SERVICE QUALITY ASSURANCE FOR THE IPTV NETWORKS

A Dissertation
Presented to
The Academic Faculty

by

Aytac Azgin

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
Aug 2013

SERVICE QUALITY ASSURANCE FOR THE IPTV NETWORKS

Approved by:

Professor Yucel Altunbasak, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Constantine Dovrolis
College of Computing
Georgia Institute of Technology

Professor Ghassan Al-Regib, Co-advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mostafa Ammar
College of Computing
Georgia Institute of Technology

Professor Gee-Kung Chang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Raheem Beyah
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 6 May 2013

To my dearest family,

my mother, Kiymet, my father, Erhan, and my sister, Ayca.

Thank you for your endless support.

ACKNOWLEDGEMENTS

I want to first thank my advisor, Prof. Yucel Altunbasak, for his early guidance and long lasting support. I greatly appreciate the experience I gained through working with him. I want to also thank my reading committee members, Prof. Ghassan Al-Regib, Prof. Gee-Kung Chang, Prof. Constantine Dovrolis, and Prof. Henry L. Owen for serving in my committee, and for their valuable feedback and constructive critiques that helped shape my research.

I want to thank Prof. David Hertling and Prof. Bonnie H. Ferri for their tremendous support during my last two years and for giving me the opportunity to further my research studies. I want to also thank Prof. Biing-Hwang Fred Juang, Prof. Chris Rozell, and especially, Prof. James H. McClellan for their valuable guidance during my teaching assistantship, and for helping me recognize the strengths needed to become a great educator.

I want to thank my friends at the Multimedia Computing and Communications Lab, and at the Center for Signal and Image Processing, for their sincere friendship and support. I want to especially thank Ali Cafer Gurbuz, Toygar Akgun, Mehmet Umut Demircin, and Erman Ayday for always being great friends.

I want to thank Robert Stacey, who was my manager at Intel Corporation, for his guidance during my internship and for introducing me to a great environment to conduct valuable research.

Finally, I want to thank my parents and my sister for always being there for me, for their neverending love, support and understanding. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of the proposed research is to design and evaluate end-to-end solutions to support the Quality of Experience (QoE) for the Internet Protocol Television (IPTV) service. IPTV is a system that integrates voice, video, and data delivery into a single Internet Protocol (IP) framework to enable interactive broadcasting services at the subscribers. It promises significant advantages for both service providers and subscribers. For instance, unlike conventional broadcasting systems, IPTV broadcasts will not be restricted by the limited number of channels in the broadcast/radio spectrum. Furthermore, IPTV will provide its subscribers with the opportunity to access and interact with a wide variety of high-quality on-demand video content over the Internet. However, these advantages come at the expense of stricter quality of service (QoS) requirements than traditional Internet applications. Since IPTV is considered as a real-time broadcast service over the Internet, the success of the IPTV service depends on the QoE perceived by the end-users. The characteristics of the video traffic as well as the high-quality requirements of the IPTV broadcast impose strict requirements on transmission delay. IPTV framework has to provide mechanisms to satisfy the stringent delay, jitter, and packet loss requirements of the IPTV service over lossy transmission channels with varying characteristics.

The proposed research focuses on error recovery and channel change latency problems in IPTV networks. Our specific aim is to develop a content delivery framework that integrates content features, IPTV application requirements, and network characteristics in such a way that the network resource utilization can be optimized for the given constraints on the user perceived service quality. To achieve the desired QoE levels, the proposed research focuses on the design of resource optimal server-based

and peer-assisted delivery techniques. First, by analyzing the tradeoffs on the use of proactive and reactive repair techniques, a solution that optimizes the error recovery overhead is proposed. Further analysis on the proposed solution is performed by also focusing on the use of multicast error recovery techniques. By investigating the tradeoffs on the use of network-assisted and client-based channel change solutions, distributed content delivery frameworks are proposed to optimize the error recovery performance. Next, bandwidth and latency tradeoffs associated with the use of concurrent delivery streams to support the IPTV channel change are analyzed, and the results are used to develop a resource-optimal channel change framework that greatly improves the latency performance in the network. For both problems studied in this research, scalability concerns for the IPTV service are addressed by properly integrating peer-based delivery techniques into server-based solutions.

CHAPTER I

INTRODUCTION

Internet Protocol Television (IPTV) is a system that is used to deliver the Internet television services across the Internet Protocol (IP) infrastructure [107, 111]. IPTV integrates voice, video and data delivery into a single IP framework to enable interactive broadcasting services to the subscribers. It promises significant advantages for both service providers and subscribers. Unlike conventional broadcasting systems, IPTV broadcasts will not be restricted by the limited number of channels in the broadcast/radio spectrum. Furthermore, IPTV will provide its subscribers with the opportunity to access and interact with a wide variety of high-quality on-demand video content over the Internet.

However, these advantages come at the expense of stricter quality of service (QoS) requirements than more traditional Internet applications, such as Voice over IP (VoIP) or Video on Demand (VoD) [51, 30]. Since IPTV is considered as a real-time broadcast service over the Internet, the success of the IPTV service depends on the quality of experience (QoE) perceived by the end-users. The IPTV system has to support a wide-range of transmission channel characteristics (i.e., varying packet loss rates and transmission delays) between the content providers and the end-users. The characteristics of video traffic as well as the high-quality requirements of IPTV broadcast impose strict requirements on transmission delay. IPTV framework has to provide mechanisms to satisfy the stringent delay, jitter, and bandwidth requirements over lossy transmission channels.

The objective of the proposed research is to design and evaluate end-to-end solutions for reliable IPTV service. The main focus of the proposed research is on the

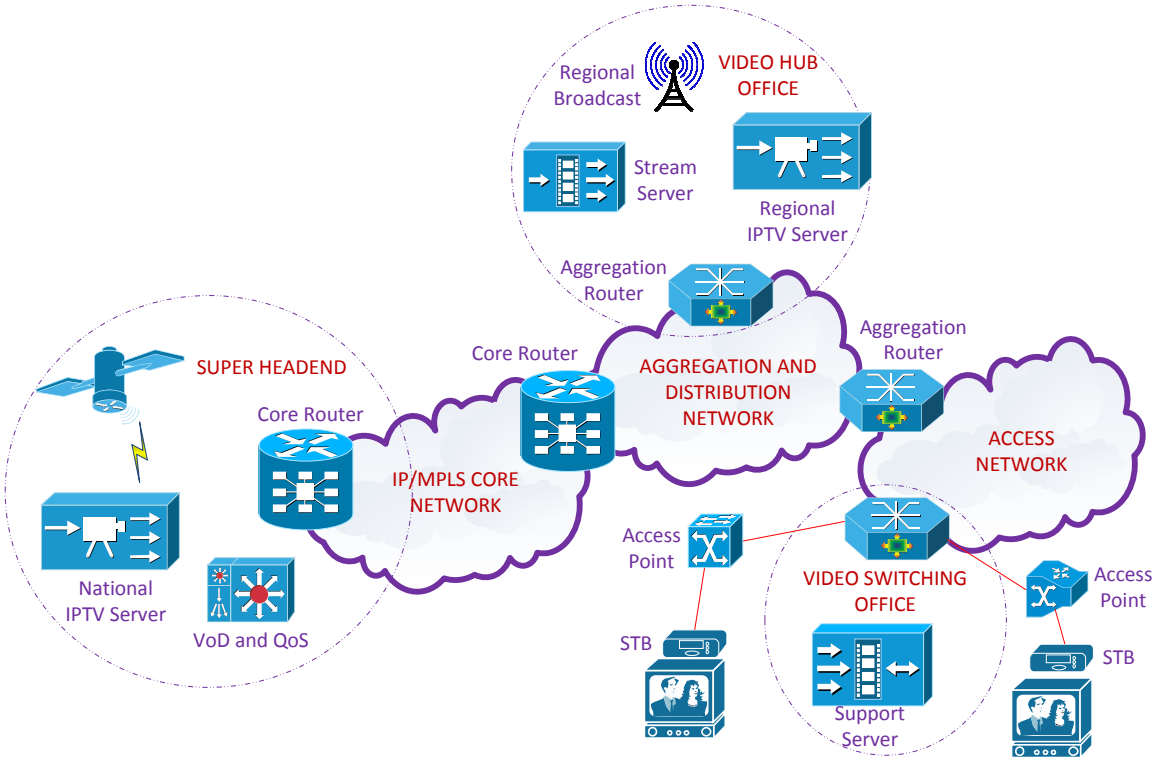


Figure 1: Architectural framework for an IPTV network.

design of (i) effective packet-loss recovery techniques to suppress the negative impact of packet losses on the perceived broadcast quality and (ii) resource-efficient channel change techniques. Our specific aim is to develop an error-recovery framework and a channel change framework that integrate content features, IPTV application requirements, and network characteristics in such a way that the network resource utilization can be optimized for the given constraints on the user-perceived quality. To achieve the desired quality of experience, the proposed research focuses on two main research topics: the design of resource optimal server-based and peer-based content delivery techniques. Within the proposed framework, first the tradeoffs on the use of proactive and reactive repair techniques are analyzed and a solution that optimizes the error recovery overhead is proposed. Further analysis on the proposed solution is performed by also focusing on the use of multicast error recovery techniques. Scalability concerns for the IPTV error recovery are addressed by integrating peer-based

error recovery techniques into the proposed server-based error recovery framework. The impact of channel dynamics on error recovery is investigated by focusing on the use of IPTV over wireless networks. Consequently, we propose resource-optimal peer-based recovery techniques for systems that cannot make efficient use of server-based error recovery. We next focus on the channel change problem in IPTV networks and propose a realistic evaluation framework to analyze the tradeoffs associated with the delivery of concurrent channel change streams. The overreliance of the concurrent delivery techniques on the availability of access network bandwidth leads us to develop a peer-assisted server-based channel change framework that effectively addresses the service quality requirements of the IPTV service during channel change without introducing significant overhead in the network.

The rest of the chapter is organized as follows. In Section 1.1 we present a brief overview of the reliable content delivery techniques and their use within the IPTV framework. In Section 1.2 we examine the channel change problem in IPTV networks, and present an overview of earlier approaches developed to minimize its impact on IPTV service quality. The organization of the thesis and our contributions are stated in Section 1.3.

1.1 Error Recovery Problems in IPTV Networks

Internet Protocol Television (IPTV) is a system to deliver broadcast-quality video content over the Internet Protocol (IP) infrastructure [106, 72]. IPTV is offered by the service providers as part of the triple play service, which consists of voice over IP (VoIP) and Internet services. Since the IPTV service supports linear broadcast, time-shifted broadcast, and video-on-demand services, there is virtually unlimited content that can be delivered to users at anytime. Furthermore, interactive services offered within the IPTV service can significantly enhance the user experience.

However, despite these advantages, IPTV presents the challenge of delivering consistent quality of experience (QoE) to end-users over the packet-switched IP networks [13, 84, 88, 7]. The key issues that significantly affect the user experience are network congestion, link-errors over the access network, slow channel change times, and insufficient feedback. To achieve the desired Visual Quality of Experience (VQE), we need to improve the video quality at the end-users while utilizing network resources as efficiently as possible. However, the expected penetration levels for the IPTV service make this a very challenging task because of scalability-related concerns. Additionally, as the IPTV content is highly sensitive to packet loss, delay, and jitter [37], supporting large-scale distribution of high-bandwidth IPTV streams requires service providers to upgrade their network infrastructure [51, 31] in a way that the level of interruption in the core and distribution networks are minimized. Furthermore, intelligent protocol designs are required to minimize the detrimental impact of packet loss and jitter, to speed up the channel change process, and to monitor the quality of experience [17, 4, 9].

To deliver satisfactory quality of experience, Digital Video Broadcast (DVB) standard suggests a maximum of one video-artifact per two-hour broadcast [1], which corresponds to target packet loss rates on the order of 10^{-6} to 10^{-7}.[1] Considering the transmission link characteristics of DSL networks, which represent the most common broadband technology along the last-mile, achieving the required target packet loss rate becomes a challenging task. Strong signal attenuation and external noises (*i.e.*, background, crosstalk, and impulse noises) have a significant impact on the link quality perceived by the end-users. The noises observed along the last-mile typically generate bit-error rates of 10^{-7}, which translates to packet loss rates on the order of 10^{-3}. The impact of crosstalk noise can be mitigated using coding techniques such

[1] If the errors are random, the upperbound on the packet loss rate is on the order of $10^{-6} - 10^{-7}$. On the other hand, if the errors occur in bursts, the upperbound on the packet loss rate is given by $10^{-5} - 10^{-6}$ [19].

as Trellis Coding [103]. The characteristics of the impulse noise (*e.g.*, non-stationary and state-dependent behaviors), on the other hand, make the statistical analysis difficult [40, 74, 75, 77]. To reduce the impact of impulse noise, cycling redundancy check (CRC) and FEC coding is applied to the incoming data at the ADSL transceiver unit (ATU). The resulting codewords are then interleaved before the Tone Ordering module encodes them into discrete multi-tone (DMT) symbols. As a result, protection can be offered up to a certain number of impulse noise events, which is referred to as the Impulse Noise Protection (INP). If the number of impulse noise events is higher than INP, then the whole interleaved block is lost. On the other hand, to minimize the impact of packet losses caused by network failure events, various technologies can be used such as Multi-protocol Label Switching Traffic Engineering (MPLS-TE) Fast Reroute or Multicast-only Fast Reroute [70, 50, 3, 110]. However, to achieve satisfactory results, we need to complement these network level approaches with the appropriate application level approaches.

In short, the protection offered by the network is limited and in certain cases may not be sufficient to support the end-to-end QoS requirements of the IPTV service. When the IPTV QoS requirements cannot be met by the network infrastructure, IPTV services need to rely on protocols that are offered above the IP layer [69, 30]. Considering the nature of the content delivered to end-users, one possible solution is the use of error concealment techniques [102]. Error concealment makes use of the residual redundancy in the multimedia stream and can achieve satisfactory results in the perceived quality [55]. However, because of concerns on decoding complexity and bandwidth overhead, currently deployed error-control mechanisms are typically based on multicast forward error correction (FEC) and unicast repair techniques (*i.e.*, automatic repeat request, ARQ).

For the multicast FEC service, application-layer forward error correction (AL-FEC) techniques are typically considered [69]. AL-FEC is an end-to-end error control

technique that operates at the application layer, the layer above the IP layer [100]. To enable its operation, IPTV stream is partitioned into source transmission blocks so that erasure coding can be applied on each block separately to create the repair packets necessary to recover from a certain amount of packet loss. There are various erasure codes that can be utilized, *e.g.*, optimal parity check and Reed-Solomon (RS) codes [82], or near-optimal Raptor codes [94]. Among these codes, Reed-Solomon codes can achieve the best recovery performance. However, RS codes are not generally preferred for the IPTV framework due to the complexity required to implement them [69]. Instead, parity check and raptor codes are the more preferred choices to implement the AL-FEC.

To implement AL-FEC within the IPTV framework, multiple approaches can be used. For instance, we can apply AL-FEC directly on the UDP flows, on the RTP packet streams, or within the MPEG-2 TS [2]. DVB specifications provide the option of incorporating AL-FEC on top of RTP or UDP. The solution proposed by the DVB standardization group to implement AL-FEC (see [10]) uses a combination of parity check based Pro-MPEG COP3 code [6] and the Raptor code [94]. Assuming a layered video service is implemented, parity check code is utilized in the base layer, whereas Raptor code is utilized in the enhancement layer. There are typically two approaches to generate the repair packets for the COP3 code, which are referred to as one dimensional and two dimensional approaches. We illustrate these approaches in Figure 2. In the given figure, the source packets, on which the AL-FEC is applied, are the RTP packets. To generate the repair packets, we first interleave the source packets. One-dimensional code is generated by applying parity check on each row or column (for instance by using the exclusive OR operation [18]). For the two-dimensional code, parity check is applied on both the row and the column to generate a total of

[2]IPTV content is typically delivered using the real-time transport protocol (RTP) [5] [3], for which the packets are generated by encapsulating multiple fixed-length (188 bytes) MPEG-2 transport stream (TS) packets [2].

$D + L$ repair packets. However, note that, certain limits are imposed on the use of the D and L parameters by the Pro-MPEG Forum, as stated in [6] [4].

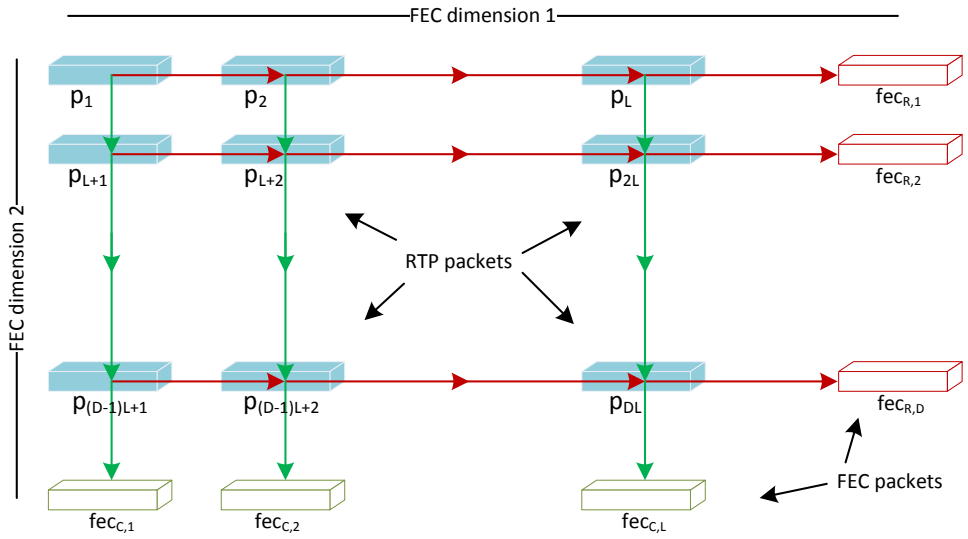


Figure 2: Application layer FEC implementation.

For the unicast repair service, the most common choices are requesting the repair packets from a dedicated retransmission server (server-based repair) [19, 80, 20] or from other users (peer-based repair)[64, 68, 67]. The former approach assumes that the retransmission server receives the source packets through the IPTV multicast and holds on to the received source packets until the timer to keep them in the cache expires. Error recovery process is initiated once the users make a recovery request to the retransmission server, which responds by sending the repair packets using unicast. Peer-based recovery approach, on the other hand, requires the repair packets to be transmitted by the users connected to the same IPTV multicast session. For the wireline networks, both approaches are shown to be effective in quickly recovering from packet losses observed along the last-mile.

To investigate the performance comparisons of multicast FEC and unicast repair techniques, various research studies have been performed. These studies essentially

[4]The limits on the values of L and D are stated as follows: $L \times D < 100$, $1 \leq L \leq 20$, and $4 \leq D \leq 20$.

focus on analyzing the tradeoffs of utilizing FEC vs. ARQ to recover from packet losses [29, 8]. These studies typically focus on the impact of three potential sources for packet loss in DSL networks: stationary noise causing random loss, impulse noise causing bursty loss, and outages. The comparisons between the FEC and ARQ approaches lead to the conclusion of ARQ being the better approach for typical loss scenarios. One reason for that is, compared to ARQ, performance of FEC has been shown to be more dependent on the loss characteristics of the network, especially if the losses are bursty in-nature. On the other hand, for the outage scenarios, it has been shown that achieving the targeted service quality levels significantly increases the error recovery costs for both FEC and ARQ.

Even though the initial IPTV-related studies have focused on the implementation of IPTV over the wireline networks, as the wireless broadband technologies became more accessible, some of the focus has shifted to the development of protocols to support IPTV traffic over the wireless networks [83, 99, 91, 54, 93, 96, 63, 104]. Some of these works focused on the reliability issues observed in a home network [112, 76]. For instance, in [112] the authors address the typical losses observed in a home network and propose modifications to the set-top box (STB) and the home gateway modules to recover from these losses using AL-FEC. In [104], the authors propose a hybrid adaptive FEC and ARQ mechanism to recover from losses observed at the last hop. In [63] the authors propose a hybrid error recovery approach, which integrates application layer FEC with MAC layer ARQ. Many of the other works focus on developing techniques to efficiently deliver the IPTV content over the WiMAX networks [26, 43, 92]. For instance, in [43] the authors propose a cooperative MIMO-based error recovery approach to support error recovery for the WiMAX networks. In [92] the authors propose a cross-layer framework, which is used to overcome multiuser channel diversity in wireless networks. To achieve the stated objective, the authors utilize multiple description coding (MDC) on scalable video bitstreams at the source

together with superposition coding (SCM) on multicast signals at the channel.

1.2 Channel Change Problems in IPTV Networks

In IPTV networks, channel change latency (or zapping delay) is another major concern in achieving the targeted Quality of Experience (QoE) levels at the user side [51, 30]. The reason for that is because, unlike the traditional broadcast systems where the viewers have immediate access to all the available channels locally, in IPTV networks, the users have immediate access to only a limited number of channels locally. [5] As a result, channel change requests oftentimes need to go through the network. As the requests and the IPTV content are delivered over the IP infrastructure, users typically experience longer latency values for the displayed content up to a few seconds, much higher than the acceptable level of ≤ 0.5 seconds [53]. Additionally, due to network dynamics (i.e., time varying characteristics of the connection quality and/or network/user resource availability), perceived latency values at the client side can vary significantly, thereby making it more difficult to achieve the desired service quality levels.

We illustrate the timing for the basis channel change operation in IPTV networks in Figure 3. The channel change process in IPTV networks can be summarized as follows. After a client makes a channel change request, the set-top box (STB) first checks whether the requested content is already available (which can happen, for instance, if the client has already subscribed to the multicast session for the requested channel). If that is the case, STB decodes the available content and starts displaying the channel. If, however, the requested content is not locally available at the STB, then an IGMP leave message is generated by the STB for the currently viewed channel

[5] The number of channels locally available at the client side depends on various factors, such as, average bandwidth requirement for an IPTV session and the downlink bandwidth availability at the access network. For instance, a $18MBps$ downlink connection at the client side can support at most six standard definition streams of $3Mbps$ delivery rate, or two high definition streams of $8Mbps$ delivery rate, or a combination of these two, i.e., one high definition stream and three standard definition streams.

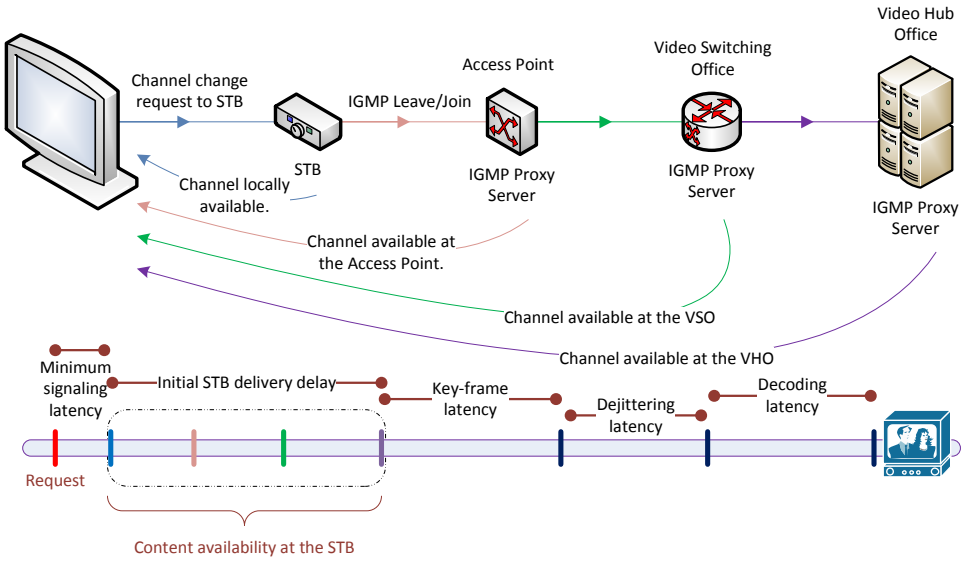


Figure 3: Basis channel change framework.

and an IGMP join message is generated for the requested session. The request message is then forwarded through the residential gateway towards the network until an IGMP proxy server is found that has immediate access to the requested content. Depending on the IGMP reports generated by the downlink servers, multicast tables may need to be updated to ensure that the requested session's multicast data is forwarded towards the client's access network. After the STB starts receiving the content for the targeted session, it first needs to wait for the delivery of the next *intra-coded* key-frame. As soon as the client finishes receiving the key-frame packets, it initiates the buffering operations-by filling the dejittering and the decoding buffers-before the received content can be displayed on the client's audio-visual (AV) equipment.

In general, there are three major factors that contribute to the perceived channel change latency at the client side [20]:

- *The first contributing factor* is the Internet Group Management Protocol (IGMP) signaling latency, which represents the latency caused by the multicast session leave and join events. The signaling latency is typically considered to have minimal impact on the overall channel change latency, since it usually has a value

of $\approx 100ms$ and its value does not show significant variations in time.

- *The second contributing factor* is the key information acquisition latency, which represents the total latency associated with the acquisition of the program specific information (PSI) and the random access point (*i.e.*, key-frame or I-frame). Here, RAP acquisition delay represents the major contributing element to the key information acquisition latency. That is because the average values for the RAP acquisition delay is considered to be within the range of $250ms$-to-$1s$, whereas the average delay to acquire the PSI tables is assumed to be around $100ms$ [95].

- *The third contributing factor* to channel change latency is the buffering latency, which corresponds to waiting times associated with the error recovery, dejittering, and decoding phases. *Error recovery phase* is used for the clients to recover from their losses using proactive FEC or reactive ARQ. Hence, buffering requirements for the error recovery phase are determined based on the user perceived quality along the downlink transmission channel and the level of recovery support provided by the network. *Dejittering phase* is used to smooth out the jitter in the received content, and, in general, has little impact on the channel change latency (especially for the content delivered over wireline transmission channels). Conversely, latency associated with the *decoding phase*, which represents the minimum time required by the decoding buffers to avoid an underflow situation, is as important as the RAP acquisition latency and is the second major contributor to the channel change latency.

In IPTV networks, implementing the basis channel change operation *as is*, without utilizing additional means to reduce the user-perceived latency, is known to cause unsatisfactory quality of experience results [30]. The main reason for that is because

the aggregate value for the RAP acquisition and buffering latencies oftentimes exceeds the minimum acceptable latency threshold. To minimize the occurrence rate of such scenarios, various studies have been performed that aim to improve the overall latency performance in IPTV networks. These studies have so far focused on proposing modifications at the content level (*i.e.*, during the video coding and processing phases), at the client level (*i.e.*, upgrades in the set-top box), and at the network level (*i.e.*, upgrades to the network infrastructure). We next give a detailed overview of these approaches.

1.2.1 Content-adaptive Solutions

Content-adaptive solutions represent the approaches that focus on the video coding aspect of the latency problem. To efficiently support the delivery of the channel change content from the head-end servers, these solutions typically require the use of additional tune-in streams to accompany each of the supported source streams. Tune-in streams are generally used to deliver lower-quality-but which enables quick channel switching-content to the end users. For instance, the architecture proposed in [23] requires the encoder to generate additional lower-quality key-frames, which are then transmitted together with the corresponding source stream. Doing so helps the channel switching clients to promptly start the decoding process for the received content, without waiting for the original key-frame packets to be received from the source stream. To improve the resource utilization efficiency associated with the delivery of the tune-in streams, source stream can be encoded with less number of RAPs [23, 48]. It is also possible to reduce the decoding latency by increasing the transmission frequency of the synchronization frames, when compared to the original GOP transmission scenario [47]. Here synchronization frames correspond to the random access points.

Note that, lower-quality tune-in streams are essentially used to synchronize faster

with the source stream, which normally occurs after the clients receive a full-resolution key-frame from the source stream. However, tune-in streams can also be used to deliver full-resolution key-frames to the channel switching clients [11, 16]. For the given scenario, channel switching client joins the source stream as soon as all the key-frame packets are received from the tune-in stream. Consequently, channel switching clients typically experience a lower-quality viewing period until the next *full-resolution* key-frame is received from the source stream [48, 46]. To minimize the drift problem that may occur during the decoding phase, key-frames transmitted along the tune-in streams can be generated with that objective in mind, for instance, by using the decoded version of the source stream to generate the tune-in streams [46].

The last set of approaches within this category focus on the specific characteristics of the video coding technique that is used to generate the IPTV content. For instance, scalable video coding (SVC) option of the $H.264$ codec can be used to generate, for each session, two data streams, one of which carries the base layer and the other carries the enhancement layer [61, 62]. Here base layer stream is typically used to gain quick, but, low-quality access to the channel change data. Since the bandwidth requirements for the base-layer streams are much smaller than that of the originally encoded source stream, channel switching client can subscribe to a number of base layer streams corresponding to, for instance, the most accessed channels or the channels adjacent to the previously viewed channel. In doing so, channel change latency for the subsequent requests can be significantly reduced at the cost of lower quality viewing during the given surfing period [6]. Upon settling on the last accessed channel, channel switching client can join the enhancement layer stream associated with the last selected session to achieve the highest quality of experience. Similar performance improvements in channel change latency can be achieved by integrating the gradual decoder refresh

[6]Surfing period refers to the period, during which the client makes frequent channel change requests.

option in *H.*264 into the tune-in streams, with minimal quality loss observed during the switch from the tune-in stream to the source stream [45].

1.2.2 Network-assisted Solutions

Network-assisted solutions represent the approaches that require the use of additional streams to directly support the channel change process at the network infrastructure. These streams are typically created at two distinct locations in the network: (*i*) at the head-end servers located at the Video Hub Office (VHO), or (*ii*) at the dedicated servers installed near the access network (*e.g.*, at the Video Switching Office (VSO)).

