video quality. Given a partition collection $\boldsymbol{P} = \{P_i : i \in N\}$, we have K_i single-rate multicast problems for each source i, for which it was shown in [11] that 1-hop and 2-hop MC trees covering the nodes in $G_k^{(i)}$ are *optimal.* Then, our problem is to maximize the average video quality by optimizing the amount of uplink capacity of each user $j \in G_k^{(i)}$ to video session i. Once this allocation is done, we can find the rates of the said 1-hop and 2-hop MC trees as shown in [11]. We can now formulate the multi-source, multi-rate video quality maximization problem with receiver partitioning using non-layered coding as,

$$\max_{u_{ij}, b_k^{(i)}, g_k^{(i)} \geq 0} \quad \sum_{i \in N} \sum_{k=1}^{K_i} |G_k^{(i)}| Q_{NL}(g_k^{(i)}) \quad \text{subject to} \tag{2.14}$$

$$g_k^{(i)} \leq b_k^{(i)}, \quad \forall i \in N, \ 1 \leq k \leq K_i \tag{2.15a}$$

$$|G_k^{(i)}| g_k^{(i)} \leq b_k^{(i)} + \sum_{j \in G_k^{(i)}} u_{ji}, \quad \forall i \in N, \forall k \tag{2.15b}$$

$$\sum_{k=1}^{K_i} b_k^{(i)} + \sum_{j \neq i} u_{ij} \leq U_i, \quad \forall i \in N \tag{2.15c}$$

$$\sum_{j:\ i \in G_k^{(j)}} g_k^{(j)} \leq D_i, \quad \forall i \in N. \tag{2.15d}$$

Again, the objective function Q_{NL} in (2.14) is a non-decreasing, concave function of the video rate. Here, $b_k^{(i)}$ and u_{ij} denote the portions of the uplink capacity of user i that is allocated to $G_k^{(i)}$, for which user i is the source; and to $G_k^{(j)}$, for which $i \in G_k^{(j)}$, respectively. (2.15a) follows since the group rate cannot be higher than the allocated source upload capacity, whereas (2.15b) follows directly from [11]. Evidently, (2.15c) and (2.15d) are the uplink and downlink capacity constraints. Once again the feasible region defined by (2.15) is convex. Hence, the formulated problem above is a non-strictly concave optimization problem with linear constraints. Similar to (2.11), it has a unique solution in the group rates $g_k^{(i)*}$, assuming the optimal solution lies where Q_{NL} is strictly concave, whereas u_{ij}^* and $b_k^{(i)*}$ are not unique. The number of variables, which depends on $\boldsymbol{P}$, is $O(n^2)$ in the worst case. For any optimal $\{u_{ij}^*, b_k^{(i)*}\}$, the MC tree rates are determined as

$$x_{ij}^* = \frac{1}{|G_k^{(i)}| - 1} u_{ji}^*, \quad x_{ii}^* = \frac{1}{|G_k^{(i)}|}(b_k^{(i)*} - \sum_{j \in G_k^{(i)}} x_{ij}^*) \tag{2.16}$$

where x_{ij} and x_{ii} are the rates of the 2-hop multicast tree rooted at node i and passing through node $j \in G_k^{(i)}$ and the 1-hop tree rooted at node i and branching out to all receivers in the group.

The difficulty with the optimal partitioned simulcast systems in P2P-MPVC is that we do not readily know the optimal $\boldsymbol{P}^*$. For n users, the number of different $\boldsymbol{P}$'s we can have is given by $(B_{n-1})^n$, where B_m is the m^{th} *Bell number*, equal to the number of ways we can partition a set with m elements. Therefore, exhaustively searching among all possible partition collections is hopeless even for a small number of users[3]. In order to overcome this difficulty, we now propose a simple heuristic algorithm to find a suitable collection of receiver partitions, as well as the group rates that can be achieved.

2.4.2 Heuristic Algorithm for Partitioning

The main idea behind the heuristic is to look for the best partition P_i^* in each video session i by assuming that the values of the uplink capacity allocation variables $\{u_{ij}^*, b_k^{(i)*}\}$ in the optimal partition collection can be approximated through the solution of a much simpler problem. We start our analysis by considering the video session hosted by user i. If we were given the optimal values $\{u_{ji}^*, b_k^{(i)*}\}$ of the upload capacities allocated by user j to video session i for each $j \in R_i$, along with the optimal value $b_k^{(i)*}$ of the upload capacity allocated by the user i to drive its receiver group k, finding P_i^* would be possible with a search in the set of all partitions of R_i, only. In other words, for each partition candidate P_i, user i would have K_i single-source, single-rate multicast sessions, where the group rate, or the multicast capacity for group k is known from [11] to be equal to

$$g_k^{(i)*} = \min(b_k^{(i)*}, \frac{b_k^{(i)*} + \sum_{j \in G_k^{(i)}} u_{ji}^*}{|G_k^{(i)}|}), \qquad (2.17)$$

leading to an average video quality of $\sum_k |G_k^{(i)}| Q_{NL}(g_k^{(i)*})$. However, since we have no such information, we approximate u_{ji}^* through the solution of

$$\max_{r_{ij} \geq 0} \quad \sum_{i \in N} \sum_{j \neq i} Q_{NL}(r_{ij}), \qquad (2.18)$$

[3]For $n = 5$, 759375 choices, for $n = 6$, 1.97×10^{10} choices

subject to the constraints in (2.4). The solution of (2.18)-(2.4) gives a loose quality *upper bound* for any non-layered video distribution scheme under the given upload and download constraints, as it does not consider whether or not the video rates can be achieved by any distribution tree. Note that the maximizer $\{r_{ij}\}$, which we denote by $\{r_{ij}^{\mathbf{B}}\}$, can be easily found with a water-filling algorithm: each source sends out equal flows to each receiver, while gradually increasing the flows at the same pace, until either all peers are downlink-saturated or there is no more upload bandwidth.

Once we obtain $\{r_{ij}^{\mathbf{B}}\}$, we can express the total upload bandwidth consumed for multicasting user i's video as $M_i \triangleq \sum_{j \in R_i} r_{ij}^{\mathbf{B}}$. Note that the benefit of using 2-hop multicast trees is that the user i can still sustain a video session even if $U_i < M_i$, by exploiting other users with abundant upload bandwidths. In this case, the additional bandwidth provided to user i's video session would simply be $M_i - U_i$. On the other hand, if $M_i < U_i$, then $U_i - M_i$ would be the additional bandwidth provided by user i to other video sessions.

We can see that the difference between M_i and U_i indicates whether user i requires or provides additional bandwidth in the optimal solution of (2.18)-(2.4). Hereafter, we classify the users as *ϵ-peers* and *α-peers*, depending on whether they require or provide the additional bandwidth. Note that if there exists an ϵ-peer, there must also exist an α-peer, otherwise we would have $\sum_{i \in N} U_i < \sum_{(i,j)} r_{ij}^{\mathbf{B}}$.

Let us now go back to approximating u_{ji}^*. We should first note that u_{ji}^* should be non-zero only if $i = j$ or if j is an α-peer and i is an ϵ-peer, since an ϵ-peer cannot provide additional bandwidth, while an α-peer does not require it. Then, for such an (α, ϵ) user pair, we approximate u_{ji}^* by $\widetilde{u^*}_{ji}$, assuming that the α-peer j provides additional bandwidth to ϵ-peer i in proportion to the total bandwidth it can provide:

$$\widetilde{u^*}_{ji} = \begin{cases} \min(U_j, M_j), & \text{if } j = i \\ (M_i - U_i)\frac{U_j - M_j}{\sum\limits_{k:\alpha} U_k - M_k}, & \text{if } j : \alpha \text{ and } i : \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{2.19}$$

Note that we have $u_{ii} \triangleq \sum_k b_k^{(i)}$, and that $b_k^{(i)}$ is still undetermined. The way we finalize our heuristic is to approximate the optimal downlink capacity $\widetilde{d}_{ji}^*$ allocated by user j to video session i, so that we can solve for the optimal $b_k^{(i)}$ and $g_k^{(i)}$ given

$\widetilde{u^*}_{ji}$ and $\widetilde{d^*_{ji}}$. This final approximation is given as

$$\widetilde{d^*_{ji}} = \frac{D_j}{n-1}. \tag{2.20}$$

Finally, we can search for the best receiver partition at each session, where we consider only the *ordered partitions* with respect to the downlink capacities. Here, a receiver partition $P_i = \{G_k^{(i)}, 1 \leq k \leq K_i\}$ is ordered with respect to the downlink capacities if we have $D_m \leq D_{m'}$ for all $m \in G_k^{(i)}$, $m' \in G_{k'}^{(i)}$ and $k < k'$. Each user i starts the search with the partition $P_i = \{R_i\}$ that includes all the receivers, and employs steepest-ascent search to find a local maximum by examining, at each step, all ordered partitions containing one more group. To evaluate an examined P_i, peer i solves for the maximum average session quality that can be achieved given the allocated upload and download capacities $\{\widetilde{u^*}_{ji}, \widetilde{d^*}_{ji}\}$ and the receiver partition P_i, by the following optimization problem.

$$\max_{g_k^{(i)}, b_k^{(i)} \geq 0} \quad \mathbb{Q}(P_i) = \sum_{k=1}^{K_i} |G_k^{(i)}| Q_{NL}(g_k^{(i)}) \quad \text{subject to} \tag{2.21}$$

$$g_k^{(i)} \leq b_k^{(i)}, \quad 1 \leq k \leq K_i \tag{2.22a}$$

$$|G_k^{(i)}| g_k^{(i)} \leq \begin{cases} b_k^{(i)} + \sum\limits_{j \in G_k^{(i)}} \widetilde{u^*}_{ji} & \text{if } |G_k^{(i)}| > 1 \\ b_k^{(i)} & \text{if } |G_k^{(i)}| = 1 \end{cases} \tag{2.22b}$$

$$\sum_{k=1}^{K_i} b_k^{(i)} \leq \widetilde{u^*}_{ii} \tag{2.22c}$$

$$g_k^{(i)} \leq \min_{j \in G_k^{(i)}} (\widetilde{d^*}_{ji}) \tag{2.22d}$$

Here, (2.22a) and (2.22b) are due to [11], while (2.22c) follows from the upload bandwidth allocated by user i to its own session. (2.22d) states that the group rate cannot be larger than the minimum allocated downlink capacity to the group. After examining each neighbor, we pick the the one with the highest average session quality $\mathbb{Q}(P_i)$ as the new local maximum candidate. The algorithm stops when there is no neighboring partition that yields a higher average session quality. Finally, the optimal tree rates can be found similar to (2.16).

Algorithm 2 Receiver partitioning heuristic (PH)

```
1: Find the solution $\{r^*_{ij}\}$ of (2.18)-(2.4)                ▷ Initialization
2: Calculate $M_i - U_i$ for each $i \in N$
3: Approximate $\{u^*_{ij}, d^*_{ij}\}$ through $\{\widetilde{u^*}_{ij}, \widetilde{d^*}_{ij}\}$
4: for all $i \in N$ do                                            ▷ Distributed phase
5:     $P_i^{(best)} \leftarrow \{R_i\}$, $\mathbb{Q}_i^{(best)} \leftarrow \mathbb{Q}(\{R_i\})$
6:     repeat
7:         $P_i^{(current)} \leftarrow P_i^{(best)}$, $\mathbb{Q}_i^{(current)} \leftarrow \mathbb{Q}_i^{(best)}$
8:         for all $P_i \in$ neighbors($P_i^{(current)}$) do
9:             Find the solution of (2.21)-(2.22)
10:            if $\mathbb{Q}(P_i) > \mathbb{Q}(P_i^{(current)})$ then
11:                $P_i^{(best)} \leftarrow P_i$, $\mathbb{Q}_i^{(best)} \leftarrow \mathbb{Q}(P_i)$
12:            end if
13:        end for
14:    until $P_i^{(current)} = P_i^{(best)}$
15: end for
```

2.5 Simulation Results

In this section, we numerically evaluate the capacity regions achievable through the layered and partitioned simulcast systems, respectively. In our simulations, we consider the layered video distribution scheme, given as the solution of (2.11)-(2.12) in Section 2.3, the partitioned simulcast system using optimal receiver partitioning, found by exhaustive search among all possible partition collections maximizing the solution of (2.14)-(2.15) in Section 2.4.1, the partitioned simulcast system using the fast heuristic algorithm presented in Algorithm 2 in Section 2.4.2, and the theoretical upper bound for both rate and quality, given as the solution of (2.18)-(2.4), along with the simulcast and the single-rate multicast schemes. Note that, simulcast is a special case of partitioned simulcast, where each receiver constitutes a group by itself. On the other hand, for multi-source single-rate multicasting, we consider the solution of [12] with the addition of the downlink constraints.

In our simulations, we first assume that the ratio of a peer's download capacity to its upload capacity is the same for all peers in the video conference, that is, $w = D_i/U_i$, for all $i \in N$. We will show results obtained with different w ratios, showing the performance trend as the system becomes less limited by the downlink capacity. In all our simulations, we assume the network is static within the time needed to perform the rate optimization.

2.5.1 Assumptions about Video Coding

The proposed layered system requires a layered coder that can generate a successively refinable bitstream that covers a large rate range and that can be divided into any number of layers at any desired rate determined by the rate allocation algorithm for the layered system. Unfortunately, with the current state of the art in scalable coding, no practical layered coders can accomplish this efficiently. For our numerical simulation of the layered system, we use the JSVM software for SVC encoding and operate it with a configuration that generates many thin layers with a very low base layer rate and a very high maximum rate, so that any desirable rate determined by the optimal rate allocation algorithm is within the range between the base layer rate and the maximum rate. Specifically, we use the combined spatio-temporal scalability option with 5 temporal layers (at frame rates 1.875, 3.75, 7.5, 15, and 30 Hz), 3 spatial layers (at frame sizes QCIF, CIF, and 4CIF), and 4 quantization levels (at QP values 40, 36, 32, 28). The layers are ordered to achieve the maximum quality improvement for each additional layer [26] based on the quality and rate models described in [25] and [31], respectively. We further assume that any rate between two successive discrete rates associated with two successive layers is achievable, and generate a continuous $Q_{LV}(r)$ curve by interpolating the achievable (Q, r) pairs. In practice, such rate points can be possibly realized by taking partial bits from the higher layer.

For the simulation of the partitioned simulcast system, we use the JM reference software for AVC [36]. For a given rate, the optimal spatial, temporal and amplitude resolutions are chosen to maximize the perceptual quality also using the methodology described in [26].

As shown in [26], the subjective quality vs. rate relation of both AVC and SVC coders can be modeled by

$$Q(r) = \frac{1 - e^{-\kappa\left(\frac{r}{r_{max}}\right)^{0.55}}}{1 - e^{-\kappa}}, \tag{2.23}$$

where r is the received video rate, r_{max} is the video rate at the highest spatial, temporal, and amplitude resolutions considered, and κ is a parameter that depends on the video characteristics and layer configuration. In video conferences, users' video sequences are likely to have similar features, therefore we associate the same $Q(r)$ function with each user. As an example, we take the "Crew" video sequence, which shows people with body motions that somewhat resemble typical motions in a conference session. The corresponding $Q(r)$ curves for both AVC and SVC encodings using the above configuration can be seen in Figure 2.4a, where the quality-rate model has the following parameters: $\kappa_{\text{SVC}} = 3.121$, $\kappa_{\text{AVC}} = 3.4$, $r_{max}^{\text{AVC}} = 2969$ Kbps and $r_{max}^{\text{SVC}} = 3515$ Kbps. As shown on Fig. 2.4b, the layered coder, denoted by "SVC 32% OH" incurs up to 35% rate overhead, with an average overhead of 32% using the BD-rate gain methodology of [37]. We should note that, generally, the layered coder incurs more coding overhead when the base layer rate is low and the number of layers is high.

When the bandwidth heterogeneity is relatively low so that the target video rate range is relatively small, it is possible to use a small number of appropriately chosen layers to generate an SVC stream with significantly lower redundancy. For example, when using only 2 spatial (CIF and 4CIF) and 5 temporal layers (1.875, 3.75, 7.5, 15 and 30 Hz) and not using amplitude scalability with a fixed QP=28, we found that the resulting SVC stream has less than 5% coding overhead with an achievable rate range of 120 Kbps - 2.25 Mbps. Finally, temporal scalability does not incur rate overhead (as the AVC coder also applies the same hierarchical temporal prediction structure in our simulations), but it only provides a limited number of rate points.

In order to evaluate the impact of the rate overhead of layered coding on the system performance, we also generate two hypothetical quality-rate functions with $\kappa_{\text{HYP},20} = 3.3$, $r_{max}^{\text{HYP},20} = 3450$ Kbps and $\kappa_{\text{HYP},10} = 3.35$, $r_{max}^{\text{HYP},10} = 3200$ Kbps, which lead to average-maximum rate overhead pairs of (20.7%, 21.9%) and (9.9%, 10.4%), respectively (Fig. 2.4). To the best of our knowledge, 20% bitrate overhead is an optimistic target in HD scalable video coding that is accepted by the industry, especially with more than 2 layers and over a large rate range. In the past, an overhead of 10% was reported for an SVC encoder with two spatial/quality layers, obtained by a complicated joint mode decision algorithm across layers [38]. With more layers, larger coding overhead is expected. Therefore, 10% overhead for fine granularity scalable coding is even a more optimistic assumption.

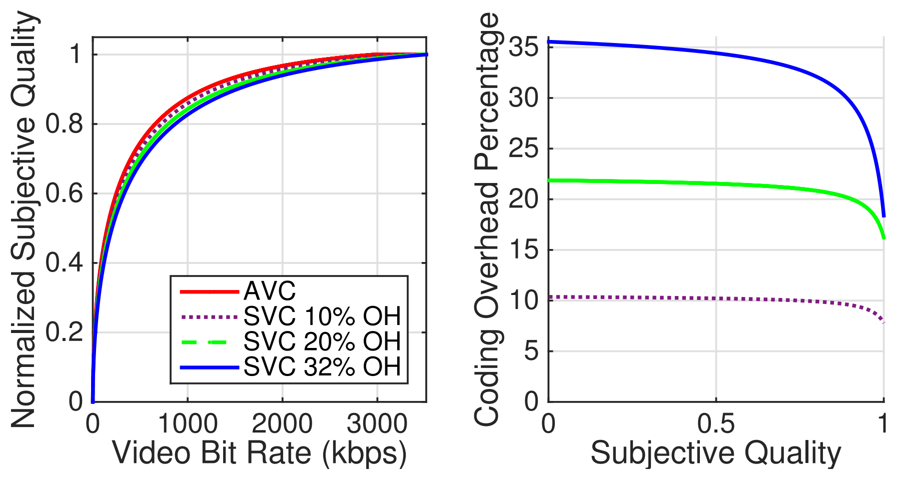

Figure 2.4: Quality vs. rate curves of the Crew sequence encoded according to AVC and SVC standards (left), additional bitrate (percentage) required by SVC to achieve the same subjective quality as AVC (right).

An alternative way to generate a fine-granularity scalable bit stream is using MPEG4-FGS [39]. However, MPEG4-FGS can provide fine granularity only in amplitude resolution for a fixed spatial and temporal resolution. Therefore, it can provide fine granularity scalability only over a small rate range. Furthermore, it has a much higher rate overhead than SVC.

2.5.2 Simulations

4-user Video Conferencing Simulations

We first examine an MPVC scenario with $n = 4$ users under different bandwidth heterogeneity conditions, focusing on the impact of the heterogeneity on the average video rates and qualities achieved in the system. Since the size of the conference is small, we are able to include the result of the optimal receiver partitioning scheme (obtained by exhaustive search) in our comparison. The results on the top and bottom of Fig. 2.5 are obtained for relatively more ($\{0.5, 2, 4, 5.5\}$ Mbps) and less heterogeneous ($\{1.5, 3, 3, 4.5\}$ Mbps) user upload capacities, respectively. In all figures, the horizontal axis is the download/upload ratio, w, which characterizes the degree of asymmetry between the upload and downlink bandwidths, and the averaging is performed over all source-receiver pairs in the video conference. Our findings are as follows.

(a) The average video quality and rate both increase with the downlink capacities under both heterogeneity conditions. As the downlinks cease to be the bandwidth bottleneck, our solutions converge to the uplink-limited-only solution in [12]. Particularly, when w is large, the video rates achieved by the layered system and the partitioning heuristic are equal. In this case, the enhancement layer trees are assigned zero rate and the video from each source is transmitted at a single rate.

(b) Due to the coding overhead, even the layered system with only 10% overhead achieves lower average video quality than partitioned simulcast, despite the fact that it delivers the highest video rates. We also consider layered video distribution without any coding overhead, which results in the best quality and rates.

(c) The optimal receiver partitioning with non-layered video distribution achieves almost the same rate and hence, almost the same quality, as the layered video distribution without coding overhead. *This result shows that, in MPVC systems where the downlinks and uplinks may both present bottlenecks, we can obtain a multi-rate solution by using optimal receiver partitioning and non-layered video without any significant performance loss in terms of the average video quality, compared even with an ideal layered video distribution scheme with no overhead.*

(d) The proposed heuristic partition algorithm yields an average video quality and rate that are very close to those obtained by optimal partitioning.

(e) Even when we do not allow more than 2 receiver groups per source, our heuristic still achieves a better received average video quality than the layered video distribution with 20% overhead, and is still competitive with the layered video distribution with 10% overhead in terms of the achieved quality for $w \leq 1.3$. For higher w values, our heuristic with at most 2 receiver groups per source outperforms the layered distribution with only 10% overhead, as well.

(f) Simulcast and single-rate distribution schemes perform poorly in face of higher heterogeneity. For simulcast, this is because there is no bandwidth sharing between the users. For the single-rate scheme, all peers, except for the one with the minimum download capacity, are starved. As expected, as the users become more homogeneous, the performance of the single-rate and simulcast schemes become more competitive.

(g) Comparing the top and bottom figures, we can see that for any w, the achieved quality and rate are both at least as high or higher for less user bandwidth heterogeneity, regardless of the distribution scheme, which underlines the difficulty caused by bandwidth heterogeneity.

To further analyze the performances of the layered video distribution (LV), non-

$\boldsymbol{r}_{\mathrm{LV}}$	1	2	3	4
1	0	500	500	500
2	167	0	1750	2000
3	167	750	0	3000
4	167	750	1750	0

$\boldsymbol{r}_{\mathrm{PH}}$	1	2	3	4
1	0	500	500	500
2	167	0	1333	1333
3	167	746	0	2994
4	167	752	2149	0

$\boldsymbol{r}_{\mathrm{SC}}$	1	2	3	4
1	0	147	176	176
2	147	0	926	927
3	176	927	0	2832
4	176	926	2832	0

$\boldsymbol{r}_{\mathrm{SR}}$	1	2	3	4
1	0	500	500	500
2	167	0	167	167
3	167	167	0	167
4	167	167	167	0

Table 2.1: Video rates for the layered video distribution (LV), non-layered video distribution through the partitioning heuristic (PH), simulcast (SC) and single-rate (SR) schemes, with $w = 1$. Rows and columns represent source and receiver indices, respectively, all in Kbps.

layered video distribution through the partitioning heuristic (PH), simulcast (SC) and single-rate (SR) schemes, we present the video rates achieved by each scheme in Tables 2.1 and 2.2, where the results for the more heterogeneous scenario are examined for $w = 1$ and $w = 2$, respectively. For $w = 2$, we can see that, although the average received rate per user are all 1 Mbps except for the single-rate solution (see upper right plot in Fig. 2.5), the achieved qualities are quiet different. Also, for the partitioning heuristic, the α-peers are users 3 and 4, which only use 1-hop trees (equivalent to simulcasting), while the ϵ-peers are users 1 and 2, which have 1 and 2 receiver groups, respectively.

6-user Video Conferencing Simulations

Next, we investigate a set of larger video conferences with $n = 6$ peers. As a result, we do not consider the optimal partitioning scheme due to its complexity. We assume that the end-users can be categorized into 4 different user classes with respect to their upload bandwidth. The considered upload capacities for these different classes are 0.5, 1, 5 and 10 Mbps, representing a typical range of uplink capacities for average WiFi, mid- and high-speed DSL and cable services [40]. We randomly pick users out

r_{LV}	1	2	3	4
1	0	500	500	500
2	333	0	1583	1583
3	333	1583	0	1583
4	333	1583	1587	0

r_{PH}	1	2	3	4
1	0	500	500	500
2	333	0	1580	1580
3	333	1333	0	1836
4	333	1333	1837	0

r_{SC}	1	2	3	4
1	0	166	167	167
2	277	0	862	862
3	344	1779	0	1877
4	379	2055	2986	0

r_{SR}	1	2	3	4
1	0	500	500	500
2	333	0	333	333
3	333	333	0	333
4	333	333	333	0

Table 2.2: Video rates for the layered video distribution (LV), non-layered video distribution through the partitioning heuristic (PH), simulcast (SC) and single-rate (SR) schemes, with $w = 2$. Rows and columns represent source and receiver indices, respectively, all in Kbps.

of these classes with a uniform distribution and generate the average quality curves for $0.1 \leq w \leq 5$. For each w, the results are obtained by averaging over 100 randomly selected bandwidth profiles with 6 peers. Figure 2.6 depicts how the average quality, rate and the number of layers/groups per source changes with respect to w. Our observations are as follows.

(a) In terms of the achieved quality, the proposed partition heuristic achieves a better performance than the layered system with 20% overhead and a similar performance as the layered system with 10% overhead, even though the average video rate of the heuristic falls below the upper bound, as seen from the upper right plot in Fig. 2.6. The quality of the layered system with 30% overhead is even lower than the quality achieved by the partition heuristic that limits the number of groups to 2.