For instance, in [49] the authors propose to combine tune-in streams, which are created at the head-end servers, with in-advance delivery of popular channels to the edge network to reduce the channel change latency. To meet the service quality requirements, channel change parameters for the given approach, that is, the set of IPTV channels delivered to the edge routers and the key-frame transmission rate for the tune-in streams, are dynamically determined based on the tradeoffs between network utilization and channel change latency.

On the other hand, the approach proposed in [30] focuses on the use of dedicated servers installed near the edge network to support the channel change process. Specifically, the dedicated server is used to implement a unicast-based burst transmission strategy to deliver accelerated high-rate data bursts to the channel switching clients to allow these clients to quickly fill up their STB buffers and initiate their decoding phases. A similar approach is proposed in [20], where the authors integrate the channel change mechanism into a previously developed service provider-driven error control framework to deliver better quality of experience to the channel switching clients [19]. Note that, these approaches typically deliver unicast bursts to channel switching clients until they fully synchronize with their corresponding source streams. As a direct consequence of this, average servicing time for the channel change requests

can be quite significant (close to 10 seconds or more). Hence, the amount of data that needs to be delivered per request can easily become overwhelming for the server, especially during globally shared surfing periods (or commercials).

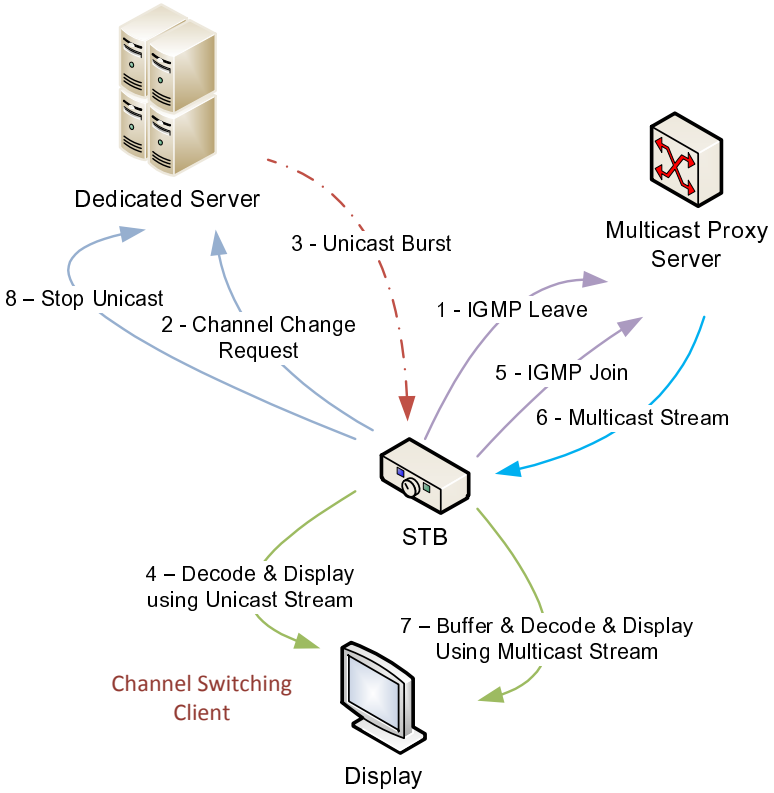


Figure 4: Channel change system that uses unicast bursts.

To address the scalability problems that may arise with the use of unicast-based delivery techniques and minimize the overhead associated with the delivery of channel change data to the clients, a few studies have adopted the multicast-based transmission strategy. For instance, in [21] dedicated server is used to multicast time-shifted replicas of the source stream to minimize the channel change latency. These channel change streams are delivered at a rate that is higher than the source multicast rate to enable for faster convergence to the source streams. In doing so, overhead associated with the delivery of these additional streams can be significantly reduced, as the server stops the delivery of inactive streams after the clients converge to the source streams. In [86] to also reduce the buffering latency the authors propose a similar

approach, which uses higher rate tune-in streams that deliver constantly shifted and reordered version of the source stream to the clients. However, since the ordering rule does not take into account the type of packets delivered along the tune-in stream, performance improvements strictly depend on the time of request. To minimize the performance loss caused by this limitation, in [85] a new ordering rule is proposed for the tune-in stream by taking into account the type of each packet transmitted along the source stream. In doing so, latency performance can be further improved (e.g., by an additional $250ms$) when compared to the approach proposed in [86].

1.2.3 Client-side Solutions

Client side solutions represent the approaches that aim to limit the system updates/upgrades to the client side, thereby allowing them to be implemented on a wider scale, potentially independent and irrespective of the support offered by the IPTV service provider. These approaches typically require the clients to prejoin a selective set of channels concurrently during the channel switching process; if the targeted session is one of the pre-joined channels, then channel change latency can be significantly reduced.

For instance, the solution proposed in [25], which is considered as one of the earliest studies on this topic, uses an adjacent-channel join approach, where the clients join to channels that are adjacent to the previously selected channel during the channel change process. If the received channel switching request targets one of the adjacent channels, then the client can immediately start watching the selected channel without experiencing additional network latency. However, if the adjacent channels do not represent the optimal set of sessions for the clients to prejoin (*i.e.*, selections are oftentimes a miss rather than a hit), then the overhead associated with the delivery of these streams can become a major concern. To limit the disruptive impact of inaccurate channel selection decisions, the set of prejoined channels can be determined

by using the channel popularity information aggregated at a dedicated rating server [60]. The accuracy of the channel switching decisions can be further improved by integrating the clients' remote control behaviors and personal preferences into the decision process [52]. However, in general, it is sufficient to use the information on channel access frequencies and viewing durations to estimate the set of channels delivered to the clients [73].

Note that, an important limiting factor for the client-side solutions is the bandwidth required to receive and deliver extra channels to the clients. If the number of IPTV clients connected to the same access point is sufficiently high, then the resulting overhead associated with the channel change process can be overwhelming. To limit this overhead, a finite-duration multichannel delivery approach can be used to limit the duration for the clients to join multiple sessions simultaneously. For instance, in [98], clients are required to join two-to-three additional multicast streams, each of which corresponds to an adjacent channel, after each channel change request. For the given approach, clients stop receiving the prejoined sessions after a predetermined idle period, during which no channel change request is made. The finite-duration multichannel delivery approach is shown to provide significant savings in the channel change overhead. This approach is further extended in [108] to also cover for the random channel switching scenarios. Note that, since finite-duration multichannel delivery approaches require the clients to join additional streams after the first channel switching request is made (*i.e.*, the request that starts a surfing period), latency perceived during the first channel change event does not change. To address this problem, in [58] the authors propose a predictive tuning approach with special emphasis on the latency experienced during the first channel switching event. To optimize the overall latency performance, clients are required to prejoin different numbers of channels during the surfing and the viewing periods. The optimal number of channels to prejoin is determined using a Semi-Markov based analysis. Resource usage efficiency

for the given framework can be further improved by incorporating the button pushing preferences into channel switching behavior analysis.

The last set of approaches considered in this category focus on channel reordering schemes. Specifically, by clustering channels with high access rates, channel switching performance can be indirectly improved by reducing the number of channel switching requests required to reach the targeted session for typical button-pushing activities (*i.e.*, up and down) [59]. This approach can be further improved by using a circular channel reordering scheme that interleaves popular channels in alternating-upward and downward-directions to evenly distribute the accessing load in either directions [79].

1.3 Organization and Contributions of the Thesis

The rest of the thesis is organized in chapters, each of which discusses a separate research problem. Introduction section within each chapter presents the motivation for the stated problem and discusses the related work. Proposed architectures and the results on the experimental and the theoretical studies are presented within the chapter bodies. Discussions on the findings for the proposed research are presented within the conclusion section of each chapter. The architectures proposed to address the error recovery problem in IPTV networks are discussed in Chapters 2-5, and the architectures proposed to address the channel change problem in IPTV networks are discussed in Chapters 6-7. The outline of the thesis and our contributions are summarized as follows.

In Chapter 2 we propose a novel performance evaluation framework for IPTV networks to develop resource efficient error control techniques. Specifically, the proposed framework is used to analyze the performance tradeoffs associated with the delivery of repair packets using proactive and reactive error recovery techniques. Then, using the basic characteristics of the proposed framework, a practical server-based error

recovery protocol is proposed to achieve the optimal tradeoffs in IPTV error recovery.

In Chapter 3 we address the error recovery problem for the correlated packet loss scenarios. For that purpose, we develop novel spatial loss correlation models that are applicable to IPTV networks. We then design an error recovery framework that is capable of exploiting the spatial correlation characteristics of the network. Finally, using the proposed packet loss models and recovery protocols, performance tradeoffs associated with the use of unicast- and multicast-based error recovery strategies are analyzed for correlated packet loss scenarios in IPTV networks.

In Chapter 4 we investigate the use of peer-based error recovery strategies within a server-based error recovery framework to address the scalability concerns in IPTV networks. For that purpose, a novel server-assisted peer-based error recovery solution is proposed to minimize the probability of the Error Recovery Server entering a non-responsive state by pushing the error recovery load towards the end-users whenever possible. The information on the perceived network characteristics is used to make resource-optimal decisions to find the recovery-peers. During the decision process, latency and fairness requirements are simultaneously evaluated to avoid overutilizing the resources at the peers and the Error Recovery Server.

In Chapter 5 we focus on the reliable delivery of IPTV content over wireless access networks and propose a novel resource-efficient cooperative multiple-input single-output (MISO) technique to deliver multicast IPTV content in WiMAX networks. To support the strict service quality requirements for the IPTV service within a cooperative diversity-driven framework, a two-phased transmission strategy is utilized which consists of a multicast phase (to deliver from the base station to the clients) and a cooperation phase (for delivery within the active client set). To maximize the session throughput for a given set of clients, an adaptive rate selection technique is proposed within the given cooperative recovery framework.

In Chapter 6 we focus on the channel change problem in IPTV networks and propose a realistic analytical framework to evaluate the performance of concurrent stream delivery-based (CSD) channel switching techniques in IPTV networks. Results from the latest statistical research on user distributions and channel switching activities in IPTV networks are utilized to develop the proposed framework and improve the accuracy of performance evaluations for CSD-based channel switching techniques.

In Chapter 7 we address the scalability-related concerns in server-based channel change frameworks by proposing a novel user-assisted server-based channel change framework. The proposed solution integrates the capabilities of a dedicated channel change server with that of the IPTV subscribers to create a resource-efficient and scalable channel change framework.

CHAPTER II

A GENERALIZED HYBRID FEC/ARQ-BASED ERROR RECOVERY FRAMEWORK FOR IPTV NETWORKS

2.1 Introduction

The success of the IPTV broadcast depends on the quality of experience (QoE) perceived by the end-users [84, 88, 51, 13]. Since IPTV is a real-time broadcast service, it is often associated with stringent quality of service (QoS) requirements. The general consensus on the minimum acceptable QoS level can be stated as *at most one perceivable error during a two-to-four hour broadcast*. Because of these strict quality requirements, strong error recovery techniques are required to protect the content that is delivered to end-users and recover from the occasional packet losses observed at different sections of the network. To guarantee the required protection levels, proactive (FEC-based) and/or reactive (ARQ-based) error recovery techniques are typically utilized [29, 69, 19].

Figure 5 depicts a general error-recovery framework, which consists of proactive forward error correction (FEC) coding and reactive repair services. In proactive FEC coding, the level of protection provided by the FEC code is selected to ensure that a substantial portion of packet losses caused by the inherent characteristics of the network (*e.g.*, Repetitive Electrical Impulse Noise (REIN) for the DSL networks) can be recovered. In reactive repair service, as illustrated in Figure 5, each unrecoverable packet loss triggers a repair request from the set-top box (STB) to the Error Recovery Server (ERServ). Reactive repair service can be utilized together with the proactive FEC coding to assist proactive error correction or without any proactive FEC coding.

We can state the characteristics of the proactive recovery as follows. Here, repair

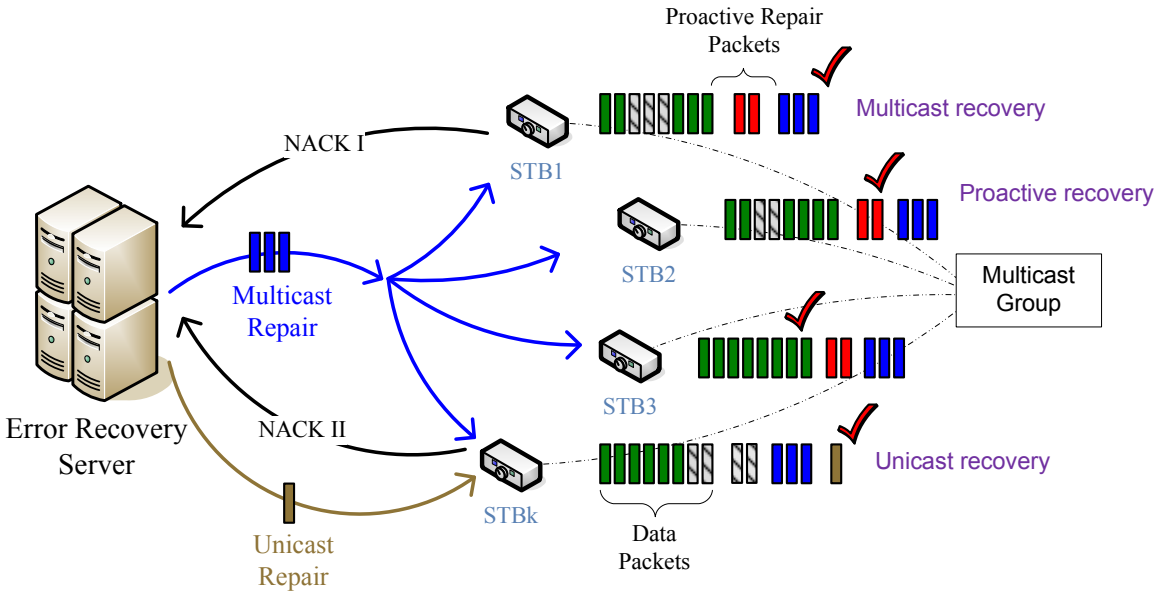


Figure 5: IPTV error control techniques.

packets are transmitted together with the source packets but over different multicast sessions. In doing so, multi-layer error protection services can be offered to any user requesting the service. The protection level for proactive repair is typically determined at the beginning of a session and preserved throughout the session. If a single multicast session is used for the delivery of proactive repair packets, then recovery strength is selected based on a highly conservative estimate of the packet loss rate, which may lead to inefficient bandwidth utilization.

Reactive repair service, on the other hand, is initiated by the end-user after a packet loss is observed. End users initiate the reactive recovery process by sending a request packet to the Error Recovery Server. After receiving a repair packet request, the Error Recovery Server responds to the request by sending a unicast repair packet to the user making the request. Reactive repair service significantly reduces the error recovery overhead, since repair packets are only transmitted after a packet loss is observed. However, effectiveness of the reactive recovery process is limited for two main reasons. First, since reactive recovery service is initiated after a packet loss is observed, round-trip-time (RTT) between an end-user and the Error Recovery

Server becomes a critical measure of success. Because of the time-varying network characteristics, there is a non-zero probability of failing to deliver the repair packet on-time. Second, since the Error Recovery Server is expected to serve thousands of users, the probability that the server becomes overloaded with too many requests has a non-negligible probability. During such periods, incoming requests experience significant delays at the Error Recovery Server, which in turn may result in late deliveries for many users. Consequently, these users would suffer from occasional quality degradations. From a quality of experience point of view, neither of these cases is acceptable.

If we compare reactive and proactive error recovery approaches, we observe that each approach is advantageous in certain scenarios and disadvantageous in others. For instance, if users observe high packet loss rates, a proactive approach would be preferred over a reactive approach, and vice versa. In short, network characteristics play a major role in finding an optimal solution. However, since network dynamics create a highly complicated and non-stationary system view at any given point in time, neither approach alone would be sufficient to reach an optimal solution. Furthermore, the rate at which the changes occur may be so fast that the system cannot take immediate action to reach optimal performance. As a result of these changes, sudden performance degradations may be observed by the end-users.

To make the IPTV system react to changes in the network state in a timely manner and preserve the end-user service quality, we need to utilize both error recovery approaches at the same time. However, utilizing both approaches at the same time without introducing limits to their usage would lead to over-utilization of the system resources. Furthermore, as the number of users increases, scalability problems may arise. In short, to effectively combat the time-varying packet-loss dynamics, it becomes necessary to implement a hybrid error recovery service, where the constraints on visual quality and resource utilization are simultaneously evaluated. In this study,

our objective is to develop a hybrid error recovery architecture that can be utilized to evaluate the performance tradeoffs associated with the use of the IPTV service. We present a detailed analysis of the proposed error recovery framework, which can be used to achieve effective and resource efficient error control in IPTV networks.

The rest of the chapter is organized as follows. In Section 2.2 we present our system model. In Section 2.3 we develop the equations necessary to analyze the performance of the proposed framework. In Section 2.4 we state the optimization problem for the given system. In Section 2.5 we present a practical error recovery solution that makes use of the proposed framework. In Section 2.6 we present the simulation results for the proposed framework. Section 2.7 concludes the chapter.

2.2 System Model

We use the IPTV system architecture shown in Figure 6 for our analysis. In the given system, an Error Recovery Server (ERServ) is placed between the Headend Server (HeServ) and the end-users. HeServ is responsible for the broadcast of source video packets towards the end-users and the ERServ, the latter of which caches the received source packets until the deadline to keep them expires. To serve all the users in the network, multiple error recovery servers are typically utilized, each of which targets a distinct set of users. The process to distribute the end-users to multiple ERServs is out of the scope of this study, hence we will omit its discussion. Throughout the rest of the chapter, we will therefore focus on the operation of a single ERServ.

Proactive error control relies on the use of application-layer FEC (AL-FEC), which uses interleaved parity coding on a group of video source packets (which we refer to as the transmission block) to create the FEC packets. This process is typically achieved on either one or two dimensions. Generated AL-FEC packets are assumed to be delivered to end-users error free (for instance, by transmitting them on a different transmission block).

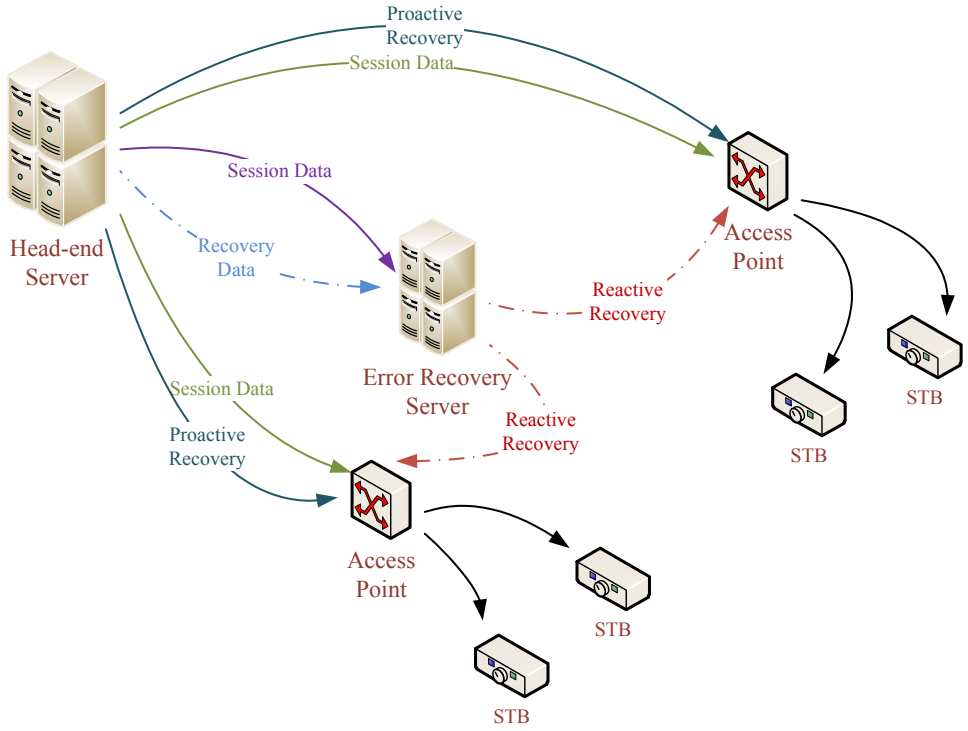


Figure 6: IPTV error recovery framework.

Reactive error recovery process initiates after an end-user decides that a source packet is lost (or cannot be recoverable). The end-user observing the loss sends a repair packet request directly to the ERServ. If more than one packet is lost within the decision period, then the end-user sends a cumulative repair packet request for all the packets lost within that group. Each request at the ERServ is processed on a first-come-first-serve basis. After the request is processed, ERServ makes the decision on how to respond to the repair packet request (*i.e.*, whether to respond, when and how to respond, etc.). In our study, we assume that the ERServ responds to each of the received repair packet requests immediately, and we assume the use of a unicast-based source packet retransmission approach at the ERServ.

To model the packet loss process we use the following approach. We assume the packet losses to be bursty, and to account for the varying situations that corresponds to differing burst-length scenarios, we consider the use of a Markov Modulated Inhomogeneous Poisson Process (MM-IPP) with K-states, where the states represent the

varying burst lengths. In the current study, we focus on a single-type burst length to develop our model. Furthermore, to have a better understanding of the underlying techniques, we will restrict the arrivals within each state to errors affecting a single packet. The arrival rates within each state will then be adjusted accordingly to emulate the worst-case scenarios for the bursty packet loss process.

In the next section, we present an in-depth analysis of the proposed error recovery framework.

2.3 Analysis of the Error Recovery Framework

The objective, here, is to minimize the error-recovery overhead for the IPTV networks under a given set of visual quality constraints. Our main focus then becomes to find the resource optimal proactive and reactive protection parameters for all users connected to the same Error Recovery Server. We use the parameter χ to represent the error-recovery overhead. The visual quality constraint refers to the minimum end-user experience required to satisfy the quality demands of an end-user. The parameter Υ is used to represent the upperbound on the visual-quality loss performance.

We state the optimization problem as follows:

$$\min \chi \ s.t. \ \Theta \leq \Upsilon \tag{1}$$

where Θ represents the visual quality-loss performance observed during the transmission period of a given IPTV multicast session.

We assume that the quality-loss metric at each user can be analyzed independently. Therefore, Θ can be expressed as a combination of the individual quality loss metrics, *i.e.*, $\theta(\nu) \leq \Upsilon$, $\forall \nu \in U$, where U represents the set of end-users and $\theta(\nu)$ represents the amount of quality degradation observed by the user ν during an active session.

The error recovery process requires the source data to be partitioned into transmission blocks. For the proactive error recovery process, each of these blocks is evaluated separately and independent of the other transmission blocks. One consequence of this

assumption is that unrecoverable packet losses observed in a transmission block has no impact on the visual quality performance of another transmission block. [1] Based on these assumptions, we can restate the optimization problem as follows:

$$\theta(\nu) = \sum_{\forall \pi \in \Pi} \theta(\nu, \pi) \leq \Upsilon, \forall \nu \in U \tag{2}$$

where $\theta(\nu, \pi)$ represents the quality loss observed by the user ν during the transmission block π, and Π represents the set of transmission blocks in a given session, $\mathfrak{S}$.

To represent the visual quality loss performance, we use the unrecoverable packet loss rate. The assumption on the inter-block independence regarding the visual quality loss performance allows us to focus on the quality performance within a single transmission block. If we take the expectation of both sides in (2), we achieve the desired form:

$$E[\theta(\nu, \pi)] \leq \frac{\Upsilon}{|\Pi|} \tag{3}$$

where the operator $|\cdot|$ represents the size of a given set.

The parameter Υ is used to represent the number of unrecoverable packet losses within a given period, *i.e.*, $\ell(\top)$ losses within $\top$ time units. If the parameter $\overline{\pi}$ represents the average duration of a transmission block, then the right hand side of (3) becomes:

$$\Upsilon = \frac{|\Pi| \times \ell(\overline{\pi}) \times \top}{\top_C} \tag{4}$$

where $\ell(\overline{\pi})$ represents the upperbound on the number of unrecoverable packet losses within a single transmission block and $\top_C$ represents the duration of an active session.

To determine the unrecoverable packet loss count within a transmission block, we can compare the delivery times of each data packet with the respective decoding

[1] Note that, in reality, depending on the location of the packet losses observed within the GOP structure, losses can have a propagative impact. This is especially true for the I-frame and the P-frame packets. Therefore, the results presented in this chapter represent an upperbound on the user perceived performance.

deadlines:

$$\phi(\nu, \pi) = |\pi| - \sum_{p_i \in \pi} I(T_A^{\nu}(p_i) \leq \tau_{\pi}^{\nu}) \tag{5}$$

where $|\pi|$ represents the total number of packets transmitted for a given transmission block π, $T_A^{\nu}(p_i)$ represents delivery time for p_i to ν, τ_{π}^{ν} represents the decoding deadline for p_i, and $I(\epsilon)$ represents the indicator function that is used to check the occurrence of an event (*i.e.*, $I(\epsilon)$ is equal to 1 if the event occurred, and 0 otherwise).

To average the impact of random events that take place within each transmission block, we take the expectation of the indicator function, which results in the following approximation:

$$E\left[I(T_A^{\nu}(p_i) \leq \tau_{\pi}^{\nu})\right] = P(T_A(\nu, p_i) \leq \tau_{\pi}^{\nu}) \tag{6}$$

The righthand side of (6) represents the cumulative distribution function (cdf) corresponding to the packet arrival times. We use the function $F_{T_A}^{\nu,p_i}(t)$ to represent the distribution of packet arrival times at the user ν. To determine the equation for this distribution function, we need to analyze all possible scenarios that lead to a successful packet delivery. Specifically, we consider the successful delivery of both the original source packet transmissions from the HeServ and the repair packet transmissions from the ERServ. In short, we use the following equation to represent the function $F_{T_A}^{\nu,p_i}(t)$:

$$F_{T_A}^{\nu,p_i}(\tau_{\pi}^{\nu}) = P_{S0}(\nu, p_i) + P_{F0}(\nu, p_i) \times P_C(\nu, p_i) \tag{7}$$

where, $P_{S0}(\nu, p_i)$ represents the probability of successfully delivering the source packet during the initial transmission from the head-end server, $P_{F0}(\nu, p_i)$ represents the probability of failing the initial delivery, and $P_C(\nu, p_i)$ represents the probability of successfully recovering from the failed, initial delivery.

To solve the equation for $F_{T_A}^{\nu,p_i}(\tau_{\pi}^{\nu})$, we follow a three-step methodology. [2]

[2]Hereafter, we will simplify the notation for $F_{T_A}^{\nu,p_i}(\tau_{\pi}^{\nu})$ and use $P_s(\nu, p_i)$ instead.

- *In the first step*, we find the initial success and failure probabilities, which is essentially based on the error function associated with the end-to-end delivery path.

- *In the second step*, we find the probability of recovering from a packet loss by treating each packet equally for the reactive recovery process (*i.e.*, we assume all packets have the same reactive recovery limit).

- *In the last step*, we combine the impact of packet delivery times and latency measures to update the equation for the reactive recovery limit; and we do this for each packet separately.

Note that, the third step is where the impact of early packet losses is integrated into the proposed calculations. In the next section, we present the methodology that is used to develop the equations required to establish a resource-optimal error recovery protocol.

2.3.1 Finding the Initial Success and Failure Probabilities:

To find $P_{S0}(\nu, p_i)$, we separate the end-to-end network into two sections: the core network and the access network. We initially focus on the packet loss events occurring in the access network. To find the error function corresponding to the access network, we need to analyze the nature of packet loss events observed along the last-mile. As DSL technology is one of the most widely used broadband access technologies, we focus on the characteristics of the DSL networks to determine the error function.

Specifically, we consider a bursty packet-loss scenario, where the burst length varies based on a memoryless discrete distribution function. We represent this distribution function with σ, *i.e.*, $N_B = \left(\sum j\sigma[j]\right)$, where N_B represents the average burst length. The arrival process for the burst-loss case is based on an Inhomogeneous Poisson Process (IPP). In our study, we analyze the impact of packet loss events by

focusing on a single transmission block. Additionally, we analyze each packet loss event independent of the previous or the future loss events. To enable the proposed analysis, we transform the IPP-based bursty packet-loss arrival process (BLP) to an IPP-based single packet-loss arrival process. For that purpose, we approximate the bursty packet-loss process using a two-state Markov modulated IPP (MM-IPP).

The arrival rate for the single packet-loss IPP is represented with $\lambda(t)$. We discretize the observation period into Λ intervals, within which we use a homogeneous arrival rate, *i.e.*, $\lambda(t) = \lambda_S(i)$, where $T_i \leq t < T_{i+1}$, $0 \leq i < \Lambda$, and $\lambda_S(i)$ represents the poisson arrival rate for the individual packet loss events within the interval $[T_i, T_{i+1})$.

To find these rates, we equate the average number of packet losses observed for both cases (single-loss and bursty-loss) within a given timeframe:

$$\sum_{j=0}^{\beta} \lambda_S(j) \times \left[T_{j+1} + \lceil t - T_{j+1} \rceil^+ - T_j\right] = \widetilde{N}_B(t) \times \overline{\lambda}_B(t) \tag{8}$$

where $\overline{\lambda}_B[t]$ represents the accumulative arrival rate for the BLP within the interval $(0, t)$, $\widetilde{N}_B(t)$ represents the weighted average of the packet loss count during a burst error period under the given boundary conditions, and β is equal to $\max_j[T_j \leq t]$.