(b) It is noteworthy that the quality achieved by the partition heuristic falls below that of the layered system with 10% overhead for $0.7 \leq w \leq 3$. This is mainly because the users with relatively higher uplink capacities in the conference (α-peers) are placed in the same group as their downlink capacities are also the highest, due to the fixed w assumption. Therefore, the bandwidth sharing between the α-peers and the ϵ-peers in the conference is reduced, leading to a decrease in the average quality in the conference. This effect becomes negligible as the downlink capacities get higher,

as the optimal partitioning choice tends to $P_i = \{R_i\}$ for any source i. For small w values, the described disadvantage in terms of the achieved rates is effectively masked by the coding overhead. In other words, both the layered system and the partitioned simulcast system fail to deliver high rates because of the small downlink capacities.
(c) We examine the average number of different video versions or layers created per source, for partitioned simulcast and layered systems, respectively. In the lower right plot in Fig. 2.6, we see that, for each scheme, the number of layers or versions generated decreases with the increasing downlink capacities, tending towards 1 (a single group or layer) for the uplink-limited-only regime. Note that, for the layered system, the number of layers assigned by the layer assignment heuristic is always $n-1$, but the optimal solution may use less layers, by assigning zero rates to higher layers. Also, in the 6-user scenario simulated, because there are only 4 bandwidth profiles, there are at most 3 different layers or groups. Nonetheless, for each w value, the number of different versions generated per source with the partitioning heuristic is always less than the number of layers used by the layered system.

Heterogeneous Uplink/Downlink Ratios

We also consider a P2P-MPVC scenario with 6 users, where the downlink/uplink ratios differ among the users. The considered uplink and downlink capacities are $\{0.5, 0.5, 0.5, 0.5, 1, 5\}$ Mbps and $\{0.5, 1, 1.5, 2, 4, 4\}$ Mbps, respectively. We will only examine the solutions of the layered video distribution with 20% overhead and the partitioned simulcast heuristic. The average video qualities achieved in the conference are $\bar{Q}_{LV} = 0.561$ and $\bar{Q}_{PH} = 0.594$ for layered system with 20% overhead and partitioned simulcast methods, whereas the video rates achieved by both systems are shown in Fig. 2.7. As can be seen from the achieved video rates, both the proposed layered and partitioned simulcast systems are able to cope with the additional downlink/uplink ratio heterogeneity in the conference. Again, in this case, the two systems achieved quite similar performance, with the partitioning system slightly better. This is consistent with the results we obtained for the 6-user scenario when all users have the same downlink/uplink ratio and there are at most 4 different bandwidth profiles among the 6 users. Note that even though all 6 users have different downlink capacities, the heuristic partitioning algorithm used only 2 groups for Sources 1 to 4.

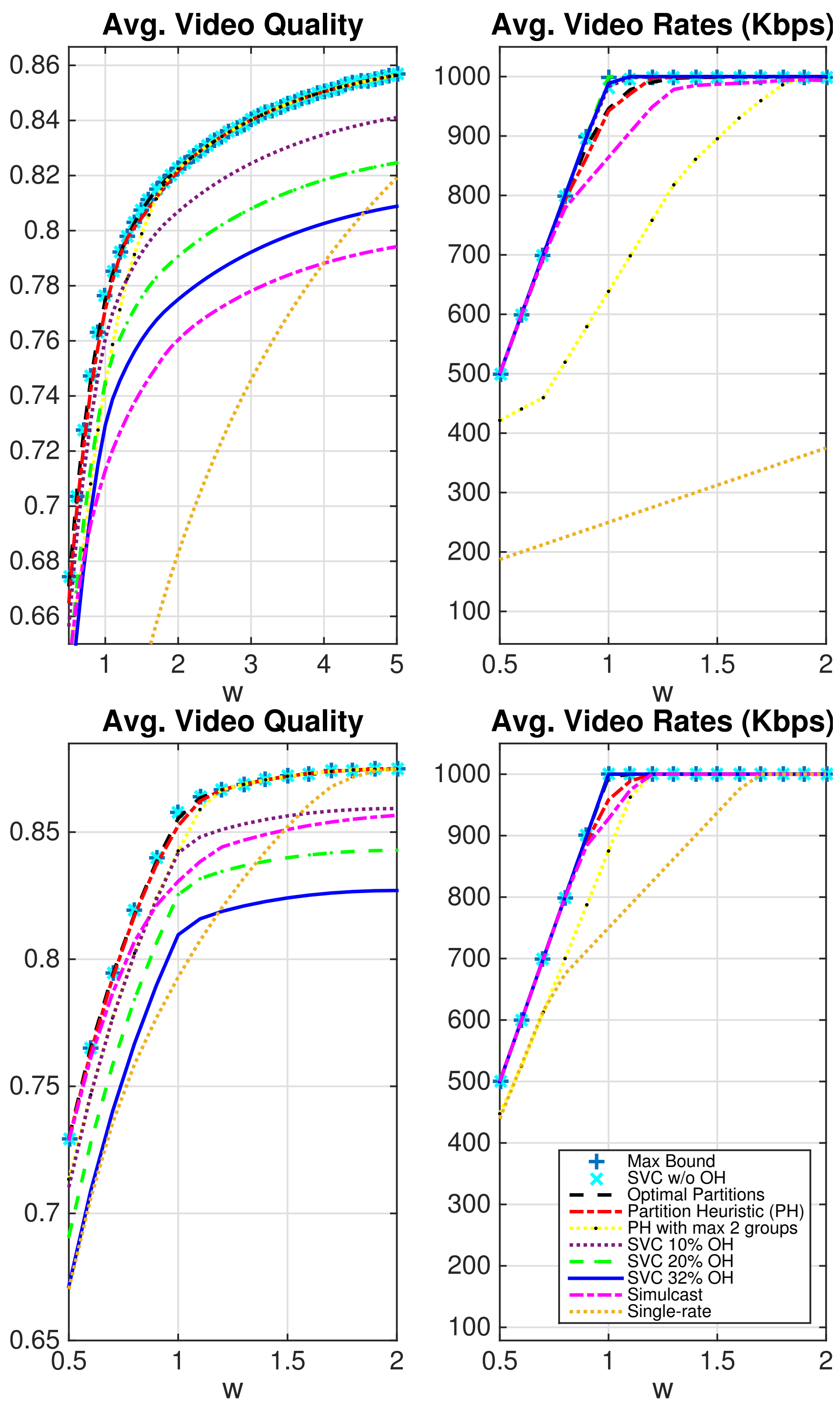

Figure 2.5: Average video quality (left) and rate (right) curves with user uplink capacities $\{0.5, 2, 4, 5.5\}$Mbps (top) and $\{1.5, 3, 3, 4.5\}$Mbps (bottom). Note that the curves for "Max Bound", "SVC w/o OH" and "Optimal Partitions" overlap on the quality plots, and the curves for "Max Bound", and all SVC cases overlap on the rate plots.

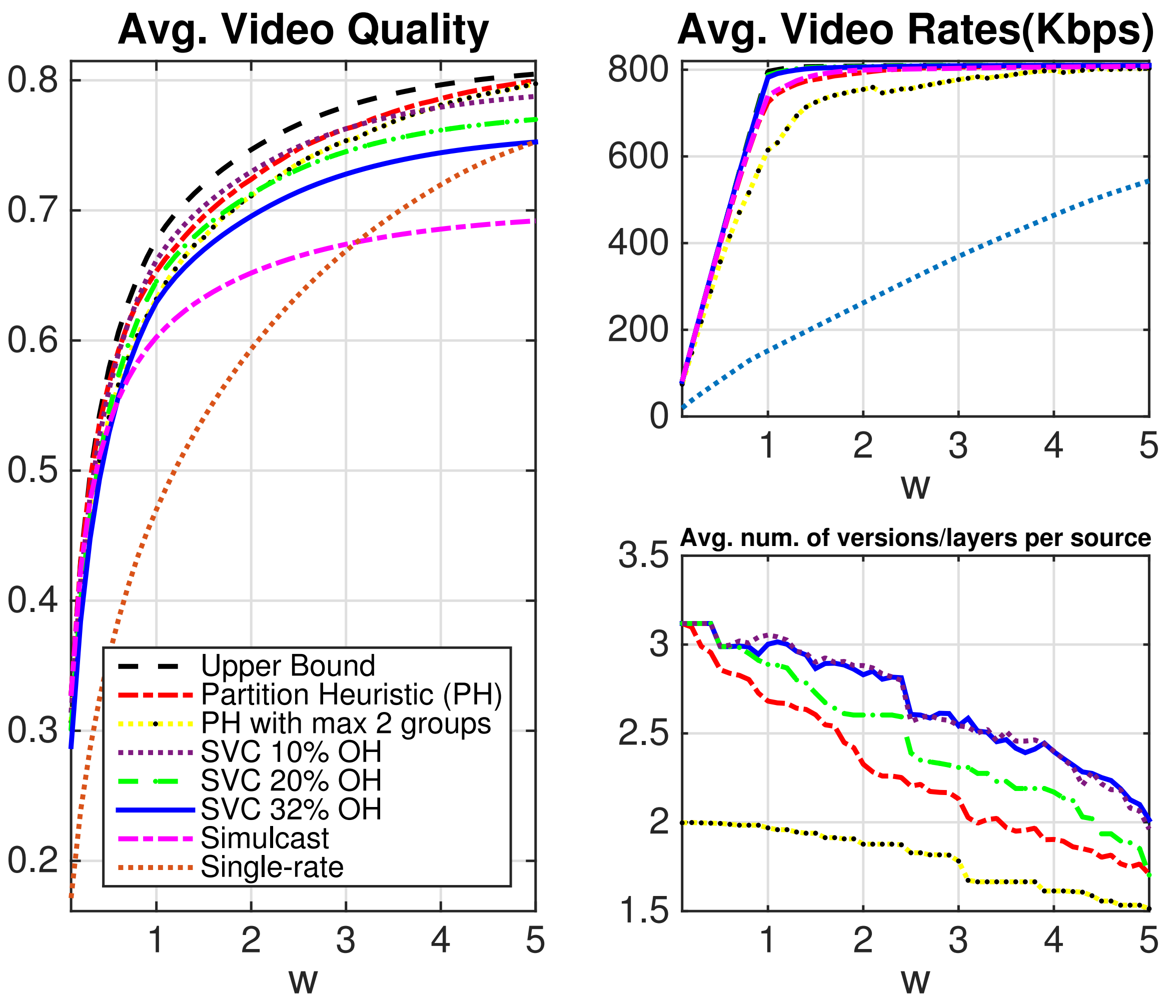

Figure 2.6: Results for 6-user P2P-MPVC, averaged over 100 random user capacity profiles: average video quality (left), average video rate received (upper right), average number of groups/layers per source (lower right)

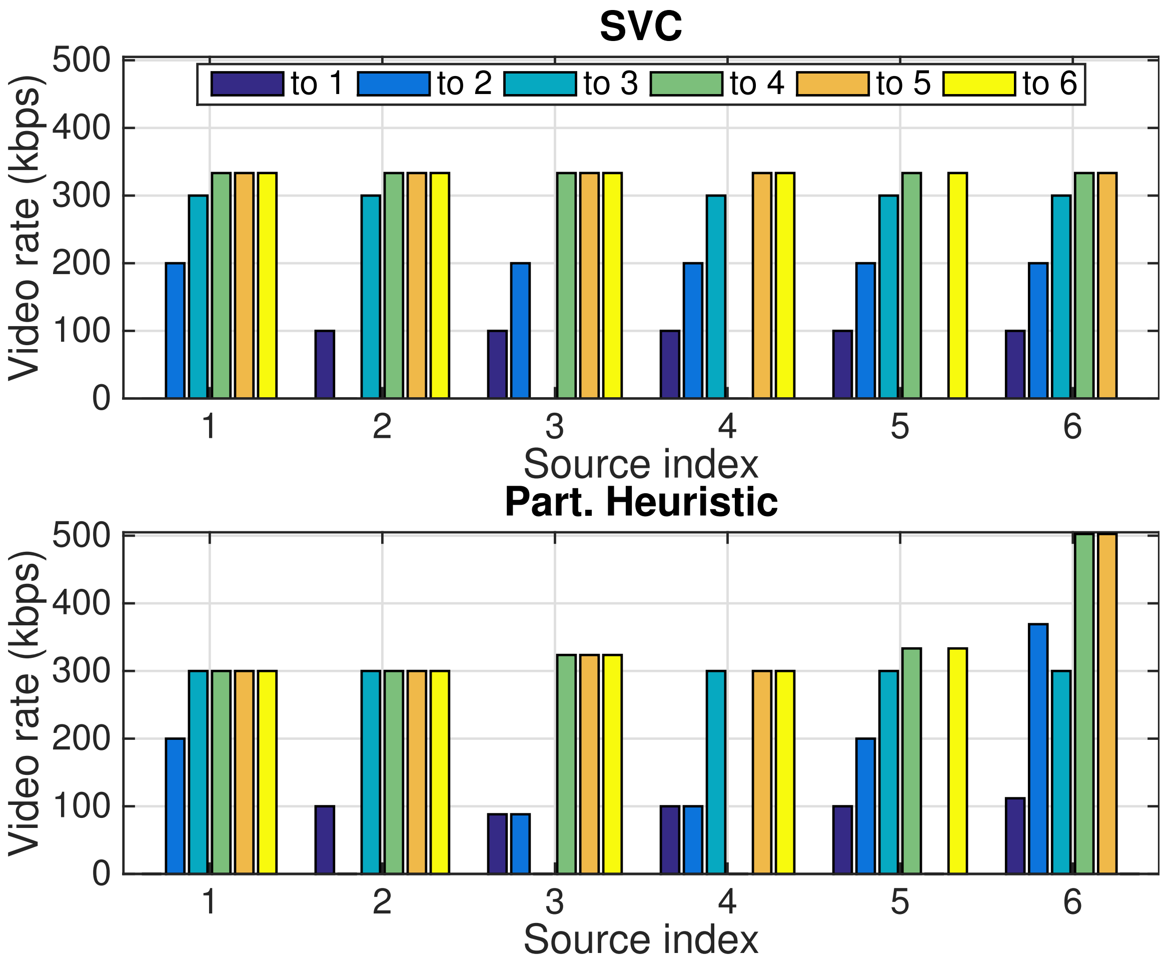

Figure 2.7: Video rates for 6-user P2P-MPVC with different w values: Uplink/downlink capacities for each user are $\{(0.5/0.5), (0.5/1), (0.5/1.5), (0.5/2), (1/4), (5/4)\}$ Mbps. Each bar represents the video rate from the corresponding source to the receiver. Note that the rates achieved by the layered system are proportional to the users' downlink capacity.

Chapter 3

Real-Time Bandwidth Estimation and Rate Adaptation for Video Calls over Cellular Networks

This chapter is organized as follows. We briefly discuss the related work on available bandwidth measurement and congestion control in Section 3.1. The overview of the proposed end-to-end system are presented in Section 3.2. The sending bitrate control design is presented in Section 3.3, where we discuss the available bandwidth measurement, future available bandwidth prediction and the determination of the next sending bitrate. In Section 3.4, we first study how to select the frames to send when the encoded video bitrate exceeds the sending bitrate determined. We then cover how to update the bit budget and the priority among the frames. Numerical simulation and emulation results comparing Rebera and FaceTime are reported in Section 3.5.

3.1 Related Work

Rate adaptation is a key problem for video transmission over best-effort networks. Most of the previous studies focus on one-way streaming of live or recorded video, where a few seconds of video buffering at the receiver can be tolerated. Due to buffering, a temporary mismatch between the video rate and the ABW does not directly impact the video playback, as long as the buffer does not drain out. The recent industry trend here is Dynamic Adaptive Streaming over HTTP (DASH) [41]. Various rate adaptation algorithms have recently been proposed [42–45], and some of them were specifically designed for wireless networks, e.g. [46, 47].

To the contrary, video call involves two-way real-time video streaming. To facilitate live interaction, video call does not have the luxury of seconds of video buffering. Consequently, mismatch between the selected video rate and the ABW will directly translate into quality degradation of video playback, such as long video frame delays, delay jitters and video freezes. Thus, for a video call, real-time bandwidth estimation and video rate adaptation are more challenging, compared with one-way video streaming.

Sending rate determination, or congestion control, has been an active area of research for decades. Window-based congestion control algorithms are the dominant form of congestion control on the Internet, which reactively adjust the window size, and hence the sending rate according to a congestion signal. TCP variants such as Tahoe and New Reno [48] use packet losses, while Vegas [49], FAST [50] and Compound [51] react to packet round trip times. However, the additive-increase multiplicative-decrease (AIMD) probing method used in TCP, along with the retransmission of every single lost packet in these protocols renders them less desirable for interactive video calls. TFRC [52] and TCP Cubic [53] control the sending rate with smaller variations, however, their delay performances deteriorate in the face of fast ABW variations. As a result, rate-based congestion control protocols are dominantly used in commercial video call applications, such as Microsoft Skype, Google Hangouts and Apple FaceTime. Nonetheless, these protocols also behave reactively when adjusting the sending rate, and therefore suffer from the same self-congestion problem over highly volatile links. Authors of [21] proposed a proactive congestion control scheme for realtime video delivery in cellular networks. They model cellular links as single-server queues emptied out by a doubly-stochastic service process. For the ABW estimation, we, unlike [21], assume no particular time-evolution model for the link capacity. Furthermore, [21] focused only on congestion control without considering video adaptation.

Adapting the video rate in real-time according to the sending rate determined is crucial for a low-delay video application. This task is usually handled at the video encoder only. However, if the rate control is not accurate and the encoded video rate exceeds the rate constraint, sending every encoded frame will cause self-congestion.

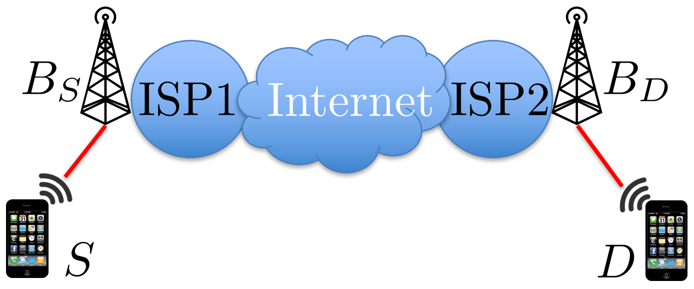

Figure 3.1: Cellular links between the mobile devices and their respective base stations are in red.

This problem can be alleviated if the video stream has temporal layering, by allowing the sender to prioritize from lower to higher layers, until the sending rate stays just below the rate constraint. However, on-the-fly decision of discarding a higher-layer frame that was encoded before the more important lower-layer frames is not trivial, since the sizes of the upcoming encoded frames are yet unknown. To the best of our knowledge, there is no published work addressing this problem.

3.2 Proposed System Overview

We examine a real-time video delivery scenario between a sender and a receiver, where at least one user is connected to a cellular network (Fig. 3.1). We denote the source device by S, the destination device by D, and the corresponding base stations by B_S and B_D, respectively. We call the directed path from S to D the forward path, and the directed path from D to S in the reverse direction the backward path. We assume that the in-network directed path (B_S, B_D) that connect the base stations has higher ABW, and constant queuing and propagation delay. Therefore, the overall ABW along the forward path (S, B_S, B_D, D) is equal to the minimum of the bandwidths along the cellular uplink (S, B_S) and cellular downlink (B_D, D).

According to the queuing model of [21], all packets destined to or sent from a given mobile device that is connected to a base station are queued up in isolated buffers, which are located on the mobile device for the uplink and in the base station for the downlink. These buffers are not shared by any other flow to or from other users; that is, there is no cross-traffic in these queues. The backlogged packets leave their respective buffers once they are successfully transmitted over the link. Thus, how fast these buffers are emptied out directly reflects the capacity of the cellular links, and consequently the end-to-end ABW.

As for the video stream, we assume that the sender uses a layered encoder so that it can easily adjust the sending rate by adjusting the number of video layers transmitted. Layered coding also enables unequal error protection; i.e., a basic level of quality can be guaranteed with high likelihood by providing more protection to the base layer. We consider only temporal layering to keep the encoding complexity and overhead at a minimum. In order to minimize the encoding delay, we further assume that the hierarchical-P structure (Fig. 3.5) is used to achieve temporal scalability. Starting with the highest temporal layer, the frames can be discarded to reduce the video rate. In the example shown in Fig. 3.5, each Group of Picture (GoP) consists of 4 frames, which leads to three temporal layers (TLs). We assume that the encoder inserts an I-frame every N frames, and we denote the time duration covering all N frames from an I-frame up to but excluding the next I-frame by an "intra-period." Then, the time duration T for an intra-period is equal to N/f, where f is the frame rate of the captured video.

We can now summarize the operation of the proposed system. Since rate control is usually performed once per intra-period in conventional video encoders, we predict the average ABW for each new intra-period. The prediction is based on the average ABWs for the past intra-periods, which are measured by the receiver and fed back to the sender. Specifically, the receiver periodically measures the ABW using the video frames that arrived within the last T-second window, and then feeds the result back to the sender. The window slides forward every Δ seconds. In order to have as fresh feedback messages as possible, we have $\Delta \ll T$. The sender, in turn, records the most recent measurement, and updates its value with the arrival of each new measurement. Then, at the beginning of the next intra-period k, the value of the most recent measurement is taken as the ABW $\tilde{c}_{k-1}$ measured during the last intra-period $k-1$. This value is input to an adaptive linear prediction filter, which then updates its prediction $\hat{c}_k$ regarding the ABW during the new intra-period k using the past bandwidth values $\tilde{c}_{k-1}, \ldots, \tilde{c}_{k-M}$. Using this prediction, the sender calculates the bit budget b_k, which is the maximum number of bits that the sender is allowed to send during this intra-period so that all the data that have been sent arrive at the receiver until the end of the intra-period with a high probability. The components of our design can be seen in Fig. 3.2.

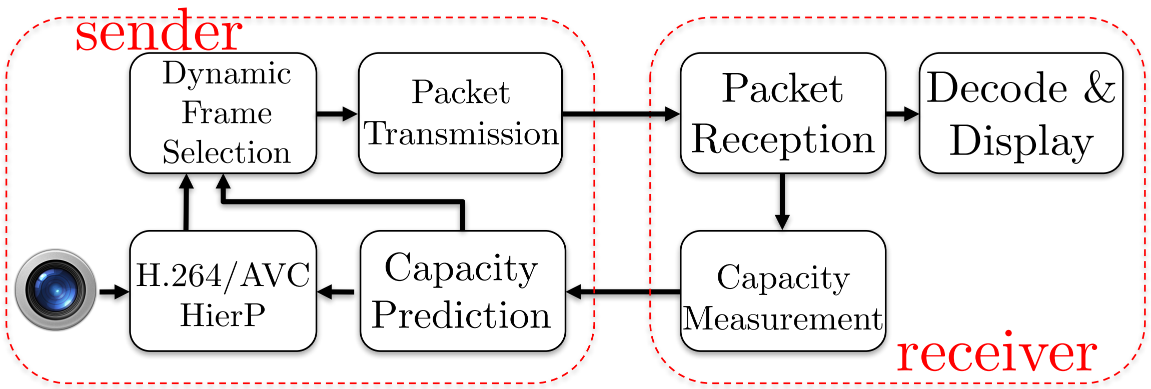

Figure 3.2: Proposed Rebera real-time video delivery system for cellular networks

3.3 Sending Rate Control

3.3.1 Measuring the Available Bandwidth

Packet pair/train methods [54] are well-known active capacity measurement schemes for finding the minimum capacity along a network path. The performance of these methods improve if there is no cross-traffic on the links, making them suitable to measure the cellular link capacity according to our model. In our system, we propose measuring the average ABW $c(t_1, t_2)$ actively at the destination, using the video frames received in $(t_1, t_2]$ as packet trains. Using the video frames as packet trains enables us to directly exploit the video data flow for capacity measurements and to avoid sending additional measurement traffic. Specifically, at the sender side, we first packetize each frame regardless of its size into $p \geq 2$ packets, and then send them together in a burst. The resulting instantaneous sending rate is likely to be higher than the instantaneous capacity of the link. As a result, packets queue up in the bottleneck; i.e. the base station buffer for the downlink or the cellular device buffer for the uplink, where they are transmitted one by one. Then, at the receiver side, we take capacity measurements $\{m_n\}$, where the sample m_n is obtained by using the arriving video frame n as a packet train. Let us denote the inter-arrival time between packet $i-1$ and i by a_i, and the size of the packet i by z_i. Then, we can calculate the capacity sampled by frame n as the following (Fig. 3.3)

$$m_n \triangleq \frac{z_2 + \cdots + z_p}{a_2 + \cdots + a_p} \triangleq \frac{Z_n}{A_n}. \tag{3.1}$$

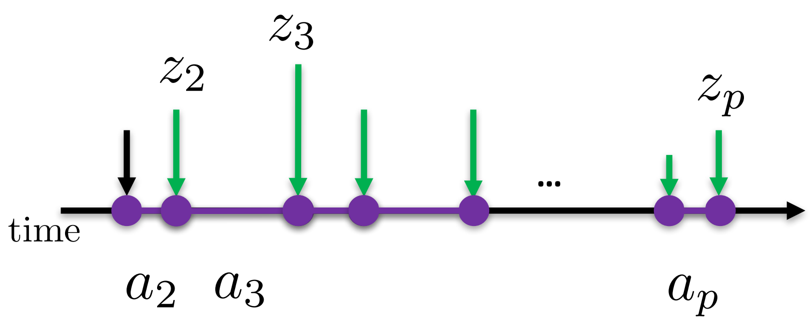

Figure 3.3: An illustration of the sizes and the inter-arrival times of the packets that make up a packet train at the receiver side.

For any time period $(t_1, t_2]$, we can estimate the average capacity $c(t_1, t_2)$ over this time simply by

$$\tilde{c}(t_1, t_2) = \frac{\sum_{n \in \mathcal{N}} Z_n}{\sum_{n \in \mathcal{N}} A_n}, \tag{3.2}$$

where $\mathcal{N}$ is the set of all frames that arrived in $(t_1, t_2]$. Note that Eq. (3.2) is equivalent to taking a weighted average of all the capacity samples in $\{m_n\}$, where the sample m_n is weighted in proportion to its measurement duration A_n with weight $w_n = A_n / \sum_{n \in \mathcal{N}} A_n$. Having completed the average capacity measurement regarding $(t_1, t_2]$, the receiver prepares a small feedback packet and sends it back to the source. Note that we are ultimately interested in measuring the ABW c_k during $(T_k, T_{k+1}]$, where T_k denotes the start of the intra-period k. However, since the sender and receiver have different clock times in general, the receiver cannot know when exactly an intra-period starts. Furthermore, the feedback packets are subject to time-varying delays in the network. In short, we cannot guarantee that the feedback packets will arrive at the sender on time for predicting the capacity of the next intra period. To address this issue, the receiver measures the average capacity within the last T seconds every Δ seconds, where $\Delta \ll T$, through a sliding window mechanism (Fig. 3.4). Each of these measurements are immediately sent back to the sender. Specifically, a measurement generated at time t is the average capacity in $(t - T, t]$, while the next measurement that is generated at $t + \Delta$ is the average bandwidth in $(t - T + \Delta, t + \Delta]$. The sender then uses the latest feedback received before T_k to predict the ABW in the next intra-period $(T_k, T_{k+1}]$. Lastly, assuming we keep the sending rate below the capacity, our measurement accuracy depends on the difference between the sending rate and the capacity of the link. If the sending rate equals, or

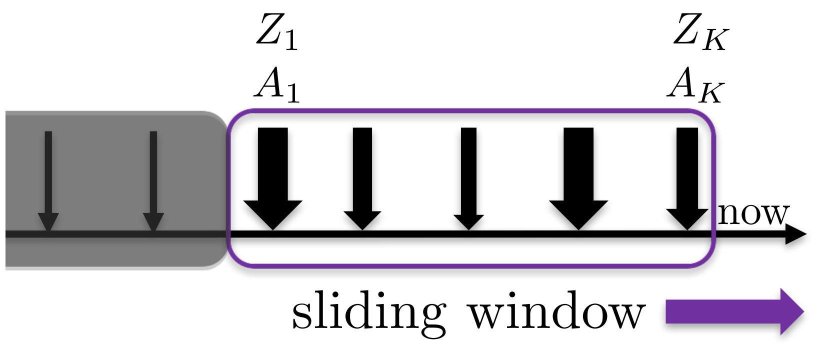

Figure 3.4: Sliding window of instantaneous bandwidth measurements, of length T seconds and sliding every Δ seconds.

by any chance, exceeds the capacity, we would have very high measurement accuracy, but this may lead to saturated links and long queueing delays, which are detrimental to video call quality.

Robustness against bursts

It is known that the cellular links occasionally experience channel outages that may last for up to several seconds, during which the capacity essentially drops to zero, and the packets in transit are backlogged in the respective buffers. As a result, the sender should stop sending any more packets as soon as an outage is detected. When the outage ends, all the packets queued up in the buffer are usually transmitted and arrive at the receiver as a burst. If the receiver uses these packets for capacity measurement, the burst rate, which is on the order of several Mbps, can severely disrupt the learning process of the predictor. In order to protect our system against these bursty measurements, we simply detect them through the sample measurement duration A_n. In our system, we consider a measurement bursty if $A_n < 10$ ms. Bursty measurements are simply discarded.

3.3.2 Predicting the Available Bandwidth

History-based forecast is a popular method for prediction [55], where the past measurement values are used to determine an estimate of the future. In this study, we perform linear prediction for history-based forecast. In particular, we chose a well-known online linear adaptive filter called the Recursive Least Squares (RLS) [22]. With each new capacity measurement regarding the last intra-period, RLS recursively updates

M	number of filter taps
λ	forgetting factor parameter
θ	initializer parameter for $\boldsymbol{P}$
$\boldsymbol{w}(k)$	filter tap vector of length M
$\boldsymbol{P}(k)$	inverse of empirical autocorrelation matrix, $M \times M$
$\boldsymbol{g}(k)$	gain vector of length M
$\widetilde{c}_k$	measured capacity
$\boldsymbol{c}(k)$	vector of M most recently measured capacity values
ϵ_k	a priori prediction error

Table 3.1: Notation regarding the RLS predictor

its filter taps of length M, and makes a prediction for the capacity during the next intra-period. One of the advantages of the RLS algorithm is that it does not require a model for its input signal, and performs minimum least-squares regression [56]. Furthermore, it can adapt to the time-varying signal statistics through its forgetting factor λ, which serves to exponentially discount the weight of the past observations, without any need for a time-evolution model for the system. The notation regarding the RLS algorithm are summarized in Table 3.1.

The periodic prediction procedure is as follows. At $t = T_{k+1}$, which is the end of the intra-period k, the most recent capacity measurement received by the sender is taken as $\tilde{c}_k$, that is, the average ABW during the intra-period k. Then, the gain vector $\boldsymbol{g}(k)$ and the a priori prediction error ϵ_k are calculated, which are then used to update the filter tap vector $\boldsymbol{w}(k)$. At this point, the linear prediction for c_{k+1} is simply

$$\hat{c}_{k+1} = \boldsymbol{w}^T(k)\boldsymbol{c}(k). \tag{3.3}$$

The step concludes by updating the inverse of the empirical autocorrelation matrix of the measured capacities. The overall procedure is summarized in Algorithm 3.

3.3.3 Determining the Sending Rate

Our ultimate goal is to ensure that all the frames sent during an intra-period finish their transmission before the start of the next one. In other words, we aim to have

Algorithm 3 Recursive Least Squares

```
1: $\boldsymbol{P}(0) = \theta^{-1}\boldsymbol{I}, \boldsymbol{w}(0) = \boldsymbol{0}, \boldsymbol{c}(k) = \boldsymbol{0}$    ▷ *Initialization*
2: **for all** intra-period $k \geq 1$ **do**
3:     $\boldsymbol{g}(k) = \frac{\lambda^{-1}\boldsymbol{P}(k-1)\boldsymbol{c}(k-1)}{1+\lambda^{-1}\boldsymbol{c}^T(k-1)\boldsymbol{P}(k-1)\boldsymbol{c}(k-1)}$
4:     $\epsilon_k = \tilde{c}_k - \boldsymbol{w}^T(k-1)\boldsymbol{c}(k-1)$
5:     $\boldsymbol{w}(k) = \boldsymbol{w}(k-1) + \epsilon_k \boldsymbol{g}(k)$
6:     $\boldsymbol{P}(k) = \lambda^{-1}[\boldsymbol{P}(k-1) - \boldsymbol{g}(k)\boldsymbol{c}^T(k-1)\boldsymbol{P}(k-1)]$
7:     $\hat{c}_{k+1} = \boldsymbol{w}^T(k)\boldsymbol{c}(k)$
8: **end for**
```

each I-frame encounter empty buffers with high probability. Let us denote our sending rate in the intra-period $k+1$ by r_{k+1}. We determine r_{k+1} such that the probability to exceed the capacity c_{k+1} is low; that is,

$$\Pr(c_{k+1} < r_{k+1}) = \delta, \tag{3.4}$$

where δ is a small tolerance parameter that characterizes our tolerance to frequent ABW overshoots. Let ϵ_{k+1} denote the ratio of the actual capacity to the prediction obtained from the RLS algorithm, i.e, $\epsilon_{k+1} = c_{k+1}/\hat{c}_{k+1}$. Then, we can rewrite Eq. (3.4) as

$$\Pr\left(\epsilon_{k+1}\hat{c}_{k+1} < r_{k+1}\right) = \Pr\left(\epsilon_{k+1} < u_{k+1}\right) = \delta, \tag{3.5}$$

where $u_{k+1} \triangleq r_{k+1}/\hat{c}_{k+1}$ is referred to as safety coefficient. This means that, for a given δ value, r_{k+1} should be set by scaling the prediction $\hat{c}_{k+1}$ by u_{k+1}, the δ-quantile of ϵ_{k+1}. In Rebera, we set $\delta = 0.05$, and calculate the running 5-percentile of ϵ_{k+1} with a moving window [57].

Handling backlogged and lost packets

Note that, keeping the sending rate below the ABW cannot be guaranteed, even with the safety margin u_k, leading to occasional packet backlogs. If we do not consider the backlogged packets while determining the sending rate, the total number of bytes backlogged in the buffers accumulate in time. In order to address this issue, the sender, through information fed back by the receiver, estimates the number q_k of bytes still in the buffers at the end of the intra-period k, by subtracting the total number of bytes received at the receiver from the total number of bytes sent so far. However, in case of packet losses, q_k would keep growing in time, since lost packets never arrive at the receiver. In order to account for the losses, we assume that the packets arrive at the destination in the order of their sequence numbers. To detect the number of lost bytes, we insert in each packet header the total number of bytes sent so far. Then, upon receiving a new packet, the receiver simply subtracts the number of bytes it has received so far from this number. The result is the number of bytes lost, which is fed back to the sender, along with the number of bytes received. The sender then determines q_k by taking the difference between the total number of bytes sent and the total number of bytes received and lost. Out-of-order packet deliveries will introduce only temporary errors to our estimates: after the delayed packets arrive at the receiver, our algorithm will automatically correct these errors in the next estimate. Combining all, we set the bit budget b_{k+1} for the intra-period $k+1$ as

$$b_{k+1} = (\hat{c}_{k+1} \times u_{k+1})T - q_k, \tag{3.6}$$

where T is the intra-period duration. *This way, we expect the network not only can finish the transmission of all video frames in intra-period* $k+1$*, but also can clean up the currently backlogged packets* q_k *by the end of intra-period* $k+1$*.*

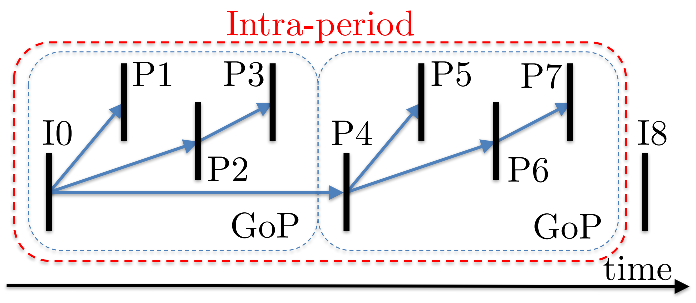

Figure 3.5: hPP prediction with $N = 8$. Blue arrows indicate the prediction directions. Here, $G = 4$, TL(1)={I0, P4}; TL(2)={P2, P6}; TL(3)={P1, P3, P5, P7}.

3.4 Real-Time Frame Selection for hPP Video

Video rate control is crucial for real-time applications over networks with time-varying bandwidth. However, accurate rate control is very challenging, especially in the very low-delay scenarios, where look-ahead and multi-pass encoding are not suitable. In spite of the extensive research in this area [58], significant mismatch between the target and actual bitrate over an intra-period can still occur [58]. In case of rate mismatch, if the video is coded with the IPP structure, all remaining frames will have to be discarded once the target budget for an intra-period is used up. When this happens early in the intra-period, the receiver experiences a relatively long freeze.

To remedy this problem, we propose to use a temporal layered encoder with the hierarchical-P coding structure, so that the sending rate can be adjusted by skipping the higher layer frames, without incurring additional encoding delay or complexity. Figure 3.5 shows an example prediction structure for the hierarchical-P encoding, which yields three temporal layers. We propose a frame selection scheme that either discards or sends each encoded frame, subject to the given bit budget b_k and frame dependencies. We assume that the video encoder runs its own rate control algorithm, but may not meet the bit budget per intra-period accurately. When the encoded bitrate exceeds the budget, an encoded frame may be dropped by the frame selection scheme so that the actual sending rate never exceeds the predicted bandwidth for an intra-period. The benefit of using the hierarchical-P structure is that the delivered video has more evenly-spread frames, whereas the IPP structure can lead to a very jittery video when some frames are dropped. With the frame selection module outside the encoder, the encoder rate control can be less conservative. This, in turn, can lead to higher bandwidth utilization.

3.4.1 Dynamic Frame Selection

Frame selection is ultimately about allocating the budget for more important (lower layer) frames. Higher layer frames can be sent only if there is available bit budget after sending the lower layer frames. However, to minimize the delays, the decision to either send or discard a given frame must be made right after it is encoded, without knowing future frame sizes. For example, in Fig. 3.5, we cannot wait to see if we can send P4 first, followed by P2 and then P1. Rather, we have to determine whether we send P1 as soon as P1 arrives. If the future frames from lower layers are large, sending frames from a current higher layer may preclude the sending of upcoming lower layer frames. On the other hand, dropping frames from higher layers when the future lower layer frames are small would underutilize the ABW.

Given an intra-period, let us label each frame with its appearance order, and denote the size and the temporal layer of the frame n by s_n and ℓ_n, respectively. Our goal is to decide, for each encoded frame n, to either send or discard it, such that the total number of frames sent at the end of the intra-period is maximized, while the mean and the variance of the time gap between the selected frames are kept small. We start our frame selection procedure by estimating frame size for each temporal layer, in order to make decisions considering the future frames. We then continue by ordering the frames in this intra-period based on their layer numbers, starting with the lowest layer, since the higher layer frames cannot be decoded with the lower layers. We denote this priority order by an ordered list π. For each newly arriving frame n, we trim π into π_n by excluding the past frames for which a decision has already been made, and the future frames that cannot be decoded at the receiver due to the previously discarded frames. π_n is basically the priority order among the eligible frames left. Next, we update the frame size estimations, as well as our estimation for the remaining bit budget. Then, we create a set E_n of frames that we expect to send according to our frame size and the remaining bit budget estimations, by greedily picking frames starting from the first frame in π_n. We stop picking the frames when the total estimated size of the frames picked reaches the estimated bit budget. Finally, if frame n is in the set E_n, we send it; otherwise it is discarded.

For frame size estimation, we assume that the frame sizes in the same temporal layer will be similar. Therefore, we keep a frame size estimate $\hat{s}_\ell$ for each layer ℓ. In this study, we use an exponentially weighted moving average (EWMA) filter with parameter γ for estimating the size of future frames in layer l using the actual sizes of the past coded frames in this layer. Note that for the base layer, we apply the above method only to the successive P-frames as the I-frame size is much larger than P-frames. We do not need to estimate the I-frame size, since we always send the I-frames. The overall algorithm is summarized in Algorithm 4.

3.4.2 Bit Budget Update

The bit budget b_k is the estimation of the total number of bits that the sender can transmit during the intra-period k without causing buffer build-up. Here, we assume that, at any time t since the start of the intra-period, $\frac{t}{T}b_k$ bits can be transmitted on average, with a mean rate of b_k/T. Thus, if the sender sends less than this amount, the unused bandwidth is wasted. In order to account for these missed transmission opportunities, we update the remaining bit budget at each step n by

$$\hat{b}_k(n) = b_k - \max\left(S_n, \frac{n}{N} b_k\right), \tag{3.7}$$

where S_n is the total number of bits sent before selecting frame n. Without updating the budget, the sender may end up sending large frames close to the end of the intra-period, which would then backlog in the buffer, and potentially delay all the packets in the next intra-period.

3.4.3 Frame Priority Order

In the frame priority list π, placing frame i before frame j means we allocate our bit budget to send frame i first, and that frame j is sent only if there is sufficient bandwidth budget to do so, after we have decided to send all the frames placed before frame j. Accordingly, lower layer frames have higher priority than the higher layer frames, which depend on the former. Within the base layer, the frames are ranked in their encoding order, as they follow the IPP coding structure. However, within an enhancement layer, *any* order of frames is decodable, since the frames from lower layers are picked before. Now, if the layer l frames are prioritized sequentially from the beginning, budget depletion results in a lower frame rate until the intra-period ends. On the other hand, if the frames are prioritized starting from the end, we may miss the transmission opportunities for the earlier frames, if the latter frames turn out smaller. Therefore, we pick the frames in multiple steps, alternating the direction in each step to strike a balance. Starting with the list of frames in the appearance order, we divide the list into two equal-sized lists at each step. We then pick the last frame from each smaller list, following the direction at that step.

3.5 Simulations and Experiments

3.5.1 Forecasting via Adaptive Filtering

We start our evaluations by motivating the use of the RLS linear adaptive filter for capacity prediction. We compare the prediction performance of the RLS with the simple and popular EWMA predictor [55]. In our experience, the filter length and the forgetting factor parameters do not significantly affect the prediction errors

Algorithm 4 Dynamic Frame Selection

```
1: $S_0 = 0$, $\pi_0 = \pi$, intra-period $k$, bit budget $b_k$
2: for all frames $n = \{0, \ldots, N-1\}$ do
3:     $\hat{s}_j \leftarrow \gamma s_n + (1-\gamma)\hat{s}_j$, for each frame $j \in \ell_n$
4:     $\hat{b}_k(n) = b_k - \max(S_n, \frac{n}{N} b_k)$
5:     Create $E_n$ from $\pi_n$, based on $\hat{\boldsymbol{s}}$ and $\widehat{b}_k(n)$
6:     if $n \in E_n$ then
7:         $S_{n+1} = S_n + s_n$ and send frame $n$
8:     else
9:         $\pi_{n+1} = \pi_n - \{$frames depending on $n\}$
10:    end if
11:    $\pi_{n+1} = \pi_n - n$
12: end for
```

	Mean (kbps)	Std (kbps)	Coeff. of Var.	Outage %
Trace 1	176	115	0.654	2.0
Trace 2	388	165	0.425	0.5
Trace 3	634	262	0.413	0.0
Trace 4	735	264	0.359	0.2
Trace 5	937	356	0.379	1.2
Trace 6	1055	501	0.475	0.1

Table 3.2: Statistics of traces used in the experiments.

provided that we choose $M < 10$ and $\lambda > 0.99$. Therefore, we have selected $M = 5$, $\lambda = 0.999$, $\theta = 0.001$ and fixed this configuration for the rest of the evaluations. We collected six real cellular link capacity traces (Fig. 3.9) following the methodology in [21], over T-Mobile 3G and HSPA networks, during different times of the day and in different campus locations. Each of these is 1066 seconds long and their statistics can be found in Table 3.2. As expected, the capacity traces are very dynamic, posing significant challenge to capacity estimation.

Over these traces, in Matlab, we perform time-series forecasting using RLS with parameters mentioned above, and the EWMA filter, where the smoothing parameter α is varied from 0 to 1. We assume that we know the past capacity values exactly. The results can be seen in Table 3.3, where "Best" and "Worst" represent the mini-

	RLS	α_{Best}	Best	α_{Worst}	Worst
Trace 1	53	0.55	55	0.05	87
Trace 2	88	0.7	90	0.05	120
Trace 3	158	0.55	157	0.05	209
Trace 4	186	0.4	178	0.05	211
Trace 5	250	0.2	235	1	293
Trace 6	244	0.4	242	0.05	291

Table 3.3: Comparing the prediction error RMS of the RLS predictor with those of the best and worst EWMA predictors with corresponding parameters. RLS, Best and Worst columns are in Kbps.

mum and maximum prediction error root-mean square (RMS) values obtained with EWMA with different smoothing parameters, respectively. We see that for all traces, prediction performance of RLS is very close to that of the best EWMA predictor, if not better, as it adapts to the statistics of the time series.

3.5.2 Dynamic Frame Selection Simulations

Next, we compare the performance of our dynamic frame selection (DFS) algorithm against the layer-push (LP) and frame-push (FP) algorithms. LP also estimates the frame size in each temporal layer using the same approach as in DFS, but then decides on the highest layer $l_{\max}$ that may be sent. In other words, only the frames from layers up to $l_{\max}$ are eligible for sending. Among these frames, following the encoding order, the algorithm sends as many frames as possible until the bit budget is exhausted. FP, on the other hand, does not consider layer information and sends as many frames as possible following their encoding order, until the bit budget is exhausted.

For each algorithm, we evaluate the total number of frames sent, the mean and the standard deviation of the resulting frame intervals, and finally the fraction of the unused bit budget. Here, a frame interval represents the temporal distance between a pair of consecutive frames that have been selected to be sent. The frame interval statistics are calculated using the fraction of time each interval lasts as the probability to observe that interval. We use the JM encoder [36] to encode the video sequence “Crew” [59] with a hierarchical-P structure having three temporal layers (GoP length=4) and intra-period of 32 frames. We used a fixed quantization parameter (QP) of 36, yielding the average bitrate of 415 kbps when all frames are included. The resulting video sequence has a frame rate of 30 fps and comprises 9 intra-periods, with an intra-period of $T = 32/30$ seconds. For the proposed algorithm, we used $\gamma = 0.75$, which was found to perform the best, and the frame priority order is $\pi = (0, 4, 8, 12, 16, 20, 24, 28, 30, 14, 6, 22, 26, 18, 10, 2, 31, 15, 7, 23, 27, 19, 11, 3, 1, 5, 9, 13, 17, 21, 25, 29)$. In these simulations, we assume that the bit budget b_k is constant for each intra-period k of the video and we want to compare the performances of the algorithms described above under different b_k values, from 10 kB to 80 kB. In Fig. 3.6, we see that FP sends the most frames by sending as many frames as possible. However, it also has the largest mean frame interval and the largest frame interval variation, making the displayed video jittery. On the other hand, the LP algorithm sends the lowest number of frames but also with lower mean frame interval and frame interval variance. The proposed DFS algorithm achieves a good compromise between sending more frames, consequently utilizing ABW more closely, and reducing the frame distance variation. In fact, DFS outperforms both methods in terms of the mean and standard deviation of the frame intervals, while sending almost as many frames as the FP. Finally, the plot in the upper right shows the fraction of the unused bandwidth for each method, where we see that the performance of DFS is very similar to FP, whereas LP is not as efficient.

3.5.3 Evaluation on the Testbed

For system evaluation, we developed a testbed to compare Rebera with popular video call applications. On this testbed (Fig. 3.7), S and D are the source and destination end-points running the video call application under test, while the nodes C_S and C_D are cellular link emulators running the CellSim software [21], respectively. The emulators are connected to each other through the campus network, and to their respective end-points via Ethernet. For cellular link emulation, we use the uplink and downlink capacity traces collected (Table 3.2). For evaluation, we report the ABW utilization, the 95-percentile one-way packet delays, and the 95-percentile one-way frame delays as the performance metrics. In order to calculate the bandwidth utilization, we count how many bytes were sent out by the video call application

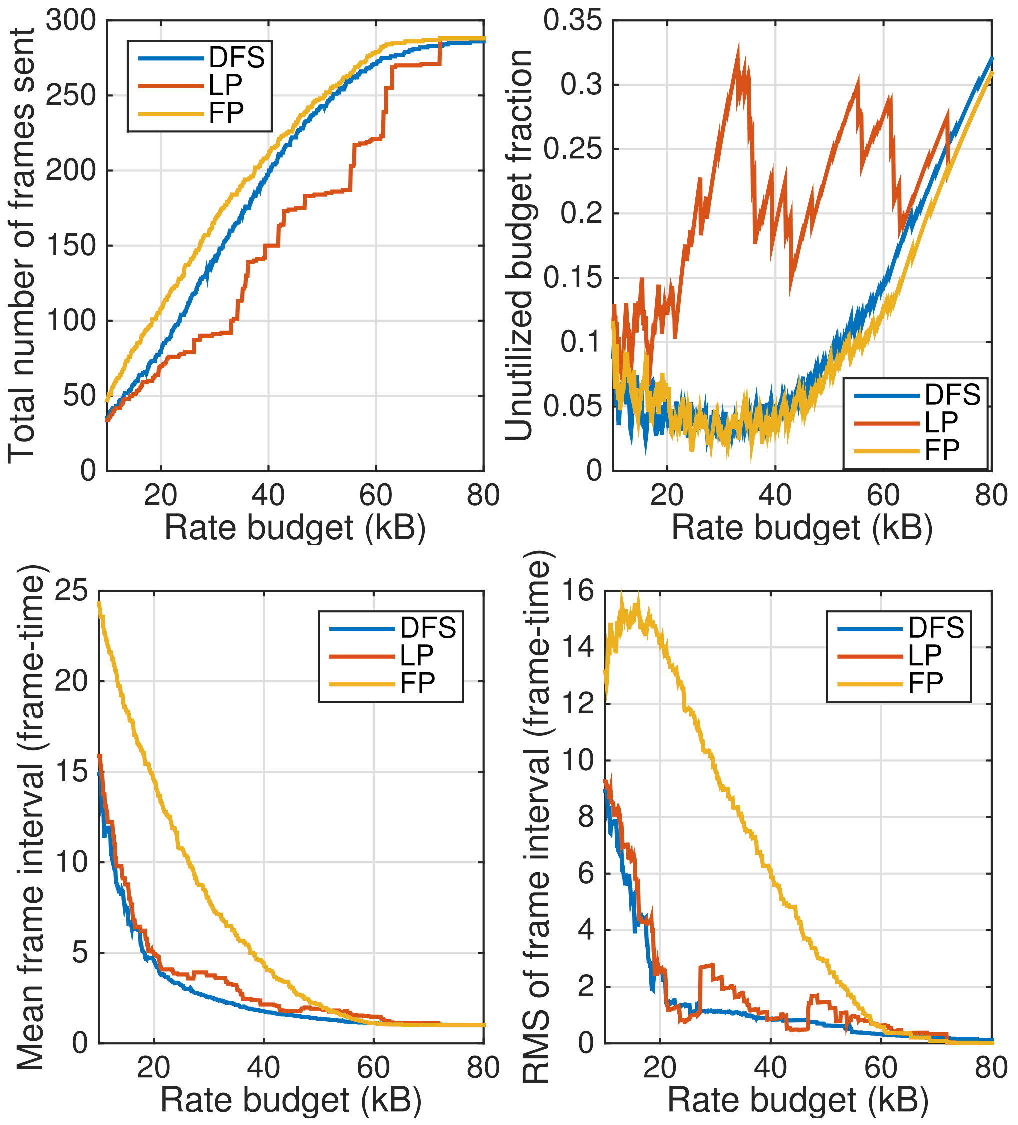

Figure 3.6: Comparison of DFS with FP and LP; number of frames sent (upper-left), unused budget (upper-right), mean and standard deviation of the frame intervals (lower-left and lower-right). Encoding frame rate is 30 Hz, thus frame-time is 1/30 sec.

under test and compare it with the minimum of the capacities of the sender link and the receiver link. The one-way end-to-end delays are collected by different means: in Rebera experiments, for each packet that made it to the receiver, the receiver sends an acknowledgement packet back to the sender over an ethernet cable on which there is no other traffic (Fig. 3.7). As a result, the measured round-trip times are almost equal to the one-way delays, enabling us to measure the delay for each individual packet and frame. In FaceTime experiments, we used Wireshark to sniff the packets on the emulator hosts. We also note that FaceTime sends voice packets even after the voice is muted, at a constant rate of 32 kbps. Rebera, on the other hand, does not send audio. In order to compensate for this in the bandwidth utilization calculations, we subtract 32 kbps from the sending rate achieved by Rebera. In each test, we loop the video sequence "Crew", which is more challenging in terms of the video rate than "Akiyo" and somewhat captures hand/arm movements present in video calls.