$\widetilde{N}_B(t)$ is found by using $\sum_{\forall j} \widetilde{N}_B^j(t) \times \sigma[j]$, where $\widetilde{N}_B^j(t)$ represents the expected number of packets to be affected by a burst error and it is given by:

$$= \begin{cases} \left[\displaystyle\sum_{l=1}^{k^-(t)} \frac{\lambda_B[l\tau_p]}{\overline{\lambda}_B[t]/(k^+(t)-l)} + \frac{\lambda_B[t] \times k^+(t)}{\overline{\lambda}_B[t]}\right] & \text{if } t \leq T_B(j) \\ \dfrac{\overline{\lambda}_B[t - T_B(j)]}{\overline{\lambda}_B[t]/j} + \displaystyle\sum_{l=1}^{j} \frac{\lambda_B[T_B(j) + l\tau_p]}{\overline{\lambda}_B[t]/(j-l)} & \text{otherwise} \end{cases} \tag{9}$$

where τ_p represents the single packet transmission time, $T_B(j)$ represents the duration of a burst error, which affects j packets consecutively, and $\overline{\lambda}_B[t]$ is the burst arrival rate within the packet transmission period that includes t (*i.e.*, $\lambda_B[t] = \int_{l\tau_p}^{t} \lambda_B(s)ds$, where $l\tau_p \leq t \leq (l+1)\tau_p$). The parameters k^+ and k^- are equal to $\lfloor t/\tau_p \rfloor$ and $\max\{k^-(t) + 1, \lceil t/\tau_p \rceil\}$.

Equation (9) makes use of the fact that, for a poisson process, within an observation interval for which an event has occurred, the location of the event is uniformly distributed. Using this information, we divide the observation interval into two non-overlapping periods, allowing us to find the average packet loss count for the given burst.

The next step in the proposed transformation process is to use the Markov modulation on the resulting arrival rates so as to emulate the distribution of packet losses within a given transmission block. For that purpose, we multiply the initially selected poisson arrival rates with the compaction ratio used to establish the Markov states. In doing so, we can limit the deviations corresponding to the probabilistic nature of events observed for the high-packet loss scenarios, while preserving the average packet loss arrival rates.

Consequently, the error function becomes equal to $\left(1 - e^{-\int_{T_{i-1}}^{T_i} \lambda_S(t)dt}\right)$. This function is mostly useful in finding the packet loss count within a given timeframe. For the other evaluation scenarios, utilizing a parameter that represents the steady state value for the packet loss rate would be more useful. To find this parameter, we approximate the BLP using an $M/G/1$ queue. For the given queueing model, the arrival rate is equal to the burst packet loss rate, $\lambda_B(t)$, and the service time is equal to the duration of a single burst. We use Ξ_E to represent the expected length of a burst-loss period and Ξ_V to represent the variance for the burst-length distribution. Then, using the steady state equations for the resulting $M/G/1$ queueing system, we can find the fraction of time that the system is in a burst-error state (γ) using $\gamma = \overline{B}/(\overline{B} + T/\overline{\lambda}_B[T])$, where $\overline{\lambda}_B[T]$ represents the accumulative arrival rate for the BLP within the interval $(0, T)$ and $\overline{B}$ represents the average duration for the burst-error state, which can be found by using $\Xi_E/(1 - (\overline{\lambda}_B[T]/T)\Xi_E)$.

We use $\gamma(\nu)$ to represent the probability of a user ν observing a packet loss. Note that, $\gamma(\nu)$ actually represents the probability that the system is in a burst-error state

at a random point in time. Combining these equations, we can express the equation for the modified error function corresponding to a packet p_i, $e_L(\nu, p_i)$, as follows:

$$e_L(\nu, p_i) = 1 - \frac{T}{\overline{\lambda}_B[T]} \left(\sum_{\forall j} j \times \tau_p \times \sigma[j] \right)^{-1} \tag{10}$$

which leads to the equation of $P_{S0}(\nu, p_i) = 1 - \gamma(\nu)$.

2.3.2 Finding the Successful Recovery Probability:

To find the probability of successfully recovering from a packet loss, $P_C(\nu, p_i)$, we use the following equation:

$$\begin{aligned} P_C(\nu, p_i) &= P_{FEC}(\nu, p_i) + P_{NFEC}(\nu, p_i) \\ &\quad \times \sum_{k=1}^{\kappa_R(\nu, p_i)} R_k(\nu, p_i) \end{aligned} \tag{11}$$

where $P_{FEC}(\nu, p_i)$ refers to the probability of recovering from the packet loss using FEC, $P_{NFEC}(\nu, p_i)$ refers to the probability of failing to recover from the packet loss using FEC (*i.e.*, $P_{NFEC}(\nu, p_i) = 1 - P_{FEC}(\nu, p_i)$), $\kappa_R(\nu, p_i)$ refers to the maximum number of times a user ν is allowed to request a repair packet for p_i [3], and $R_k(\nu, p_i)$ refers to the success probability for the reactive recovery process, which can be expressed using the following equation:

$$R_k(\nu, p_i) = \begin{cases} P_{S1}(\nu, p_i) & \text{if } k = 1 \\ \prod_{j=1}^{k-1} P_{Fj}(\nu, p_i) \times P_{Sk}(\nu, p_i) & \text{if } k > 1 \end{cases} \tag{12}$$

Here $P_{Sk}(\nu, p_i)$ represents the probability of successfully delivering the kth retransmission of p_i to ν and $P_{Fk}(\nu, p_i)$ refers to the probability of failing to deliver the kth retransmission of p_i to ν (*i.e.*, $P_{Sk}(\nu, p_i) = 1 - P_{Fk}(\nu, p_i)$).

[3] The maximum number of retransmissions is found by using (i) the limits imposed on the operation of the Error Recovery Server (*e.g.*, the upperbound on the retransmission requests that can be accepted for a particular user at the ERServ) and (ii) the restrictions imposed by the link between the user and the ERServ (*e.g.*, end-to-end delay).

We can find the equation for $P_{FEC}(\nu, p_i)$ by recursively checking the state of delivery for the previously transmitted packets, which results in the following equation:

$$P_{FEC}(\nu, p_i) = \begin{cases} 1 & \text{if } 0 < i \leq \kappa_P(\nu) \\ 1 - \sum_{j=\kappa_P(\nu)}^{i-1} e_{\nu,j} \times P_{\nu,j}^{(u)}(\kappa_P(\nu) - 1) \times \Omega_{fec}^{(1)}(\nu, p_j) & \text{if } \kappa_P(\nu) < i \leq |\pi| \end{cases} \tag{13}$$

where $\kappa_P(\nu)$ represents the proactive recovery strength, $\Omega_{fec}^{(1)}(\nu, p_j)$ represents the impact of non-distinct packet losses[4], $P_{\nu,i}^{(u)}(\kappa_P(\nu) - 1)$ represents the probability of $\{\kappa_P(\nu) - 1\}$ unrestricted packet losses [5] happening before p_i is transmitted by the access point, and $e_{\nu,j}$ is equal to $1 - e^{-\int_{T_{j-1}}^{T_j} \lambda_S(t)dt}$.

The approach that is used to determine the equation for $\Omega_{fec}^{(1)}(\nu, p_j)$ is presented in the Appendix. We can state the equation for $P_{\nu,i}^{(u)}(\kappa_P(\nu) - 1)$ as follows:

$$P_{\nu,i}^{(u)}(\kappa_P(\nu) - 1) = \frac{e^{-\int_0^{T_{i-1}} \lambda_S(t)dt} \times \Big[(\kappa_P(\nu) - 1)!\Big]^{-1}}{\left[\int_0^{T_{i-1}} \lambda_S(t)dt\right]^{1-\kappa_P(\nu)}} \tag{14}$$

To illustrate the importance of these equations, we created a simple packet loss scenario where the loss events are generated using the beta distribution with parameters ($\alpha = 1, \beta = 5$). We focus on an interval of $100ms$, which represents the size of a transmission block. We use $5ms$ as the packet interarrival time (τ_p) (*i.e.*, a transmission block consists of 20 packets). For the simulations, we vary the packet loss rate (γ). In Figure 7, we show the results for $\gamma = 0.01$, and in Figure 8, we show the results for $\gamma = 0.05$. For each of these cases, we varied the FEC strength using values taken from the interval [1,5]. We observe from the results that FEC alone is sufficient for full recovery as long as the FEC strength is kept at or above 3. However, FEC protection of 3 packets per transmission block suggests a constant error

[4]If multiple packet loss events occur during a single packet's transmission period, we consider these events as non-distinct packet loss events.

[5]An unrestricted packet loss scenario covers both distinct and non-distinct packet loss scenarios.

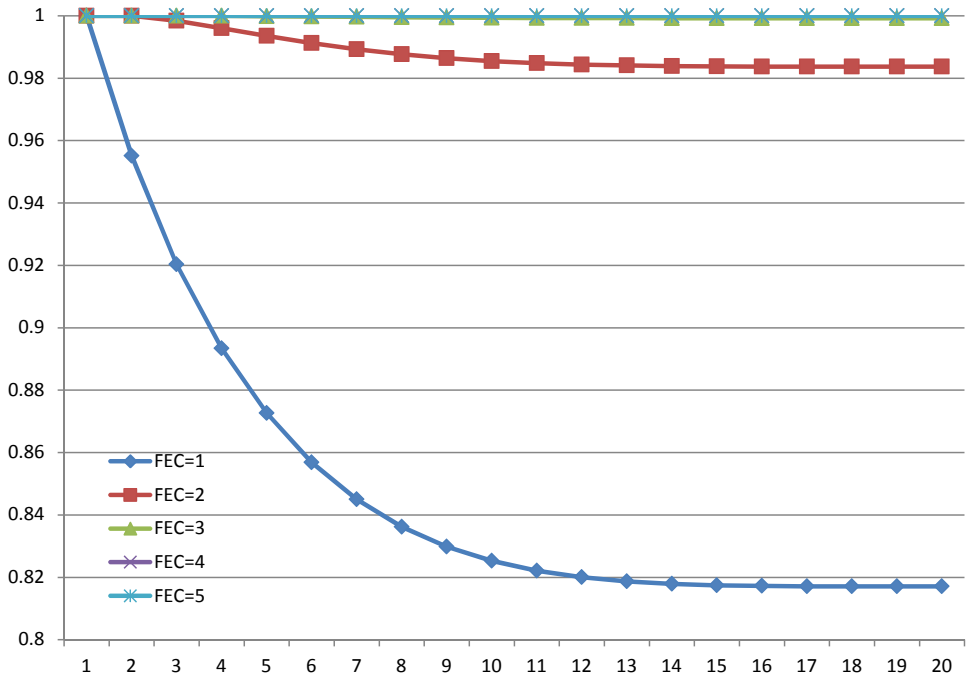


Figure 7: Probability of proactive recovery, when $\gamma = 0.01$.

recovery overhead of 15%. To reduce this overhead, we can limit the FEC strength. For instance, by reducing the FEC strength to 1, we can reduce the error recovery overhead to 5%. The results also show that the probability of full recovery using only proactive recovery decreases significantly as the error rate increases. Hence, to compensate for the limitations imposed by the proactive recovery, we need to use reactive repair techniques.

The next step to determine the probability of successful recovery is to find the equations corresponding to the functions $P_{Sk}(\nu, p_i)$ and $P_{Fk}(\nu, p_i)$. These functions represent the success and failure probabilities for the kth retransmission attempt, where $1 \leq k \leq \min(\varsigma(\nu, p_i), \kappa_R(\nu, p_i))$. Here, $\varsigma(\nu, p_i)$ is used to represent the maximum number of retransmissions allowed for p_i, and its value depends on two variables: time left until the decoding deadline and the capabilities of the Error Recovery Server. Initially, we assume this limit to be independent of the other retransmission attempts taking place within the same transmission block.

To find the impact of the remaining time on the success rate of the retransmissions,

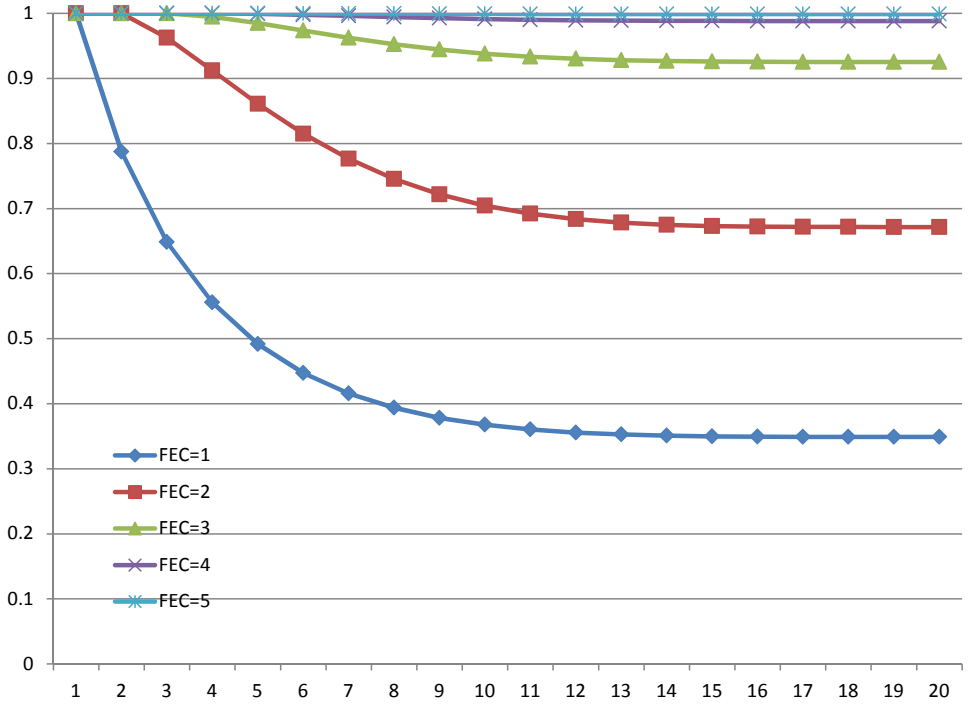


Figure 8: Probability of proactive recovery, when $\gamma = 0.05$.

we use the parameter $\Delta_k(\nu, p_i)$, which represents the expected delivery time for the kth retransmission. Then, the upperbound on the number of retransmissions can be found as follows:

$$\varsigma_{\hat{1}}(\nu, p_i) = \max\{k : \Delta_k(\nu, p_i) < \tau_\pi^\nu < \Delta_{k+1}(\nu, p_i)\} \tag{15}$$

To estimate the arrival times for the repair packets, we use the cumulative distribution function corresponding to the arrival times for the repair packets. We represent this function with $A(\delta_k(\nu, p_i), \Delta_k(\nu, p_i))$. We can then estimate the probability of failure using the probability density function corresponding to $A(\cdot)$, which we refer to as $a(\cdot)$, using the following equation:

$$P_{Fk}(\nu, p_i) = \int_t P_{loss}(t) \times a(t) \times dt \tag{16}$$

where $P_{loss}(t)$ represents the probability of losing the repair packet that is expected to be delivered by the time t.

2.3.3 Finding an Accurate Estimate For $\kappa_R(\nu)$:

So far, the retransmission limit is assumed to be independent of the previous packet losses and the retransmission requests. However, because of the servicing limitations at the Error Recovery Server (ERServ), it is possible to have a strict limit on the number of requests that the ERServ can respond within a specific timeframe (which also suggests a limit on per user requests). We use the parameter $\varpi(\nu)$ to represent this limit, which actually refers to the maximum number of repair packet requests ν can send to the ERServ during each transmission block.

We use the parameter $\varpi_{max}(\nu, p_i)$ to represent the maximum number of requests that can be sent for p_i, and we find its value using $\varpi_{max}(\nu, p_i) = \varpi(\nu) - \varpi(\nu, \overrightarrow{p_i})$, where $\varpi(\nu, \overrightarrow{p_i})$ represents the number of requests sent for packets that are transmitted before p_i and belong to the set $\{p_1, p_2, \cdots, p_{i-1}\}$ [6].

To find the retransmission limit for each packet, we use the information on delivery times for all packets that are delivered earlier than the delivery time of the given packet. We use the vector $\vec{\mathbf{u}}_{\nu k}$ to represent the transmission order set for the kth source packet at the user ν. This set includes the source packet and all possible repair packet transmissions. The comparison point we use for the ordering is the expected delivery time for each possible combination of source and repair packet transmissions within a transmission block [7]. We also create a subset of $\vec{\mathbf{u}}_{\nu k}$, which is referred to as $\vec{\mathbf{u}}^r_{\nu k}$, from the elements that corresponds to the repair packet transmissions only.

The number of retransmission requests that ν can send for p_i before the lth retransmission of p_k is calculated using the following equation:

$$\hat{u}^k_{\nu i}(l) \;=\; \sum_{\forall j>1} (j-1) \times \Big(I[\vec{\mathbf{u}}^r_{\nu i}(j-1) < \vec{\mathbf{u}}^r_{\nu k}(l)] \times I[\vec{\mathbf{u}}^r_{\nu i}(j) > \vec{\mathbf{u}}^r_{\nu k}(l)] \Big) \qquad (17)$$

To integrate the impact of proactive recovery, we use the early packet loss scenarios

[6] $\varpi(\nu, \overrightarrow{p_i}) = \sum_{j=1}^{i-1} \varpi(\nu, p_j)$, where $\varpi(\nu, p_i)$ represents the requests sent for p_i.

[7] The order within the given sets is an event-triggered one, that is, unless a packet failure occurs the next event cannot be initiated.

$$w^k_{\nu,\kappa_P}(l) = \begin{cases} \sum_{s=\kappa_P}^{k-1} \frac{\mu_s(\kappa_P)}{\mu(k,\kappa_P)} \sum_{r_{s+1}=0}^{\hat{u}^k_{\nu s}(l)} \cdots \sum_{r_\eta=0}^{\hat{u}^k_{\nu\eta}(l)} I\left[l + \sum_{\substack{\forall p_j \neq p_k \\ j \geq s}} r_j \leq \kappa_R(\nu) \right] & \text{if } k > \kappa_P \\ \times \prod_{\substack{\forall p_i \neq p_k \\ i \geq s}} \acute{P}^i_\nu(r_i, \hat{u}^k_{\nu i}(l)) & \\ 0 & \text{otherwise} \end{cases}$$

that are correctable using proactive repair. We use (18) to determine the limits associated with each repair packet, where $\acute{P}^i_\nu(r_i, \hat{u}^k_{\nu i}(l))$ is given as follows:

$$\acute{P}^i_\nu(r_i, \hat{u}^k_{\nu i}(l)) = P^{(i)}_{S\nu}(r_i)^{I[r_i \neq \hat{u}^k_{\nu i}(l)]} \times \prod_{j=1}^{r_i} P^{(i)}_{F\nu}(r_j) \tag{18}$$

Here $P^{(i)}_{S\nu}(r_j)$ refers to the probability of successfully delivering the r_jth retransmission of p_i to ν and $\mu(k,\kappa_P)$ is found by using $\sum_{s=\kappa_P}^{k-1} \mu_s(\kappa_P)$ where $\mu_s(\kappa_P)$ is given as follows:

$$\mu_s(\kappa_P) = \left(\frac{(s-1)!}{(s-\kappa_P)!(\kappa_P - 1)!} \right) \times \gamma^{\kappa_P} \times (1-\gamma)^{(s-\kappa_P)} \tag{19}$$

The methodology used to develop (18) can be explained as follows. Here, the initial selection focuses on the packet $p_{[\kappa]}$ on which the last FEC packet is used. Then, we search for the packet loss scenarios that would lead to $p_{[\kappa]}$ being lost. After these scenarios are determined, we sum up the occurrence probabilities of these scenarios and we report the result as $\mu_{[\kappa]}$. Then, we focus on the second part of packet loss scenarios that take place after $p_{[\kappa]}$. In this case, we determine the possible scenarios for which the desired retry count is attainable for the initially considered packet.

To illustrate the impact of these parameters, we simulated (18) using the following scenario: transmission block length of 10-packets, γ selected from the set of { 0.005, 0.01, 0.05, 0.1 }, end-to-end delay of $20ms$ between the end-user and the ERServ, global retransmission limit of 5 requests per transmission block, a packet interarrival time of $5ms$, and a decoding deadline of $125ms$. In Figure 9, we show the results for

the first packet in the transmission sequence [8]. In the figure, the x-axis corresponds to the retransmission attempts that are selected from the interval of [1,5]. From these results, we observe that as the retry count increases, the probability of ever requesting another retransmission decreases. The rate of decrease depends on the steady-state error probabilities.

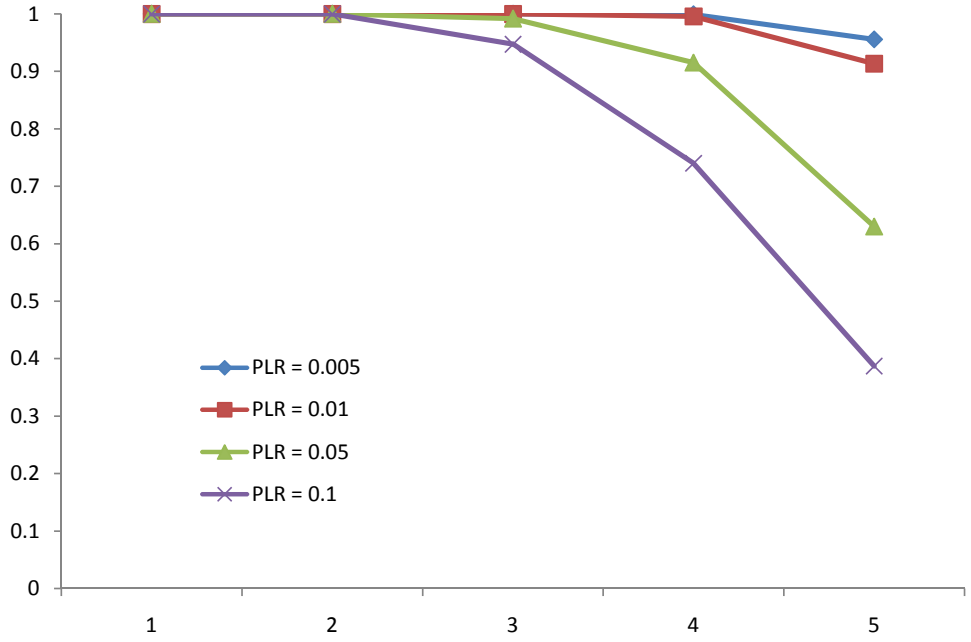


Figure 9: Retransmission probability distribution.

To finalize the equation corresponding to a successful recovery, we use these packet-specific retransmission limits, which are selected from $\vec{w}_\nu$, as weight factors within the final equation. These weights represent the potential impact of the earlier retransmission attempts. In other words, they represent the probability of ever reaching the given retransmission count. To find the value of this weight parameter, we use the probability of having at least one retry chance left at the time of making the request for the repair packet.

[8]Since the first packet always has the highest probability of utilizing its maximum allowed retry count compared to all the other packets coming after in sequence, the results on the first packet represent the upperbound on the retry usage probability.

Consequently, we rewrite the equation for $P_C(\nu, p_i)$ as follows:

$$P_C(\nu, p_i) = P_{FEC}(\nu, p_i) + P_{NFEC}(\nu, p_i) \times \sum_{k=1}^{\kappa_R(\nu,p_i)} \omega_\nu^i(k) \times R_k(\nu, p_i) \quad (20)$$

We can then integrate (20) into (7) to finalize the equation for the probability of a successful recovery.

2.4 Optimization Problem

Using the functions we developed earlier, we can restate the optimization problem initially defined in (1) as follows:

$$\min \quad \sum_{\nu} H(\nu, \pi) \quad (21)$$

$$s.t. \quad \Big[\Gamma - G(\nu, \pi)\Big] < 0, \ \forall \nu \quad (22)$$

where the constant Γ is equal to $\big(|\pi| - \ell(\overline{\pi})\top/\top_{sim}\big)$, and the functions G and H correspond to the following equations:

$$H(\nu, \pi) = \sum_{p_i \in \pi} \Bigg[\overline{\varnothing}_{P,i}(\nu) \times P_{FEC}(\nu, p_i) + \overline{\varnothing}_{R,i}(\nu) \times P_{NFEC}(\nu, p_i) \times \sum_{k=1}^{\kappa_R(\nu,p_i)} k \times \omega_\nu^i(k) \times R_k(\nu, p_i) \Bigg] \quad (23)$$

$$G(\nu, \pi) = \sum_{p_i \in \pi} \Bigg[P_{S0}(\nu, p_i) + P_{F0}(\nu, p_i) \times \Big(P_{FEC}(\nu, p_i) + P_{NFEC}(\nu, p_i) \times \sum_{k=1}^{\kappa_R(\nu,p_i)} \omega_\nu^i(k) \times R_k(\nu, p_i) \Big) \Bigg] \quad (24)$$

where $\overline{\varnothing}_{P,i}(\nu)$ represents the proactive recovery overhead and $\overline{\varnothing}_{R,i}(\nu)$ represents the reactive recovery overhead.

If the error recovery parameters were to take their values from a continuous set, then a quick solution to the optimization problem would be found using the Lagrangian in the continuous space. However, since the parameters take their values from the discrete space, we cannot use the first-order derivatives to find the necessary

conditions, unlike the case for the continuous space. Instead, we need to use the discrete Lagrangian function, $L_d(\kappa_P, \kappa_R, \lambda, \mu)$ to solve the optimization problem (see [101] for details). Then, we can state the discrete Lagrangian function as follows:

$$\begin{aligned} L_d(\kappa_P, \kappa_R, \lambda, \mu) &= \sum_{\nu} H(\nu, \pi) + \sum_{\nu} \lambda_\nu \times K\Big(\Gamma - G(\nu, \pi)\Big) \\ &\quad + \sum_{\nu} \mu_\nu \times K\Big(\max(0, \Gamma - G(\nu, \pi))\Big) \end{aligned} \tag{25}$$

In the above equation, K represents a continuous function that is used to transform the function G. If $K(x)$ is a continuous function satisfying $K(x) = 0 \iff x = 0$, and is non-negative (or non-positive), then in the discrete space, the following is true: the saddle points (*i.e.*, local minima satisfying the discrete-space first-order conditions) and the constrained local minima in the discrete space correspond to the same set of points. One possible choice for K, which satisfies the requirement at $x = 0$, is the square function. Also note that the max function is used to convert the inequality constraint into an equality constraint [90].

In short, to solve the discrete Lagrangian optimization problem, it is sufficient to find the discrete saddle points. We refer to the input variable set using the vector $\mathbf{x}$, where $\mathbf{x} = \{\kappa_P(1), \cdots, \kappa_P(\nu), \kappa_R(1), \cdots, \kappa_R(\nu)\}$. We use the vectors λ and μ to represent the Lagrange parameter sets, *i.e.*, $\lambda = [\lambda_1 \cdots \lambda_N]^T$ and $\mu = [\mu_1 \cdots \mu_N]^T$. We use the parameters C_λ and C_μ to represent the positive adjustment factors, which are used to control how fast the Lagrange multipliers change (where $C_\lambda = [c_{\lambda 1} \cdots c_{\lambda N}] \otimes I_N$ and $C_\mu = [c_{\mu 1} \cdots c_{\mu N}] \otimes I_N$ [9]). To represent the incremental functions that correspond to the change in Lagrange parameters, we use K_λ and K_μ:

$$K_{\lambda,\nu} = K(\Gamma - G(\nu, \pi)) \tag{26}$$

$$K_{\mu,\nu} = K(\max[0, \Gamma - G(\nu, \pi)]) \tag{27}$$

To find the saddle point, the following iterative discrete first-order method is used

[9] $\otimes$ represents time vector-multiply operator and I_N represents the identity matrix of size N

[105]:

$$\mathbf{x}^{k+1} = \mathbf{x}^k \oplus \triangle_x L_d(\mathbf{x}^k, \lambda^k, \mu^k) \quad (28)$$

$$\lambda^{k+1} = \lambda^k + C_\lambda \otimes K_\lambda \quad (29)$$

$$\mu^{k+1} = \mu^k + C_\mu \otimes K_\mu \quad (30)$$

where $\oplus$ represents the vector-add operator, and $\triangle_x$ represents the direction for the maximum potential drop. In the next section, we present a practical approach that can be used to find a near-optimal solution.

2.5 A Centralized Server-based Error Recovery Protocol

To minimize the error recovery overhead, it is of critical importance for the Error Recovery Server to react to packet losses in a timely manner. If the response is delayed for too long, further chances to request additional retransmissions may be lost. By having more chances to implement the reactive repair process, we can increase the probability of a successful recovery. Therefore, to increase the reactive repair success rate, we need to minimize the servicing overhead at the Error Recovery Server (*i.e.*, both processing and queuing overheads).

If the number of users connected to the same Error Recovery Server is small, imposing a limit on the maximum number of repair requests for each user within a specific time-frame may not be necessary. The reason for this is because the repair requests that have been accumulated at the Error Recovery Server will not have a significant impact on the servicing overhead. However, as the number of users connected to the same Error Recovery Server increases, requests will start to experience significant delays, thereby limiting the number of requests that can be served. Consequently, end-users that mostly rely on the reactive repair process to recover from the observed packet losses will fail to recover from these losses. These tradeoffs, which are considered for the reactive repair process, represents our starting point to design a practical resource-optimal error recovery protocol.

We start by defining a new metric to represent the maximum number of requests that the Error Recovery Server is capable of serving within a time-frame of T. We refer to this metric as $\overline{\kappa}_R(T)$. If T_i represents the duration of the ith transmission block, then $\overline{\kappa}_R(T_i)$ represents the upper-bound on the number of repair packet requests that the users connected to the same Error Recovery Server can make (*i.e.*, $\sum_{\forall\nu} \kappa_{R,i}(\nu) \leq \overline{\kappa}_R(T_i)$).