Rebera is able to encode the video in real-time thanks to the open source x264 video encoder [24]. This allows us to change the video rate according to the predicted ABW, for each new intra-period, using x264's rate control module. We have modified x264's code, so that the encoded video has a hierarchical-P coding structure, by changing the reference frames used before encoding each frame according to the H.264/AVC standard. Specifically, in our modification, the GoP length is set to 4, giving rise to 3 temporal layers. In all our experiments in the lab, the minimum and maximum encoding rates were set as 200 kbps and 3 Mbps, respectively. The video and RLS parameters used are the same as in Sections 3.5.2 and 3.5.1. Specifically, the encoding frame rate is 30 Hz, and the intra-period length T is 32 frames, or 1.066 seconds. The initial sending rate is set to 120 kbps. In each experiment, we evaluate the sending rate over consecutive periods of T seconds. Please note that FaceTime may not be using a constant intra-period, let alone the same intra-period T as Rebera. Furthermore, FaceTime's sending rate is, in general, the sum of FEC and the video data rates. In order to feed the same looped test video into FaceTime, we used the ManyCam [60] virtual webcam on Mac OS 10.10.4[1].

Evaluation with Piecewise Constant Bandwidth

In this experiment, we use a piecewise constant bandwidth trace, with steps of 100 kbps lasting 100 seconds, between 300 and 600 kbps. We set the packet loss rate to

[1]Detailed explanations can be found online at [61].

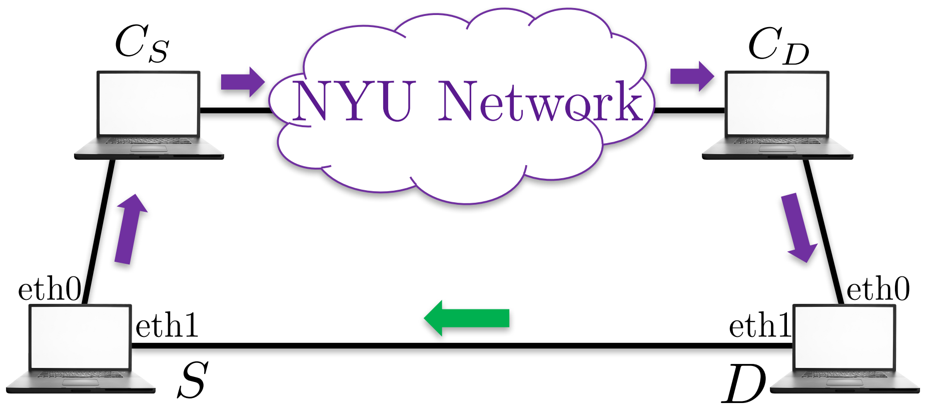

Figure 3.7: Illustration of the testbed. Purple arrows indicate the flow direction of the video, whereas the acks follow the green arrow from D to S.

zero. In Fig. 3.8, we can see Rebera's (i) measured bandwidth, (ii) rate reduction due to the estimated number of backlogged bits, (iii) overall budget and (iv) the sending rate, along with FaceTime's sending rate. On average, the bandwidth utilization of Rebera is 86.21%, while FaceTime achieves a utilization of 78.78%. Moreover, we can observe that, Rebera is able to measure the current bandwidth very accurately, and thus react to the changes in the bandwidth rapidly.

Evaluation with Cellular Capacity Traces

In this set of experiments, we use cellular bandwidth traces (Fig. 3.9) to emulate the cellular links. Each experiment lasts for 1000 intra-periods. We first present the results involving a single cellular link along the end-to-end path. Specifically, we start by examining the particular scenario where the sender is connected through a cellular network, and traces 5 and 6 were used to emulate the forward and backward end-to-end ABW, respectively. The receiver is assumed to have a wired connection. In the top plot of Fig. 3.10, we present Rebera's and FaceTime's sending rates over time. Here, Rebera achieves a forward bandwidth utilization of 75.6%, while FaceTime's utilization is 65.2%. Furthermore, 95-percentile packet and frame delays are observed to be 204 and 232 ms for Rebera, and 307 and 380 ms for FaceTime. The empirical packet and frame delay CDFs for both systems can be seen in Fig. 3.11.

Similarly, we employ traces 2, 3, 4 and 5 as the forward, and traces 1 and 6 for the backward end-to-end ABW. Results are summarized as bandwidth utilization, 95-percentile packet and frame delay tuples in Tables 3.4. We can see that in all

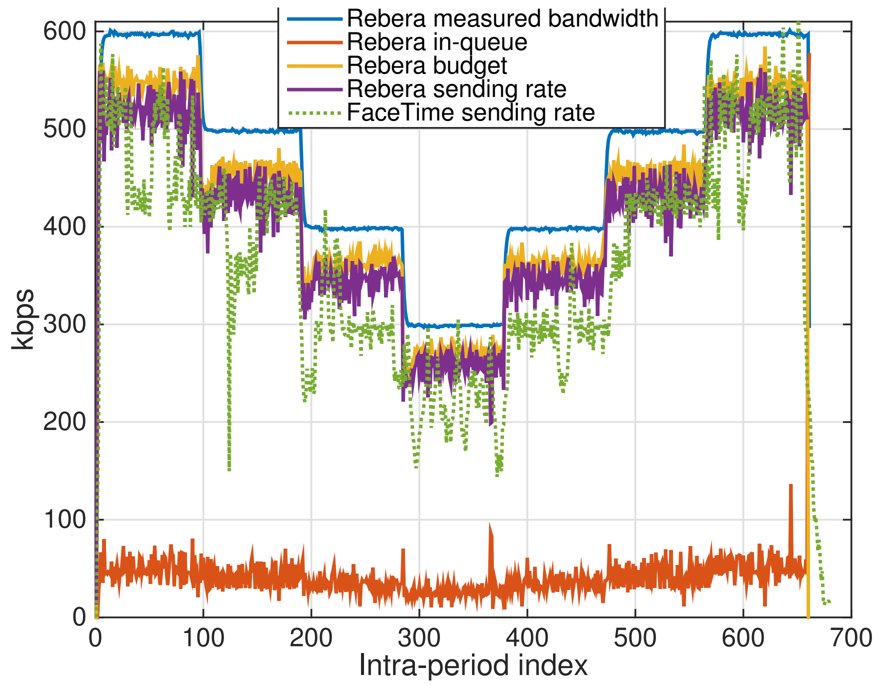

Figure 3.8: Sending rate of Rebera and FaceTime under piecewise-constant ABW.

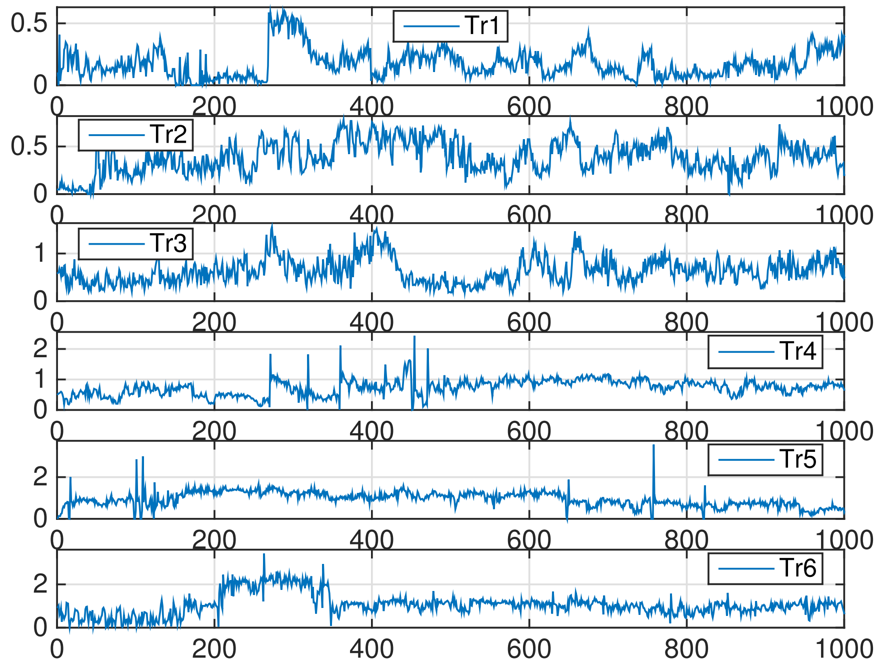

Figure 3.9: Traces used in the experiments. Vertical axis: capacity (Mbps), horizontal axis: intra-period index. Traces 2, 3, 4 and 5 are used as forward capacities, 1 and 6 are used as backward capacities.

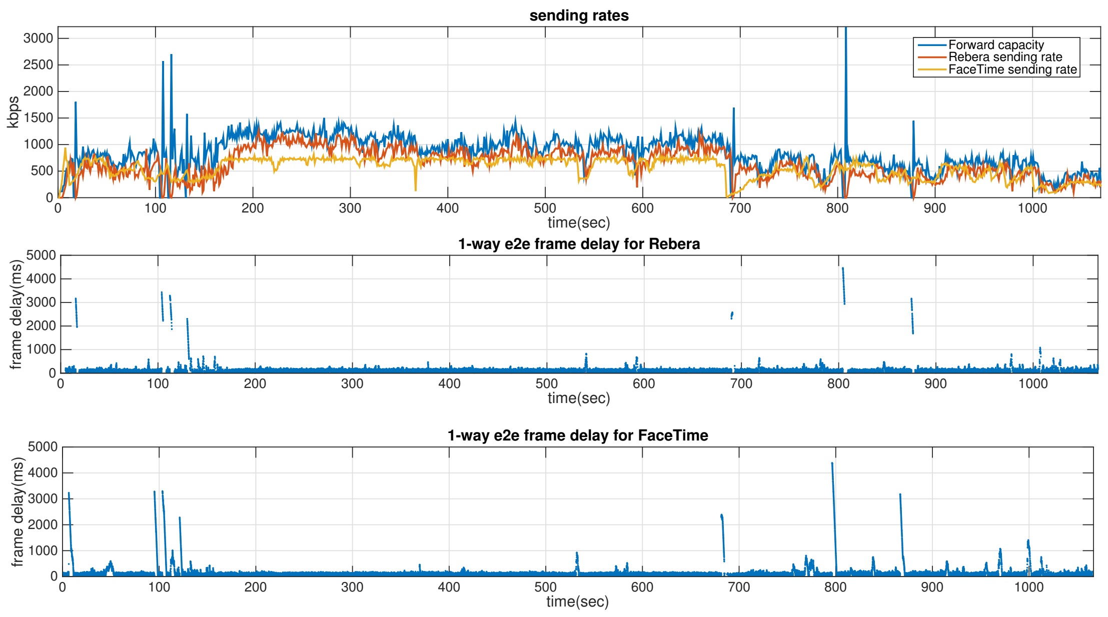

Figure 3.10: Video sending rates of Rebera and FaceTime (top) over a single cellular link with traces 5 & 6 as the forward & backward capacities, respectively. We present the one-way end-to-end frame delays of Rebera (middle) and FaceTime (bottom), where the horizontal axis represents the sending time of each frame. The avg. ABW utilization and 95-perc. packet and frame delay are in the bottom Table 3.4, row 4.

experiments, Rebera achieves a higher utilization of the forward ABW with shorter delays. Averaged over these experiments, Rebera provides 20.5% higher bandwidth utilization compared to FaceTime, and a reduction of 122 ms and 102 ms in the 95-percentile packet and frame queueing delays, respectively. When a more challenging backward capacity (trace 1 in Table 3.4) is used for the backward path, the information fed back to the sender side undergo a longer delay for both Rebera and FaceTime, decreasing the ABW utilization of both systems. FaceTime's delay performance also degrades, whereas Rebera is still able to provide similar delays.

Next, we consider the double-link scenarios, where both users are connected over different cellular links. We first assume there exists three different cellular connections, which we denote by A, B and C, where the uplink and downlink ABW pairs for each connection are given as traces 5 and 6, traces 3 and 4, and traces 1 and 2, respectively. In other words, connection A provides the highest mean ABW, while the connection C provides the lowest. We evaluate all six cases for which the sender and the receiver have different connections. The results can be seen in Table 3.5.

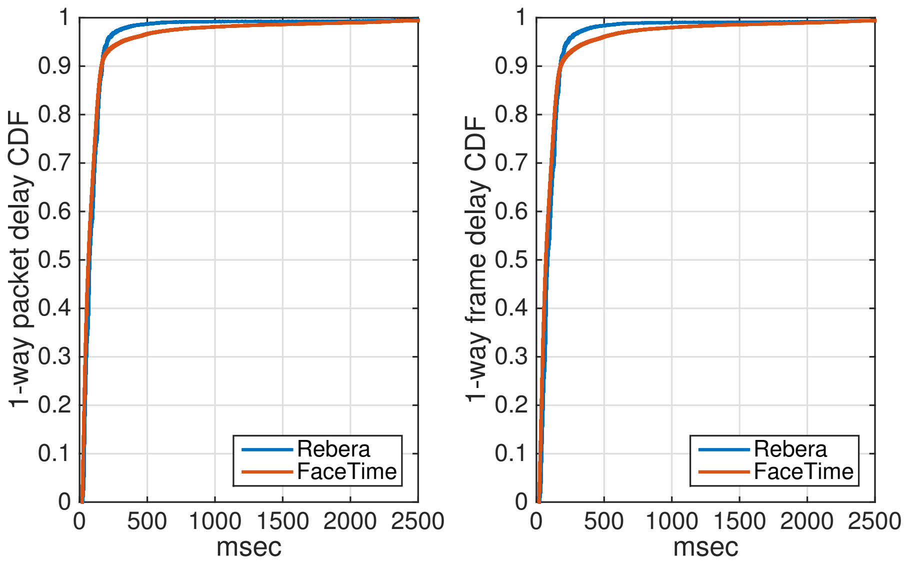

Figure 3.11: Empirical packet (left) and frame (right) delay CDFs of Rebera and FaceTime, where forward and backward bandwidths were emulated via traces 5 and 6, respectively.

Fwd Cap.	**Rebera(%,ms,ms)**	**FaceTime(%,ms,ms)**
Trace 2	68.8, 402, 402	58.1, 447, 415
Trace 3	63.8, 338, 334	32.7, 631, 528
Trace 4	73.5, 177, 201	63.0, 383, 392
Trace 5	71.8, 216, 243	58.7, 317, 341
Average	69.5, 283, 295	53.1, 444, 419

Fwd. Cap.	**Rebera(%,ms,ms)**	**FaceTime(%,ms,ms)**
Trace 2	69.5, 381, 394	59.2, 426, 406
Trace 3	66.4, 307, 313	61.9, 341, 349
Trace 4	76.1, 168, 189	70.6, 276, 307
Trace 5	75.6, 204, 232	65.2, 307, 380
Average	71.9, 265, 282	64.2, 337, 360

Table 3.4: Evaluation over single cellular link, using trace 1 (top) and trace 6 (bottom) as the backward capacity. Reported values are bandwidth utilization percentage, 95-perc. packet delay, and 95-perc. frame delay, respectively.

	Rebera(%,ms,ms)	**FaceTime(%,ms,ms)**
A to B	60.5, 300, 312	47.9, 529, 508
B to A	58.1, 483, 498	48.5, 485, 483
A to C	59.9, 432, 442	44.0, 588, 518
C to A	61.3, 1066, 1019	43.3, 1278, 851
B to C	61.2, 416, 419	28.2, 1180, 996
C to B	59.5, 804, 809	44.0, 3230, 1090
Average	60.1, 583, 583	42.6, 1215, 741

Table 3.5: Evaluation when both users are connected over different cellular networks. Reported values are bandwidth utilization percentage, 95-perc. packet delay, and 95-perc. frame delay, respectively.

In all scenarios, Rebera provides a significantly higher ABW utilization, while still delivering shorter packet and frame delays on average and in most cases.

Lastly, we extend our comparison of Rebera and FaceTime for double-link scenarios, using capacity traces in [21] that have been gathered from three different cellular operators in the US. In Fig. 3.12, we present two emulation experiments, each of which demonstrates the sending bitrates of Rebera and FaceTime, along with the end-to-end ABW. We can see that, Rebera's sending rate is below the capacity, and above the sending rate of FaceTime in each one. The results for all possible cellular trace combinations, merged with the previous results in Table 3.5, can be seen in Fig. 3.13. On average, Rebera provides 23% more ABW utilization, with 2 sec. reduction in the 95-percentile frame delays.

Effect of the Tolerance Parameter

Next, we investigate the effect of the tolerance parameter δ in Section 3.3.3 on Rebera. We vary δ from 0.05 up to 0.5, and record the utilization and 95-percentile packet delays in Table 3.6. Having a larger δ value means the system is willing to tolerate more frequent capacity overshoots, and hence more frequent large packet and frame delays, in exchange for higher bandwidth utilization, which could be the case for video applications with less stringent delay requirements.

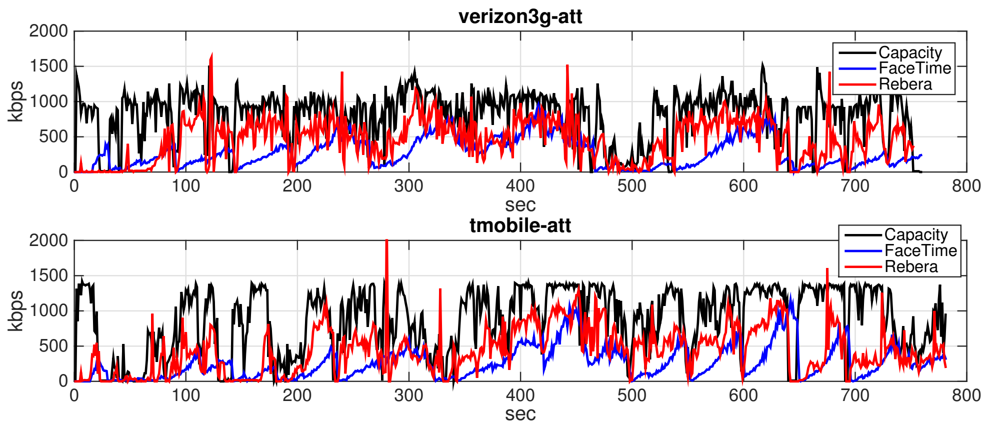

Figure 3.12: Example double-link emulations with cellular capacity traces with uplink / downlink traces: Verizon 3G / AT&T (top), T-Mobile / AT&T (bottom)

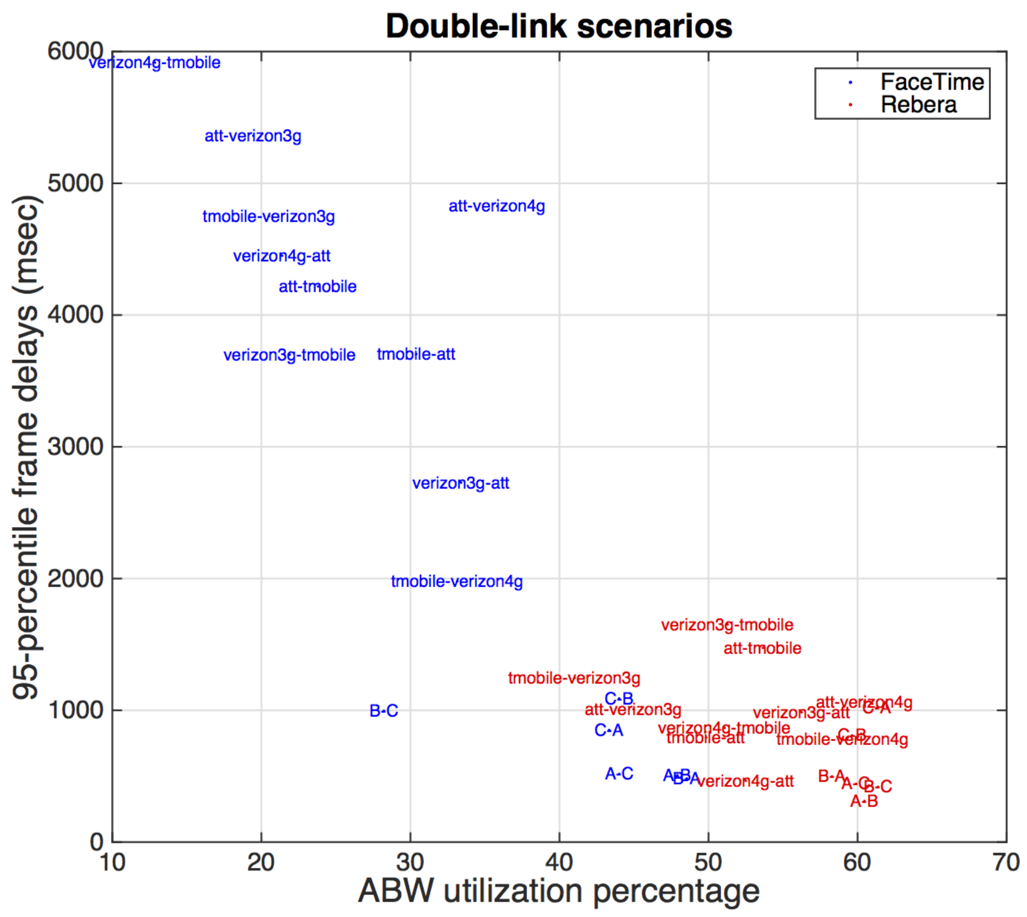

Figure 3.13: All double-link evaluation results. Each point labeled "x-y" in the plane represents an emulation experiment with uplink trace x and downlink trace y.

δ	0.05	0.1	0.2	0.5
ABW utilization (%)	66.4	69.6	72.7	75.36
95-perc. packet delay (ms)	307	354	371	486
95-perc. frame delay (ms)	313	367	403	516

Table 3.6: Effect of the tolerance parameter on Rebera over single cellular link. Forward-backward capacity: traces 3-6

Rebera Bahavior in Presence of Packet Loss

The purpose of this evaluation is to demonstrate that Rebera can still track the link capacity in the presence of packet loss. Note that additional studies are needed to investigate the error resilience provided by the temporal layering, and how to further improve it through unequal error protection. To examine the performance of Rebera in presence of packet loss, we employ CellSim to introduce random iid losses. We tested Rebera when the packet loss rate is 5% and 10%, and the results are given in Table 3.7. Although not significantly, the ABW utilization drops with the loss rate, as there are fewer packets crossing the links. Furthermore, the delays experienced by the received frames reduce, since there is less backlog in the buffers.

Packet loss rate	0%	5%	10%
ABW utilization (%)	66.4	63.4	61.1
95-perc. packet delay (ms)	307	281	286

Table 3.7: Effect of the packet losses on Rebera over single cellular link. Forward-backward capacities: traces 3-6

3.5.4 Evaluation over Cellular Networks

Finally, we evaluate Rebera over a real cellular network. The setup we used for this experiment can be seen in Fig. 3.14. Here, a mobile device (Motorola Nexus 6) is tethered to the sender host via USB, acting as a modem. The sender is stationary during the experiments, which last for 10 minutes. The receiver host is inside the campus network, and has a public IP address. The experiment was done over the

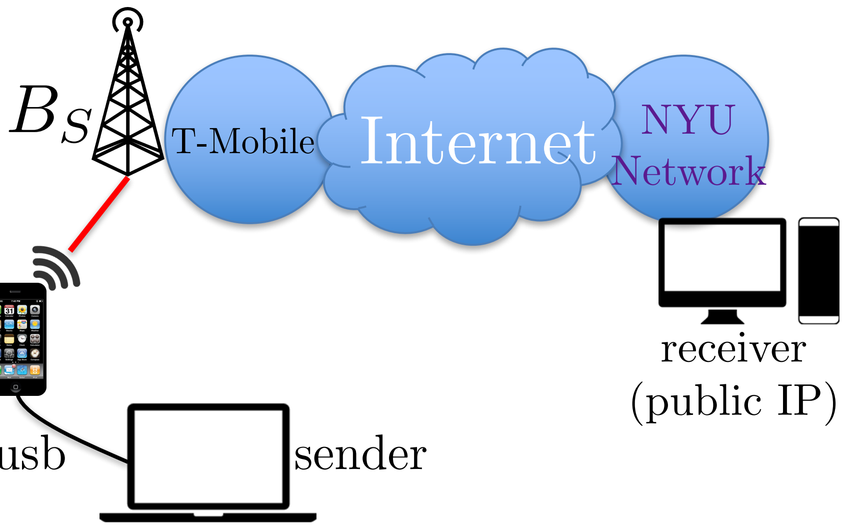

Figure 3.14: Setup for experiments in-the-wild

T-Mobile network, using LTE and HSPA on December 4, 2015 at 6 PM. In Fig. 3.15, we can see that, in the HSPA uplink, which provides an average ABW of 1.055 Mbps, the outages may last as long as 20 seconds. On the other hand, LTE provides an almost outage-free uplink channel, with an average ABW of 5.95 Mbps. Table 3.8 summarizes the experiment results. Note that during the experiment over LTE, Rebera's maximum encoding rate is set as 10 Mbps. This change serves as a means to utilize the ABW better, and is not necessary for Rebera's operation. The empirical packet delay distributions for Rebera using either access technology is given in Fig. 3.16.

Rebera	HSPA	LTE
average sending rate (Mbps)	0.81	5.17
95-perc. packet delay (ms)	221	105
95-perc. frame delay (ms)	298	137

Table 3.8: Rebera's performance over T-Mobile with HSPA and LTE technologies

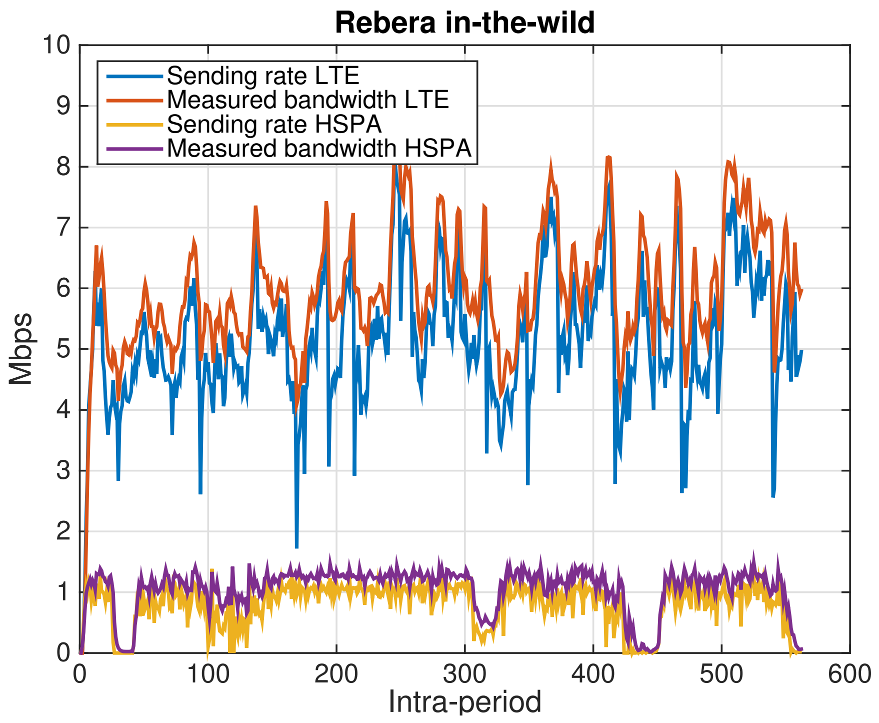

Figure 3.15: Rebera sending rate and measured bandwidth over LTE and HSPA

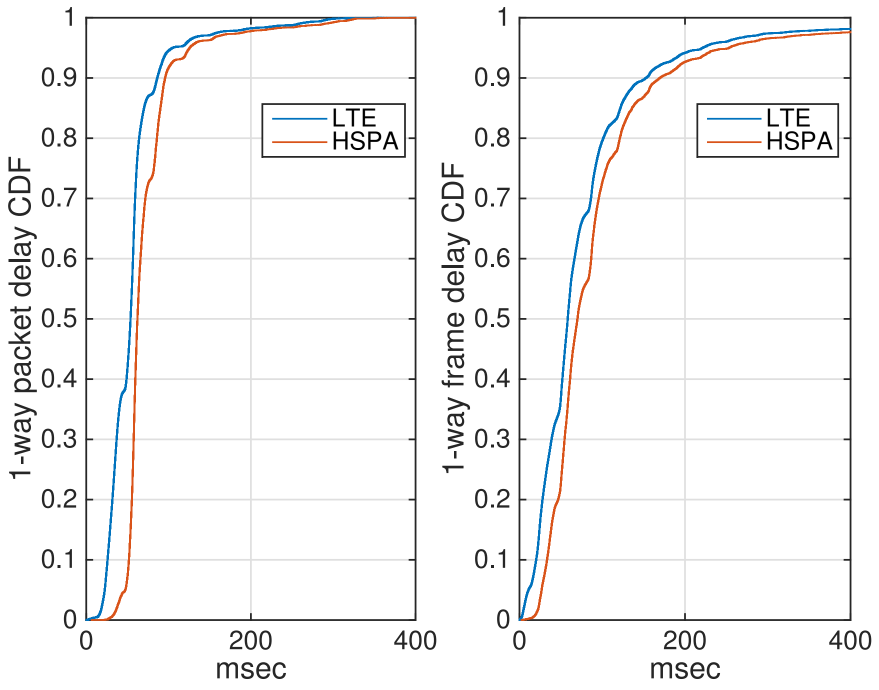

Figure 3.16: One-way packet and frame delay CDFs of Rebera over LTE and HSPA

Chapter 4

Perceptual Quality Maximization and Packet Loss Resiliency through Joint Optimization of Video Encoding and FEC Parameters

This chapter is organized as follows. We briefly discuss the related work on packet loss resiliency and video quality maximization in Section 4.1. The overview of the end-to-end system and the perceptual quality maximization problem are presented in Section 4.2. The means to determine the frame-level FEC redundancy rates is presented in Section 4.3, where we cast the combinatorial optimization problem and discuss the solution heuristics to be used in the case of independent and bursty packet losses. Extensive numerical evaluations under a range of sending bitrate constraints and different packet loss parameters are reported in Section 4.4, along with comparisons between the hierarchical-P and the IPP...P coding structures.

4.1 Related Work

Video transmission over unreliable networks requires both source and FEC coding, in general. Since infinite-length source or FEC coding blocks are not realizable, the *separation principle* [62], which states that the source and channel coding can be designed independently, cannot be used without performance loss in real-time video delivery applications. As a remedy, joint source-channel coding (JSCC), aiming to minimize the end-to-end source distortion, has been studied extensively for video delivery over lossy links or networks. Many papers, such as [63] and [64], focus on applying FEC on the PHY layer, assuming a particular network structure, while others focus on general end-to-end scenarios. Much of the existing work on this problem differs in (i) the video coding methods considered, (ii) the video distortion or quality metrics used, (iii) the total amount of video distortion caused by the unrecoverable packet losses, and (iv) the video data unit over which the chosen metric is estimated, and on which the FEC is applied. Traditionally, when layered video is considered, each layer is assumed to reduce the total MSE distortion by a certain amount that is known [65], or estimated [66, 67]. The additional distortion due to packet losses are mostly considered to be additive [66], however the nonlinear effect of bursty losses have been considered in [68]. In [69], a decision-tree-based subjective quality model is developed, without considering the optimization of the video encoding parameters, including eFR. From the delay perspective, the existing studies mostly do not address the delay requirement of video calls, as the FEC is traditionally applied on a fraction of a single or multiple consecutive GoPs [70, 71], where the receiver must wait for the entire GoP or sub-GoP to recover a frame, leading to either unacceptable delays or distortions. A few papers [72, 73] explored frame-level FEC strategies by maximizing the expected number of decoded frames under a total FEC bitrate constraint and assuming IPP coding structure. To the best of our knowledge, no prior work has studied frame-level FEC for hPP, nor studied the gains achieved by adapting the frame rate and FEC together.

4.2 Maximizing the Received Perceptual Quality

We consider a video call scenario between a source $\mathcal{S}$ and a destination $\mathcal{D}$ over an unreliable network, in which packets may get lost. The directed paths from $\mathcal{S}$ to $\mathcal{D}$ and $\mathcal{D}$ to $\mathcal{S}$ are called the forward and the backward paths, respectively. We assume that the available bandwidth on the forward path is predicted by a congestion control module at the sender, and based on this prediction, a safe maximum sending rate is determined to ensure low queuing delay[1]. We further assume that the packet loss

[1]The design of such a sending bitrate control mechanism (congestion control) that ensures low frame delays has been studied in Chapter 3.

events on the forward path are correlated in general. Similar to the previous work, we use the Gilbert model to describe the packet loss process on the end-to-end path. The arrival state of each packet, that is, whether it is lost or not, is represented by 0 or 1 in a discrete-time binary Markov chain, respectively. The corresponding state transition probabilities are

$$\Pr(0|1) = \xi_{0|1} \qquad \Pr(1|0) = \xi_{1|0}.$$

Then, the mean packet loss rate ϵ and the mean burst length λ are

$$\epsilon = \frac{\xi_{0|1}}{\xi_{0|1} + \xi_{1|0}} \qquad \lambda = \frac{1}{\xi_{1|0}}.$$

In this study, we consider only temporal layering to keep the layered encoding complexity and overhead at a minimum. Our goal is to explore the performance of both temporal-layered and non-temporal-layered encoding for video calls over lossy networks. We assume that the encoder inserts an I-frame every T seconds, which is called an *intra-period*. We employ the IPP structure in Fig. 4.1 for non-layered encoding, and the hPP structure in Fig. 4.2 for layered encoding with $L \geq 1$ temporal layers (TLs), where $G = 2^{L-1}$ is the length of a group of pictures (GoP). Note that, hPP reduces to IPP for $L = 1$. The number N of encoded frames that belong to a given intra-period is equal to $T \cdot f_e$, where f_e denotes the eFR in that intra-period.

Next, frame-level FEC is applied to protect the compressed video stream. The sender packetizes frame i of a given intra-period into $k_i = \lceil z_i/B \rceil$ network packets, where z_i is the size of the frame i and B is the maximum payload size. Then, Reed-Solomon coding [74] is applied across all packets of each individual frame i, by creating m_i redundancy packets based on k_i source packets. The redundancy rate for frame i is then $r_i = m_i/(k_i + m_i)$. By applying FEC across the packets of a single frame, instead of across packets belonging to multiple frames, we avoid further FEC decoding delay at the receiver.

When FEC fails and the current frame cannot be recovered, it is discarded along with its descendants in the dependency tree. In the meantime, the last decoded frame is frozen on the screen. This error concealment technique results in irregular distances between displayed frames that are free of artifacts, as opposed to regular intervals between displayed frames with noticeable artifacts that grow towards the end of the

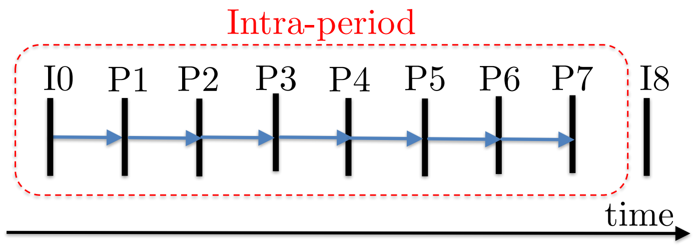

Figure 4.1: IPP prediction with $N = 8$. Blue arrows indicate the prediction directions.

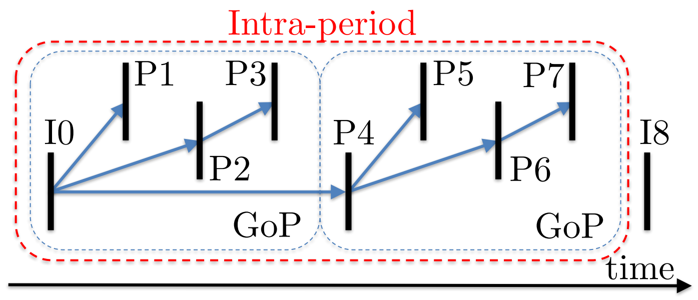

Figure 4.2: hPP prediction with $N = 8$. Blue arrows indicate the prediction directions. Here, $G = 4$, TL(1)={I0, P4}; TL(2)={P2, P6}; TL(3)={P1, P3, P5, P7}.

intra-period. Such frame-distance irregularity is less severe in hPP, since a playback freeze due to a frame loss in a temporal enhancement layer only lasts until the next lower-layer frame is decoded, providing higher resilience. For example, in Figure 4.2, if I0 and P1 are decoded and P2 suffers an unrecoverable packet loss, P2 and P3 are both discarded, regardless of whether P3 arrives or not. If P4 arrives, we can decode it from I0. In this case, P1, which has been kept on screen until now, will be replaced by P4. However, the same frame loss will make all subsequent frames P2-P7 non-decodable in Figure 4.1. Note that this error resilience is obtained at non-negligible expense of coding efficiency.

To measure the quality of the video displayed at the receiver, we use the perceptual quality model in [25]. According to this model, the perceptual quality of a decoded video sequence can be written as a function of the spatial, temporal and the amplitude resolution of the video. Then, keeping the spatial resolution constant, we have

$$\begin{aligned} Q(q, f_d) &= \mathrm{NQQ}(q) \cdot \mathrm{NQT}(f_d) \\ &= \frac{1 - e^{-\alpha_q \frac{q_{\min}}{q}}}{1 - e^{-\alpha_q}} \cdot \frac{1 - e^{-\alpha_f \left(\frac{f_d}{f_{\max}}\right)^{0.63}}}{1 - e^{-\alpha_f}} \end{aligned} \tag{4.1}$$

where q is the QS and f_d is the dFR, whereas α_q and α_f are parameters that depend on the video characteristics, and $q_{\min}$ and $f_{\max}$ are minimum QS and the maximum frame rate values considered, for which $Q(q_{\min}, f_{\max}) = 1$.

We can now summarize the operation of the proposed video call system. Since the video bitrate control is usually performed once per intra-period in conventional video encoders, we assume that the congestion control module determines the SBR R_S every intra-period [29]. In the meantime, packet loss process is monitored by the receiver, which keeps an estimate of the packet loss parameters and periodically feeds them back to the sender over the backward path $\mathcal{D} - \mathcal{S}$. When there is no packet loss, we have $f_d = f_e$, therefore the sender optimizes q and f_e such that the perceptual quality Q is maximized, under the sending rate constraint [26]. However, in the presence of random packet losses, f_d is a random variable that depends on the vector $\boldsymbol{m} = [m_1, \ldots, m_N]^T$ of the number of redundant FEC packets for each frame, frame size vector $\boldsymbol{k} = [k_1, \ldots, k_N]^T$, number L of temporal layers and finally the eFR f_e. Then, given the estimated loss parameters $\xi_{0|1}$ and $\xi_{1|0}$, along with the sending bitrate R_S, the sender is tasked with optimizing q, f_e and $\boldsymbol{m}$, so as to maximize the mean perceptual quality $\mathbb{E}(Q) \triangleq \overline{Q}$ of the decoded video for each new intra-period by solving the following problem.

$$\begin{aligned} \max_{q, f_e, \boldsymbol{m}} \quad & \overline{Q} = \mathrm{NQQ}(q) \cdot \overline{\mathrm{NQT}}(f_d(q, f_e, \boldsymbol{m})) \\ \text{s.t.} \quad & R(q, f_e) + \frac{B}{T} \sum_{i=1}^{N} m_i \leq R_{\mathrm{S}} \\ & q > 0,\ m_i \in \mathbb{N},\ f_e \in \mathcal{F} \end{aligned} \tag{4.2}$$

Here, $\mathcal{F}$ is the set of frame rates considered, and $R(q, f_e)$ is the bitrate of the encoded video, which is estimated by the rate model in [31] as follows.

$$R(q, f_e) = R_{\max} \left(\frac{q_{\min}}{q} \right)^{\beta_q} \left(\frac{f_e}{f_{\max}} \right)^{\beta_f} \tag{4.3}$$

Problem (4.2) is a mixed integer programming problem, and is hard to solve in general. Since $\mathcal{F}$ is a small set, optimizing f_e via exhaustive search in $\mathcal{F}$ is feasible. Then, for a particular value of f_e, the task is to optimize q and $\boldsymbol{m}$. Note that, even though q is continuous, the resulting frame size vectors $\{\boldsymbol{k}\}$ are discrete due to packetization. To predict distinct frame size vectors systematically, we assume that the frames in the same temporal layer have equal sizes, and develop a mean frame size prediction $\hat{z}_l$ per temporal layer l that depends on the target bitrate R linearly, given the eFR f_e, as well as the video contents. Accordingly, we choose to replace the optimization variable q with the video bitrate R, using Eq. (4.3). Next, we perform a hill climbing search in video bitrates $R \leq R_S$, however without using a predetermined step-size. Instead, we consider $R \leq R_S$ that result in distinct frame size vectors, which are estimated by the aforementioned model for a (f_e, R) tuple. In a bitrate interval $(R_0, R_1]$ that maps to the same prediction $\hat{\boldsymbol{k}} = \lceil \hat{\boldsymbol{z}}/B \rceil$, the highest bitrate R_1 leads to the maximum perceptual quality. The hill-climbing search is carried out over such suprema, starting from R_S and decreasing (line 8). At each step of hill climbing, NQQ $(q(f_e, R))$ is easily calculated from Eqs. (4.1) and (4.3) using the current R and the given f_e. Then, all that remains is to determine the FEC allocation vector $\boldsymbol{m}$ subject to the first constraint in Eq. (4.2) to maximize $\overline{\text{NQT}}(f_d(q, f_e, \boldsymbol{m}))$. The entire procedure is summarized in Alg. 5. Note that, $\boldsymbol{m}$ is determined from its previous value for speed-up (line 11). In the next section, we will focus on this procedure, that is, how $\overline{\text{NQT}}$ can be maximized.

Algorithm 5 Frame Rate and Video Bitrate Opt.

```
 1: Inputs: $\epsilon$, $\lambda$, $R_S$, $L$, $\mathcal{F}$, $T$, $B$, $\alpha_q$, $\alpha_f$, $\beta_q$, $\beta_f$, $f_{\max}$, $q_{\min}$, $R_{\max}$
 2: Outputs: $f_e^*$, $R^*$, $\boldsymbol{m}^*$
 3: $Q^* \leftarrow 0$
 4: for all $f_e \in \mathcal{F}$ do                                   ▷ Exhaustive search
 5:     $N \leftarrow T \times f_e$, $R \leftarrow R_S$, $Q \leftarrow 0$, $\boldsymbol{m} \leftarrow \boldsymbol{0}$, $\boldsymbol{k} \leftarrow \boldsymbol{0}$
 6:     do                                                            ▷ Hill-climbing begins
 7:         $Q_{\max} \leftarrow Q$, $R_{\text{best}} \leftarrow R$, $\boldsymbol{m}_{\text{best}} \leftarrow \boldsymbol{m}$
 8:         $R \leftarrow \max_{0 \leq R' \leq R} R'$ s.t. $\hat{\boldsymbol{k}}(R') \neq \boldsymbol{k}$           ▷ Next R
 9:         $\boldsymbol{k} \leftarrow \hat{\boldsymbol{k}}(R)$                                         ▷ Corresponding k
10:         $M \leftarrow \lfloor (R_S - R)T/B \rfloor - \sum_{i=1}^{N} m_i$
11:         $(\boldsymbol{m}, \overline{\text{NQT}}) \leftarrow$greedyFEC$(M, \boldsymbol{k}, \boldsymbol{m}, L)$
12:         $Q \leftarrow \text{NQQ}(q(f_e, R)) \times \overline{\text{NQT}}$
13:     while $Q > Q_{\max}$ and $R > 0$
14:     if $Q_{\max} > Q^*$ then
15:         $f_e^* \leftarrow f_e$, $R^* \leftarrow R_{\text{best}}$, $\boldsymbol{m}^* \leftarrow \boldsymbol{m}_{\text{best}}$
16:     end if
17: end for
```

4.3 Determining the Frame-Level FEC Rates

Given the video bitrate R, we can create at most $M = \lfloor (R_S - R)T/B \rfloor$ FEC redundancy packets of size B to protect the frames in the intra-period. Let d_i denote the Bernoulli random variable that assumes the value 1 if the frame i is decoded at the receiver and 0 otherwise, and let $\mathbb{I}$ be the indicator function. Then, the total number of decoded frames is given by $D = \sum_{i=1}^{N} \mathbb{I}_{d_i=1}$, and we can write

$$0 \leq f_d = \frac{D}{T} \leq f_e = \frac{N}{T}.$$

Ideally, we would like to optimize the FEC packet distribution $\boldsymbol{m}$, such that $\overline{\text{NQT}}(f_d)$ is maximized.

$$\begin{aligned} \max_{\boldsymbol{m}} \quad & \overline{\text{NQT}} = \sum_{n=0}^{N} \Pr(D=n)\text{NQT}(\frac{n}{T}) \\ & \text{subject to } \sum_{i=1}^{N} m_i \leq M \text{ and } 0 \leq m_i \end{aligned} \tag{4.4}$$

Problem (4.4) aims to find the best way to distribute M redundancy packets on N frames, where the search space is $C(M+N-1, M)$. Therefore, exhaustive search becomes infeasible even for small values of M. As a solution, we propose to use a greedy hill-climbing algorithm. We begin by assuming that only a single FEC packet is given, and increment the number of available FEC packets by one at each step. Next, we search among all the frames to determine which one should be protected with the newly added FEC packet. In other words, at step $n \leq M$, we search among the neighbors of the best solution $\boldsymbol{m}^*$ found so far, where a neighbor $\boldsymbol{m'}$ of a vector $\boldsymbol{m}$ has the same components except for $m'_j = m_j + 1$. We can speed up this process by skipping the frames in the same layer for which the (k_i, m_i) tuple has already been evaluated. For each neighbor, we calculate a score; $\overline{\text{NQT}}$ in case of iid packet losses or $\overline{f_d}$ for Markovian packet losses, and then pick the neighbor that leads to the highest value as the new solution candidate. The reason for the choice of a different search space for the Markovian packet losses is the additional computational complexity. The algorithm, which is summarized in Alg. 6, terminates in $O(NM)$ iterations.

Next, we show how to calculate $\overline{\text{NQT}}$ and $\overline{f_d}$ for iid and Markovian packet losses, respectively.

Algorithm 6 greedyFEC - FEC packet distribution

```
1: Inputs: $M$, $\boldsymbol{k}$, $\boldsymbol{m}$, $L$
2: Outputs: $\boldsymbol{m}$, $\overline{\text{NQT}}$
3: $\boldsymbol{m}^* \leftarrow \boldsymbol{m}$, $\overline{\text{NQT}}^* \leftarrow \overline{\text{NQT}}(\boldsymbol{m}^*)$        ▷ initial solution
4: while $M > 0$ do        ▷ distribute $M$ FEC packets
5:     $\boldsymbol{h} \leftarrow \boldsymbol{m}^*$
6:     for $1 \leq i \leq N$ do        ▷ over $N$ frames
7:         $\boldsymbol{m} \leftarrow \boldsymbol{h}$ and $m_i \leftarrow h_i + 1$
8:         if $\overline{\text{NQT}}(\boldsymbol{m}) > \overline{\text{NQT}}^*$ then
9:             $\overline{\text{NQT}}^* \leftarrow \overline{\text{NQT}}(\boldsymbol{m})$
10:            $\boldsymbol{m}^* \leftarrow \boldsymbol{m}$
11:        end if
12:    end for
13:    $M \leftarrow M - 1$
14: end while
```

4.3.1 Independent, Identically Distributed Packet Losses

Our goal in this section is to calculate $\overline{\text{NQT}}$, given the FEC packet distribution $\boldsymbol{m}$ and that the packet losses are iid. From Eq. (4.4), this calculation is straight-forward once the probability distribution $P_D(n)$ of the number D of decoded frames is known, for each $n \in \{0, 1, \ldots, N\}$. In turn, $\Pr(D = n)$ can only be calculated by considering all possible decoding patterns in the intra-period that result in the decoding of exactly n frames. Note that, each of these decoding patterns corresponds to a particular sub-tree of the frame dependency tree $\mathcal{T}$, having n nodes, and with the I-frame as the root node.