Next, we determine the optimal proactive and reactive error recovery parameters for each user, independently, using the equations that are developed for the proposed hybrid FEC/ARQ-based error recovery framework. Specifically, for each user we determine the parameter set $\Psi(\nu)$ as follows:

$$\Psi_\nu = \{\psi_\nu(j)\} \tag{31}$$

$$\text{s.t.} \quad \psi_\nu(j) = \left\{\kappa_P^{(j)}(\nu), \kappa_R^{(j)}(\nu), Q_\nu(j)\right\} \tag{32}$$

$$O_R\left(\kappa_P^{(j)}(\nu)\right) = \kappa_R^{(j)}(\nu) \tag{33}$$

$$Q_\nu(j) = Q(\kappa_P^{(j)}, \kappa_R^{(j)}, e_\nu) \tag{34}$$

where the function $O(\kappa_P)$ is used to find the optimal reactive error recovery parameter, when the proactive error recovery parameter is equal to κ_P, and $Q_\nu(j)$ represents the error recovery overhead associated with the jth error recovery scenario. We express the function $Q_\nu(j)$ using the overhead function H (which is defined in Section 2.4) as follows:

$$Q_\nu(j) = H(\nu, \pi | \kappa_P(\nu) = \kappa_P^{(j)}, \kappa_R(\nu) = \kappa_R^{(j)}) \tag{35}$$

We initialize the proposed algorithm by comparing the optimal reactive error recovery parameters to the operational capacity of the Error Recovery Server, referred to as ξ, using the equation $R(N) \times (1/\overline{\kappa}_R)$, where $R(N)$ represents the cumulative retry count for all users belonging to the multicast set N. $R(N)$ is a function of the initially selected optimal error recovery parameters (*i.e.*, $\kappa_P^{(0)}$ and $\kappa_R^{(0)}$ [10]), and the

[10] Under normal conditions, the 0th case should correspond to the case of only reactive recovery(no

user-specific error distribution functions. We want to make sure that the probability of ξ being larger than one stays below a certain threshold, $\xi_{\min}$. To not be overly restrictive, we can assume that less than one violation takes place within a time-frame of T_ξ. Then, we select the threshold $\xi_{\min}$ accordingly, instead of making it equal to zero, *i.e.*, $\xi_{\min} = 1/T_\xi - \epsilon_\xi$, where ϵ_ξ is a positive metric that is used to vary the level of strictness for our estimates.

To represent the probability of the total retransmission count staying below the server-set threshold, we define the function $F_{R(N)}(\overline{\kappa}_R)$, *i.e.*, $F_{R(N)}(\overline{\kappa}_R) = P(R(N) < \overline{\kappa}_R)$. To satisfy the server-side constraints regarding the retransmission count, we need to satisfy the condition set by the following inequality:

$$F_{R(N)}(\overline{\kappa}_R) \geq 1 - \xi_{\min} \tag{36}$$

The equation for $R(N)$ is given by $L_{\overline{\pi}} - \sum_{\forall \nu} \kappa_P(\nu)$, where $L_{\overline{\pi}}$ represents the total source and repair packet loss count within a transmission block of size $\overline{\pi}$ (*i.e.*, $L_{\overline{\pi}} = L_{\overline{\pi}}(src) + L_{\overline{\pi}}(rty)$, where $L_{\overline{\pi}}(src)$ is the source packet loss count and $L_{\overline{\pi}}(rty)$ is the repair packet loss count). We can estimate the value of the source packet loss count using the packet loss distribution function which is based on the Inhomogeneous Poisson Process:

$$\lambda(\overline{\pi}(src)) = \int_0^{\overline{\pi}} \left(\sum_{\forall \nu \in N} \lambda_\nu(u) \right) du \tag{37}$$

If the poisson rates can be discretized, then the source packet loss count can be found using the following equation:

$$\lambda(\overline{\pi}(src)) = \sum_{j=0}^{\overline{\beta}} \left(\sum_{\forall \nu \in N} \lambda_{S,\nu}(j) \right) \times \left[T_{j+1} + \lceil \overline{\pi} - T_{j+1} \rceil^+ - T_j \right] \tag{38}$$

where T_j represents the occurrence time for the jth rate-change, and $\overline{\beta}$ represents the total number of rate-change points. We find the rate-change points by combining all

FEC packet is used). However, if the error rate is higher than a certain threshold, then the 0th case requires the use of FEC packets and it becomes a joint-recovery scenario. The reason for this is because in some extreme cases, FEC-usage may have positive impact on the error recovery overhead.

the user specific rate-change point sets into a single set.

Note that, in reality, delivery times vary from one user to another when the timing is based on a reference timer, thereby making it difficult to form a synchronous set of rate-change points. However, this synchronization problem mostly affects the instantaneous packet loss rates rather than the cumulative packet loss rates. The reason for this is because when losses are considered, non-overlapping periods for the same transmission block can be compensated with the last-section of the previous transmission block or the first-section of the subsequent transmission block. In short, synchronous packet loss rates will be used (i) to represent the instantaneous packet loss rates, and (ii) to find the cumulative packet loss rate.

Then, using the characteristics of the poisson process, we can find the probability of observing k packet losses within a timeframe of $\overline{\pi}$ as follows:

$$P(L_{\overline{\pi}}(src) = k) = \frac{e^{-\lambda(\overline{\pi}(src))} \times \lambda(\overline{\pi}(src))^k}{k!} \tag{39}$$

Next, we need to find the reactive repair packet loss rate, which is used to estimate the impact of the additional recovery requests. For that purpose, we define a new metric that represents the weight of the additional retransmissions. We refer to this metric as l_ω and find its value using the following equation:

$$l_\omega = \frac{\sum\limits_{\forall \nu \in N} \gamma_\nu \times l_{\omega,\nu}}{\sum\limits_{\forall \nu \in N} \gamma_\nu} \tag{40}$$

where the value for user-dependent parameter $l_{\omega,\nu}$ is found as follows:

$$l_{\omega,\nu} = |\pi|^{-1} \times \sum_{j=\kappa_{P,\nu}+1}^{|\pi|} \sum_{k=1}^{\kappa^j_{P,\nu}} k \times \gamma_\nu^k \times (1-\gamma_\nu) \times \omega_\nu^j(k) \tag{41}$$

Using the parameter $l_{\omega,\nu}$, we can state the equation for $L_{\overline{\pi}}(rty)$ using $\big(L_{\overline{\pi}}(src) - \sum_{\forall \nu} \kappa_P(\nu)\big) \times (l_\omega - 1)$. This equation is then used to restate the equation for $R(N)$, which takes the form of $l_\omega \times \big(L_{\overline{\pi}}(src) - \sum_{\forall \nu} \kappa_P(\nu)\big)$.

Consequently, the equation for $F_{R(N)}(\overline{\kappa}_R)$ is stated as follows:

$$F_{R(N)}(\overline{\kappa}_R) = \sum_{k=0}^{\sum_{\forall\nu}\kappa_P(\nu)+\lfloor\overline{\kappa}_R/l_\omega\rfloor} \frac{e^{-\lambda(\overline{\pi}(src))} \times \lambda(\overline{\pi}(src))^k}{k!} \tag{42}$$

Algorithm 1 Error recovery parameter initialization.

$s = 0$;
$\forall\nu \in N$ **do** $s_\nu = s$;
find $\sigma = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$
$\sigma' = \sigma$;
while $\sigma' < 1 - \xi_{\min}$ **do**
  find $u_{\min} = \min_{\nu\in N}\left(\left[Q_\nu^{(\min(s_\nu,s)+1)} - Q_\nu^{(s_\nu)}\right] > \epsilon_q\right)$
  if $u_{\min} = \emptyset$ **then**
    $s = s + 1$;
  else
    update σ'
    $s_{u_{\min}} = s_{u_{\min}} + 1$;
  end if
end while

The pseudocode that is used to initialize the error recovery parameters is shown in Algorithm 1. Starting with the user that observes the minimum overhead increase, we adjust the FEC strengths step-by-step so as to minimize the change in the accumulative overhead. Note that when the FEC strength is increased, the increase we observe in the error recovery overhead depends on the distribution of packet losses observed over different transmission blocks. The overhead metric that we use in our estimations takes into account the packet loss distribution.

The first parameter update for the error recovery process takes place during the channel initialization phase. Using the initialization process, we optimally allocate the network resources based on the initial network state. Then, to account for the variations in the network state, we implement three other update procedures corresponding to join, leave, and error-update events. Next, we explain the parameter update procedures for each of these events.

2.5.1 Event I: Session Join

When a user joins an active multicast session, we first check the impact of this join-event on the overall network utilization. During the initialization phase, if the constraints set by the Error Recovery Server are not violated, then the error recovery parameters selected for the new user can be used without any problem. On the other hand, if the join-event violates the server-side constraints, then the error recovery sources need to be reallocated. For this purpose, we use Algorithm 2, which allows the system to increment the error recovery parameters starting from the level determined during the initialization phase. This process is essentially a continuum of the initial setup phase except for the initial error recovery parameters selected for the new user.

Algorithm 2 Resource update after u' joins the active receiver set.

$s = \min_{\forall \nu \in N} s_\nu$;
$N = N \cup u'$
$s_{u'} = 0$;
find $\sigma = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$
if $\sigma \geq 1 - \xi_{\min}$ **then**
  do Nothing.
else
  $\sigma' = \sigma$;
  while $\sigma' < 1 - \xi_{\min}$ **do**
    find $u_{\min} = \min_{\nu \in N} \left(\left[Q_\nu^{(\min(s_\nu, s)+1)} - Q_\nu^{(s_\nu)} \right] > \epsilon_q \right)$
    if $u_{\min} = \emptyset$ **then**
      $s = s + 1$;
      if $s_{u'} < s$ **then**
        $s_{u'} = s_{u'} + 1$;
      end if
    else
      update σ'
      $s_{u_{\min}} = s_{u_{\min}} + 1$;
    end if
  end while
end if

2.5.2 Event II: Session Leave

When a user leaves an active channel, the portion of the error recovery resources used by this user needs to be freed. Then, to facilitate optimal resource utilization, we redistribute the resources that are freed as a result of the leave-event among the current active users. Typically, the most efficient redistribution approach is to deliver these resources to as many users as possible. As the number of end-users that take advantage of this resource availability increases, the amount of resources allocated for the proactive error control decreases. For this purpose, we reverse the operation of the original resource allocation policy. Specifically, we target the end-users that observe the maximum decrease in the error recovery overhead by having one-step decrease in the proactive recovery strength. To further improve the fairness, we restrict the search process to users with the highest proactive recovery strength. We present the pseudocode for the proposed local resource reallocation approach in Algorithm 3.

Algorithm 3 Resource update after u' leaves active receiver set.

$N = N \backslash u'$;
$s^* = \max_{\forall \nu \in N} s_\nu$;
$N(s^*) = \{\forall \nu : s_\nu = s^*\}$;
find $\sigma = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$
$\sigma' = \sigma$;
while $\sigma' \geq 1 - \xi_{\min}$ **do**
  find $u_{\max} = \max_{\nu \in N(s^*)} \left(\left[Q_\nu^{(s^*)} - Q_\nu^{(s^*-1)} \right] > \epsilon_q \right)$
  if $u_{\max} = \emptyset$ **then**
    $s^* = s^* - 1$;
    $N(s^*) = \{\forall \nu : s_\nu = s^*\}$;
  else
    $s_{u_{\max}} = s^* - 1$;
    $N(s^*) = N(s^*) \backslash u_{\max}$;
    update σ'
  end if
end while

2.5.3 Event III: Session Update

The last case we investigate is the error update event, which refers to a change in the average error rate observed by an end-user. Since error rate has a direct impact on how the optimal error recovery parameters are selected, a change in the perceived channel state (*i.e.*, error rate) requires these parameters to be updated. Specifically, if the error rate increases, then the optimal proactive and/or reactive parameters also increase, and vice versa. Regardless of the nature of these changes, we may need to update the error recovery parameters for the given active user set. The pseudocode for the required procedure is given in Algorithm 4.

Algorithm 4 Integrating the packet loss rate variations observed by u.

$\Psi_{u,pre} = \Psi_u$ & $\Psi_u \rightarrow \Psi_{u,cur}$;
$s_{\min} = \min_{\forall \nu \in N} s_\nu, s_{\max} = \max_{\forall \nu \in N} s_\nu$;
$s_{u,pre} = s_u$;
if $\gamma_{cur}(u) > \gamma_{pre}(u)$ **then**
  $s_u = s_{\max}$;
  $\sigma = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$;
  if $\sigma \geq 1 - \xi_{\min}$ **then**
    while $\sigma \geq 1 - \xi_{\min}$ **do**
      $s_u = s_u - 1$;
      $\sigma = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$;
    end while
    $s_u = s_u + 1$;
  else
    initiate `Update Process1`
  end if
else
  $\sigma_{cur} = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$;
  while $\sigma_{cur} \geq 1 - \xi_{\min}$ **and** $s_u \geq 0$ **do**
    $\sigma = \sigma_{cur}$;
    $s_u = s_u - 1$;
    $\sigma_{cur} = F_{R(\{s_\nu\})}(\overline{\kappa}_R)$;
  end while
  $s_u = s_u + 1$;
  if $s_u = 0$ **then**
    if $\sigma \geq 1 - \xi_{\min}$ **then**
      initiate `Update Process2`
    end if
  end if
end if

The proposed update procedure uses two child processes to finalize the update corresponding to each possible scenario, that is, error rate increase or decrease. `Update Process1` refers to the join-event update process, which is used to redistribute the resources currently in use, and `Update Process2` refers to the leave-event update process, which is used to redistribute the freed resources within the active user set.

2.6 Performance Analysis

In this section, we evaluate the performance of the proposed error recovery framework using a simulation based study. We implement the proposed error recovery framework in Matlab. The simulation parameters are chosen as follows. To evaluate the overhead performance for the worst-case scenarios, the number of users is varied between 1×10^3 and 1×10^4. To generate the packet loss events, Poisson process is used. Packet loss rates are chosen independently from the range $\left(1 \times 10^{-3}, 2 \times 10^{-1}\right)$ using the log-domain approach, which assigns the packet loss rate values using the uniform distribution in the log-domain. To be specific, $\log(p_{loss})$, which represents the logarithmic value for the packet loss rate, is selected uniformly from the interval $\big(\log(p_{loss,\min}), \log(p_{loss,\max})\big)$, where the values for the minimum packet loss rate, $p_{loss,\min}$, and the maximum packet loss rate, $p_{loss,\max}$, are varied depending on the simulation scenario. [11]

IPTV multicast transmission rate is given by $3Mbps$. We assume each IPTV packet to have a length of $1356Bytes$. Error Recovery Server is assumed to have a transmission capacity of $100Mbps$, which suggests a servicing rate of at most 9218 single repair packet requests per second. [12] To minimize the latency associated with the error recovery process, we use a transmission block size of $100ms$, and an

[11] Note that, assigning the packet loss rate values in the log-domain allows us to skew the packet loss distribution towards the low loss probability state, thereby creating a much more realistic scenario in regards to the users' expected packet loss rates.

[12] Maximum value for the servicing capacity is achieved when the processing delay at the server becomes negligible when compared to the transmission delay. For the sake of simplicity, in our simulations, we assume the previous statement to be true.

initial startup latency of $160ms$, which gives the client an additional $60ms$ to recover from its losses during a given transmission block. We assume an end-to-end delay of $50ms$ between the client and the Error Recovery Server, which is typically sufficient to recover from all the losses a client observes within a transmission block. The reported results are the average of 10 simulation runs, each of which represents a 4-hour long IPTV broadcast.

In our simulations, we focus on two critical performance metrics: recovery cost and the level of fairness in the distribution of the recovery costs. Here, with the recovery cost metric, we refer to the average error recovery overhead observed at the client side. Since one crucial objective of the proposed recovery framework is to minimize the probability of a server overload, as long as the given objective is met, the resulting overhead at the server side does not constitute a critical concern in our performance evaluations. The second performance metric is used to analyze how the recovery load is distributed among the users, each of which is characterized with a different packet loss function.

We show the first set of results in Figure 10, which depicts the average overhead observed at the client side as the number of users connected to the same Error Recovery Server is varied between $1,000$ and $10,000$. Here, log-normalized assignments for the packet loss rates resulted in an average packet loss rate of 6.3×10^{-2}. The same figure also shows the results for three different scenarios: (i) when proactive recovery is not used (*Approach 1*), (ii) when clients are assigned FEC rates based on the mean packet loss rate (*Approach 2*) [13], and (iii) when clients are assigned FEC rates based on the worst case scenario (*Approach 3*), *i.e.*, to ensure that, for the given simulation scenario, no more than 1 packet loss is observed during a 4-hour long IPTV broadcast

[13] Assuming that n_i represents the average number of packet losses the ith client observes within a transmission block, for Approach 2, the proactive recovery strength is selected by using $\kappa_p = \lceil n_i \rceil$, where $\lceil \cdot \rceil$ represents the ceiling operation. Here, ceiling operation is chosen over rounding up n_i to the nearest integer, as the former achieved significantly better results in the servicing capacity when compared to the latter approach.

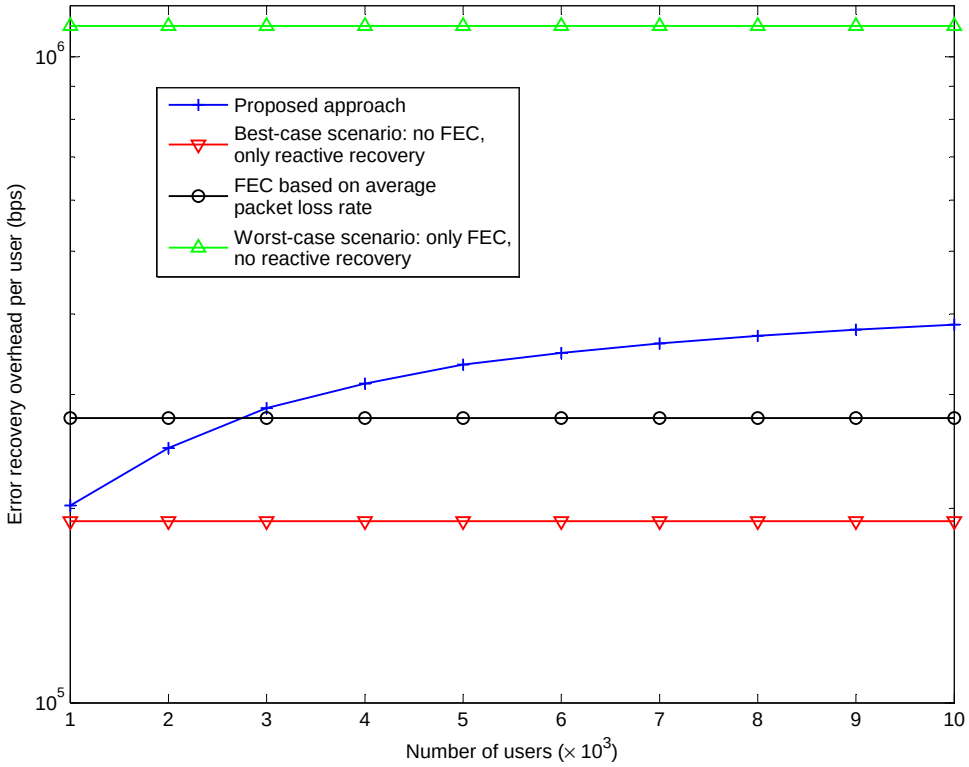


Figure 10: Recovery overhead performance for various recovery approaches.

when only proactive recovery is used.

The first approach represents the ideal (or the best-case) scenario, from the users' perspective, since it achieves the least overhead at the client side. However, its performance depends on the transmission capacity at the Error Recovery Server. When the number of requests received during a specific time-frame exceeds the servicing threshold at the Error Recovery Server, then the users may experience significant performance degradations. Hence it is mostly useful when the number of users connected to the server is sufficiently small, or when the accumulative packet loss rate is not significant. The second approach is typically effective when the number of users is manageable by the server. The third approach is mostly effective when the number of requests is significantly higher than the servicing capacity at the Error Recovery Server.

For the given simulation scenario, we observe the following comparative results

for the error recovery overhead:

- *Approach 1* achieves the least per-user overhead at the client side, 191.36$Kbps$, since a client receives a single repair packet per each packet lost. However, to ensure that the quality requirements for the IPTV service are met at the client side, we need to increase the server capacity to more than 200$Mbps$ when $N = 1000$.

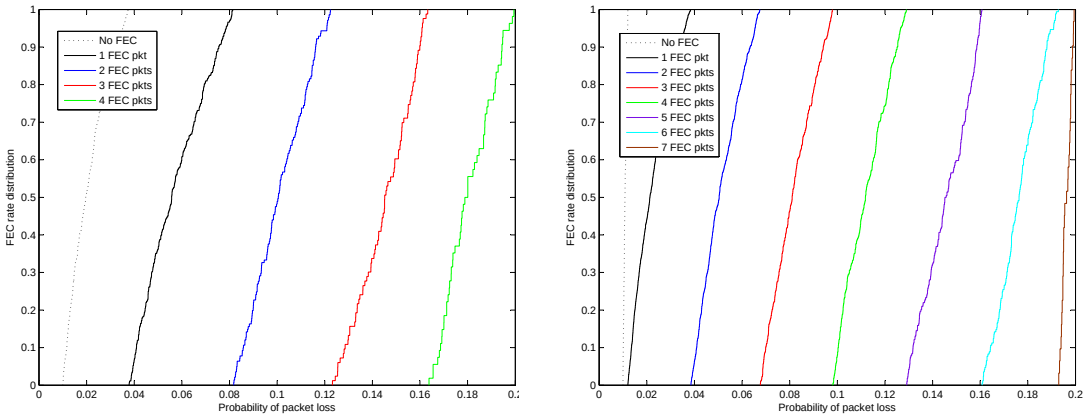
- *Approach 2* results in an overhead of 275.84$Kbps$, which represents a 44% increase when compared to the ideal scenario. The second approach can only achieve the service quality objectives when $N \leq 2000$. When the number of users is increased to 3000, the server becomes overloaded more than 15% of time, suggesting a critical threshold for the servicing capacity. As the number of users is increased further, we start to observe a near-continual overload state at the server side. Hence, Approach 2 fails to satisfy the service quality requirements for the IPTV clients for most of the considered scenarios.

- *Approach 3* results in an overhead of 1.116$Mbps$, which suggests close to six times increase in the per-user recovery overhead when compared to Approach 1, and a 37% increase in the minimum required bandwidth usage over the 3$Mbps$-rate for the IPTV multicast. Even though the third approach succeeds in satisfying the packet loss requirements for the IPTV service, for all the considered scenarios, bandwidth requirements at the client side exceed the typically permitted levels, leading to highly inefficient resource utilization results.

- Proposed error recovery framework results in an average overhead of 322.3$Kbps$, which is only a 68.6% increase in the overhead when compared to the ideal scenario. Since the proposed framework evaluates the impact of proactive and reactive recovery schemes on the system together as a whole, by jointly considering the requirements at the clients and the server side, it can successfully

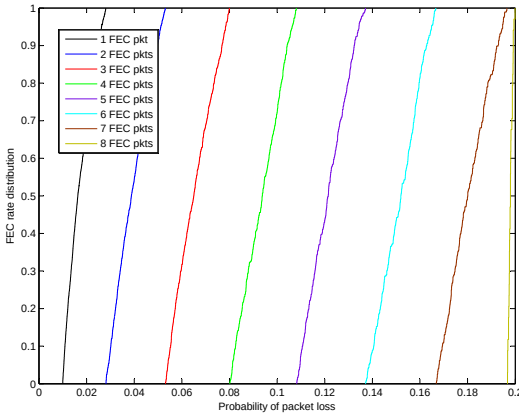
achieve the quality objectives with limited increase in the error recovery overhead.

In short, comparing the results of the proposed framework to that of Approach 2, we can make the following observations: allocating resources for the clients independently (i) overestimates the bandwidth requirements for the low loss and/or sparsely connected scenarios and (ii) underestimates the bandwidth requirements for the high loss and/or densely connected scenarios.

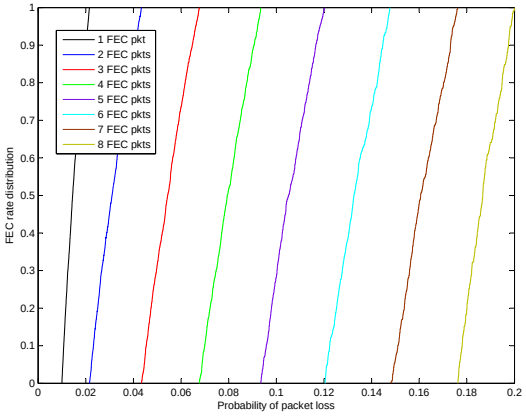


(a) 1000 users

(b) 3000 users



(c) 5000 users



(d) 8000 users

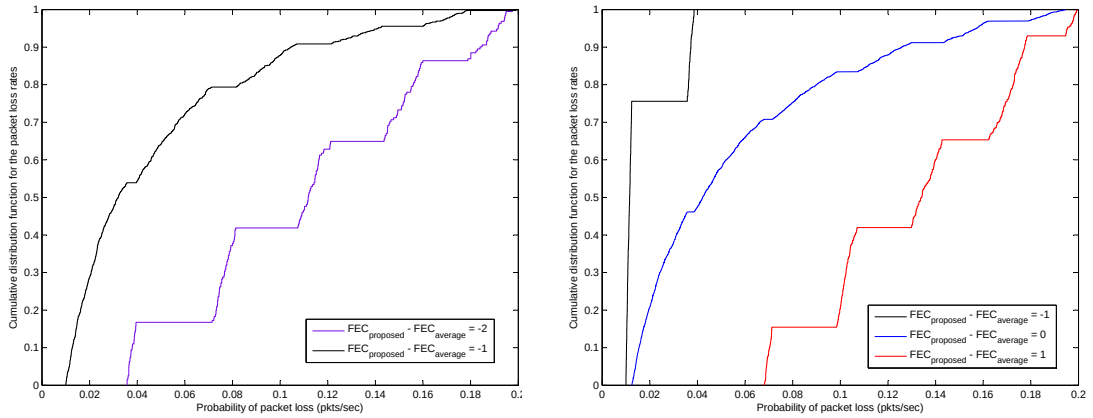
Figure 11: Distribution of FEC packets to IPTV clients.

Next, in Figure 11, we illustrate the relationship between the selected FEC rates and the user perceived packet loss rates as we vary the number of users. We observe that, for all the cases considered, the distribution for the FEC rates assigned to the

users closely follow the packet loss rates experienced by the users. If the packet loss rate experienced by a user is less than a specific threshold, then such user is not considered for proactive recovery. If, on the other hand, the packet loss rate experienced by a user is higher than the given threshold, then the FEC rates are increased to the level which limits the server overload occurrences to no more than one instance during the considered continual viewing period.

We also observe in Figure 11 that increasing the system load had limited impact on the worst-case scenario for the FEC rate allocations (e.g., when $N = 3000$, the maximum value for the FEC rate is 7 packets per transmission block, whereas, when $N = 8000$, the maximum value for the FEC rate is 8 packets per transmission block). Note that, here, increasing the number of users essentially shifts the FEC curves towards the lower loss regions to minimize the increase in recovery overhead per user. In doing so, we can fairly distribute the recovery overhead to the users that experience varying packet loss rates.

To further investigate the fairness characteristics of the proposed framework, we compare the resulting assignments for each user to the respective assignments when the second approach is implemented. Specifically, we want to determine the relationship between the level of shifting we observe for the FEC rate assignments and the user perceived packet loss rates. We show these results in Figure 12. In general, we expect the users with lowest packet loss rates to observe the least amount of changes in their assigned FEC rate values. The results shown in Figure 12 validates these expectations. We observe similar trends for all the considered scenarios. As the users start to observe smaller packet loss rates, the changes these users observe in their FEC rate assignments also decrease, and vice versa. In short, the proposed framework is capable of deciding on the proactive recovery strengths and distributing the recovery resources in a fair manner.



(a) 1000 users

(b) 3000 users

(c) 5000 users

(d) 8000 users

Figure 12: Change in the number of selected FEC packets when compared to the average loss-rate based FEC assignment scenario.

2.7 Conclusions

In this chapter, we presented a general error recovery framework for the IPTV networks by focusing on the cumulative impact of transmission medium, error recovery parameters, session latency, and transmission delays with the objective of designing a resource efficient error recovery protocol that is capable of achieving the service quality requirements for the IPTV service. We started the chapter by giving an in-depth theoretical analysis for the proposed error recovery system. Using the proposed system, we then developed an optimization problem to find the error recovery parameters that require the least amount of network resources while allowing the service quality to stay above a certain threshold. To solve the optimization problem, we specifically focused on the servicing requirements at the Error Recovery Server, and developed a practical recovery protocol that aims to minimize the probability of server overload while using the least amount of resources at the clients. We evaluated the performance of the proposed error recovery protocol using a simulation based study and observed significant improvements in per-client resource utilization with the proposed scheme. The proposed framework is also shown to be effective in fairly distributing the resources to the clients.

CHAPTER III

SUPPORTING RELIABLE CONTENT DELIVERY IN GROUP CORRELATED IPTV NETWORKS

3.1 Introduction

In Chapter 2, we proposed a generalized hybrid FEC/ARQ-based error recovery framework for IPTV networks that relies on the use of a dedicated server, which is installed between the content delivery server and the end users. We referred to this server as the Error Recovery Server (ERServ). The solution we proposed in Chapter 2 is based on the assumption that the users observe uncorrelated packet losses, which allowed us to pursue a unicast-based transmission strategy to achieve the optimal solution.

However, if the users start to observe correlated packet losses, then the unicast-based strategy will no longer be sufficient to achieve the optimal solution. Especially if the number of users connected to an ERServ is on the order of thousands and the average error rate is high, then the number of repair packets an ERServ may need to transmit within a short time-frame can easily exceed the manageable levels. As a result, the dedicated server may become overloaded with too many request messages and enter a non-responsive state. This may lead to many requests getting dropped at the entry to the server, and the ones being admitted to the server to experience significant delays. In such scenarios, a multicast-based recovery may become the better and the more effective option in responding to the error recovery requests in a timely manner compared to unicast-based recovery. Therefore, in networks where the users observe correlated packet losses, it becomes crucial to investigate the tradeoffs between unicast- and multicast-based recovery techniques so as to determine the most

suitable approach to maximize the servicing capacity of the network.