When the packet losses are independent Bernoulli events with probability ϵ, frame arrivals are also independent, but with non-identical distributions due to the unequal error protection. Let a_i denote the Bernoulli random variable taking on the value 1 if the encoded frame i successfully arrives at the receiver. Note that d_i and a_i are different events, since an arriving frame cannot be decoded without its reference frame. Then, the arrival probability for frame i is

$$\Pr(a_i = 1) = \sum_{j=0}^{m_i} \binom{k_i + m_i}{j} \epsilon^j (1-\epsilon)^{k_i + m_i - j},$$

that is, the probability of having at most m_i packet losses for frame i. Now, let $\mathcal{T}_i$ be the sub-tree of the coding structure $\mathcal{T}$, with its root as frame i. Furthermore, let $\mathcal{P}_i$

be the reference frame from which frame i is predicted, and $\mathcal{C}_i$ be the set of frames predicted from frame i. Then, if D_i denotes the number of decoded frames in $\mathcal{T}_i$ *given* that $\mathcal{P}_i$ is decoded, the distribution of D_i can be written recursively as follows.

$$\Pr(D_i = n) = \begin{cases} \Pr(a_i = 0) & n = 0 \\ \Pr(a_i = 1)\Pr(\sum\limits_{j \in \mathcal{C}_i} D_j = n - 1), & n > 0 \end{cases} \tag{4.5}$$

Since the frame arrivals are independent due to iid packet losses, and there is no decoding dependency across $\mathcal{T}_j$ for $j \in \mathcal{C}_i$, the distribution on the right hand side of Eq. (4.5) can be computed via convolution of the distributions of D_j, $j \in \mathcal{C}_i$. If frame i belongs to the highest temporal layer L, we have $\Pr(D_i = n) = \Pr(a_i = n)$. Finally, since $D = D_1$ by definition, $\Pr(D = n)$ can be found recursively from Eq. (4.5) and taking convolutions, as summarized in Alg. 7.

Algorithm 7 calcPMF (for iid loss)

```
1: Inputs: Frame index $i, \Pr(a_j = 1)$ for $1 \le j \le N$
2: Output: $P_{D_i}$
3: if $i \in \mathrm{TL}(L)$ then                                ▷ Highest layer
4:     $P_{D_i} = [\Pr(a_i = 0), \Pr(a_i = 1)]$
5: else                                                          ▷ Lower layer
6:     $c = 0$
7:     for all $j$ predicted from $i$ do
8:         if $c = 0$ then $c =$ calcPMF($j$)
9:         else $c = c \, * \,$ calcPMF($j$)                     ▷ convolution
10:        end if
11:    end for
12:    $P_{D_i} = [\Pr(a_i = 0), \Pr(a_i = 1) \times c]$
13: end if
```

4.3.2 Markovian Packet Losses

When the packet losses are Markovian and hence bursty, the frame arrivals become dependent, preventing us from using Alg. 7 to determine the end-to-end perceptual quality. Instead, we have to go through each possible frame decoding pattern, and calculate the corresponding probability of occurrence. As mentioned in Sec. 4.3.1, this is equivalent to enumerating all sub-trees of the frame dependency tree $\mathcal{T}$ with the

I-frame as the root node, and becomes computationally cumbersome [2] for practical values of N and L. We circumvent this problem by maximizing the mean number of decoded frames $\overline{f_d}$ instead of $\overline{\mathrm{NQT}}$. We have

$$\overline{f_d} = \frac{\mathbb{E}(D)}{T} = \frac{1}{T}\sum_{i=1}^{N}\mathbb{E}(\mathbb{I}_{d_i=1}) = \frac{1}{T}\sum_{i=1}^{N}\Pr(d_i = 1). \tag{4.6}$$

Then, the decoding probability for frame i can be expanded as the multiplication of conditional arrival probabilities.

$$\Pr(d_i = 1) = \prod_{j\in\mathcal{A}_i}\Pr(a_j = 1|a_u = 1, \forall u \in A_j) \triangleq \prod_{j\in\mathcal{A}_i} p(j) \tag{4.7}$$

Here, $\mathcal{A}_i$ is the set of ancestors of frame i in the coding structure. The conditional frame arrival probability $p(j)$ can be expanded [75] by further conditioning on the arrival state of frame j's first packet, which, in turn, depends on the arrival state of its reference frame's last packet. Towards this end, let $a_j^{(n)}$ denote the arrival state of the n^{th} packet that belongs to frame j. In particular, we consider the conditional probability $p_s(j)$ that the frame j arrives *and* the arrival state of its last packet is s, given that all the ancestors of frame j have arrived. Then, $p_s(j)$ is given by the following.

$$p_s(j) = \pi_0(j)\mathbb{I}_{m_j-2+s\geq 0}\sum\nolimits_{\ell=0}^{m_j-2+s} L_s(\ell, k_j + m_j - 2) \\ + \pi_1(j)\sum_{\ell=0}^{m_j-1+s} R_s(k_j + m_j - 2 - \ell, k_j + m_j - 2). \tag{4.8}$$

In Eq. (4.8), we first condition on $a_j^{(1)}$ using the probability $\pi_s(j)$.

$$\pi_s(j) = \Pr(a_j^{(1)} = s \mid a_u = 1, \forall u \in \mathcal{A}_j) \tag{4.9}$$

Additionally, $R_s(u, v)$ and $L_s(u, v)$ denote the probabilities that there will be *exactly* u received or lost packets in the next $v \geq u$ packets, which are then followed by a packet in state s, given that the first packet is received or lost, respectively. The upper-triangle matrices R_0, R_1, L_0 and L_1 can be pre-computed easily via memoiza-

[2] When $N = 32$, there are 2.015×10^6 and 3.187×10^6 sub-trees for $L = 3$ and $L = 4$, respectively.

tion [75], starting from the base case $(0,0)$. The final step is to calculate $\pi_s(j)$. It holds that

$$\begin{bmatrix} \pi_0(j) \\ \pi_1(j) \end{bmatrix} = \begin{bmatrix} 1-\xi_{1|0} & \xi_{0|1} \\ \xi_{1|0} & 1-\xi_{0|1} \end{bmatrix}^{\kappa+1} \begin{bmatrix} p_0(\mathcal{P}_j) \\ p_1(\mathcal{P}_j) \end{bmatrix}, \tag{4.10}$$

where κ is the number of packets between the reference frame $\mathcal{P}_j$'s last packet and frame j's first. Starting from frame 1 up to frame N, we can calculate $p_s(j)$ for all frame $1 \leq j \leq N$ using Eqs. (4.8), (4.9) and (4.10). Then, $p(j)$ is simply given by

$$p(j) = p_0(j) + p_1(j).$$

4.4 Evaluations

In this section, we evaluate the performance of the proposed system. We present the maximized mean end-to-end Q-STAR value $\overline{Q}$ and the maximizer eFR f_e and the FEC redundancy rate $r \triangleq 1 - R/R_S$ determined by our scheme for different video sequences, subject to iid and bursty packet losses, considering both hPP and IPP structures. In each evaluation, 10-second-long video sequences "Crew", "City", "Harbour" and "Soccer" are tested [59], while R_S is varied from 100 kbps to 1.6 Mbps with 30 kbps stepsizes. Due to space constraints, we present only the results obtained with "Harbour" and "City" in most cases, please see [76] for all results. For hPP, we also examine the mean FEC redundancy rate $r(l)$ within layer l, given by $r(l) \triangleq \frac{1}{n_l}\sum_{i\in \mathrm{TL}(l)} r_i$, where n_l is the number P-frames that belong to layer l in the intra-period. Clearly, $1 + \sum_{l=1}^{L} n_l = N$.

Encoder settings

We use the x264 encoder with "High" profile, "very fast" preset and tuned for "zero latency" [77], and choose its one-pass ABR rate control algorithm, suitable for ultra-low delay scenarios. For each video, the picture resolution is 4CIF, and the frame rate is selected from $\mathcal{F} = \{15\,\mathrm{Hz}, 30\,\mathrm{Hz}\}$. To generate a particular hPP structure, we modified the x264 encoder by altering the reference frames used before each frame encoding [76] according to the H.264/AVC standard [78]. In our modification, the GoP length is set to $G = 4$, leading to $L = 3$ temporal layers as shown in Fig. 4.2. QP-cascading is performed during encoding. An I-frame is inserted every $T =$

	α_q	α_f	β_q^{hPP}	β_f^{hPP}	β_q^{IPP}	β_f^{IPP}	q_{min}
Crew	4.51	3.09	1.061	0.707	1.064	0.662	22.271
City	7.25	4.10	1.142	0.471	1.247	0.449	18.206
Harbour	9.65	2.83	1.320	0.584	1.461	0.489	34.301
Soccer	9.31	2.23	1.194	0.598	1.196	0.579	20.019

Table 4.1: QSTAR and RSTAR parameters for IPP and hPP with 3 TLs. x264 settings: "High" profile, "very fast" preset, "zero latency" tuning. $f_{\text{max}} = 30\,\text{Hz}$.

16/15 seconds, meaning there are $N = 32$ and $N = 16$ frames in an intra-period for $f_e = 30\,\text{Hz}$ and $f_e = 15\,\text{Hz}$, respectively. To generate an IPP structure, x264 is used without any modification.

Q-STAR and R-STAR parameters

As described above, we have $f_{\text{max}} = 30$ Hz and $R_{\text{max}} = 1.6$ Mbps. The Q-STAR parameters α_q and α_f are taken from [25]. To derive the R-STAR parameters β_q and β_f for either IPP or hPP, we encode each video sequence with the corresponding encoder at target bitrates varied from 100 kbps to 1.6 Mbps with 30 kbps stepsizes, at each $f_e \in \mathcal{F}$. At each target bitrate, the average QP in each frame is obtained and converted[3] to QS, which are then averaged over all frames to determine the mean q. Using these q values, the actual video bitrates R and the f_e used, we derive β_q and β_f via curve-fitting with respect to Eq. 4.3. Finally, the parameter q_{min}, which is used to evaluate *both* IPP and hPP for fair comparison, corresponds to the minimal QS obtained with the IPP structure at $R = R_{\text{max}}$ and $f_e = 30$ Hz. Parameter values are listed in Table 4.1. The corresponding $Q(R)$ curves for both hPP and IPP can be seen in Fig. 4.3.

Modeling the Frame Sizes

As mentioned in Section 4.2, we develop a model for the average frame size $\hat{z}(l, R)$ in each temporal layer l, which depends on the encoding bitrate R. Towards this end, we make use of the encoded video sequences described above. At each target bitrate R, we normalize the size z_i of each P-frame i with respect to the size z_0 of the I-frame

[3] In H.264, $\text{QP} = 4 + 6\log_2(q)$.

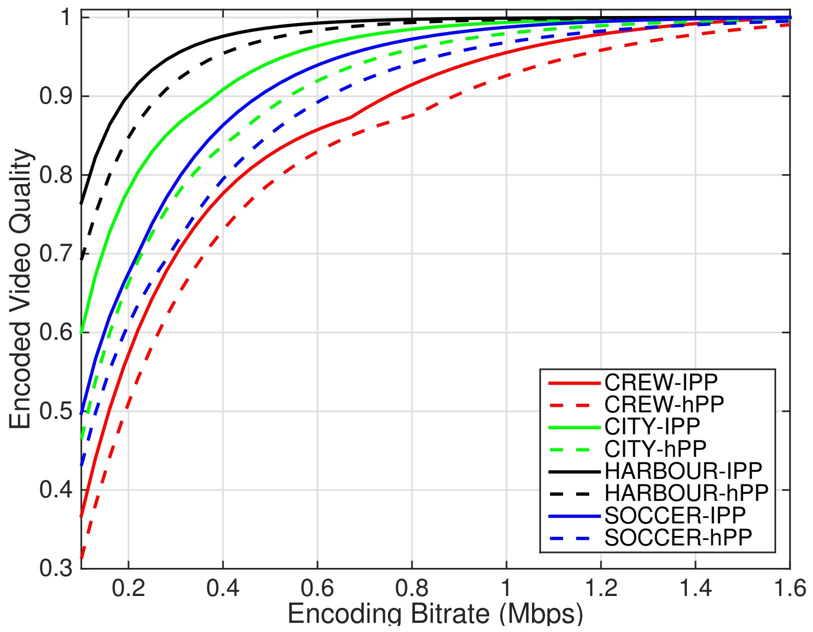

Figure 4.3: Normalized Q-STAR perceptual quality models of video sequences with respect to the bitrate, using IPP (bold) and hPP (dashed) structures.

within the same intra-period, and find the average normalized frame size $\tilde{z}(l, R)$ in each temporal layer l over the full duration of the video. The mean and the coefficient of variation of $\tilde{z}(l, R)$ in each layer l over the target bitrate range $R \in [100, 1600]$ kbps is given in Table 4.2. We can see that, for each video sequence, the normalized mean frame size $\tilde{z}(l)$ in layer l shows little variation with the target bitrate, which means that it can be modeled independent of R.

Next in Fig. 4.4, we plot $\tilde{z}(l)$ against the temporal prediction distance τ_l within each layer l. By examining the general trend of how $\tilde{z}(l)$ changes with the temporal prediction distance τ_l, we propose the following model.

$$\tilde{z}(l) = 1 - e^{-\theta \tau_l^{\eta}} \tag{4.11}$$

The parameters θ and η can be found by curve-fitting in Figure 4.4. Once the normalized mean frame size $\tilde{z}(l)$ is predicted, we can estimate the actual size $\hat{z}(l, R)$ of a layer-l P-frame at a particular bitrate R by distributing the total number of bits RT in the intra-period among the frames proportional to $\tilde{z}(l)$.

$$\hat{z}(l, R) = RT \frac{\tilde{z}(l)}{1 + \sum_{l=1}^{L} n_l \tilde{z}(l)} \tag{4.12}$$

$f_e = 30\,\text{Hz}$	**TL(1)**	**TL(2)**	**TL(3)**
Crew	0.559 / 0.037	0.451 / 0.035	0.361 / 0.031
City	0.444 / 0.063	0.382 / 0.057	0.299 / 0.057
Harbour	0.462 / 0.079	0.402 / 0.100	0.300 / 0.111
Soccer	0.508 / 0.101	0.426 / 0.109	0.326 / 0.098
$f_e = 15\,\text{Hz}$	**TL(1)**	**TL(2)**	**TL(3)**
Crew	0.815 / 0.030	0.733 / 0.025	0.611 / 0.028
City	0.670 / 0.049	0.594 / 0.061	0.508 / 0.057
Harbour	0.730 / 0.036	0.641 / 0.023	0.543 / 0.026
Soccer	0.802 / 0.029	0.733 / 0.039	0.549 / 0.061

Table 4.2: Mean / Coefficients of Variation of $\tilde{z}(l)$ for $l \in \{1, 2, 3\}$, over $R \in [100, 1600]$ kbps

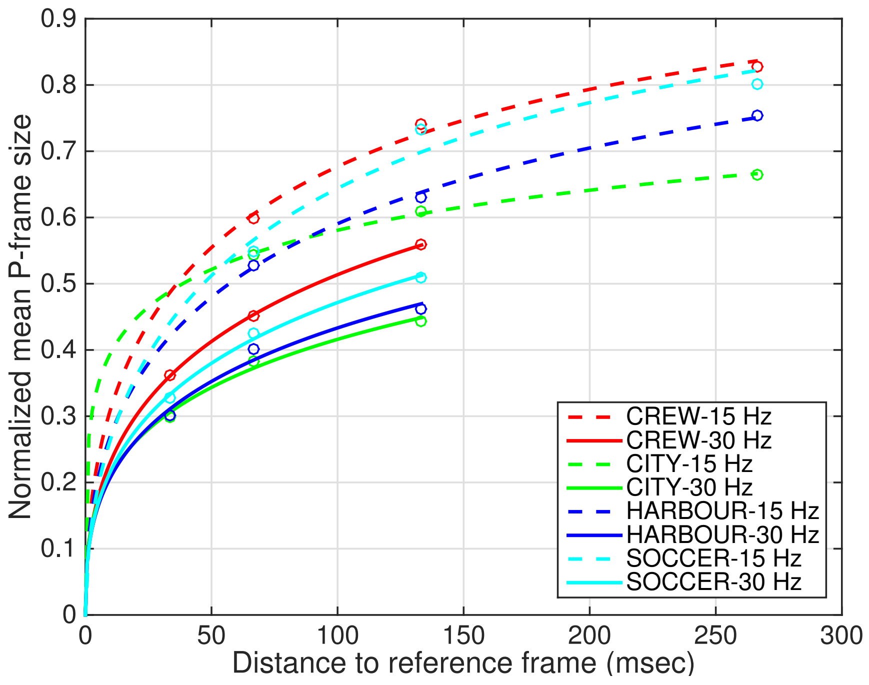

Figure 4.4: Normalized mean frame sizes (circles) and the corresponding fits (bold and dashed lines) with respect to the temporal distance to their reference frames. $L = 3$ and for $f_e = 15$, $\tau_1 = 266.6$ ms, while for $f_e = 30$, $\tau_1 = 133.3$ ms. We have $\tau_1 = 2\tau_2 = 4\tau_3$.

The frame sizes in units of packets is given by $\hat{k}(l, R) = \lceil \hat{z}(l, R)/B \rceil$. In our evaluations, the payload size was chosen to be $B = 200$ bytes.

4.4.1 Evaluations for iid Packet Losses

We begin our evaluations with iid losses, for which we set $\xi_{0|1} + \xi_{1|0} = 1$. Packet loss rate ϵ is varied from 0.05 to 0.2.

Using the hPP structure

In Fig. 4.5, the Q-STAR scores $\overline{Q}_{\text{hPP}}$ that are achieved with the hPP structure are shown for each video. We see that $\overline{Q}_{\text{hPP}}$ drops further as ϵ increases. The relative Q-STAR scores normalized with respect to the lossless case are shown in Fig. 4.6, where the largest percentage-wise quality drop happens at low SBR values. For a given video sequence and the packet loss rate ϵ, the FEC redundancy rate $r_{\text{hPP}}(\epsilon)$ is roughly independent of R_S (Fig. 4.7), and only fluctuates due to the discrete nature of the problem that arises due to packetization. For Harbour, when $R_S \geq 790$ kbps, FEC ensures that the $\overline{Q}_{\text{hPP}}$ is at least 96% of what could be achieved without packet losses (Fig. 4.6), since the perceptual quality model for this sequence is largely insensitive to increases in QS, but sensitive to decreases in dFR. As a result, r_{hPP} is higher for Harbour, trading off the less important amplitude resolution with the more important end-to-end temporal resolution.

From Table 4.3, we observe that the SBR range, on which the preferred eFR is 15 Hz instead of 30 Hz, gets wider as the packet losses get more severe. For Harbour, eFR jumps to 30 Hz at 300 kbps or less; whereas for Crew, it happens after 820 kbps, since the perceptual quality model for Crew is less sensitive to dFR. In Fig. 4.9, we show how the layer redundancy rates $r(l)$ change with R_S for each layer l. The redundancy rates of higher layers are smaller in general, indicating unequal error protection. However, as R_S rises, the redundancy rate for each layer converges on the FEC redundancy rate $r_{\text{hPP}}(\epsilon)$.

Finally, we plot the FEC redundancy rate r against the packet loss rate ϵ in Fig. 4.8. We see that r can be modeled as an affine function of ϵ, for *both* hPP and IPP structures:

$$r(\epsilon) = a \cdot \epsilon + b \tag{4.13}$$

The corresponding model parameters for Eq. (4.13) that are obtained by fitting to

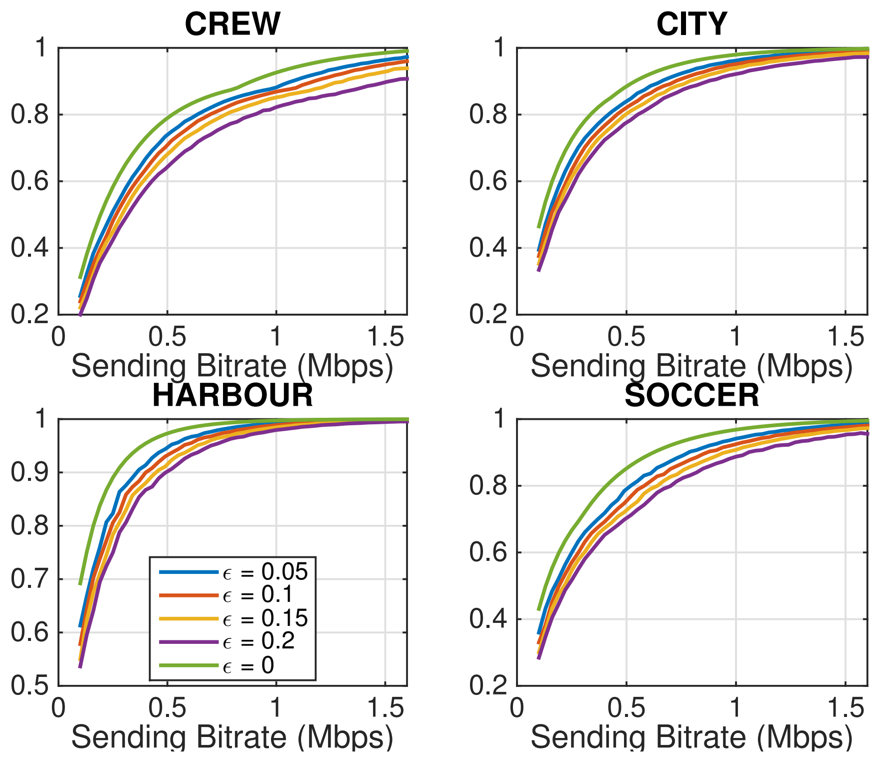

Figure 4.5: Q-STAR scores achieved for iid packet losses, using hPP with 3 TLs.

our evaluation results are given in Table 4.4. A similar affine model has also been observed in [79] for Skype [3], where it was found that $r_{\text{Skype}} = a_{\text{Skype}} \cdot \epsilon + b_{\text{Skype}} = 4.5\epsilon + 15$. Skype's offset parameter b_{Skype} is close to our findings in Table 4.4; still, its multiplicative constant a_{Skype} is much higher, suggesting a room for improvement.

Using the IPP structure

IPP provides higher bitrate efficiency compared to hPP (Fig. 4.3), but lacks the resiliency offered by temporal layering. In terms of the achieved Q-STAR scores, we have $\overline{Q}_{\text{IPP}} > \overline{Q}_{\text{hPP}}$ for all scenarios in Fig. 4.10. Further inspection reveals that the IPP structure delivers similar dFR as hPP, but at the cost of more FEC bits, i.e., $r_{\text{IPP}} > r_{\text{hPP}}$ (Fig. 4.8). Especially at low video bitrates, IPP can have smaller QS due to hPP's relatively high coding overhead, and thus achieves higher $\overline{Q}$.

It may appear that the IPP structure outperforms hPP from the Q-STAR metric perspective. However, it is worth noting that, whenever a frame is lost, the rest of the frames in the intra-period is rendered undecodable in the IPP structure, leading to prolonged frame freezing, whereas the hPP structure is able to continue the stream

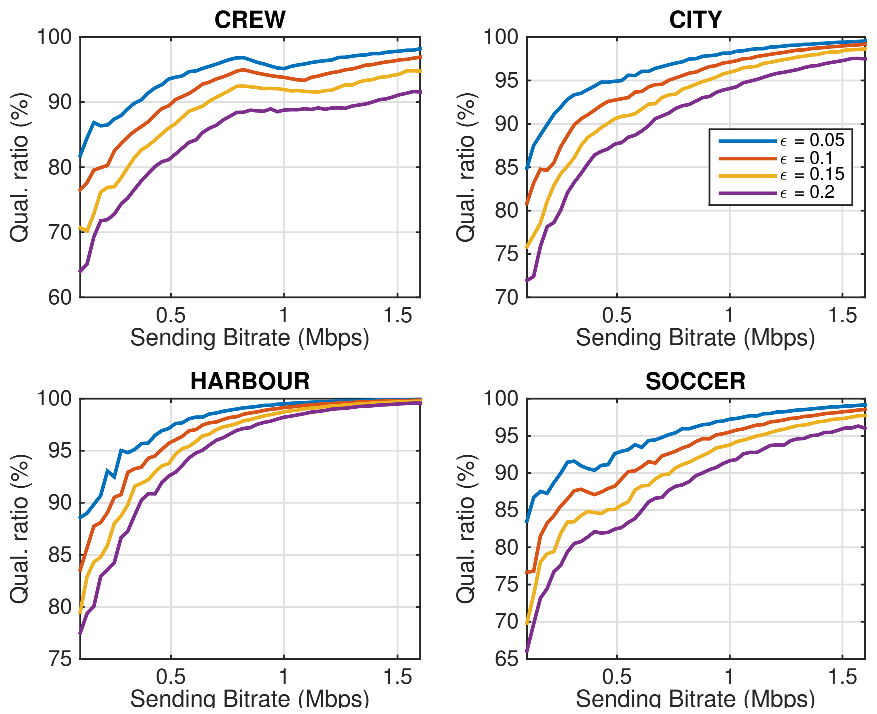

Figure 4.6: Relative Q-STAR scores (%) with respect to the lossless case for iid packet losses, using hPP with 3 TLs.

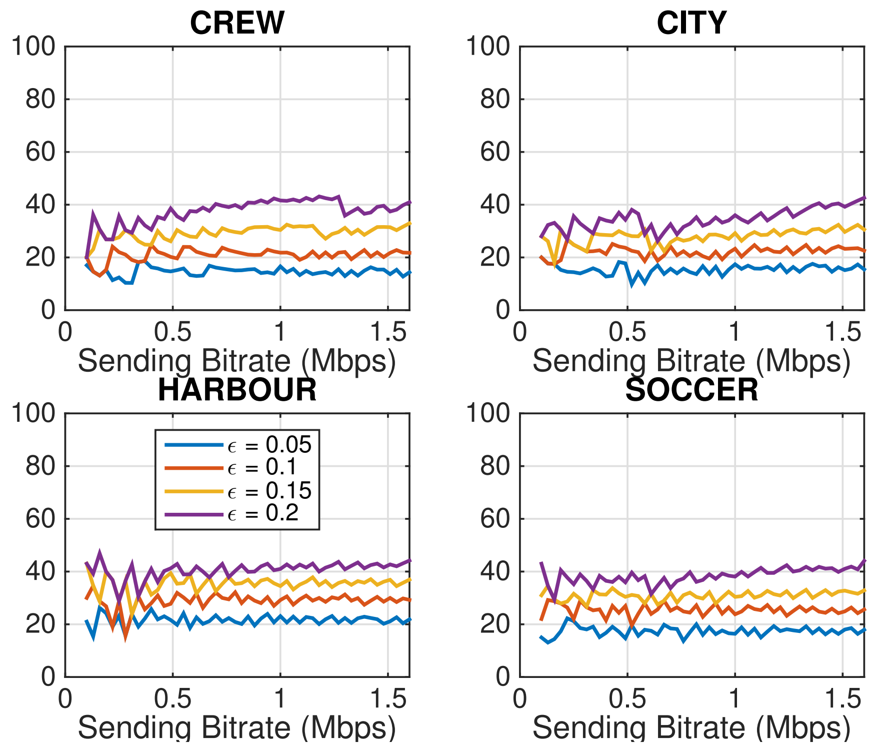

Figure 4.7: FEC redundancy rates (%) for iid packet losses, using hPP with 3 TLs.