Therefore, in this chapter, our objective is to investigate the impact of different levels of correlations among the users' packet loss processes on the error recovery overhead when multicast- and unicast-based recovery techniques are evaluated together. To achieve this objective, we first propose the group loss correlation model to generate spatially correlated packet loss events. We test our approach using three different loss processes, namely, the Poisson process, K-state Markov-modulated Poisson Process (MMPP) and the 2-state Discrete Time Markov Chain (DTMC). We then propose a simple, yet, effective approach to integrate the parity-check-based Application-layer Forward Error Correction packets into the reactive multicast-based recovery process and investigate its effectiveness in reducing the error recovery overhead, thereby improving the servicing capacity of the network.

The rest of the chapter is organized as follows. In Section 3.2 we present our system model. In Section 3.3 we simulate the proposed error recovery framework and analyze its performance in various scenarios. Section 3.4 concludes the chapter.

3.2 System Model

We consider an IPTV system that consists of a single Error Recovery Server (ERServ) and N users [1], each of which is connected to a single IPTV multicast session. Assuming that the ERServ serves κ multicast sessions in total, then we have $\sum_{i=1}^{\kappa} |N_i| = N$, where N_i represents the set of users connected to the ith multicast session.

To evaluate the end-to-end delivery performance of data traffic over the access links, various packet loss models have been considered. For instance, for the DSL-based access networks (which represent the most widely used access network for the IPTV clients), multi-state semi-Markov model with mixed exponential and Pareto distributions is considered to achieve the best results in terms of accurately representing

[1] Hereafter, we will use the terms user and client interchangeably.

the realistic conditions. However, because of the difficulties involved in the practical implementation of such models to evaluate the networking performance, some simplifications have been made to these models by using either a two-state Markov model (with the good and bad states representing the error-free and erroneous periods) or a model based on the Poisson process (*i.e.*, exponentially distributed interarrival times for the error-bursts). These simplified models are typically used to implement uncorrelated packet loss scenarios (*e.g.*, Poisson process) or temporally correlated bursty packet loss scenarios (*e.g.*, Markov model). Note that, these approaches are mainly proposed to represent independent (and user-specific) packet loss scenarios. Therefore, neither approach is alone sufficient to generate spatially correlated data sets, which correspond to the packet loss observations of different clients.

Here, to form the spatially correlated packet loss scenarios, we introduce an approach that we refer to as the group loss correlation model, which utilizes an aggregate packet loss model to project the correlation statistics onto a given user set. For this purpose, we consider three different packet loss models. The first model uses Poisson distribution to generate the individual single packet loss events. The second model is based on a K-state Markov-modulated Poisson Process (MMPP) [34, 39], which also generates the packet loss events on a packet-by-packet basis. Finally, the third model is based on the two-state Discrete Time Markov Chain (DTMC), which is also known as the Gilbert-Elliot (GE) model [36, 33]. We use the third model to generate correlated and bursty packet loss events.

Next we present the discussion on each of these models and explain the methodology we use to make the necessary transition from the aggregate packet loss process to the individual ones, each of which represents the group correlated packet loss process for each user.

3.2.1 Poisson Process

To generate the packet loss events at each user, we use the following procedure. Assuming that $\lambda_{\nu_j}^{(i)}$ represents the arrival rate for the packet loss events at a user ν_j, where $\nu_j \in N_i$, then we use the equation $\lambda_T^{(i)} = \sum_{\forall \nu_j \in N_i} \lambda_{\nu_j}^{(i)}$ to represent the aggregate packet loss rate for users connected to the ith multicast session.

To determine the value of the group correlation metric for the set N_i, we use the following equation:

$$\rho_G^{(i)} = \frac{\lambda_T^{(i)} - \lambda_G^{(i)}}{\lambda_T^{(i)}} \tag{43}$$

where $\rho_G^{(i)}$ represents the group correlation metric and $\lambda_G^{(i)}$ represents the arrival rate (or packet loss event generation rate) for the group correlation loss process. [2]

Next, we use the packet loss arrival rate for the group loss model-selected based on the desired loss correlation ratio-to individually generate the packet loss events at each user. For that purpose, we create a user-specific binomial parameter to represent the occurrence probability of a loss event. We refer to this binomial parameter using $p_j^{(i)}$, $\forall \nu_j \in N_i$. We can find the value of $p_j^{(i)}$ by using the characteristics of the poisson process. Specifically, we focus on packet loss events that take place during the transmission period of a single packet (referred to as τ_p), and by equating, for each user, the packet loss probability in group correlation model with the individual packet loss probability, we obtain the following equality:

$$e^{-\lambda_j^{(i)}\tau_p} = \sum_{k=0}^{\infty} \frac{e^{-\lambda_G^{(i)}\tau_p} \times (\lambda_G^{(i)}\tau_p)^k}{k!} \times (1 - p_j^{(i)})^k \tag{44}$$

which leads to $p_j^{(i)} = \lambda_j^{(i)}/\lambda_G^{(i)}$.

Figure 13 illustrates the impact of different group correlation metrics on the distribution of pairwise correlated losses [3], when the individual packet loss rates for

[2] The feasible set for the correlation metric values needs to ensure that $\lambda_G^{(i)} \geq \lambda_j$, $\forall \nu_j \in N_i$. For instance, if $\lambda_{\nu_j} = \lambda_{\nu_k}$, $\forall \{\nu_j, \nu_k\} \in N_i$, then $\rho_G^{(i)} \leq (|N_i| - 1)/|N_i|$.

[3] Pairwise correlated loss rate represents the ratio of the number of pairwise correlated losses to the total number of packet losses observed at any given user.

the users are randomly selected from the interval $(10^{-2}, 10^{-1})$. [4] Compared to the uncorrelated loss scenario, when ρ_G is assigned a value close to or less than 0.5, no dramatic changes are observed in the distribution of pairwise correlated losses. As we continue to increase the value of ρ_G, geometric growth observed for the resulting ratios becomes more noticeable. Consequently, when the value of ρ_G becomes close to 0.9, we start to observe evenly distributed pairwise correlated loss ratios for the given user set.

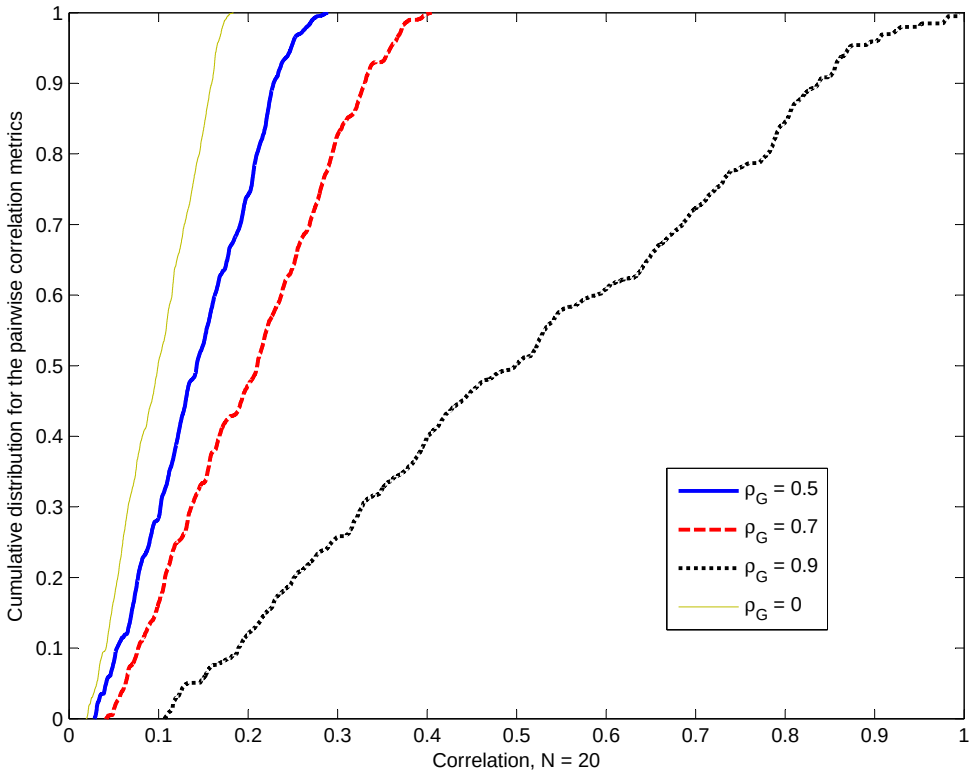


Figure 13: Pairwise correlation results.

3.2.2 Markov-modulated Poisson Process (MMPP)

The second approach to generate the group correlated loss process is based on the doubly stochastic Markov-modulated Poisson Process (MMPP) [34]. To generate

[4] Unless otherwise stated, in our simulations, packet loss rates for all the clients are selected from an interval of $(10^{-2}, 10^{-1})$.

the packet loss events using a K-state MMPP process, we utilize K distinct Poisson processes, each of which is represented with an arrival rate of λ_i, where $i \leq K$, and for which the transitions from one state (or process) to another are triggered based on the underlying Markov process. Here, the sojourn times and the state transition probabilities are determined by the Markov process, whereas the arrivals within each state are determined by the corresponding Poisson process.

We use the following methodology to generate the MMPP-based correlated packet loss events. We start by focusing on a specific group of users, which is represented with the set N_i. Let us assume that the number of states for the underlying Markov model is given as K_i and the generator matrix $G^{(i)}$ corresponding to the generalized process is also known beforehand, *i.e.*, $[G^{(i)}]_{j,k} = p_{jk}$ if $j \neq k$ and $[G^{(i)}]_{j,j} = -p_j = -\sum_{\forall k \neq j} p_{jk}$, where $p_{j,k}$ represents the transition rate from state s_j to state s_k, and $1/p_j$ represents the mean sojourn time for s_j. We can then use the following set of equations to find the steady state probabilities associated with each available state:

$$\pi_j^{(i)} \times p_j^{(i)} = \sum_{\substack{k \leq K_i \\ k \neq j}} \pi_k^{(i)} \times p_{kj}^{(i)} \text{ and } \sum_{j \leq K_i} \pi_j^{(i)} = 1 \tag{45}$$

where $\pi_j^{(i)}$ represents the steady state probability corresponding to state s_j (where $j \leq K_i$).

Let us also assume that the individual packet loss rates at each client within N_i to be known. The proposed model suggests that the state transitions at the clients follow the transitions observed for the generalized process, thereby allowing us to have direct access to the state transition probability information at each client. [5] Next, at each client $\nu_j \in N_i$, and for each state s_k (where $k \leq K_i$), we randomly assign a loss

[5]Note that, the actual superposed generalized process may need to consist of $\prod_{\forall i \in N_i} K_i$ states. Further simplifications can be made to reduce the size of the generalized process [41]. However, accurately capturing correlations among the individual processes using (simplified) superposed processes may not be very practical. Instead, we can assume the individual processes as the superposition of two processes, an uncorrelated one and a correlated one. In our study, we essentially focus our attention on the latter.

rate parameter, referred to as $\alpha_{j,k}^{(i)}$, selected from an interval of $(0, 1)$.

To determine the values for the actual packet loss rates associated with each channel state, we use a (unit) loss rate metric, which is represented with the parameter $\mu_j^{(i)}$ for client $\nu_j \in N_i$. It then becomes sufficient to solve the following equation to determine the state-dependent loss event generation rates:

$$\lambda_{j,k}^{(i)} = \mu_j^{(i)} \times \sum_{k \leq K_i} \alpha_{j,k}^{(i)} \times \pi_k^{(i)} \tag{46}$$

We determine the state-dependent loss rates for the generalized process using a similar procedure. To be specific, we first determine the average loss rate for the generalized loss process ($\lambda_G^{(i)}$) based on the selected correlation ratio ($\rho_G^{(i)}$) using equation (43). After we find the value for $\lambda_G^{(i)}$, we assign random weights to each available state s_k (*i.e.*, $\alpha_{G,k}^{(i)}$, where $k \leq K_i$) using the almost same procedure as before, with a slight difference observed in the estimation of the unit loss rate metric. Specifically, for each state s_j, we identify the maximum valued loss event generation rate that is utilized by the given set of users, *i.e.*, $\lambda_{\max,k}^{(i)} = \max_{\forall \nu_j \in N_i} \lambda_{j,k}^{(i)}$. In the next step, we define the state-dependent loss event generation rates for the generalized process using the following equation:

$$\lambda_{G,k}^{(i)} = \lambda_{\max,k}^{(i)} + \alpha_{G,k}^{(i)} \times \mu_G^{(i)} \tag{47}$$

where $k \leq K_i$.

For example, in the case of 2-state availability, *i.e.*, $K_i = 2$, then we can simplify the equation for $\mu_j^{(i)}$ as follows:

$$\mu_j^{(i)} = \frac{\lambda_j^{(i)} \times (p_1^{(i)} + p_2^{(i)})}{\alpha_{j,1}^{(i)} \times p_2^{(i)} + \alpha_{j,2}^{(i)} \times p_1^{(i)}} \tag{48}$$

Then, the weight for the generalized loss process is determined as follows:

$$\mu_G^{(i)} = \frac{\lambda_G^{(i)} \times (p_1^{(i)} + p_2^{(i)}) - \lambda_{\max,1}^{(i)} \times p_2^{(i)} - \lambda_{\max,2}^{(i)} \times p_1}{\alpha_{G,1}^{(i)} \times p_2^{(i)} + \alpha_{G,2}^{(i)} \times p_1^{(i)}} \tag{49}$$

After we determine the state-dependent (packet loss) event generation rates for both the individual processes and the generalized process, we can utilize the approach presented in the previous section for the Poisson loss scenario. Specifically, at each observation instance for the packet loss events corresponding to the generalized process, we can find the probability of a client ν_j to also observe a packet loss event at the same instance, $p_{L,j}$, using the following equation:

$$p_{L,j}^{(i)} = \frac{\lambda_{j,k}^{(i)}}{\lambda_{G,k}^{(i)}} \tag{50}$$

where k represents the index for the active channel state (*i.e.*, $k \leq K_i$).

3.2.3 Two-state Discrete Time Markov Chain (DTMC)

The third proposed group correlation model is based on the well-known Gilbert-Elliot (GE) model, which has been extensively studied to model the bursty loss scenarios at the bit level [32] or the packet level [38]. The Gilbert-Elliot model represents a two state Markov chain in which the two states represent a good state (with low error rate) and a bad state (with high error rate). A further simplified version of this model (which is generally referred to as the Gilbert model) considers a loss rate of 0 in the good state and a loss rate of 1 in the bad state, which suggests the following: all the loss events occur when the channel is in the bad state, and any self transition during the bad state triggers bursty losses.

For the considered packet loss model, the 2×2 state transition matrix is given as $\begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$, where the parameter p represents the probability of making a transition from the good state to the bad state, and the parameter q represents the probability of making a transition from the bad state to the good state. The Gilbert-Elliot model parameters are typically determined using the statistical data obtained through analyzing observations of preferably long durations. Simplifications have been made to acquire approximate model parameters with less information. For

instance, for the Gilbert model, it is sufficient to know the expected burst length, which is represented with the parameter $\hat{L}_B$, and the average loss rate, which is represented with the parameter p_L [6], to determine the values for the state transition probabilities as follows:

$$q = \frac{1}{\hat{L}_B} \text{ and } p = \frac{p_L \times q}{(1 - p_L)} \tag{51}$$

Therefore, by choosing the mean burst length and the packet loss rates associated with the given transmission channel, we can determine the parameters corresponding to the given two state on-off based Markov model. [7]

We generate the correlated loss events using the Gilbert model as follows. We start by defining the initial parameter values corresponding to the perceived loss events (*i.e.*, mean burst length and packet loss rate) for each user in N_i. Next, we select the desired correlation ratio for the given set of clients, *i.e.*, $\rho_G^{(i)}$. The selected correlation ratio is then used to find the average loss rate for the generalized loss process as follows:

$$p_{L,G}^{(i)} = (1 - \rho_G^{(i)}) \times \sum_{\forall \nu_j \in N_i} p_{L,j}^{(i)} \tag{52}$$

To find the value for the mean burst length corresponding to the generalized loss process, which is referred to as $\hat{L}_B^{(i)}$, we use the following approximation. We assume the mean burst length for the correlated loss scenario to be equal to the mean burst length for the uncorrelated loss scenario (*i.e.*, when the given set of clients observe independent packet losses). Then, using $\hat{L}_B^{(i)}$ and $p_{L,G}^{(i)}$, we can determine the Markov state transition rates for the generalized process.

Next, we need to determine the user-specific Markov state parameters. For that purpose, we make the following assumption: *client losses occur only during the loss periods associated with the generalized process.* We also allow each client to observe

[6]Note that, if the actual loss process is based on the Poisson model, then, for client i, $p_{L,i}$ equals $\lambda_i \times l_P / W_M$, where l_P represents the packet length and W_M represents the IPTV multicast rate.

[7]Note that, for DSL-based access networks, a bursty error period typically lasts for $8ms$, which suggests 2-to-3 packets long bursty loss periods for the SD-IPTV broadcasts.

independent packet losses within these loss periods, associated with the generalized process. [8] By using the steady state probabilities for the group loss model, we can update the values for the individual client loss probabilities. Specifically, we use the following equations to find the Markov model parameters for each user ν_j in N_i:

$$q_j^{(i)} = \frac{1}{\hat{L}_{B,j}^{(i)}} \text{ and } p_j^{(i)} = \frac{p_{L,j}^{(i)} \times q_j^{(i)}}{W_M^{(i)} \times p_G^{(i)}/(l_P^{(i)} \times (p_G^{(i)} + q_G^{(i)})) - p_{L,j}^{(i)}} \tag{53}$$

3.2.4 Analyzing the Correlation Statistics

Next, to more accurately capture the correlation statistics for the given group correlation model, we focus on the two approaches used by Yajnik *et.al* in [109]. The first approach is based on finding the covariance measure using the following equation:

$$Cov_{j,k}^{(i)} = \frac{n_e(I_j^{(i)}, I_k^{(i)})}{n_p^{(i)} - 1} - \mu_{I_j^{(i)}} \times \mu_{I_k^{(i)}} \tag{54}$$

where $I_j^{(i)}$ and $I_k^{(i)}$ represent the binary valued variables that correspond to the loss events observed at $\{\nu_j, \nu_k\} \in N_i$ [9], $n_e(I_j^{(i)}, I_k^{(i)})$ represents the total number of simultaneous loss events observed by $\{\nu_j, \nu_k\}$ out of $n_p^{(i)}$ source packet transmissions, and $\mu_{I_j^{(i)}}$ represents the expected packet loss ratio for ν_j.

Then, for the group correlation model corresponding to the Poisson-based packet loss scenario, the expected value for the covariance metric is found as follows:

$$E_1[Cov_{m,n}^{(i)}] = \sum_{l=1}^{\infty} \frac{(\lambda_G^{(i)}\tau_p)^l \times e^{-\lambda_G^{(i)}\tau_p}/l!}{(1 - (\bar{p}_m^{(i)})^l)^{-1}/(1 - (\bar{p}_n^{(i)})^l)} - \frac{\lambda_m^{(i)}\lambda_n^{(i)}}{\tau_p^{-2}} \tag{55}$$

where $\{\nu_m, \nu_n\} \in N_i$ and $\bar{p}_m^{(i)}$ equals $1 - p_m^{(i)}$.

We can extend the above results to find the values for the covariance metrics in the MMPP-based loss scenarios as follows:

$$E_2[Cov_{m,n}^{(i)}] = \sum_{k \le K_i} \pi_k \times \sum_{l=1}^{\infty} \frac{(\lambda_{G,k}^{(i)}\tau_p)^l \times e^{-\lambda_{G,k}^{(i)}\tau_p}/l!}{(1 - (\bar{p}_{m,k}^{(i)})^l)^{-1}/(1 - (\bar{p}_{n,k}^{(i)})^l)} - \frac{\lambda_{m,k}^{(i)}\lambda_{n,k}^{(i)}}{\tau_p^{-2}} \tag{56}$$

[8] Therefore, the correlated loss processes are essentially formed by restricting the time frames for which the client losses can occur.

[9] $I_j^{(i)}$ equals 1 if ν_j observes a packet loss, and equals 0 otherwise.

where π_k represents the steady state probability corresponding to state s_k.

For the Gilbert-Elliot model, the expected values for the covariance metrics can be obtained by using the following equation:

$$E_3[Cov_{m,n}^{(i)}] = \sum_{b=1}^{\infty} \frac{q_G^{(i)}(1-q_G^{(i)})^{b-1}}{(\hat{L}_B^{(i)}(1+q_G^{(i)}/p_G^{(i)}))} \sum_{k=1}^{b} \frac{(b-1)!}{(k-1)!(b-k)!} \frac{(\pi_{m,2}^{(i)}\pi_{n,2}^{(i)})^k}{(\pi_{m,2}^{(i)}\bar{\pi}_{n,2}^{(i)} + \pi_{n,2}^{(i)}\bar{\pi}_{m,2}^{(i)})^{k-b}} - p_{L,m}^{(i)} p_{L,n}^{(i)} \quad (57)$$

where $\pi_{j,2}^{(i)}$ represents, for client $\nu_j \in N_i$, the steady state probability of being in the bad state and $\bar{\pi}_{i.2}^{(i)} = 1 - \pi_{i.2}^{(i)}$.

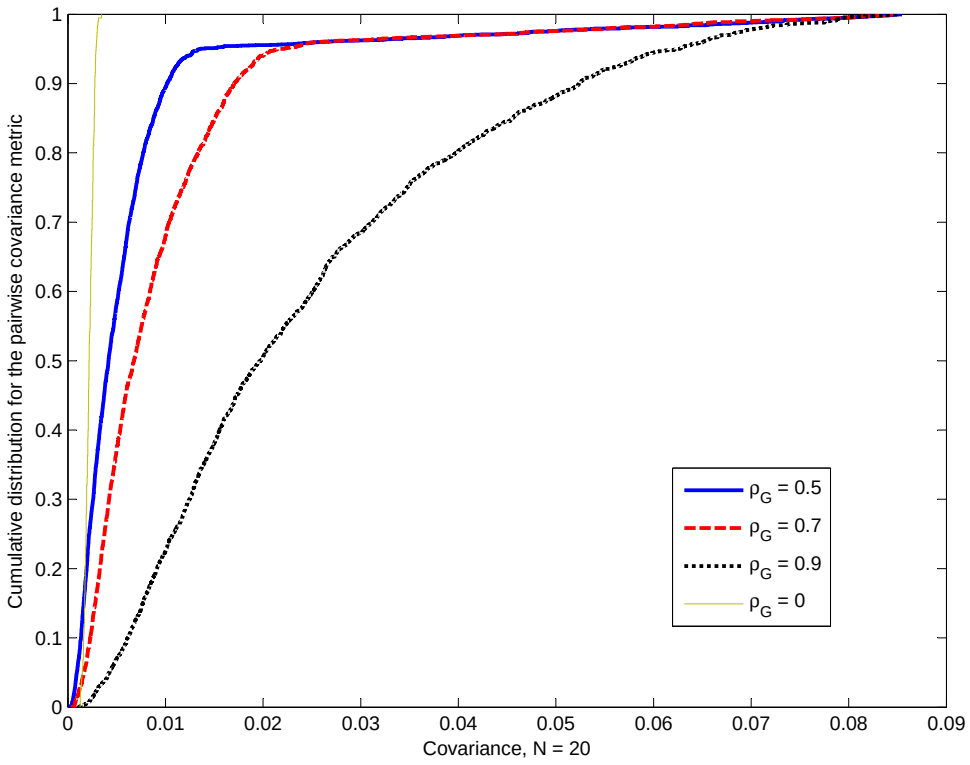


Figure 14: Comparative group loss results based on the covariance metric, when Poisson process is used.

Figure 14 demonstrates the relationship between ρ_G and the cumulative distribution for the covariance metrics, when $N_i = 20$. [10] If we compare the results from

[10]Note that, the convergence effect observed beyond the 0.9 value line for the $\rho_G > 0$ scenarios is created by the autovariance effect. We included the autocovariance results to illustrate their relationship to the crosscovariance results.

Figure 14 to the results from Figure 13, we observe that the distribution for the covariance metrics gives a more clear representation of the actual impact of the correlation metric, ρ_G. We observe similar results for the more general 2-state MMPP model, as shown in Figure 15, and the Gilbert-Elliot model, as shown in Figure 16.

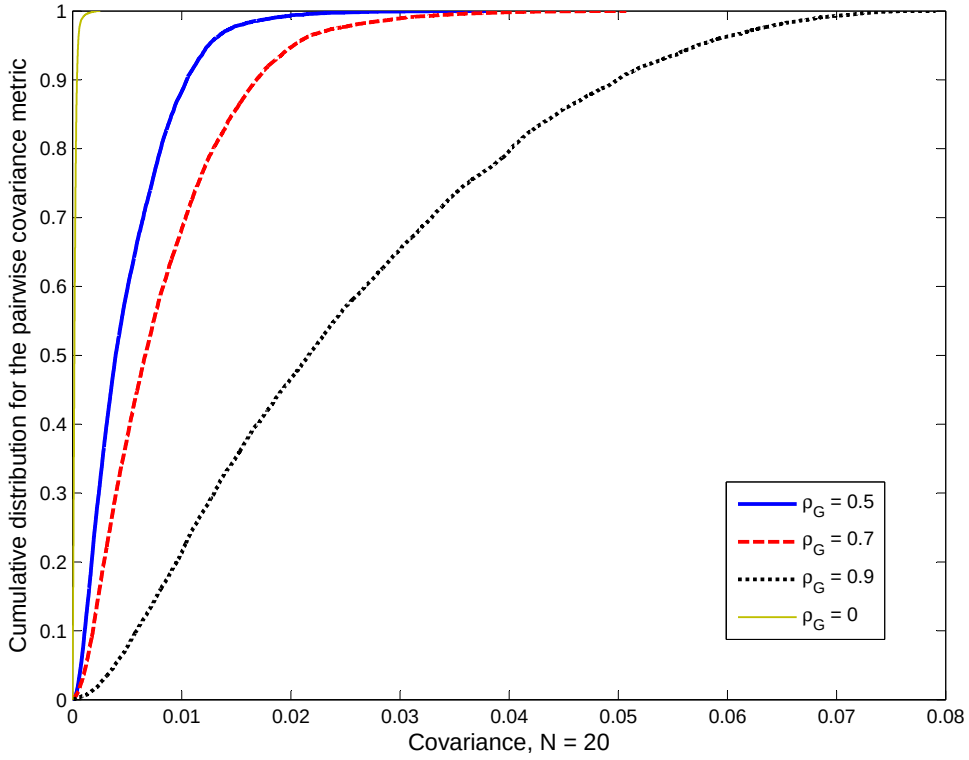


Figure 15: Comparative group loss results based on the covariance metric, when 2-state MMPP model is used.

The second approach to quantify the correlation statistics considered in [109] is based on finding the distribution for the simultaneous packet loss count. Figure 17 shows the results for the second approach where the Poisson-process-based loss correlation model is used. Here, the results also reflect our earlier observations for the covariance metrics. The results also suggest that when the value assigned for ρ_G is not very high, we may observe a noticeable increase in the user overhead. However, if we group the source packets into recovery blocks, we may observe the opposite. In Figure 18, we show the distribution of packet loss events when five consecutively

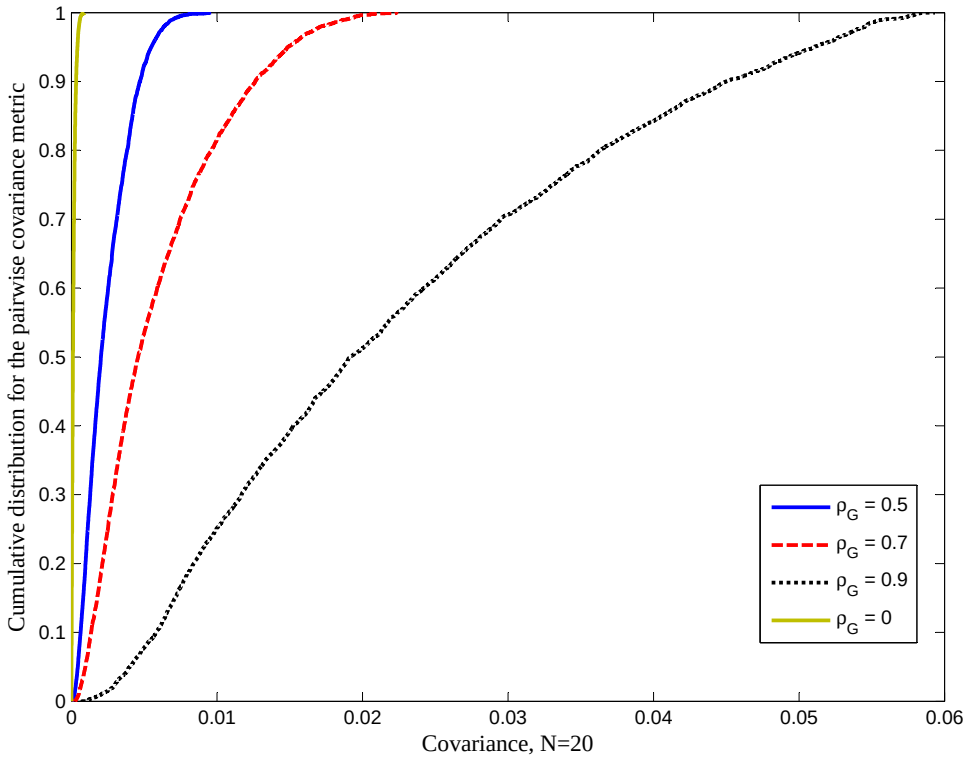


Figure 16: Comparative group loss results based on the covariance metric, when 2-state DTMC model is used.

transmitted packets are evaluated together. We observe that, in many cases, more than half of the users observe at least one packet loss during the transmission time of a source block of five packets.