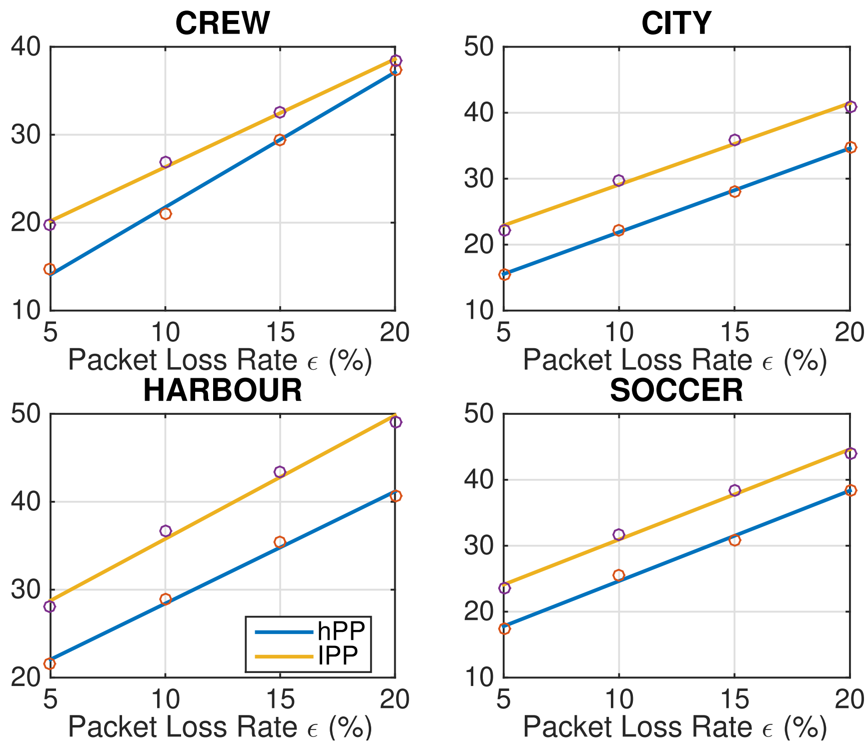

Figure 4.8: FEC redundancy rates (%) for iid packet losses. (hPP with 3 TLs)

hPP	$\epsilon = 0$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$
Crew	0.82	1	1.09	1.18	1.3
City	0.46	0.52	0.55	0.58	0.61
Harbour	0.13	0.16	0.22	0.22	0.25
Soccer	0.31	0.4	0.46	0.52	0.55
IPP	$\epsilon = 0$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$
Crew	0.67	0.94	1.03	1.12	1.21
City	0.37	0.55	0.64	0.7	0.73
Harbour	$\leq$0.1	0.16	0.22	0.25	0.28
Soccer	0.22	0.4	0.46	0.49	0.55

Table 4.3: Sending bitrate (Mbps) at which preferred f_e jumps from 15 Hz to 30 Hz, for iid packet losses. hPP with 3 TLs.

	a_{hPP}	b_{hPP}	a_{IPP}	b_{IPP}
Crew	1.534	6.417	1.288	13.540
City	1.272	9.178	1.207	17.276
Harbour	1.274	15.689	1.320	22.339
Soccer	1.371	10.936	1.294	18.170

Table 4.4: Affine model parameters of the FEC redundancy rate for iid losses

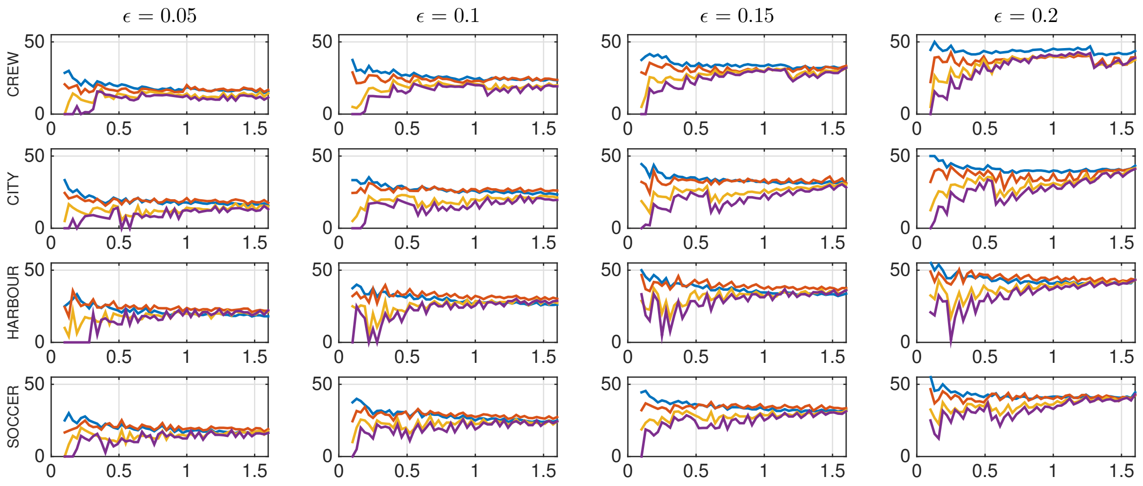

Figure 4.9: Layer redundancy rates (%) for iid packet losses and using hPP structure with 3 TLs, showing I-frame (blue), TL(1) (red), TL(2) (yellow), and TL(3) (purple). Columns correspond to different ϵ values (see top), rows correspond to different video sequences (see left).

with the lower layer frames if the lost frame belongs to enhancement layers. To test this claim, we conduct 10^5 simulations for each (R_S, ϵ) pair and for each coding structure, and we determine[4] the sample mean $\hat{\mu}_\tau$ and the sample standard deviation $\hat{\sigma}_\tau$ of the time-averaged temporal distance τ between the decoded frames for both IPP and hPP. We define τ, for a particular frame decoding pattern, as the weighted average of inter-frame distances, with weights equal to the time fraction that a particular inter-frame distance is observed. As an example, in Fig. 4.1, if P5, P6 and P7 were discarded, then the observed frame distance would be 1 unit in the first half of the intra-period, and 4 units in the rest, assuming the next I-frame is decoded. In this case, the time-averaged frame distance is $\tau = 0.5 \cdot 1 + 0.5 \cdot 4 = 2.5$ units.

[4] μ_x and σ_x denote the mean and the standard deviation of the random variable x, respectively.

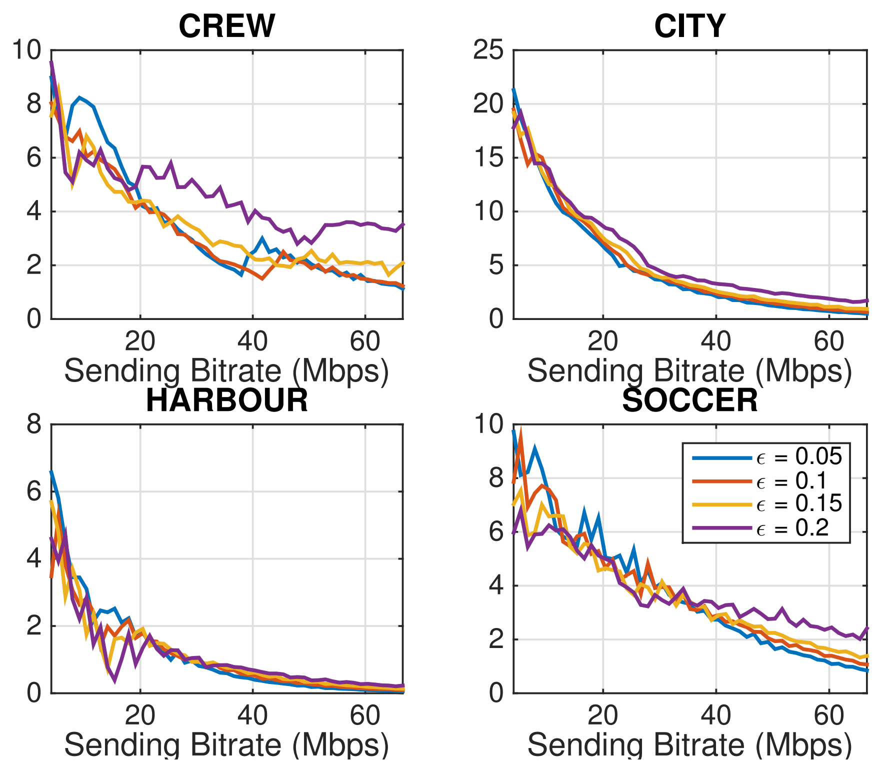

Figure 4.10: Q-STAR reduction (%) relative to IPP, $1 - \overline{Q}_{\text{hPP}}/\overline{Q}_{\text{IPP}}$. For iid losses.

Among distinct frame decoding patterns with the same number of decoded frames, τ is minimized by having minimal frame distance variation.

The differences in the sample mean $\hat{\mu}_\tau^{\text{IPP}} - \hat{\mu}_\tau^{\text{hPP}}$ and the sample standard deviation $\hat{\sigma}_\tau^{\text{IPP}} - \hat{\sigma}_\tau^{\text{hPP}}$ in milliseconds can be seen in Figures 4.11 and 4.12. We can observe that, hPP presents smaller mean and variance in the frame intervals at sending bitrates up to 750 kbps; in other words, the decoded frames can be displayed with more regular intervals in this bitrate range. Ultimately, we conjecture that taking advantage of the temporal layering in hPP is advantageous for small sending bitrates. However, as the sending bitrate grows larger, FEC can be applied more efficiently, and temporal layering loses its edge for protection.

4.4.2 Evaluations for Bursty Packet Losses

For bursty packet losses, we focus on the effects of the average burst length λ at a constant packet loss rate. We set $\epsilon = 0.1$ and try $\lambda = \{2, 5, 10, 50\}$. To investigate the layer redundancies better and make a clear comparison between IPP and hPP,

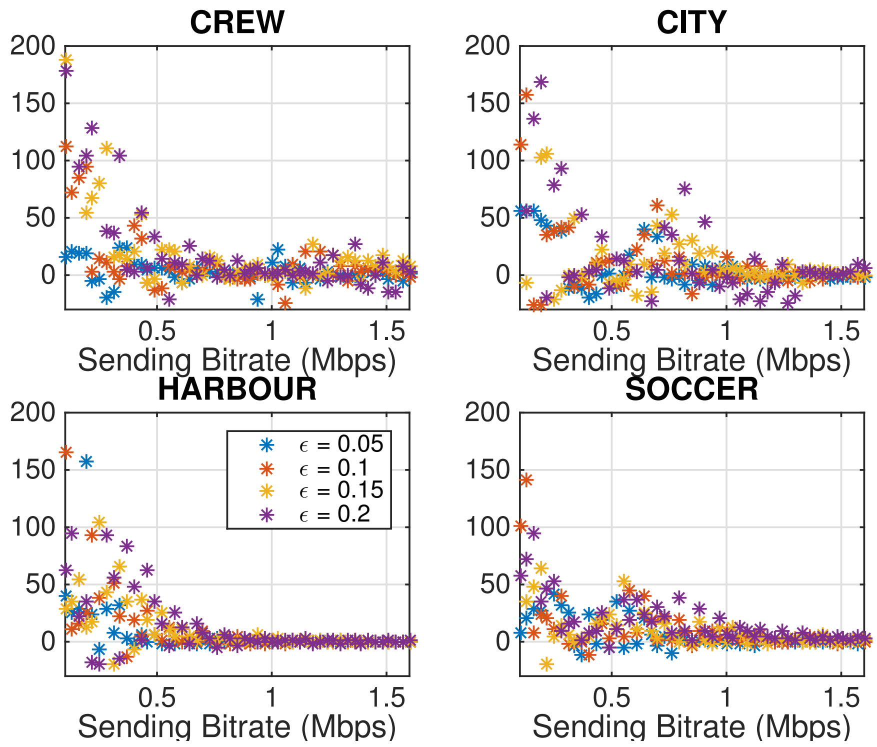

Figure 4.11: Diff. of mean frame intervals $\bar{\tau}_{\text{IPP}} - \bar{\tau}_{\text{hPP}}$ (msec) for iid packet losses

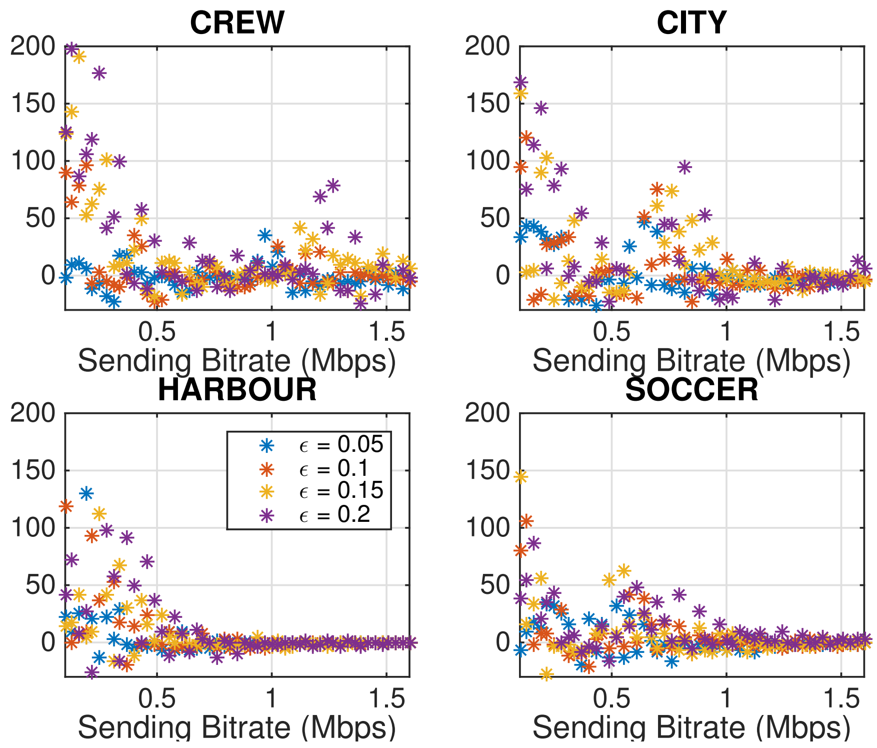

Figure 4.12: Diff. of std. of frame intervals $\sigma_{\tau}^{\text{IPP}} - \sigma_{\tau}^{\text{hPP}}$ (msec) for iid packet losses

we use a constant $f_e = 15\,\text{Hz}$ in all our evaluations.

Using the hPP structure

Figure 4.13 shows that the achieved Q-STAR scores $\overline{Q}_{\text{hPP}}$ drop further as the average burst length λ increases. Unlike the case of iid losses, FEC redundancy rate r_{hPP} varies across the SBR range in Figure 4.14[5]. Particularly, when $\lambda = 50$ packets, r_{hPP} is much smaller for average block lengths up to 25 packets ($R_S \leq 600$ kbps). Clearly, frame losses that occur due to a burst of packet losses are particularly severe when the bursts are long and the FEC code blocks are short. In such cases, preventing frame losses, given that there *is* a packet loss, may require quite large FEC redundancy rates, leaving low bitrate for encoding the video. On the other hand, keeping ϵ constant and increasing λ also increases the average number of consecutive packet arrivals, which equals $1/\xi_{0|1}$. This means that the probability that there *is* a packet loss within an intra-period diminishes, as the average burst length λ gets larger and the sending bitrate R_S gets smaller. As a consequence, the optimal FEC redundancy rate r_{hPP} is small for high λ and low R_S values, but grows with R_S, until the average block length surpasses λ. At this point, our scheme starts to use FEC redundancy rates that are roughly proportional to the average burst length.

Lastly, Figures 4.15 and 4.16 show that the gap between the layer redundancies $r(l)$ get narrower with increasing SBR, and wider with longer bursts. Specifically in Fig. 4.16, we plot the layer redundancies against the average burst length for $R_S = 1600$ kbps. We can see that the FEC redundancies for the less important enhancement layers diminish with the growing burst length. When the bursts are sufficiently long, only the base layer is protected with FEC, and the redundancies reach as high as 80%.

Using the IPP structure

Figure 4.17 shows that, when the packet losses are bursty, the IPP structure loses its advantage if the average burst length is close to the average block length. In the same figure, the horizontal axis is replaced again with the average FEC code block length in packets. When the FEC blocks are significantly shorter compared to the average

[5]In the horizontal axis of this plot, we replaced SBR with the avg. FEC code block length that is proportional to SBR, where each FEC block corresponds to a frame due to frame-level FEC coding.

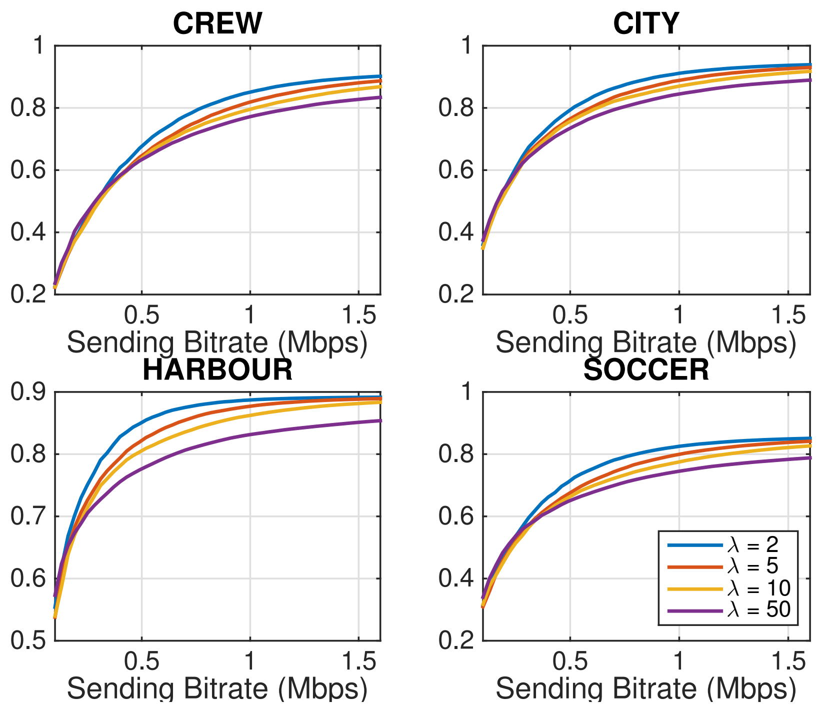

Figure 4.13: Q-STAR scores achieved for bursty packet losses with $\epsilon = 0.1$, using hPP with 3 TLs and $f_e = 15\,\text{Hz}$.

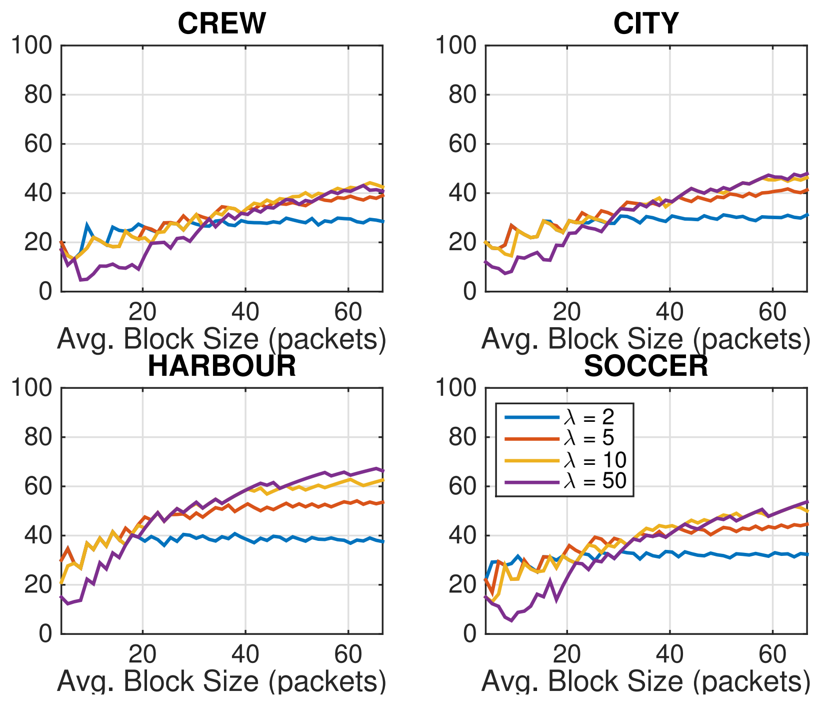

Figure 4.14: FEC redundancy rates (%) for bursty packet losses with $\epsilon = 0.1$, using hPP with 3 TLs. Avg. FEC block size is calculated for each R_S, with $B = 200$ Bytes.

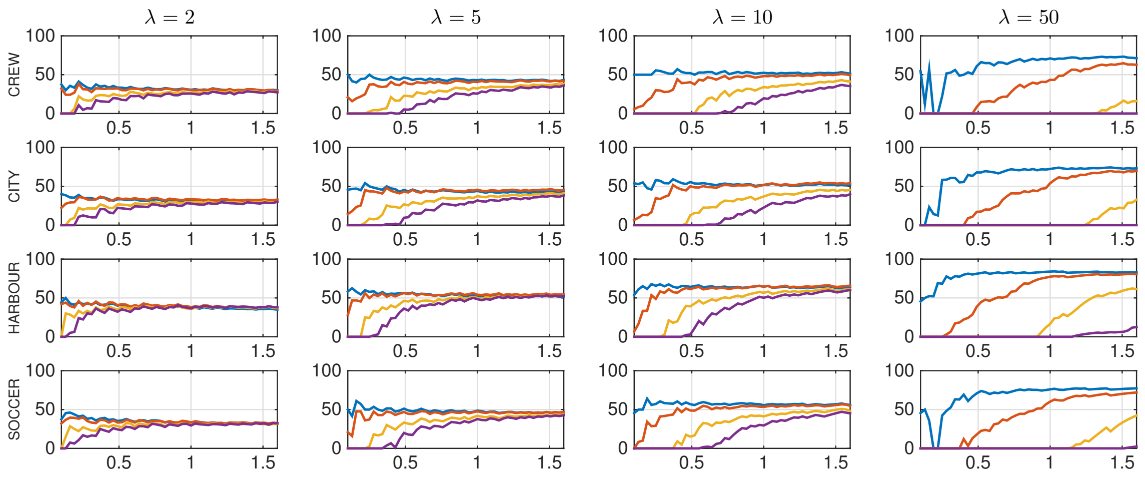

Figure 4.15: Layer redundancy rates (%) for bursty packet losses and using hPP structure with 3 TLs, showing I-frame (blue), TL(1) (red), TL(2) (yellow), and TL(3) (purple). Horizontal axis is SBR (Mbps). Columns correspond to different λ values (top), rows correspond to different video sequences (left).

burst length, many frames are likely to get lost. In this case, neither IPP nor hPP is particularly effective at preventing the frame losses, and IPP gains the upper hand due to the high coding overhead of hPP at low bitrates. As the average block length grows, hPP provides higher dFR than IPP, and thus achieves higher Q-STAR scores despite its coding overhead. This is because, if a frame gets lost, all other frames in the IPP structure are lost as well, while the hPP structure can pick up from the next GoP at the latest, as long as no base layer frame is lost. This hints at a disadvantage of the IPP structure when burst length is close to the average frame size. If the block length is further increased well beyond the burst length, the difference between IPP and hPP in terms of the delivered dFR decreases. This, coupled with the diminishing coding overhead of hPP at high bitrates, results in similar Q-STAR scores in the high SBR regime.

Similar to before, we conduct 10^5 simulations for each (R_S, λ) pair and for each coding structure, to compare the decoded frame distance statistics. We can observe from Figures 4.18 and 4.19 that, hPP presents significantly smaller mean and variance in the frame intervals in the whole SBR range. Ultimately, we conjecture that using the hPP structure is advantageous when the average FEC block length is greater than the average burst length.