3.2.5 Group-based Recovery Approach

Therefore, to exploit the overhead efficiency for multicast-based recovery in block transmission scenarios, we propose a joint recovery approach that exploits the advantage introduced by the use of Application Layer FEC (AL-FEC) packets and the decision thresholds within a multicast-based recovery framework. Here, multicast decision threshold represents the minimum number of repair requests ERServ needs to receive for a specific packet to initiate a multicast-based recovery for the given packet. If the number of received requests for the given packet is less than the multicast threshold, then ERServ uses unicast-based recovery, otherwise, it uses

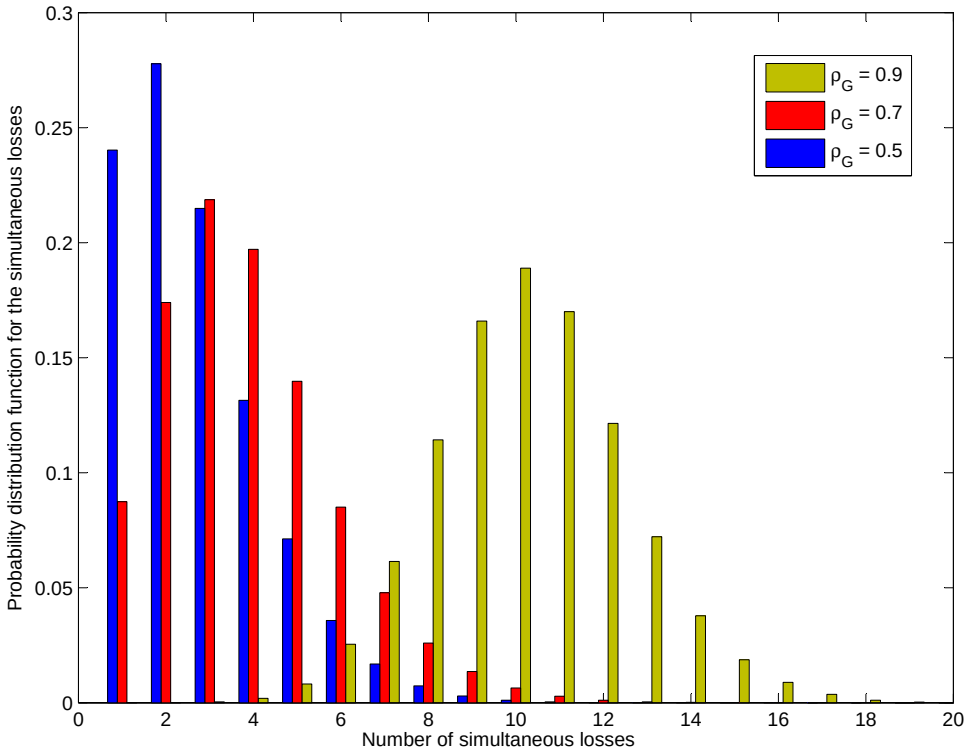


Figure 17: Probability distribution for packet loss events, when $N_i = 20$ and $L = 5$.

multicast-based recovery.

The proposed AL-FEC-based multicast recovery process is essentially based on the grouping of the source packets into L_{fec} sized recovery blocks. Each of these source packet blocks is protected by a single AL-FEC packet. Therefore, each received request is evaluated together with all the other received requests corresponding to the same recovery block.

To initiate the error recovery process, ERServ does an initial multicast of the AL-FEC packet associated with recovery block corresponding to the received requests. If the received AL-FEC packet is not sufficient to recover from the losses observed at all the concerned users (*e.g.*, for users observing multiple failed deliveries within the same recovery block), then the multicast threshold metric is used to decide whether to perform a multicast- or a unicast-based retransmission. Specifically, if the number of non-recovered losses for a given packet (*i.e.*, losses that cannot be recovered using the

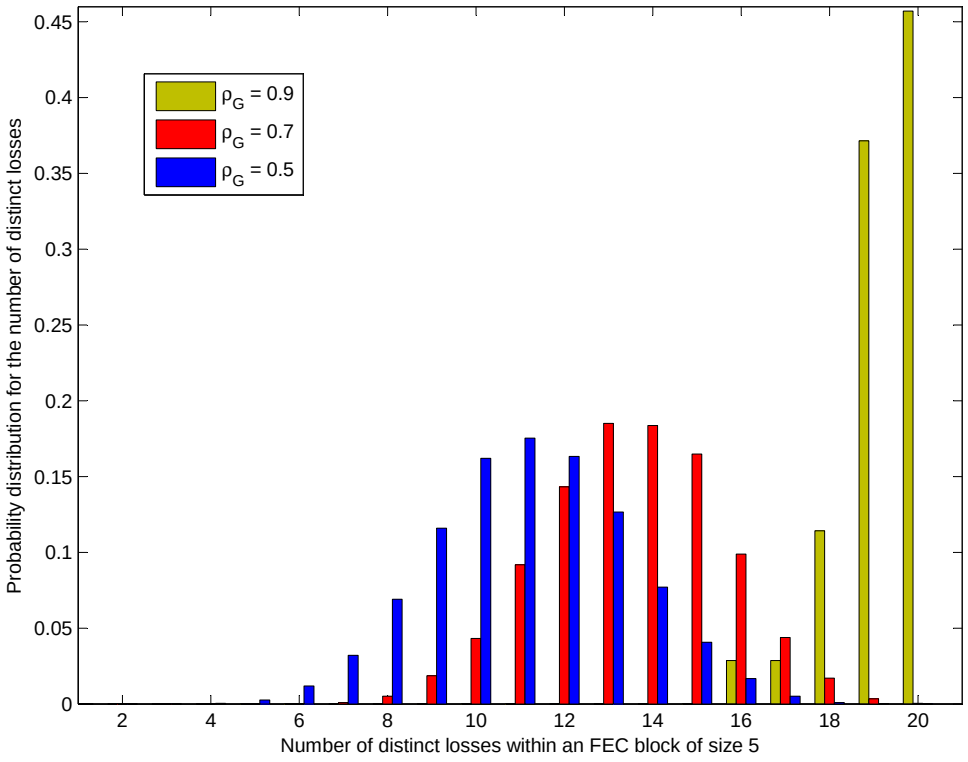


Figure 18: Probability distribution for packet loss events, when $N_i = 20$ and $L = 5$.

AL-FEC packet) is higher than the multicast threshold, then the ERServ performs a multicast-based recovery by transmitting the source packet to all the users within that multicast group. Otherwise, the ERServ performs a unicast-based recovery by sending the source packet to only the users that require additional repair packets to recover from their losses.

3.3 Performance Analysis

In this section, we evaluate the impact of correlated user losses on the IPTV error recovery performance. We specifically compare the performances of multicast- and unicast-based recovery approaches by measuring the error recovery overhead at the ERServ and the end-users. The performance evaluations presented in this section mainly focus on the results corresponding to the Poisson process-based group correlation loss model, as the given model is sufficient to illustrate the impact of the proposed error recovery technique in IPTV networks.

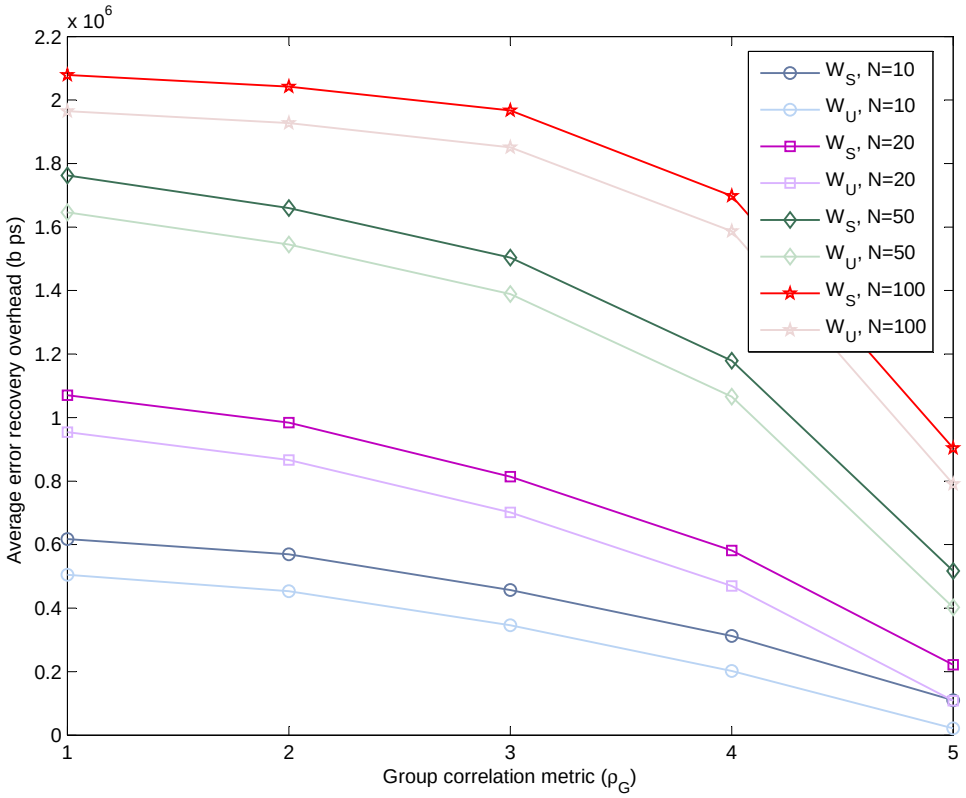


Figure 19: Impact of the size of correlated user set on the error recovery overhead.

The first set of results, as shown in Figure 19, illustrate the dependence of multicast recovery overhead on ρ_G and N_i. We observe that, as the size of the user set increases, so does the multicast recovery overhead at the user side. To minimize this overhead, we need to keep the size of multicast recovery groups small.

Table 1: Error Recovery Overhead (in Mbps) when $L_{fec} = 5$

	$N_i = 20$			$N_i = 50$		
ρ_G	0.5	0.7	0.9	0.5	0.7	0.9
$W_S^{(u)}$	2.24	2.25	2.28	5.75	5.73	5.66
$W_S^{(m)}$	0.894	0.594	0.223	1.51	1.18	0.512
$W_U^{(m)}$	0.782	0.482	0.109	1.39	1.07	0.399
$W_{S,fec}^{(m)}$	0.414	0.428	0.269	0.294	0.470	0.458
$W_{U,fec}^{(m)}$	0.339	0.334	0.114	0.260	0.410	0.361

Next, we study the impact of using AL-FEC, when the length of the recovery block equals five. The comparative results for unicast recovery and multicast recovery

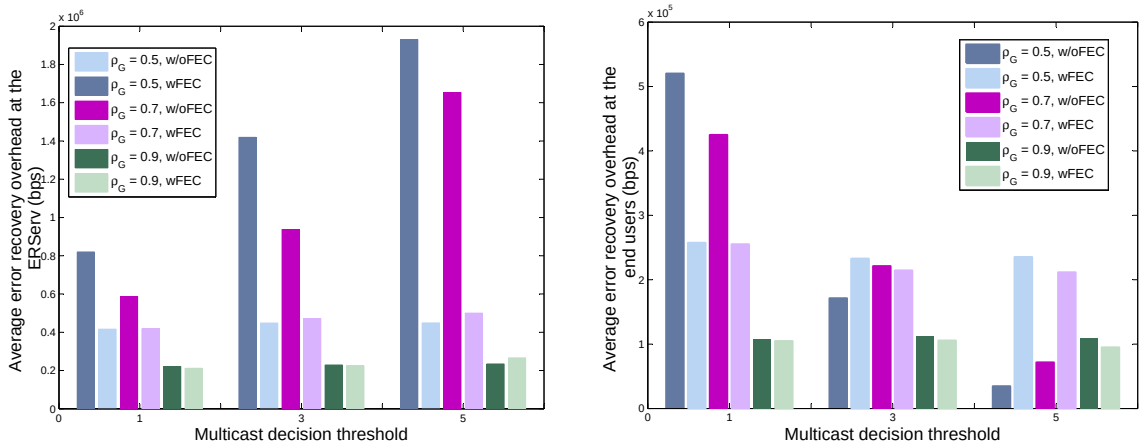
with/without FEC are shown in Table 1.[11] We observed significant improvements when using FEC-based multicast, especially when ρ_G is not assigned a very high value. However, the advantage of using FEC disappears when packet loss events are shared by most of the users. We also observed that, as we increase the size of the user set, the performance of FEC-based multicast started to show better results for low-to-mid loss correlation scenarios. This is caused by the increased rate of loss events that occur within an FEC-protected block. Consequently, by allowing more users to take advantage of the FEC-based multicast, we can effectively reduce the number of further required retransmission requests.

We next analyze the impact of FEC-based multicast and multicast decision threshold (δ_{thr}) on the error recovery performance. The results are shown in Figure 20 for 20 users and in Figure 21 for 50 users. In our simulations, δ_{thr} is selected from the set $\{1, 3, 5\}$. We achieved the best performance tradeoffs with regards to the recovery overhead when FEC-based multicast is employed. For the non-FEC-based multicast recovery, the resulting performance strongly depends on the value of δ_{thr}, whereas utilizing an FEC-based initial multicast minimizes the need for dynamically varying the multicast decision thresholds.

For the non-FEC-based multicast recovery scenario, we also observed significant fluctuations in the error recovery performance as we varied the values of ρ_G and N_i. Here, FEC-based multicast can be used to reduce the dependence of the error recovery performance on the varying network conditions, and, in doing so, improve the scalability performance of the IPTV networks.

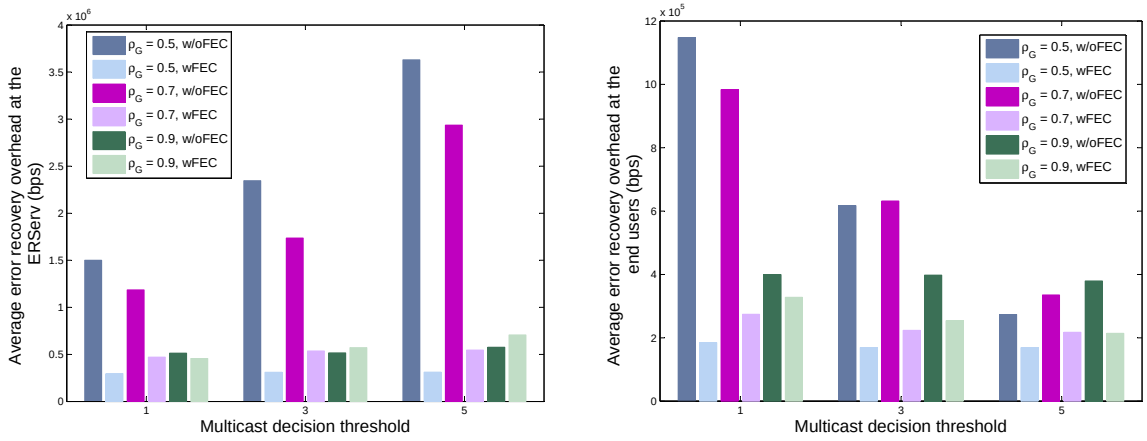
Next we study the impact of varying block sizes on the FEC-based multicast recovery. The results are shown in Figure 22, when $L_{fec} \in \{5, 10\}$ and $\delta_{thr} = 0$. We observed that regardless of the size of the user set, increasing the FEC-block size does

[11] $W_{loc}^{(app)}$ represents the overhead at *loc* (server, S, or, user, U) for *app*-based recovery (unicast, U, or, multicast, M).



(a) Server overhead (b) User overhead

Figure 20: Impact of decision threshold on error recovery overhead when $N_i = 20$.



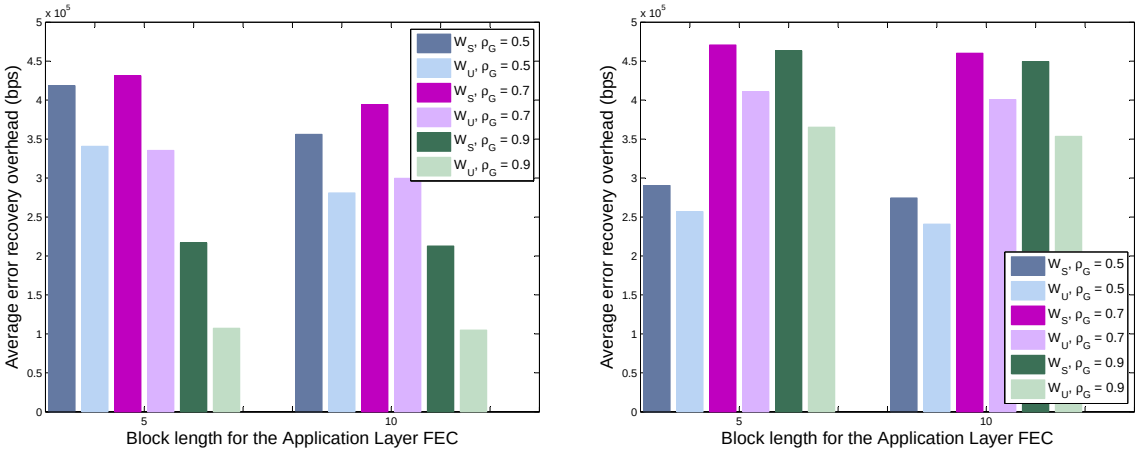
(a) Server overhead (b) User overhead

Figure 21: Impact of decision threshold on error recovery overhead when $N_i = 50$.

not significantly improve the recovery performance. However, as shown in Figure 23, at higher FEC-block sizes, error recovery performance becomes more dependent on the value of decision threshold, δ_{thr}.

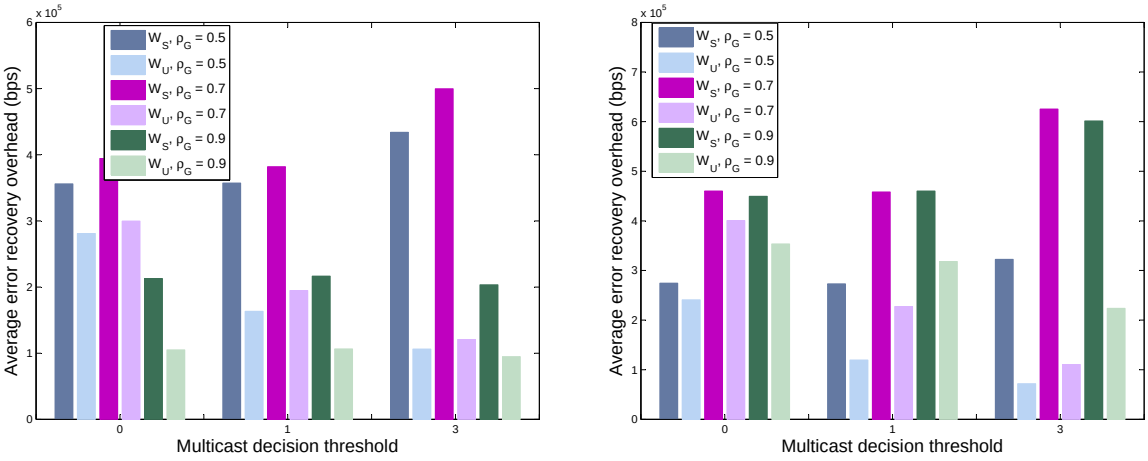
3.4 Conclusions

In this chapter, we studied the impact of spatially correlated packet losses on the error recovery performance in IPTV networks. We proposed three different group correlation loss models to create spatially correlated packet loss events to investigate the



(a) N_i=20 (b) N_i=50

Figure 22: Impact of FEC block size on error recovery overhead.



(a) N_i=20 (b) N_i=50

Figure 23: Impact of multicast decision threshold when FEC block size equals 10.

error recovery tradeoffs in IPTV networks at different correlation levels. Furthermore, to exploit the correlated losses in the network, we proposed an FEC-based multicast approach to improve the scalability performance of the IPTV network while limiting the overhead at the client side. The simulation results showed significant performance improvements for the proposed joint error recovery framework for group correlated IPTV networks.

CHAPTER IV

A SERVER-ASSISTED PEER-BASED ERROR RECOVERY FRAMEWORK FOR IPTV NETWORKS

4.1 Introduction

In the previous chapters, we investigated the use of server-based error recovery solutions to combat the lossy nature of the transmission medium along the last mile in, first, uncorrelated, and then, correlated packet loss scenarios. In these chapters, through theoretical and experimental studies, we showed the effectiveness of using server-based recovery techniques to achieve the desired service quality levels in IPTV networks. However, in general, there is a strong correlation between the performance of server-based solutions and the system capacity. Since server-based solutions oftentimes utilize unicast-based delivery techniques, as the number of users connected to an Error Recovery Server increases, scalability problems may arise quickly. In such scenarios, because of an increased likelihood of receiving too many requests within a short time-frame, requests at the Error Recovery Server would experience significant queueing delays. Since IPTV requires low latency to attain the acceptable service quality levels, additional queueing delays may prevent the repair packets to be delivered in time for decoding. Considering the strict packet loss requirements for the IPTV service, end-users may observe unacceptable performance degradations.

To limit the problems encountered during server-based recovery, peer-based recovery techniques are proposed (*e.g.*, [64]) that allow the users to send their error recovery requests to their peers (*i.e.*, users connected to the same multicast session). For instance, in [64], each user, observing a packet loss, sends its repair request to a few selected peers to maximize the probability of a successful recovery. Error recovery

server is only utilized when the number of peers is less than a threshold. However, since multiple peers are needed to repair a single packet, if the users observe high packet loss rates, repair packet traffic can significantly overburden the uplink connection at the end-users.

To limit the performance degradation impact of high packet loss rates on the recovery peers, in this chapter, we propose a server-assisted peer-based error recovery technique, which utilizes the Error Recovery Server as a backup for the peer-recovery process instead of probing multiple peers for a single packet loss. In the proposed approach, Error Recovery Server is utilized to help recover lost packets whenever peer-based recovery fails to achieve its objective. We introduce a recovery framework that uses control message exchanges within the network to increase the efficiency of the error recovery process. We also propose a fair resource allocation technique that efficiently integrates user capabilities into the user selection process. We analyze the proposed peer-based recovery model and develop equations required to evaluate its performance. We essentially measure how the proposed approach fares against server-based recovery in terms of recovery overhead, success rate, scalability, and latency. We show that we can significantly improve the scalability performance of the error recovery process without introducing significant overhead with respect to the latency and peer usage ratio performance measures.

The rest of the chapter is organized as follows. In Section 4.2 we present the system model for the proposed peer-based recovery framework. Section 4.3 presents a detailed analysis of the proposed error recovery architecture. Performance evaluations are presented in Section 4.4. Section 4.5 concludes the chapter.

4.2 System Model

The proposed system consists of a single Error Recovery Server, S, and N users. These users are connected to one of the M multicast sessions that are supported by

S. In Figure 24, we illustrate the basic message exchange process that is utilized by the end-users to initiate, proceed with, and finalize the error recovery process. Note that, each recovery attempt involves the Error Recovery Server, the user making the request, and the selected recovery-peer. We can explain the basic operation for the error recovery process as follows.

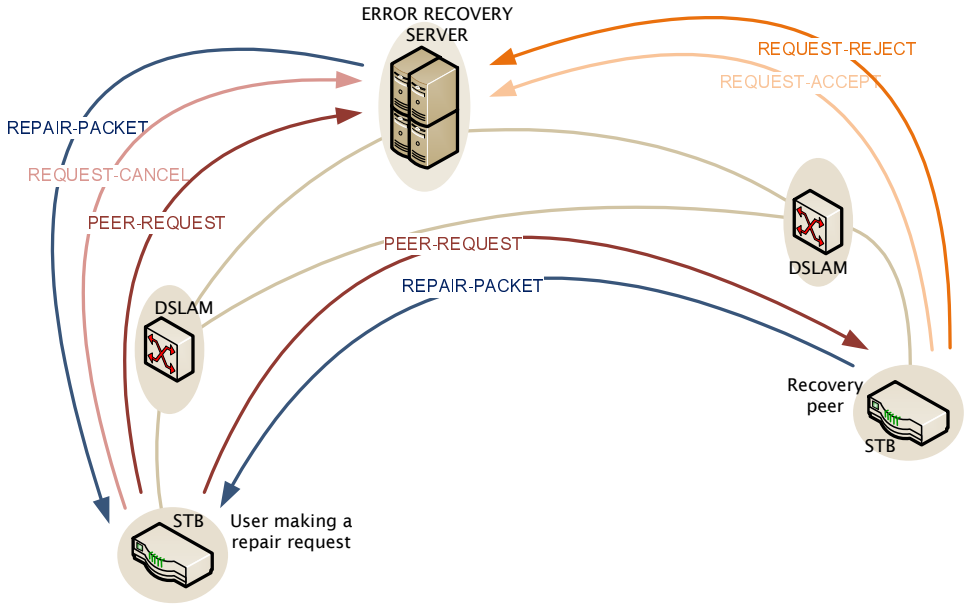


Figure 24: Message exchanges for the peer-based error recovery framework.

The protocol initiates after a packet loss is observed. We refer to the user observing the loss as ν. The first objective is to find ν's recovery-peer. For that purpose, ν uses a potential list of candidate peers (which is generated using the information received from the Error Recovery Server) and selects the recovery-peer using a weighted assignment procedure, for which the details (*i.e.*, how the weights are assigned) are presented in the next section. After the recovery-peer is selected, ν prepares the *peer-request* packet, using the information on the loss and the recovery-peer. Then, ν sends peer-request to both the recovery-peer and the Error Recovery Server. By sending peer-request packet to also the Error Recovery Server, we enable the timely process for the server to act as a recovery-peer, if needed (*e.g.*, when the initial recovery-peer has left the multicast session and ν is not informed of that before making its

request). In doing so, we can limit the problems that may arise because of having outdated information on the other peers. Additionally, we can enable quick recovery if peer-request packet fails to reach its destination. For this purpose, we associate each peer-request packet with a timeout period and we require each recovery-peer to send a *request-update* packet to the Error Recovery Server as soon as the recovery request is received and accepted. Consequently, the proposed approach can minimize the need to probe multiple end-users in response to a single packet loss. Furthermore, it allows the recovery-peer to reject a received request for not having sufficient resources to respond to the request in time. In short, the recovery-peer uses two types of request-update messages to send to Error Recovery Server: (*i*) *request-accept* and (*ii*) *request-reject*.

If the request is accepted by the recovery-peer, it sends *request-accept* packet to the Error Recovery Server and repair packet to ν, which then acknowledges the delivery by sending a *request-cancel* message to the Error Recovery Server. However, if the Error Recovery Server does not receive the *request-cancel* message in time, then the server assumes peer-based recovery failed, and proceeds with the recovery process by sending the repair packet directly to ν. Similarly, if the request is rejected by the recovery-peer, or the recovery-peer fails to update the Error Recovery Server in time, then the Error Recovery Server finalizes the recovery process by sending the repair packet to ν. The proposed approach can limit the disruptive effects of losing the repair packet before it reaches its destination, while also avoiding unnecessary repair packet transmissions by the Error Recovery Server.

Note that since the overhead associated with the transmission of a control packet is considered to be negligibly small when compared to the size of a repair packet (approximately 25 times smaller), with the proposed message exchanges, total recovery overhead at the Error Recovery Server can be significantly reduced. Furthermore, by assigning a higher priority to control message transmissions on the uplink channel

compared to repair packet transmissions, we can allow the Error Recovery Server to select a reasonably small value for the timeout period without observing a noticeable increase in the incorrect decision rate [1].

4.3 Performance Analysis

In this section, we present the technical details on the proposed peer-based error recovery framework and analyze its performance. We start by establishing an upper bound on each user's contribution to the error recovery process, after taking into account the user specific resource limitations. Then, we integrate the impact of delivery latency into the recovery framework to finalize the expected values for user contributions to achieve a fair resource allocation. Lastly, we determine the equations that represent the error recovery overhead and the success rate for the proposed error recovery framework.

4.3.1 Impact of Bandwidth Constraints

To determine each user's contribution to the peer-based error recovery process, we need to find the maximum number of requests each user can serve, which is given as the minimum of peer-request receiving rate and request-update transmission rate:

$$n_i^* = \min(n_i^u, n_i^d) \tag{58}$$

where n_i^* represents the maximum number of repair packet requests that i can serve within unit time-frame, n_i^u represents the request-update transmission rate, and n_i^d represents the peer-request receive rate. Our goal in this section is to find the values of n_i^d and n_i^u. For that purpose, we need to find the equations that represent bandwidth constraints on the downlink and uplink channels.

We use d_i and u_i to represent the downlink and the uplink bandwidth constraints

[1] If the number of packets that belong to the same protection period (or transmission block), is sufficiently high, then a high timeout value can be selected without affecting the probability of a successful delivery.

for i. Downlink bandwidth is reserved for the delivery of source packets, repair packets, and peer-recovery requests, whereas uplink bandwidth is reserved for the transmission of repair packets and control messages.

We use $\beta_{sd}^{(i)}$ and $\beta_{er}^{(i)}$ to represent the portion of downlink bandwidth that is used for data transmissions and error recovery, *i.e.*, $d_i = \beta_{sd}^{(i)} + \beta_{er}^{(i)}$. Since $\beta_{sd}^{(i)}$ is assumed as a constant-valued parameter, our focus will be on $\beta_{er}^{(i)}$, which is a summation of repair packet load, $\beta_{er}^{r,i}$, and control message load, $\beta_{er}^{q,i}$, *i.e.*, $\beta_{er}^{(i)} = \beta_{er}^{r,i} + \beta_{er}^{q,i}$. We use d_i^* to represent the upperbound on $\beta_{er}^{(i)}$, and l_r (or l_q) to represent the size of a repair (or control) packet. Using these parameters, we achieve the inequality of $(l_r n_{r,i}^d + l_q n_{q,i}^d) \leq d_i^*$, where $n_{r,i}^d$ and $n_{q,i}^d$ represent the average number of repair packets and control messages transmitted within unit time-frame.