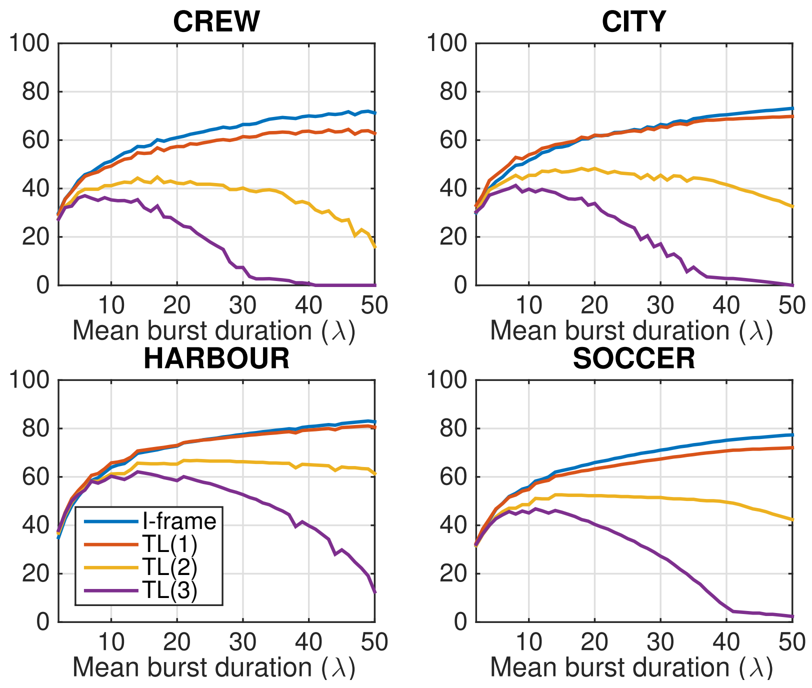

Figure 4.16: Layer redundancy rates (%) for bursty packet losses at $R_S = 1600$ kbps (avg. FEC block size = 67 packets) and using hPP structure with 3 TLs.

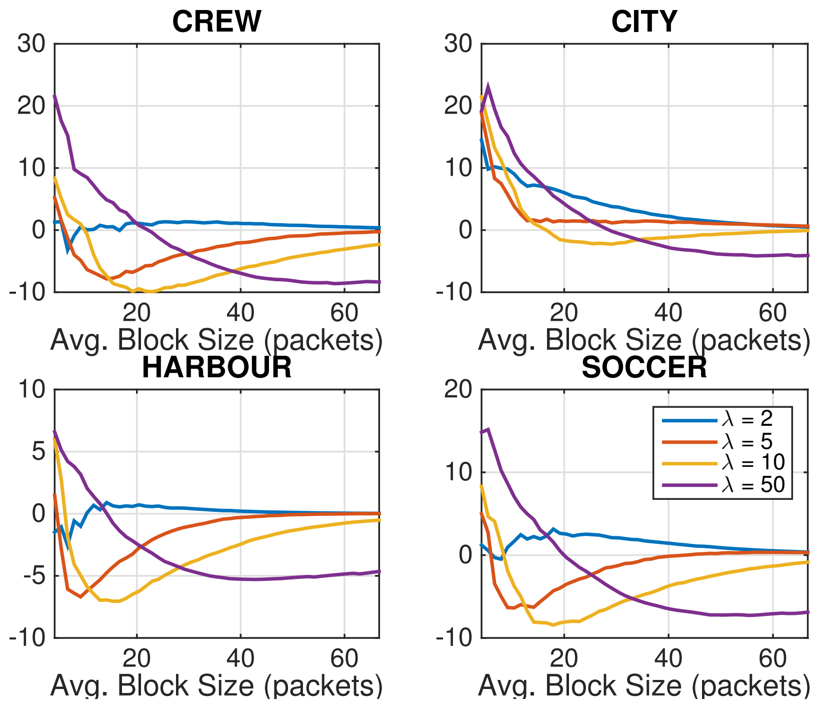

Figure 4.17: Q-STAR reduction (%) relative to IPP, $1-\overline{Q}_{\mathrm{hPP}}/\overline{Q}_{\mathrm{IPP}}$. For bursty losses with $\epsilon = 0.1$. Avg. block size is calculated for each R_S, with $B = 200$ Bytes.

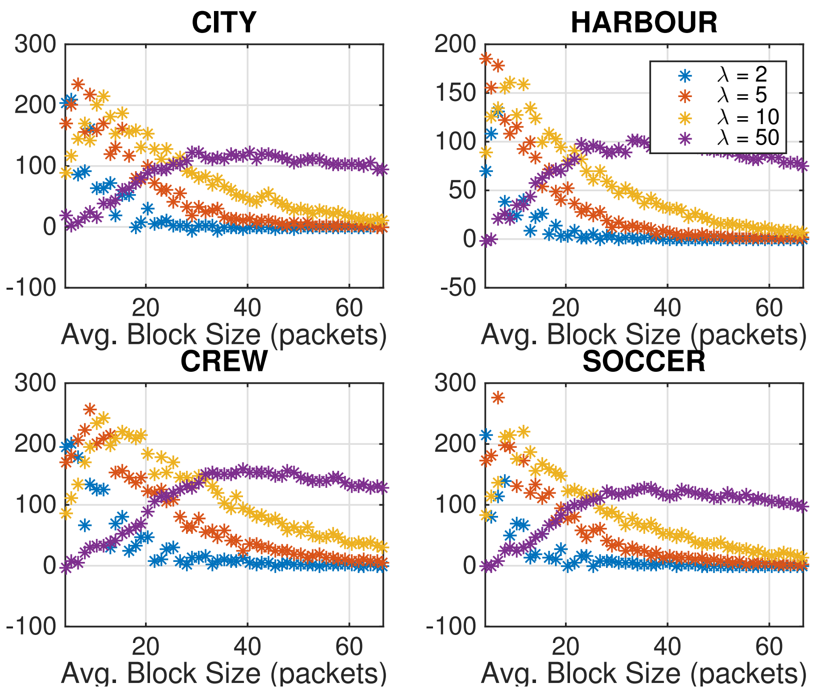

Figure 4.18: Difference of mean frame intervals $\bar{\tau}_{\text{IPP}} - \bar{\tau}_{\text{hPP}}$ (msec) with IPP and hPP for bursty losses with $\epsilon = 0.1$, 10^5 simulations.

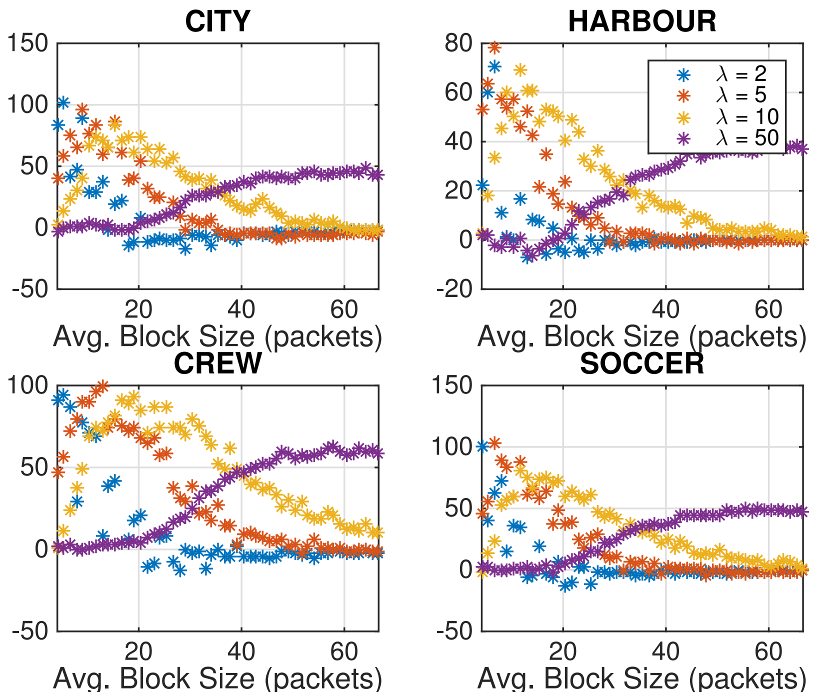

Figure 4.19: Difference of standard deviations of frame intervals $\sigma_\tau^{\text{IPP}} - \sigma_\tau^{\text{hPP}}$ (msec) with IPP and hPP for bursty losses with $\epsilon = 0.1$, 10^5 simulations.

Chapter 5

Conclusions and Future Work

In this thesis, we sought means to improve the quality of experience for the users of real-time video communication systems. Towards this goal, we have identified problems regarding multi-party video conferencing and video calling, and proposed solution methods that are in the application layer.

In Chapter 2, we tackled P2P-MPVC systems, in which, using layered coding is the “go-to” method to deal with peer bandwidth heterogeneity, due to its scalability, reduced complexity with respect to simulcast, and error resiliency. However, it is also well known that layered coders incur bitrate overheads. Alternatively, one can partition receivers of the same source to multiple groups and distribute single-layer video in each one, at the computation cost of multiple encodings. In this study, we have investigated the problem of rate allocation and multicast tree construction in P2P-MPVC systems for the layered and partitioned simulcast systems, under both uplink and downlink capacity constraints, and assuming no packet loss. We have shown that any distribution tree can be reduced to a collection of 1-hop and 2-hop trees, allowing us to consider only such trees in the search for multicast distribution trees, without losing optimality, while constraining the delay to at most 2 hops. Leveraging on this, we have designed a layer assignment heuristic and then determined the corresponding optimal rates for the layered system, assuming the coder can generate a fine granularity scalable stream that can be divided into any number of layers. For the partitioned simulcast approach, we have formulated the optimal receiver partitioning problem and proposed a simple partitioning algorithm that also determines the group rates for each source. Our simulations show that the video rates in both systems are very close to the theoretical bounds, and the quality performance of the partitioned simulcast system is competitive with the “ideal” layered system with no rate overhead in the 4-user case, and is better than the the layered system even if the overhead is only 10% in the 4-user case and 20% in the 6-user case. With 6 users, the partitioning system provides similar performance as the layered system with 10% overhead. Recognizing the significant complexity associated with the multiple encodings required by the partitioned simulcast system, we further evaluated the achievable performance by the partitioned simulcast system when only two receiver groups are allowed. We found that the average received video quality achieved by the partitioned simulcast system with only two groups is quite similar to, and even better than that of the layered system with 10% overhead in the 4-user case and with 20% overhead in the 6-user case. The proposed design of the layered system assumes the quality-rate function of the layered coder is known over the entire rate range, and uses this knowledge to determine the optimal layer rates. In practice, the quality-rate function of a layered coder depends on the desired rates of each layer. To circumvent this chicken-and-egg problem, in our simulations, we generate a bitstream with many thin layers, starting with a very low base layer rate. Such a fine-granularity scalable coder unfortunately has a large rate overhead, limiting the performance of the layered system severely. One possible way to improve the layered system is by considering a layered coder with a limited number of layers (equal or less than the maximum number of receivers) and using a quality-rate function that depends on the rates of all layers. This will however make the solution of optimal layer rate allocation problem much harder, as the optimization problem will no longer be convex. However, heuristic algorithms may be developed to yield approximate solutions. Lastly, the analysis we presented here ignores packet losses, which can severely impact both systems' performance. A real system must explicitly consider how to handle packet losses for either design. We studied the maximization of the received perceptual quality in the presence of packet losses in Chapter 4, the solution of which can be used to extend this chapter's analysis, by refining the achievable quality-rate curves for given packet loss parameters.

In Chapter 3, we have studied video calling over cellular networks, where adaptation to fast-changing network bandwidth and packet delay is of utmost importance. In this chapter, we proposed a new real-time video delivery system, Rebera, designed for cellular networks. Rebera's proactive congestion controller uses the video frames to actively measure the capacity of cellular links, and through these measurements makes a safe forecast for future capacity values, using the well-known adaptive filtering techniques. Through its dynamic frame selection module designed for temporal layered streams, Rebera ensures that its video sending rate never violates the forecast by discarding higher layer frames, thereby preventing self-congestion, and reducing the packet and consequently the frame delays. Our experiments showed that Rebera is able to deliver higher bandwidth utilization and shorter packet and frame delays compared with Apple's FaceTime on average.

In Chapter 4, we have investigated the packet loss resiliency and video quality maximization problems for video calls. Real-time video delivery is prone to packet losses in the network. Achieving minimal latency and satisfactory QoE for such applications under high packet loss rate and long loss burst scenarios requires a joint optimization of video coding methods applied, video resolutions used, and frame-level FEC code rates within individual video frames. In this study, leveraging on prior studies, we have used a perceptual video quality model, Q-STAR, that accounts for the effect of QS and the decoded frame rate separately, where dFR is simply approximated by the number of decodable frames per second. We explored methods to find the encoding frame rate, the optimal video bitrate, and the optimal FEC redundancy rate for each frame, by explicitly considering trade-offs between choosing different encoding frame rates and mean QS values that achieve a target video bitrate. We further considered two predictive encoding structures, namely the hierarchical-P and the more traditional IPP...P. Hierarchical-P enables temporal layering, which enables correct decoding of future frames in case an enhancement layer frame is lost, however, suffers from the coding overhead. IPP...P, on the other hand, presents no overhead, but cannot recover from frame losses until the next intra-period and therefore introduces relatively long frame freezes when used together with the frame-copying error concealment method. Considering the finite-length FEC blocks, we formulated the quality maximization problem as a combinatorial optimization, and solved it using a combination of exhaustive search, hill-climbing and greedy methods. We have shown that, in case of iid losses up to 20%, the optimal FEC bitrate percentage can be approximated as an affine function of the packet loss rate. Furthermore, IPP structure achieves higher Q-STAR scores at all sending rates. This, however, does not necessarily mean that the actual perceptual quality with IPP is higher, because the IPP structure is more likely to lead to uneven frame intervals, reflected by longer mean frame intervals and higher variance of frame intervals at low sending rates. In case of bursty Markovian losses, we show that the IPP structure loses its edge that it gained through its higher video coding efficiency, and hPP is favored when the average FEC block size is greater than the average burst length. Our work can be improved by carrying out a series of extensive subjective video quality tests, the results of which can be used to extend the Q-STAR model so that the precise effect of irregular frame arrivals on the perceived video quality can be established. Furthermore, one can potentially design an online algorithm that determines the encoding frame rate, the target video bitrate and the FEC redundancies, provided that the Q-STAR and R-STAR model parameters are estimated in an online fashion, possibly from certain features in the raw video, as has been done in [25].

Bibliography

[1] A. S. Tanenbaum and D. J. Wetherall, *Computer networks.* Pearson, 2011.

[2] ITU-T, "G.114 - Series G: Transmission Systems and Media, Digital Systems and Networks," *One-way transmission time*, vol. 18, 2000.

[3] Microsoft. Skype. [Online]. Available: https://www.skype.com/en

[4] Google Inc. Hangouts. [Online]. Available: https://hangouts.google.com

[5] ooVoo Homepage. [Online]. Available: http://www.oovoo.com

[6] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video Telephony for End-consumers: Measurement Study of Google+, iChat, and Skype," in *ACM Internet Measurement Conference*, 2012.

[7] "SVC-Based Conferencing Solutions Deployment Guide," Polycom Inc., Tech. Rep., 2014, http://goo.gl/Gtm71c.

[8] A. Eleftheriadis, "SVC and Video Communications," Vidyo Inc., White Paper, 2011, http://goo.gl/A3rxwe.

[9] "Tolly Test Report on AvayaLive^{TM} Video Cloud Service," AvayaLive, Tech. Rep., 2015. [Online]. Available: http://goo.gl/vPUCYL

[10] M. Westerlund and S. Wenger, "RTP Topologies," IETF Network Working Group, Internet-Draft, 2015, https://tools.ietf.org/html/draft-ietf-avtcore-rtp-topologies-update-07#section-3.7.

[11] J. Li, P. Chou, and C. Zhang, "Mutualcast: An efficient mechanism for one-to-many content distribution," in *ACM SIGCOMM Asia Workshop*, 2005.

[12] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, “Utility Maximization in Peer-to-Peer Systems,” in *Proceedings of ACM SIGMETRICS*, vol. 36, 2008.

[13] M. Ponec, S. Sengupta, M. Chen, J. Li, and P. A. Chou, “Optimizing Multi-Rate Peer-to-Peer Video Conferencing Applications,” *IEEE Transactions on Multimedia*, vol. 13, no. 5, pp. 856–868, 2011.

[14] C. Liang, M. Zhao, and Y. Liu, “Optimal Bandwidth Sharing in Multiswarm Multiparty P2P Video Conferencing Systems,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1704–1716, 2011.

[15] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, “Celerity: A Low-Delay Multiparty Conferencing Solution,” in *Proceedings of the ACM Multimedia*, 2011.

[16] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, 2007.

[17] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H. 264/AVC Video Coding Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[18] Apple. FaceTime. [Online]. Available: https://support.apple.com/en-us/HT204380

[19] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, “Profiling Skype Video Calls: Rate Control and Video Quality,” in *Proceedings of IEEE INFOCOM*, 2012.

[20] H. Lundin, S. Holmer, and H. Alvestrand, “A Google Congestion Control Algorithm for Real-Time Communication on the World Wide Web,” IETF Network Working Group, Internet-Draft, 2011.

[21] K. Winstein, A. Sivaraman, and H. Balakrishnan, “Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks,” in *USENIX Symposium on Networked Systems Design and Implementation*, 2013, pp. 459–471.

[22] S. Haykin, *Adaptive Filter Theory*, 5th ed. Pearson, 2014.

[23] D. Hong, M. Horowitz, A. Eleftheriadis, and T. Wiegand, "H.264 Hierarchical-P Coding in the Context of Ultra-low Delay, Low Complexity Applications," in *Picture Coding Symposium*. IEEE, 2010, pp. 146–149.

[24] VideoLAN. x264, the best H.264/AVC encoder. [Online]. Available: http://www.videolan.org/developers/x264.html

[25] Y.-F. Ou, Y. Xue, and Y. Wang, "Q-STAR: A Perceptual Video Quality Model Considering Impact of Spatial, Temporal, and Amplitude Resolutions," *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2473–2486, 2014.

[26] H. Hu, Z. Ma, and Y. Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation," in *Proceedings of IEEE ICIP*. IEEE, 2012.

[27] W. Chen, L. Ma, and C.-C. Shen, "Congestion-Aware MAC Layer Adaptation to Improve Video Telephony over Wi-Fi," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 5s, p. 83, 2016.

[28] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers," *Working draft, W3C*, vol. 91, 2012.

[29] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, and J. Lyu, "Real-time Bandwidth Prediction and Rate Adaptation for Video Calls over Cellular Networks," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016.

[30] L. De Cicco, G. Carlucci, and S. Mascolo, "Experimental Investigation of the Google Congestion Control for Real-time Flows," in *Proceedings of the ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*. ACM, 2013.

[31] Z. Ma, F. C. A. Fernandes, and Y. Wang, "Analytical Rate Model for Compressed Video Considering Impacts of Spatial, Temporal and Amplitude Resolutions," in *ICME Workshops*. IEEE, 2013, pp. 1–6.

[32] X. Xiao, Y. Shi, Y. Gao, and Q. Zhang, "Layer P2P: A New Data Scheduling Approach for Layered Streaming in Heterogeneous Networks," in *Proceedings of IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009.

[33] K.-L. Hua, G.-M. Chiu, H.-K. Pao, and Y.-C. Cheng, "An Efficient Scheduling Algorithm For Scalable Video Streaming Over P2P Networks," *Computer Networks*, vol. 57, no. 14, pp. 2856–2868, 2013.

[34] X. Lan, N. Zheng, J. Xue, X. Wu, and B. Gao, "A Peer-To-Peer Architecture for Efficient Live Scalable Media Streaming on Internet," in *Proceedings of the ACM Multimedia*. ACM, 2007, pp. 783–786.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[36] H.264/AVC Software Coordination. JM Software v19.0. [Online]. Available: http://iphome.hhi.de/suehring/tml/

[37] G. Bjøntegaard, "Calculation Of Average PSNR Differences Between RD Curves," ITU-T, Tech. Rep., 2001.

[38] H. Schwarz and T. Wiegand, "RD Optimized Multi-layer Encoder Control for SVC," in *Proceedings of IEEE ICIP*. IEEE, 2007.

[39] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, 2001.

[40] I. Canadi, P. Barford, and J. Sommers, "Revisiting Broadband Performance," in *Internet Measurement Conference*. ACM, 2012.

[41] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming over the Internet," *IEEE MultiMedia*, no. 4, pp. 62–67, 2011.

[42] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 145–156.

[43] G. Tian and Y. Liu, "Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming," in *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2012.

[44] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM.* ACM, 2014, pp. 187–198.

[45] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication.* ACM, 2015, pp. 325–338.

[46] W. Seo and R. Zimmermann, "Efficient Video Uploading From Mobile Devices In Support Of HTTP Streaming," *ACM Multimedia Systems*, 2012.

[47] G. Tian and Y. Liu, "On Adaptive Http Streaming to Mobile Devices," in *Packet Video Workshop.* IEEE, 2013.

[48] V. Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM computer communication review*, vol. 18, no. 4. ACM, 1988, pp. 314–329.

[49] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End-to-end Congestion Avoidance on a Global Internet," *IEEE JSAC*, vol. 13, no. 8, pp. 1465–1480, 1995.

[50] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking (ToN)*, vol. 14, no. 6, pp. 1246–1259, 2006.

[51] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-speed and Long Distance Networks," in *Proceedings of IEEE INFOCOM*, 2006.

[52] S. Floyd, M. Handley, J. Padhye, and J. Widmer, *Equation-Based Congestion Control For Unicast Applications.* ACM, 2000, vol. 30, no. 4.

[53] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[54] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, And Tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, 2003.

[55] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 145–156, 2005.

[56] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications.* John Wiley & Sons, 2013.

[57] A. Arasu and G. S. Manku, "Approximate Counts And Quantiles Over Sliding Windows," in *Proceedings of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems.* ACM, 2004, pp. 286–296.

[58] Y. Liu, Z. G. Li, and Y. C. Soh, "A Novel Rate Control Scheme for Low Delay Video Communication of H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 1, pp. 68–78, 2007.

[59] C. Montgomery. Xiph.org Test Media. [Online]. Available: https://media.xiph.org/video/derf/

[60] Visicom Media. ManyCam. [Online]. Available: https://manycam.com

[61] NYU Video Lab. Rebera Project Page. [Online]. Available: http://vision.poly.edu/index.html/index.php/HomePage/Rebera

[62] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley & Sons, 2012.

[63] C.-H. Lin, C.-K. Shieh, and W.-S. Hwang, "An Access Point-Based FEC Mechanism for Video Transmission over Wireless LANs," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 195–206, 2013.

[64] C. Hellge, D. Gomez-Barquero, T. Schierl, and T. Wiegand, "Layer-Aware Forward Error Correction for Mobile Broadcast of Layered Media," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 551–562, 2011.

[65] N. M. Freris, C.-H. Hsu, J. P. Singh, and X. Zhu, “Distortion-Aware Scalable Video Streaming to Multinetwork Clients,” *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 469–481, 2013.

[66] 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, 2000.

[67] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, “Streaming Mobile Cloud Gaming Video over TCP with Adaptive Source-FEC Coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 10, 2016.

[68] Y. J. Liang, J. G. Apostolopoulos, and B. Girod, “Analysis of Packet Loss for Compressed Video: Effect of Burst Losses and Correlation Between Error Frames,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 7, pp. 861–874, 2008.

[69] A. Hameed, R. Dai, and B. Balas, “A Decision-Tree-Based Perceptual Video Quality Prediction Model and Its Application in FEC for Wireless Multimedia Communications,” *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 764–774, 2016.

[70] J. Xiao, T. Tillo, C. Lin, and Y. Zhao, “Dynamic Sub-GOP Forward Error Correction Code for Real-time Video Applications,” *IEEE Transactions on Multimedia*, vol. 14, no. 4, pp. 1298–1308, 2012.

[71] E. Maani and A. K. Katsaggelos, “Unequal Error Protection for Robust Streaming of Scalable Video over Packet Lossy Networks,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 3, pp. 407–416, 2010.

[72] H. Wu, M. Claypool, and R. Kinicki, “Adjusting Forward Error Correction with Temporal Scaling for TCP-friendly Streaming MPEG,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 1, no. 4, pp. 315–337, 2005.

[73] C.-H. Shih, C.-I. Kuo, and Y.-K. Chou, “Frame-based Forward Error Correction Using Content-dependent Coding for Video Streaming Applications,” *Computer Networks*, vol. 105, pp. 89–98, 2016.

[74] R. E. Blahut, *Theory and Practice of Error Control Codes.* Addison-Wesley Reading (Ma) etc., 1983, vol. 126.

[75] P. Frossard, "FEC Performance in Multimedia Streaming," *IEEE Communications Letters*, vol. 5, no. 3, pp. 122–124, 2001.

[76] NYU Video Lab. FEC4RT Project Page. [Online]. Available: http://vision.poly.edu/index.html/index.php/HomePage/Fec4rt

[77] L. Merritt and R. Vanam. (2006) x264: A High Performance H.264/AVC Encoder.

[78] I. E. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia.* John Wiley & Sons, 2004.

[79] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Modeling and Analysis of Skype Video Calls: Rate Control and Video Quality," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1446–1457, 2013.

List of Publications

1. E. Kurdoglu, Y. Liu, and Y. Wang, *"Dealing with User Heterogeneity in P2P Multi-party Video Conferencing: Layered Coding Versus Receiver Partitioning"*, in Proc. of Communication and Networking Techniques for Contemporary Video Workshop (in conjunction with INFOCOM), 2014 / Toronto, Canada

2. E. Kurdoglu, Y. Liu, and Y. Wang, *"Dealing with User Heterogeneity in P2P Multi-party Video Conferencing: Layered Distribution Versus Partitioned Simulcast"*, in IEEE Transactions on Multimedia, vol. 18, no. 1, 2016

3. E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, J. Lyu, *"Real-time Bandwidth Prediction and Rate Adaptation for Video Calls over Cellular Networks"*, in Proc. of ACM MMSys, 2016 / Klagenfurt, Austria

4. E. Kurdoglu, Y. Liu, Y. Wang, *"Perceptual Quality Maximization for Video Calls with Packet Losses by Optimizing FEC, Frame Rate and Quantization"*, submitted to IEEE Transactions on Multimedia