To find the maximum number of requests that can be received on the downlink channel, we use $E[n_{r,i}^d]$, which represents the expected number of repair packets received within unit time-frame. We can find the value of $E[n_{r,i}^d]$ using packet loss rates and the expected number of repair packets that are received for each repair request. Consequently, we state the upperbound on the value of n_i^d (or $n_{q,i}^d$) as follows:

$$n_i^d = (1/l_q) \times \left(d_i^* - l_r E[n_{r,i}^d]\right) \tag{59}$$

We use a similar methodology to find the number of repair packets that can be transmitted on the uplink channel, which is reserved for the transmission of repair packets and two types of control messages (*i.e.*, *request-update* and *request-peer* [2]). We use $\beta_{er,u}^{r,i}$ and $\beta_{er,u}^{q,i}$ to represent the portion of uplink bandwidth that is used for repair and control packet transmissions (*i.e.*, $u_i = \beta_{er,u}^{r,i} + \beta_{er,u}^{q,i}$). To find the limits on uplink bandwidth utilization, we assume that the number of *request-update* messages transmitted by a recovery-peer equals the number of repair packets the same user transmits [3]. To estimate the number of *request-peer* messages transmitted within unit

[2] *Request-update* is sent by the recovery-peer, whereas *request-peer* is sent by the user observing the loss.

[3] It is possible for uplink channel to be congested causing extra delay for incoming requests. If

time-frame, we use the information on the average packet loss rate for the downlink channel, which is represented with $\lambda_i^{(d)}$. The following inequality is then used to find the upperbound on the number of repair packets that user i can transmit within unit time-frame:

$$n_i^u \leq \big(u_i^* - 3\lambda_i^{(d)} l_q\big)/\big(l_r + l_q\big) \tag{60}$$

Next, we integrate the maximum values of n_i^d and n_i^u into (58) to find the servicing rate for each user as a recovery-peer. These servicing rates are used to select the recovery-peers based on the following approach. Assume that $M_i[n]$ represents the set of recovery-peers [4] that can serve i's request for the nth source packet on time. Then i selects j as its recovery-peer to recover from the loss of the nth source packet with probability $p_{i,j}[n]$:

$$p_{i,j}[n] = \Big[\sum_{\forall k \in M_i[n]} (n_k^*/n_j^*)\Big]^{-1} \tag{61}$$

Therefore, the average probability of selecting j as i's recovery-peer, $p_{i,j}$, is given by $\big(\sum_{\forall n \in \pi} p_{i,j}[n]/|\pi|\big)$, where $|\pi|$ represents the size of transmission block π.

Note that, because of the varying round-trip-times between different user pairs, recovery-peer sets differ from one packet transmission time to another. As a result, there is a non-negligible probability of underutilizing some of the recovery-peers, which may lead to overutilization of recovery resources at some other recovery-peers, including the Error Recovery Server. We need to take into account these differences in our decision process so that we can improve the accuracy of our calculations as well as the fairness during the peer-assignment process. The next subsection proposes a solution to minimize the impact of time-varying peer-recovery sets.

---

such congestion leads to a failed delivery, then recovery-peer (i) stops the delivery attempt for the repair packet and (ii) informs the Error Recovery Server regarding its decision.

[4] The recovery-peer set also includes the Error Recovery Server, for which the expected servicing rate is given as $w_S = n_0^* = \lceil \sum_{\forall j \in M_i} \lambda_j^{(d)} - \sum_{\forall j \in M_i} n_j^* \rceil^+$, where $\lceil x \rceil^+ = \max(x, 0)$.

4.3.2 Parameter Update to Improve Fairness

Assume that for a given transmission block there are K distinct periods with different recovery-peer sets. We use U_k to represent the recovery-peer set during the kth period [5]. To allocate recovery resources using the initially assigned rates, we use Algorithm 5, which shows the parameter update procedure performed at a single user.

To evenly distribute the error recovery resources among the recovery-peers, our goal is to select the peers, which cannot be used during the later periods, at a higher rate, whenever it is possible. We refer to the recovery-peers that cannot be used during the subsequent transmission periods as departing peers. Therefore, rate assignment process initiates with the selection of the departing peers' rates. To do this, we increase their rates in such a way that, on average, they can be selected at the desired rates. After the rates for the departing peers are selected, we distribute the unused portion of the overall error recovery bandwidth to other recovery-peers based on their initially assigned rates. If it is not possible for the departing peers to get assigned at the desired rates, we maximize their overall assignment rates by only selecting them as recovery-peers during the given transmission period. After we assign the rates to all possible recovery-peers, we update non-departing peers' rates by subtracting the already assigned portion during the current transmission period.

The resulting weight parameters are then used to find the peer selection rates for user ν, *i.e.*, $p_{\nu,j}[n] = w_{\nu,j}^{(l)}$, where n represents a packet sequence number from the given transmission block, l is given by $\sum_{1\leq k\leq l-1} T_{\nu,k} \leq n\tau < \sum_{1\leq k\leq l} T_{\nu,k}$, and τ represents the inter-arrival time for the source packets.

To further improve the fairness performance of the rate assignment process and achieve a more scalable error recovery, we propose an approach that utilizes the impact of relative user distances on usage rates. We can explain our reasoning for the selected

[5]Note that $U_{k-1} \supset U_k \supset U_{k+1}$.

approach as follows. Compared to nearby peers, far-distant peers are more likely to be used at a rate less than their fair shares, leading to the Error Recovery Server being utilized more than needed. Furthermore, we start observing discrepancy between usage rates of near-by and far-away peers. To improve the scalability performance, we need to avoid overutilizing recovery resources at the Error Recovery Server, as long as the resources at the recovery peers are sufficient to support the error recovery process. To achieve this objective, we increase the initially assigned weights for the nearby peers using the following methodology.

Assume that the current usage rate for user i is given by ω_i [6] and the desired usage rate is given by ω_i^*. Then, for user i, the unused portion of the recovery bandwidth, which is referred to as $\beta_u^{(i)}$, is found by using:

$$\beta_u^{(i)} = (\omega_i^* - \omega_i) \times \frac{\sum_{\forall j \in M} \lambda_j}{\sum_{\forall j \in M} \omega_j^*} \tag{62}$$

We redistribute these unused recovery resources to the nearby users, since they are the ones that are utilized during the latter sections of a transmission block. We represent the users that can send a request to user j during the last k sections of a transmission block using the set $U_R^{(j,k)}$. The rate of increase in j's usage ratio by i that belongs to the set $U_R^{(j,k)}$ is given by:

$$\beta_u^{(j,i)} = \Big[\sum_{\forall i \in U_R^{(j,k)}} (\lambda_i / \beta_u^{(j)}) \Big]^{-1} \tag{63}$$

We can then find the usage ratio that user i can redirect from the Error Recovery Server to its recovery-peers by using:

$$\widetilde{\omega}_i = \sum_{\forall j \in M} \beta_u^{(j,i)} \tag{64}$$

where $i \in M$.

[6] The value of ω_i is found by using $(\sum_{\forall j} (\lambda_j^{(d)} \times p_{j,i}))$.

$$\Omega_U^{(D)} = \sum_{\forall i \in M} \frac{l_r \breve{\lambda}_i^{(d)}}{N^2} \Bigg(\sum_{\substack{\forall m \\ N_m \leq 1}} \frac{\lambda_i^{(d)}}{N_m^{-1}} + \sum_{\substack{\forall m \\ N_m > 1}} N_m \Bigg[\lambda_i^{(d)} \Big(\widetilde{\omega}_S^{(i)} + \breve{\widetilde{\omega}}_S^{(i)} \sum_{\substack{\forall j \in M \\ j \neq i}} \bar{p}_{i,j} \breve{\lambda}_j^{(d)} \Big[\zeta_m^{(j)} + \breve{\zeta}_m^{(j)} \times \big(\varrho_j + \breve{\varrho}_j \breve{\lambda}_j^{(u)} \big) \Big] \Big) + \sum_{\substack{\forall j \in U \\ j \neq i}} \frac{l_q}{l_r} \lambda_j^{(d)} \gamma_{j,i} \widetilde{p}_{j,i} \Bigg] \Bigg) \tag{68}$$

$$\Omega_U^{(U)} = \sum_{\forall i \in M} \frac{l_q}{N^2} \Bigg(\sum_{\substack{\forall m \\ N_m \leq 1}} \frac{\lambda_i^{(d)}}{N_m^{-1}} + \sum_{\substack{\forall m \\ N_m > 1}} \frac{\breve{\lambda}_i^{(d)}}{N_m^{-1}} \Bigg[\lambda_i^{(d)} \Bigg(\widetilde{\omega}_S^{(i)} + \breve{\widetilde{\omega}}_S^{(i)} \Bigg[2 + \sum_{\substack{\forall j \in U_m \\ j \neq i}} \bar{p}_{i,j} \breve{\lambda}_j^{(d)} \breve{\zeta}_m^{(j)} \breve{\varrho}_j \breve{\lambda}_j^{(u)} \Bigg] \Bigg) + \sum_{\substack{\forall j \in U_m \\ j \neq i}} \lambda_j^{(d)} \bar{p}_{j,i} \breve{\zeta}_m^{(i)} \Big(1 + \frac{l_r}{l_q} \breve{\varrho}_i \Big) \Bigg] \Bigg) \tag{69}$$

If the value of $\widetilde{\omega}_i$ is greater than the value of $\omega_S^{(i)}$, then we adjust the extra usage parameter targeting a recovery peer by using:

$$\widetilde{\omega}_{i,j} = \beta_u^{(j,i)} \times (\omega_S^{(i)} / \widetilde{\omega}_i) \tag{65}$$

On the other hand, if the value of $\widetilde{\omega}_i$ is less than the value of $\omega_S^{(i)}$, then the extra usage parameter targeting the recovery-peer j becomes $\widetilde{\omega}_{i,j} = \beta_u^{(j,i)}$. Then, the final usage ratio for the Error Recovery Server by user i becomes $\widetilde{\omega}_S^{(i)} = \lceil \omega_S^{(i)} - \widetilde{\omega}_i \rceil^+$.

Consequently, the equations that represent the finalized usage rates for the Error Recovery Server and the recovery-peers are given by:

$$\widetilde{\omega}_S = \sum_{\forall i \in M} \lambda_i^{(d)} \times \widetilde{\omega}_S^{(i)} \tag{66}$$

and

$$\widetilde{p}_{i,j} = p_{i,j} + I(\omega_S^{(i)} \leq \widetilde{\omega}_i) \times \omega_S^{(i)} \times (\beta_u^{(j,i)} / \widetilde{\omega}_i) + I(\omega_S^{(i)} > \widetilde{\omega}_i) \times \beta_u^{(j,i)} \tag{67}$$

4.3.3 Error Recovery Overhead and Probability of a Successful Recovery

In this section, we find the equations that represent the error recovery overhead and the probability of a successful recovery using the updated assignment rates for the recovery-peers. For the sake of simplicity, we assume that the request packets

transmitted on the uplink channel experience no error. [7] Hence, we observe two distinct packet loss scenarios: (i) losing all types of packets on the downlink channel, and (ii) losing repair packets on the uplink channel.

Using the delivery scenarios illustrated in Figure 24, we find the average recovery overhead at the end-users by using (68) and (69), where $\Omega_U^{(D)}$ (or $\Omega_U^{(D)}$) represents the error recovery overhead on the downlink (or uplink) channel, $\zeta_m^{(j)}$ represents the probability of recovery-peer j leaving the multicast session m, ϱ_j represents the probability of rejecting a received recovery request (which can happen when the queueing time at the recovery-peer exceeds the maximum allowed delay [8]), N_m represents the number of users within m, $\breve{x} = (1 - x)$, and $\bar{p}_{i,j} = \widetilde{p}_{i,j}/(\sum_{\forall j} \widetilde{p}_{i,j})$.

To find the value of N_m, we assume that the users served by an Error Recovery Server are distributed to each multicast session using the Zipf distribution, which is found as an accurate model to represent the popularity of channels in a large IPTV system [81]:

$$f(r; s, M) = \frac{1/r^s}{\sum_{1 \leq l \leq M}(1/l^s)} = \Big[\sum_{1 \leq l \leq M} (r^s/l^s) \Big]^{-1} \tag{70}$$

where r represents the popularity rank of a given channel and s represents the value of the exponent that characterizes the given distribution.

Therefore, the expected number of users belonging to the mth multicast session, which is considered as the r_mth most popular multicast session, is given by:

$$E[N_m] = \frac{\left(N/r_m^s\right)}{\sum_{1 \leq l \leq M} \left(1/l^s\right)} \tag{71}$$

To find the downlink overhead at the Error Recovery Server, Ω_S, we focus on two recovery scenarios: (i) *indirect recovery*, where the server assists an ongoing peer-recovery process, and (ii) *direct recovery*, where a user makes a direct recovery request to the server.

[7] Since the request packets can be protected without introducing significant overhead, their first-hop delivery is assumed to be error-free.

[8] To find the value of ϱ_j, we can model the uplink queue at each recovery-peer using the $M/D/1$ queueing model. Due to space considerations, we leave its discussion to a future work.

$$\Omega_S^{(D)} = \sum_{\forall i \in M} \frac{\lambda_i^{(d)} \times l_r}{N} \times \Bigg[\sum_{\substack{\forall m \\ N_m \leq 1}} N_m + \sum_{\substack{\forall m \\ N_m > 1}} N_m \times \Bigg(\widetilde{\omega}_S^{(i)} + \breve{\widetilde{\omega}}_S^{(i)} \times \sum_{\substack{\forall j \in m \\ j \neq i}} \bar{p}_{i,j} \times \Big[\zeta_m^{(j)} + \breve{\zeta}_m^{(j)} \times \varrho_j \Big] \Bigg) \Bigg] \tag{72}$$

$$\Omega_S^{(U)} = \sum_{\forall i \in M} \frac{\lambda_i^{(d)} \times l_q}{N} \times \Bigg[\sum_{\substack{\forall m \\ N_m \leq 1}} N_m + \sum_{\substack{\forall m \\ N_m > 1}} N_m \times \Bigg(\widetilde{\omega}_S^{(i)} + \breve{\widetilde{\omega}}_S^{(i)} \times \sum_{\substack{\forall j \in m \\ j \neq i}} \bar{p}_{i,j} \times \Big[\lambda_j^{(d)} + \breve{\lambda}_j^{(d)} \times \Big(2\zeta_m^{(j)} + \breve{\zeta}_m^{(j)} \big[2 \times \varrho_j + 3 \times \breve{\lambda}_j^{(u)} \times \breve{\varrho}_j \big] \Big) \Big] \Bigg) \Bigg] \tag{73}$$

Indirect recovery occurs when the selected recovery peer cannot transmit the repair packet, because of not receiving a request-peer message or not having the resources to support the recovery. *Direct recovery*, on the other hand, occurs when a user makes a direct request to the Error Recovery Server, for reasons stated earlier. To find the uplink overhead at the Error Recovery Server, we use the number of control packets that are expected to be received by the Error Recovery Server. Consequently, the resulting overhead equations are given by (72) and (73).

To find the expected probability of a failed repair packet delivery, we use the following equation, which is simplified by assuming equal packet loss rates for all the users:

$$e_r = \frac{\lambda^{(d)}}{N} \times \Bigg[\sum_{\substack{\forall m \\ N_m \leq 1}} N_m + \sum_{\substack{\forall m \\ N_m > 1}} N_m \times \Bigg(\widetilde{\omega}_S + \breve{\widetilde{\omega}}_S \times \Big[\lambda^{(d)} + \breve{\lambda}^{(d)} \times \zeta + \breve{\lambda}^{(d)} \times \breve{\zeta} \times \Big(\varrho + \breve{\varrho} \times \big[\lambda^{(u)} + \breve{\lambda}^{(u)} \times \lambda^{(d)} \big] \Big) \Big] \Bigg) \Bigg] \tag{74}$$

4.4 Performance Analysis

To find the performance of the weight assignment algorithm, we use a simulation model, in which the delay parameters are distributed randomly to a given set of users.

For that purpose, we use an approach that separates the end-to-end link between any two nodes in the network into two components that operates similarly for all the users in the network: link from the user to the access point and link between two access points. The delay component for the former link is referred to as the local-delay, whereas the delay component for the latter link is referred to as the network-delay. For our calculations, Error Recovery Server acts similar to an access point. We assume that the delay between an end-user and its access point is the same for all users and the delay between two access points is a function of the distance in-between. To find a valid set of delay parameters, we create an $A \times A$ sized network [9] and distribute the users uniformly within the given region, while placing the Error Recovery Server at the center. For a square shaped network, the average distance to the Error Recovery Server, $\overline{d}_{u,S}$, is given by:

$$\overline{d}_{u,S} = (A/6)\left[\sqrt{2} + \ln\left(1 + \sqrt{2}\right)\right] \tag{75}$$

Similarly, the average distance between two users, $\overline{d}_{u,u}$, is given by:

$$\overline{d}_{u,u} = (A/5)\left[(\sqrt{2} + 2)/5 + \ln(\sqrt{2} + 1) - 2 \times \ln(\sqrt{2} - 1)\right] \tag{76}$$

Next, we assign a specific value to the average forward-trip-time (FTT) between an end-user and the Error Recovery Server, which is referred to as δ, (*e.g.*, $\delta = 20ms$). We assume that, for the Error Recovery Server, ratio between local-delay, δ_l, and the average end-to-end delay, which is referred to as ϕ, is a parameter of choice. We can then find the value of δ_l using $\left(\phi \times \delta\right)$, the value of unit-distance network-delay metric, d_v, using $\left(\delta \times (1 - \phi)/\overline{d}_{u,S}\right)$, and the value of end-to-end delay $d_{i,j}$, using $\varpi_{i,j} \times \delta_l + d_v \times D_{i,j}$, where $D_{i,j}$ represents the distance between the ith and jth nodes, and $\varpi_{i,j}$ represents a constant that equals 1 when one of the nodes is the Error Recovery Server and 2 when both nodes are end-users.

[9] The size of the network does not have any impact in our calculations, since the only important measure is the relative distance.

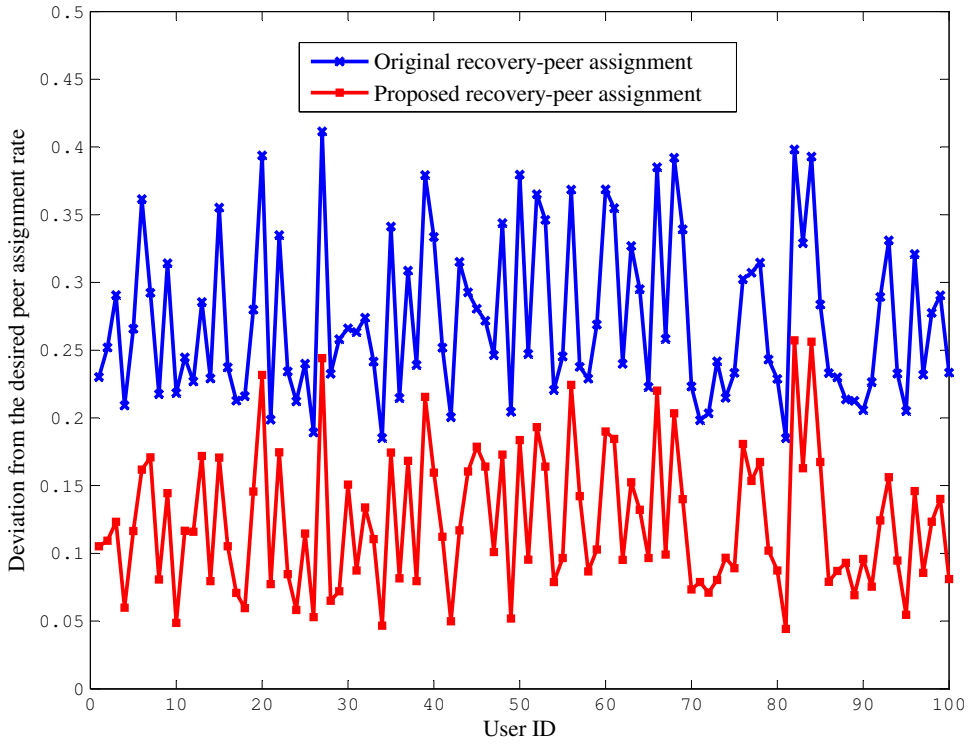


Figure 25: Deviation from the originally assigned rates, when $N = 100$.

In Figure 25, we show how much the proposed weight distribution algorithm and the original assignment policy deviate from the desired results. For the simulation, the parameters are selected as follows: $N = 100$, $\delta = 20ms$, $\phi = 0.5$, $T_0 = 100ms$, $|\pi| = 20$, $\Delta = 150ms$ (Δ represents the maximum latency), and the user weights are randomly selected from the interval $[1, 20]$. Note that, deviation from the desired rate represents the usage rate that cannot be utilized at the recovery-peers. As the deviation increases, the usage rate at the Error Recovery Server also increases. Using the proposed algorithm, we observe an initial improvement of $50\% - 60\%$. If we integrate the proposed weight reassignment process, which targets nearby users that are reachable during the last $20ms$ of a transmission block, into Algorithm 5, then the improvements increase to 85% (which suggests an average server utilization rate of $3 - 4\%$).

In Figure 26 we compare the expected error rate performance of the proposed approach with the server-based recovery approach at various values for the probability

of a recovery-peer leaving the multicast session, when the users observe the same packet loss rates (0.5×10^{-3}). We assume that the users are distributed to 200 multicast sessions [10]. We observe that the proposed peer-based recovery approach outperforms the server-based recovery approach because of the multi-step recovery strategy implemented to recover the lost packets, *i.e.*, Error Recovery Server transmits the repair packet if the recovery-peer fails to deliver the repair packet. We also observe that as the rate of leaving a multicast session increases, the probability of failing to deliver the repair packet also increases. That is because, the system at that point reverts to a server-based error recovery system.

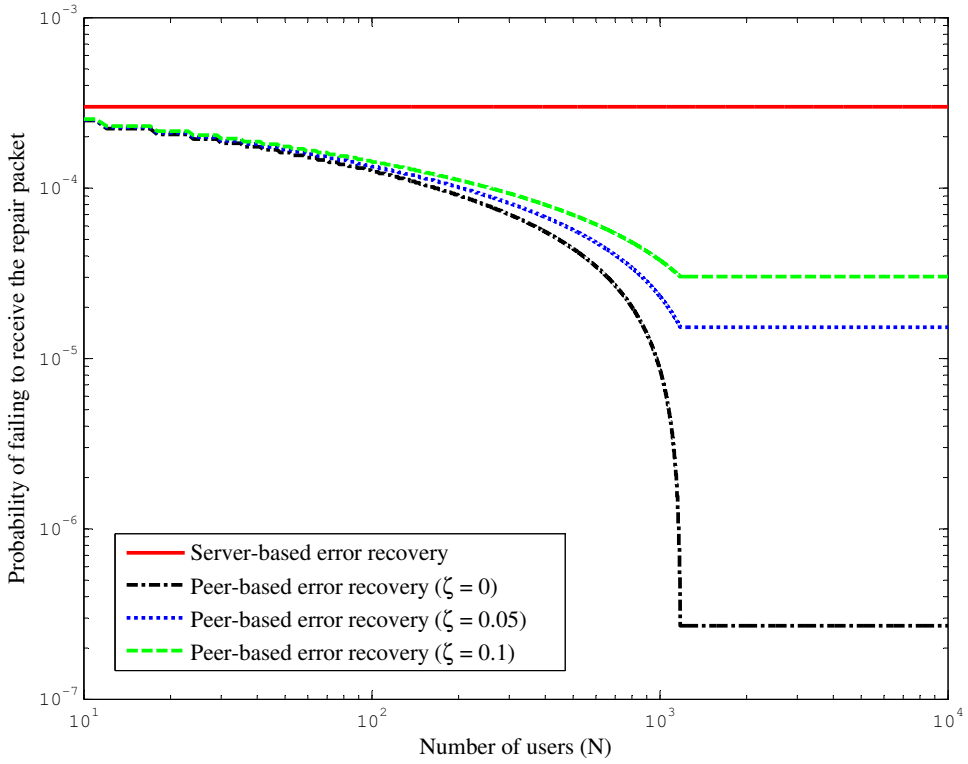


Figure 26: Probability of failing to deliver the repair packet, when ζ is varied.

In Figure 27, we present the theoretical results that show the overhead at the Error Recovery Server. The proposed approach achieves significant improvements in server

[10]Changing the number of multicast sessions does not affect the limiting values for the error recovery performance. It only affects the point the limiting values are reached, *i.e.*, the lower the number of sessions is, the lower the user count for the system is to reach the limiting values.

usage efficiency compared to a server-based error recovery approach. Specifically, under the same limitations for the transmission capacity, the proposed peer-assisted recovery approach can support 10 times more users when compared to a solely server-based recovery approach. We also observe that, as the number of users increases, the majority of the error recovery traffic occurs on the uplink channel, which still stays at a manageable level.

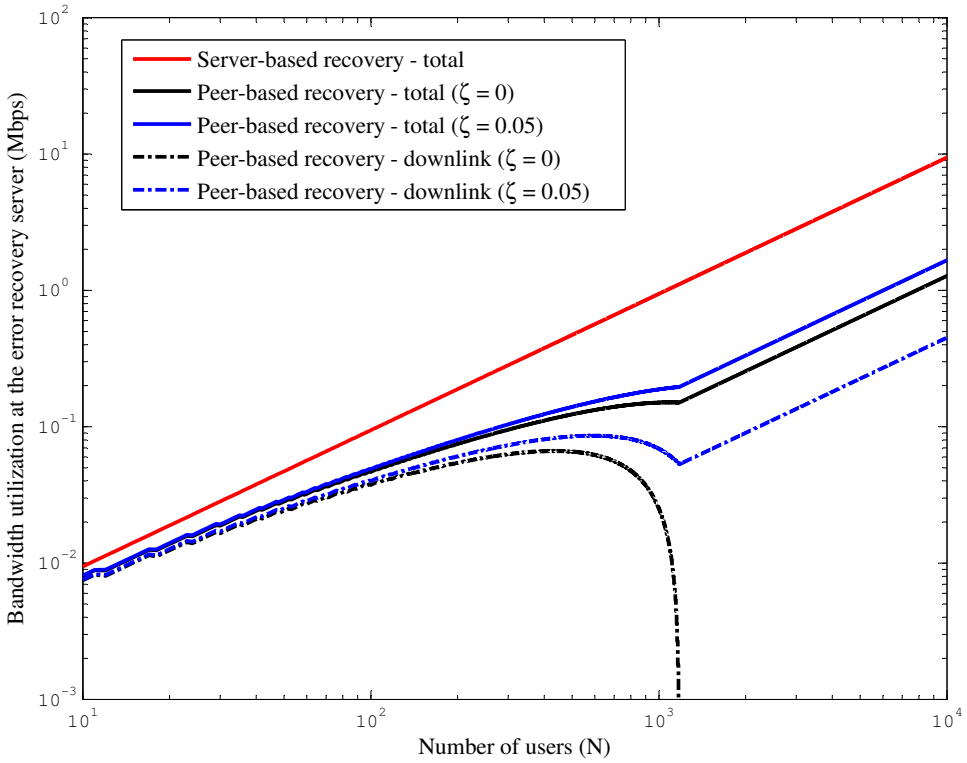


Figure 27: Overhead at the Error Recovery Server.

In Figure 28, we present the theoretical results that show the downlink and uplink error recovery overhead at the end-users when the probability of leaving a session is varied. As the number of users connected to the same Error Recovery Server increases, the downlink overhead stays almost the same, whereas the uplink overhead increases until it reaches a certain limit. The reason for observing only a slight increase in the downlink overhead is because the size of a control packet is much smaller than the size of a repair packet, hence the increased number of requests does not affect

the end result significantly. On the other hand, as the number of users increases, the expected number of users with no recovery peer decreases, causing the increase observed in the uplink overhead. In Figure 28, the results on the left represent the overhead for mostly server-based recovery, whereas the results on the right represent the overhead for mostly peer-based recovery. Also note that, changing the value of ζ does not have a significant impact of the final results.

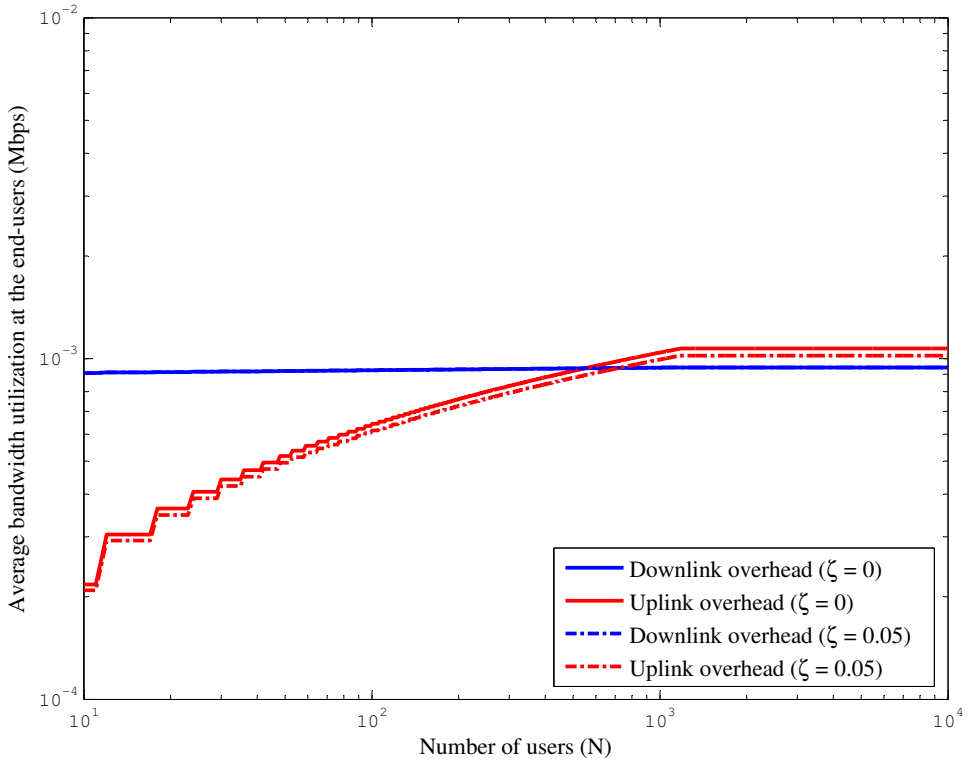


Figure 28: Error recovery overhead at the end-users.

In Figure 29, we show the level of change observed in the recovery latency measure as the number of users connected to a multicast session is increased and the value of ϕ is varied. Compared to the $40ms$ average round-trip-time required to receive the repair packet from the Error Recovery Server, additional delay incurred during the peer-based recovery phase is not significant. We also need to note the tradeoff between latency and scalability. Unlike a pure peer-based architecture, the proposed

framework supports lower latency threshold values. If the latency measure is decreased, then nearby users and the Error Recovery Server are utilized more often. As a result, we observe an increase in the usage rate for the Error Recovery Server. However, with a large enough transmission block size, we can limit the usage rate of the Error Recovery Server by limiting its use to packets that are expected to be delivered near the decoding deadline.

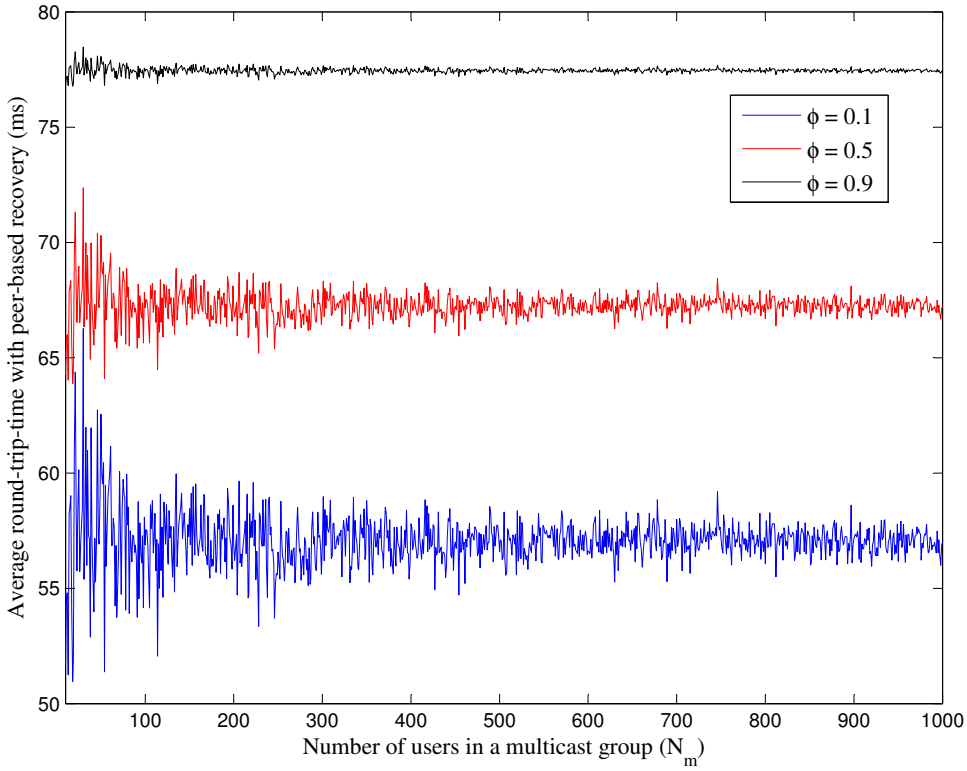


Figure 29: Impact of peer-based recovery on round-trip-times.

4.5 Conclusions

In this chapter, we presented a server-assisted peer-based error recovery framework for the IPTV networks. By introducing a reliable message exchange procedure, we have limited the use of Error Recovery Server by fairly distributing the recovery load to end-users. Our analysis showed the significant advantages of utilizing a peer-based framework, as it significantly improved the scalability and error recovery performance

of IPTV networks while limiting the recovery overhead and latency.

Algorithm 5 Weight initialization algorithm at user ν.

$i = 1$;
$W_{\nu,0} = \sum_{\forall u \in U_{\nu,1}} \omega_{\nu,u}$;
$T_{\nu,0} = \sum_{\forall i \leq K} T_{\nu,i}$;
$\omega^*_{\nu,u} = \omega_{\nu,u}/W_{\nu,0}$ and $\breve{\omega}_{\nu,u} = 0$, $\forall u \in U_{\nu,1}$;
while $i < K$ **do**
  $\widehat{U}_{\nu,i} = U_{\nu,i} \setminus U_{\nu,i+1}$;
  $\widehat{W}_{\nu,i} = \sum_{\forall u \in \widehat{U}_{\nu,i}} \omega_{\nu,u}$;
  $\widetilde{W}_{\nu,i} = \sum_{\forall u \in \widehat{U}_{\nu,i}} (\omega^*_{\nu,u} - \breve{\omega}_{\nu,u})$;
  $W_{\nu,i+1} = \sum_{\forall u \in U_{\nu,i+1}} \omega_{\nu,u}$;
  $r_{\nu,i} = T_{\nu,i}/T_{\nu,0}$;
  $\alpha_{\nu,i} = \widetilde{W}_{\nu,i}/r_{\nu,i}$;
  if $\alpha_{\nu,i} < 1$ **then**
    find $\omega^{(i)}_{\nu,u} = \alpha_{\nu,i}\omega_{\nu,u}/\widehat{W}_{\nu,i}$, $\forall u \in \widehat{U}_{\nu,i}$;
    $\widehat{\alpha}_{\nu,i} = 1 - \alpha_{\nu,i}$;
    find $\delta\omega^{(i)}_{\nu,u} = \widehat{\alpha}_{\nu,i}\omega_{\nu,u}/W_{\nu,i+1}$, $\forall u \in U_{\nu,i+1}$;
    if $\delta\omega^{(i)}_{\nu,u} r_{\nu,i} + \breve{\omega}_{\nu,u} \leq \omega^*_{\nu,u}$ **then**
      $\omega^{(i)}_{\nu,u} = \delta\omega^{(i)}_{\nu,u}$;
    else
      $\omega^{(i)}_{\nu,u} = (\omega^*_{\nu,u} - \breve{\omega}_{\nu,u})/r_{\nu,i}$;
    end if
    update $\breve{\omega}_{\nu,u} = \breve{\omega}_{\nu,u} + \omega^{(i)}_{\nu,u} r_{\nu,i}$, $\forall u \in U_{\nu,i+1}$;
  else
    find $\omega^{(i)}_{\nu,u} = \omega_{\nu,u}/\widehat{W}_{\nu,i}$, $\forall u \in \widehat{U}_{\nu,i}$;
    set $\omega^{(i)}_{\nu,u} = 0$, $\forall u \in U_{\nu,i+1}$;
  end if
end while
$\widehat{W}_{\nu,K} = \sum_{\forall u \in U_{\nu,K}} \omega_{\nu,u}$;
$\widetilde{W}_{\nu,K} = \sum_{\forall u \in \widehat{U}_{\nu,K}} (\omega^*_{\nu,u} - \breve{\omega}_{\nu,u})$;
$\alpha_{\nu,K} = \widetilde{W}_{\nu,K}/(T_{\nu,K}/T_{\nu,0})$;
find $\omega^{(K)}_{\nu,u} = \min(\alpha_{\nu,K}, 1)\omega_{\nu,u}/\widehat{W}_{\nu,K}$, $\forall u \in U_{\nu,K}$;
set $\omega^{(K)}_{\nu,S} = 1 - \min(\alpha_{\nu,K}, 1)$;

CHAPTER V

COOPERATIVE DIVERSITY-DRIVEN ERROR RECOVERY FRAMEWORK FOR WIRELESS IPTV NETWORKS

5.1 Introduction

In the previous chapters, we focused on the design of error recovery techniques to support reliable delivery of IPTV content over wireline access networks (with specific emphasis on the DSL-based access networks). Similar to DSL- and cable-based high-capacity last-mile access technologies, next-generation wireless broadband metropolitan area networks also present end-users with an opportunity to access high-speed Internet over the broadband wireless channel (*e.g.*, WiMAX) [91, 83]. WiMAX, for instance, appear as a promising technology to deliver IPTV content to end-users, because of its features, such as the high servicing capacity which makes it possible for a single WiMAX cell to carry many users receiving the same IPTV content creating a significant potential for the distribution of the IPTV content over WiMAX networks.

To meet the service quality requirements of users requesting IPTV content over the WiMAX networks, we need robust and resource efficient content delivery techniques [28, 26]. However, because of the characteristics of the wireless channel, *e.g.*, high packet loss rates and high delays, the typical approaches proposed to support reliable IPTV communications, which utilize a server-based recovery framework, may not be suitable for the current scenario. If we focus on the strengths of the WiMAX technology, we can see that it would be possible to utilize a peer-based recovery approach that can effectively address the inadequacies of employing server-based recovery techniques. Peer-based recovery allows the users to recover from their losses by

communicating with other users within their network. Peer-based recovery not only reduces the recovery response time, but also reduces the overhead of going through the base station to retrieve the recovery data.

To achieve peer-based recovery within a wireless framework, we use cooperative diversity, which refers to the use of virtual multiple-input-multiple-output (MIMO) arrays, where the users represent the elements of an array [78, 56, 89]. In a cooperative diversity-driven error recovery framework, reliable users act as a transmit array for unreliable users. With cooperative diversity, we can significantly improve the resource utilization efficiency in the network [87, 24, 27, 14, 15, 22, 65, 42, 66, 57, 71]. In [44], the authors propose a cooperative diversity-driven peer-based error recovery approach, which uses a two-phase transmission technique to improve the reliability of IPTV content delivery. The first phase is reserved for the base station's multicast. The second phase is reserved for cooperative communications, during which the users that have successfully received the content from the base station cooperatively transmit to also make the other users reliable. Cooperation allows the unreliable users to recover from their losses without the need to go through the base station once again. Consequently, temporary impairments introduced by the wireless channel can be overcome by using multiple transmission sources that utilize diverse transmission channels.

However, considering the practical difficulties involved in supporting efficient power control for a pure virtual MIMO system, it may be desirable to reduce the virtual MIMO system to a virtual multiple-input-single-output (MISO) system, where the cooperative transmitters target a single user instead of a group of users. By implementing a MISO-based error recovery system, power-control techniques can be utilized more efficiently to further limit the burden on the transmitting peers from an energy usage point of view.

In this chapter, our objective is to investigate the impact of using a virtual MISO

system by proposing a resource-efficient content delivery framework. Within a two-phase cooperative communication framework, we propose a rate selection algorithm that is used to find resource-optimal cooperation sets and cooperation parameters. The cooperation sets are formed by distributing the successful receivers from the first phase into multiple groups, each of which targets and transmits to a distinct user during the cooperation phase. In doing so, each group can simultaneously deliver the desired content to the users that failed to receive the content during the first phase. The proposed approach is then compared to cooperative MIMO and direct transmission techniques to measure its performance, in terms of throughput, delay, overhead, and energy usage, with respect to these approaches.

The rest of the chapter is organized as follows. In Section 5.2, we present the network model. Section 5.3 presents the channel model. We present the proposed resource allocation algorithm in Section 5.4. We evaluate the performance of the proposed content delivery framework in Section 5.5. Section 5.6 concludes the chapter.

5.2 Network Model

We consider a WiMAX network that consists of a base station, which is referred to as S, and N users. We focus on the performance of a single multicast group within this network, which we refer to as M, *i.e.*, and we assume $|M| = N$ [1]. At the media access control (MAC) layer, the transmissions are grouped into superframes, each of which consists of multiple frames. An IPTV multicast session is assigned one or more frames within a superframe. To allocate resources over the downlink and uplink channels, time division duplexing (TDD) is used. At the physical layer, orthogonal frequency division multiplexing (OFDM) is used. Access to the transmission medium is granted using time division multiplexing (TDM) at the base station. Base station

[1] $|X|$ represents the size of the set X.

broadcasts the framing structure to the subscriber nodes using downlink map (DL-MAP) messages. To inform the base station regarding the perceived transmission quality in-between, subscriber stations utilize the Channel Quality Indicator channel for their feedbacks.

To reliably deliver the IPTV content to end-users, based on the principles of co-operative communications, we use a two phase transmission strategy. The first phase is reserved for the base station's multicast to the subscriber nodes that belong to the given session. However, because of the varying wireless channel characteristics, only a subset of the targeted users can successfully receive the data transmitted during this multicast. To help the users that have failed to receive the initial multicast recover from their losses, a second transmission phase is utilized. During the second transmission phase, which we refer to as the *cooperation phase*, successful receivers from the *multicast phase* cooperatively transmit to the failed users (from the multicast phase) to also make them reliable. Also note that, here, the formation of the transmitter sets is based on the decode and forward approach. Hence, a user can become a cooperative transmitter, only if it has successfully received the IPTV multicast during the first phase. For the given multicast group, M, we use M_R to represent the cooperative transmitter set and M_{NR} to represent the cooperative receiver set [2].

To determine the cooperation sets and assign the cooperative transmitters to each of the receivers, we utilize a rate-adaptive transmission strategy. Specifically, base station uses the channel quality information to select the transmission rate that will be used during the initial multicast. Note that, transmission rate selected for the initial multicast determines the set of users that can or cannot successfully receive the initial IPTV multicast. The sets of reliable and unreliable users are then used to optimize the performance during the *cooperation phase*. Therefore, to achieve an overall acceptable performance, initially selected transmission rate needs to ensure

[2]Note that the subscript R represents reliable, and the subscript NR represents non-reliable.

that, on average, a certain number of users can become reliable by the end of the first phase.

As the first transmission phase ends, base station requests feedback from the users associated with the given multicast session. The feedback received from the multicast group is then used by the base station to determine the cooperation parameters, *i.e.*, cooperation groups and transmission rates, that will be used by the users within each of the selected cooperation groups. Before making the final decision, we assume that the base station utilizes the statistical data on the average signal-to-noise ratio observed by each possible transmitter-receiver pair within the given multicast set. Specifically, we assume the base station to approximate the average SNR values using the location information for each of these users. Furthermore, because of utilizing a MISO-based cooperation strategy, we assume the base station to select the cooperation parameters based on the fact that a cooperative transmitter can only target a single receiver. After the cooperation parameters are determined by the base station, it multicasts this information to the targeted users, which also signals the beginning of the second transmission phase.

5.3 Channel Model

We assume the wireless channel fading to follow the Rayleigh distribution, *i.e.*, received signal-to-noise ratio (SNR) is assumed to be exponentially distributed. To determine the average SNR, we use the information on the transmit power and the distance between the transmitter and the receiver stations.

5.3.1 Success Probability During the Multicast Phase

Since the received SNR is assumed to be exponentially distributed, probability of successfully delivering the multicast data during the first transmission phase is determined by using the following equation [12]:

$$P_{s,1}(i) = e^{-(2^{R^{(1)}}-1)/\bar{\xi}_i} \tag{77}$$

where $P_{s,1}(i)$ represents the probability of a receiver i successfully receiving the multicast data, $R^{(1)}$ represents the multicast transmission rate used by the base station, and $\bar{\xi}_i$ represents the average SNR for the channel between the base station and i. Note that (77) is derived from Shannon's theorem (*i.e.*, for unit bandwidth, to support transmission rate R, received signal-to-noise ratio needs to satisfy the following inequality $\xi \geq 2^R - 1$).

5.3.2 Success Probability During the Cooperation Phase

To find the probability of a successful recovery during the cooperation phase, we start by assuming independent fading distributions for each of the channels that is used to transmit data (*i.e.*, channels between each cooperative transmitter and its targeted receiver). In doing so, we can approximate the distribution that corresponds to the received SNR during the cooperation phase using the following equation, which represents the distribution corresponding to the sum of independent exponential variables [3]:

$$f_{S_X^{(l)}}(x) = \left[\prod_{\forall i \in C(l)} \lambda_{i,l}\right] \sum_{\forall k \in C(l)} \frac{e^{-\lambda_{k,l}x}}{\prod\limits_{\substack{\forall j \in C(l) \\ j \neq k}} (\lambda_{j,l} - \lambda_{k,l})} \tag{78}$$

where $\lambda_{i,l}$ represents the inverse of the average SNR for the channel between the cooperative transmitter i and its targeted receiver l, $C(l)$ represents the set of cooperative transmitters targeting l, and $S_X^{(l)}$ represents the total received SNR at l (*i.e.*, $S_X^{(l)} = \sum_{\forall i \in C(l)} 1/\lambda_{i,l}$).

Let us assume that the parameter $\bar{\xi}_{ij}$ represents the average signal-to-noise ratio for the channel between receiver i and its transmitter j. Therefore, if we insert the

[3] For the given distribution to be valid, pairwise SNR values, which correspond to each possible combination for the cooperation process, need to be different, which typically suggest the user distances to have distinct values.

related SNR parameters into (78), we obtain the following equation:

$$f(\xi_i) = \sum_{k \in C(i)} \left[\frac{e^{-\xi_i/\bar{\xi}_{ik}}}{\bar{\xi}_{ik}} \times \prod_{\substack{\forall l \in C(i) \\ l \neq k}} \frac{\bar{\xi}_{ik}}{\bar{\xi}_{ik} - \bar{\xi}_{il}} \right] \tag{79}$$

If the transmission rate equals R, then to estimate the probability of a successful transmission during the second phase, we can use the following equation:

$$p_s^{(i)}(R) = 1 - \sum_{k \in C(i)} \left[(1 - e^{-(2^R - 1)/\bar{\xi}_{ik}}) \prod_{\substack{\forall l \in C(i) \\ l \neq k}} \frac{1}{1 - \bar{\xi}_{il}/\bar{\xi}_{ik}} \right] \tag{80}$$

If we assume equal gain for all the transmit-receive pairs, we can further approximate the average SNR, $\bar{\xi}_{ij}$, using

$$\bar{\xi}_{ij} = \frac{P_{t,j} \times G}{N_0 \times d_{ij}^{\kappa}} \tag{81}$$

where $P_{t,j}$ represents the transmit power used by the transmitter j, κ represents the path loss coefficient for the propagation medium, and N_0 represents the noise power.

Consequently, we can state the equation that represents the probability of success during the cooperative transmission phase as follows:

$$p_s^{(i)}(R) = 1 - \sum_{k \in C(i)} (1 - e^{-C_R d_{ik}^{\kappa}/P_{t,k}}) \prod_{\substack{\forall l \in C(i) \\ l \neq k}} \left[1 - \frac{P_{t,l}/d_{il}^{\kappa}}{P_{t,k}/d_{ik}^{\kappa}} \right]^{-1} \tag{82}$$

where C_R equals $\big((2^R - 1)N_0/G\big)$. Note that, if no transmit power control technique is employed, then the cooperative transmitters are all assumed to use the same transmit power (*i.e.*, $P_{t,j} = P_t^{(i)}$, $\forall j \in C(i)$), which can further simplify (82) to become as follows:

$$p_s^{(i)}(R) = 1 - \sum_{k \in C(i)} \left[\left(1 - e^{-\widetilde{C}_R d_{ik}^{\kappa}} \right) \prod_{\substack{\forall l \in C(i) \\ l \neq k}} \frac{d_{il}^{\kappa}}{d_{il}^{\kappa} - d_{ik}^{\kappa}} \right] \tag{83}$$

where $\widetilde{C}_R$ equals to C_R/P_t.

5.4 Proposed Rate Adaptation Framework

To determine the resource-optimal multicast and cooperative transmission rates, we need to utilize a dynamic rate selection algorithm that takes into account the perceived

channel characteristics in the network. To find these rates, we start by focusing on the operation during the cooperation phase. Assume that the set of users that became reliable during the initial multicast is given, which is defined as M_R in Section 5.2. First, we divide the set of cooperative users within M_R into m subgroups, where m equals the number of receivers for the second phase, *i.e.*, $m = |M_{NR}| = |M| - |M_R|$. Note that because of using a single multicast session in our calculations, we have $|M| = N$. Using the basic characteristics of cooperative MISO, *i.e.*, each transmitter within M_R is assigned to a single receiver within M_{NR}, we have the following, $\sum_{\forall i} |M_{C,i}| = |M_R|$. We can represent the formation of these groups with a single parameter, which is defined as coverage ratio ϱ (see [44]) and represents the average success (or reliability) rate during the first phase:

$$\varrho = |M_R|/N \tag{84}$$

The value of coverage ratio varies based on the transmission rate used for the initial multicast. For instance, if the base station uses a lower transmission rate, then we can increase the average number of users that can successfully decode the multicast data, thereby increasing the coverage ratio, and vice versa.

After the coverage ratio is selected, we can find the transmitters for the cooperation phase, M_R, which is then used to approximate the average probability of success during the cooperation phase using the following equation:

$$P_S^{(2)} = \sum_{<M_R>} P_{M_R} \prod_{\forall i \in M_{NR}} p_s^{(i)}(C(i), R_{C(i)}^{(2)}) \tag{85}$$

where $< M_R >$ represents the set of possible cooperative group formations, P_{M_R} represents the probability of observing a given cooperative grouping, and $p_s^{(i)}(C(i), R_{NR}^{(2)})$ represents the probability of successfully receiving the cooperatively transmitted data when the cooperative transmission rate used by i's cooperative transmitters within $C(i)$ equals $R_{C(i)}^{(2)}$.

The cooperative transmission rate that satisfies the constraints on the successful delivery rate while also achieving the maximum session throughput during the second phase, $R_{NR}^{(2)}$, is found as follows [4]:

$$R_{NR}^{(2)} = \min_{i \in M_{NR}} R_{opt,i} \tag{86}$$

where $R_{opt,i}$ represents the maximum cooperative transmission rate that can be used by $C(i)$.

We use the probability of success measure, which is referred to as δ_S, as the optimality criteria during the rate selection process. Specifically, an optimal rate selection process needs to ensure that the average probability of success during the cooperation phase is kept at or above δ_S. The value of the success threshold determines the extra overhead required to satisfy the minimum service quality requirements associated with the given IPTV multicast session. If a lower value is selected, we would observe an increase in the number of users that would request retransmission from the base station. Considering the delay constraints imposed by the IPTV service, we cannot allow too many retransmission attempts through the base station. Each user can reserve additional bandwidth for the delivery of proactive recovery packets, however, considering the wireless channel characteristics, the associated cost may be significantly higher than the desired overhead. For these reasons, we need to assign a sufficiently high value to δ_S without significantly increasing the expected duration for the cooperation phase, *i.e.*, in our calculations, we assume that $\delta_S = 0.99$.

After selecting the success threshold metric, we can find the maximum allowed transmission rate for each user within M_{NR} as follows [5]:

$$R_{opt,i} = \max_{R} \left(p_s^{(i)}(R) \geq \delta_S \right) \tag{87}$$

[4]Note that the selected cooperative transmission rate is expected to be shared by all the cooperative transmitters to maximize the energy efficiency during the cooperation phase.

[5]Note that, any transmission rate that is lower than $R_{opt,i}$ is a feasible solution.

Note that, since the optimal transmission rates are determined by the set of cooperative transmitters, the process to assign cooperative transmitters to each receiver, before the second phase starts, carries a critical importance to increase the efficiency of the cooperation phase. For that purpose, we propose an efficient rate assignment technique which is initiated after the base station finishes collecting information on the targeted users' delivery status at the end of the initial multicast. The operation of the proposed rate selection algorithm is explained as follows.

5.4.1 Proposed Algorithm to Achieve Dynamic Rate Selection

We initialize the cooperation sets by assigning each transmitter (within M_R) to the receiver (within M_{NR}) with the highest average channel quality in-between. In practice, this suggests that users within M_R join the cooperation set of the closest receiver. To maximize the cooperative transmission rate, we need to increase the rate that is observed by the users with the lowest cooperation gain. For that purpose, we use the following observation: the highest cooperation gain is achieved when the receivers are assigned transmitters within short distance, which suggests that far-distance transmitters can be removed from the cooperation sets with no significant effect on the initially selected cooperative transmission rate. Therefore, to improve the optimal rate, we reassign the long distance transmitters that initially target receivers with high transmission rates to receivers that observe low transmission rates. Algorithm 6 presents the proposed suboptimal recursive search procedure that is used to reassign the transmitters.

The reassignment process starts with the selection of the receiver that is the target of the lowest cooperative transmission rate. We refer to this receiver using $n_{\min(S)}$. Next, we search for the close distance transmitters that are not initially assigned to the cooperation set of $n_{\min(S)}$. We analyze the impact of reassigning these transmitters, on both $n_{\min(S)}$ and the originally assigned receiver ν. If the optimal rate for ν does

Algorithm 6 Cooperation set formation.

find $n_{\min(S)} = \underset{\forall n_i \in M_{NR}}{\operatorname{argmin}} R_{opt}(n_i)$;
$R_{min} = R_{opt}(n_{\min(S)})$;
$\hat{n} = n_{\min(S)}$;
update-status = **true**;
$n_{cur} = \hat{n}$
while *update-status* = **true do**
  chk-loc = 0; //Position in proximity-ordered reliable neighbor set
  while $n_{cur} = \hat{n}$ **do**
    chk-loc = *chk-loc* +1;
    $ctx_{cur} = N_{ord}(n_{cur}, chk\text{-}loc)$; //ID of the reliable neighbor at the given position
    $rxn = Rx(ctx_{cur})$; // ID of ctx_{cur}'s initial target
    $C(rxn, ctx_{cur}) = 0$; // Reset cooperation status between rxn and ctx_{cur}
    update $R_{opt}(rxn)$;
    if $R_{opt}(rxn) > R_{min}$ **then**
      $Rx(ctx_{cur}) = n_{cur}$;
      $C(n_{cur}, ctx_{cur}) = 1$;
      update $R_{opt}(n_{cur})$
      $\hat{n} = \underset{\forall n_i \in M_{NR}}{\operatorname{argmin}} R_{opt}(n_i)$;
    else
      $C(rxn, ctx_{cur}) = 1$;
    end if
  end while
  if $\hat{n} = n_{cur}$ **then**
    update-status = **false**; // No update for cooperation set is possible
  end if
end while
$R^{(2)}_{NR} = \min_{\forall n_i \in M_{NR}} R_{opt}(n_i)$;

not become lower than the optimal rate selected for $n_{\min(S)}$ before the reassignment, we accept the selected reassignment. We repeat this search for $n_{\min(S)}$ as long as it remains as the receiver that experiences the lowest transmission rate. If at some point during the execution of the algorithm another receiver $n'_{\min(S)}$ replaces $n_{\min(S)}$ as the receiver with the lowest transmission rate, then we initiate a new recursive search process, this time for $n'_{\min(S)}$.

We continue to execute the above reassignment process, as long as the process continues to improve the globally optimal rate. We stop the search process when

no further changes can be made, *i.e.*, the receiver that experiences the minimum transmission rate cannot improve its rate any further and the receiver with the minimum optimal rate cannot be changed. The last selected optimal rate becomes the cooperative transmission rate that will be used during the second phase. Note that the cooperative transmitter set that is used for the recursive search process depends on the transmission rate that is used during the initial multicast, *i.e.*, varying the coverage ratio also varies the effectiveness of the proposed recursive search process.

5.4.2 Finding the Expected Values for Effective Throughput and Recovery Overhead

Consequently, using the multicast and cooperative transmission rates, we can determine the session throughput that is experienced by a receiver i as follows:

$$\mathsf{T}_i = \frac{R_R^{(1)}}{T_f/T_R^{(1)}} \times \left(P_{S,i}^{(1)}(R_R) + \left[1 - P_{S,i}^{(1)}(R_R)\right] \times P_{S,i}^{(2)}(R_{NR}) \right) \tag{88}$$

where T_R represents the duration of the initial multicast phase and T_f represents the duration of a superframe. If multiple frames are assigned to the same multicast session within a superframe, then we sum up the throughput achieved during each transmission frame.

To find the error recovery overhead associated with the cooperative recovery process, we use the following equation:

$$\mathsf{O} = \sum_{\forall 1 \in M} \frac{R_R^{(1)}}{T_f/T_R^{(1)}} \times \left[1 - P_{S,i}^{(2)}(R_{NR})\right] \times \left[1 - P_{S,i}^{(1)}(R_R)\right] \tag{89}$$

Here, error recovery overhead represents the additional transmissions required to compensate for the losses that cannot be recovered during the cooperation phase. Retransmission process utilizes a unicast-based approach. To be specific, repair packets are transmitted directly from the source or the designated Error Recovery Server to the end-user. Users can further minimize the latency associated with the error recovery process by using proactive recovery.

5.5 Performance Analysis

In this section, using simulations, we evaluate the performance of the proposed error recovery system by investigating the impact of initial rate selection on session throughput, recovery overhead, delivery latency, and energy utilization. The proposed framework is compared to a cooperative MIMO system, where we have a single cooperative transmitter set simultaneously targeting all the unreliable users, and a non-cooperative direct transmission system, where we use unicast-based error recovery (*i.e.*, retransmissions going through the base station). Note that for the direct transmission system, we limit the number of failures during the initial multicast by keeping the average success ratio above a specific threshold, which is achieved by selecting the multicast transmission rate accordingly.

We carried out our simulations using a circular shaped network with a radius of $8000m$. The subscriber nodes are distributed randomly within the network region with the base station being placed at the center. The simulations present the results for a single multicast session. In doing so, we can observe the average IPTV multicast performance, when the multicast session is selected randomly by the base station. We assume that the number of users connected to the given multicast session equals 50 [6]. We assume a transmit power to noise power ratio of 171dB for the channel between base station and subscriber nodes, and a transmit power to noise power ratio of 162.8dB for the channel between any two subscriber nodes. The path loss coefficient is given as 4.375.

For the simulations, we used 100 different topologies, and simulated each topology 10 times using different random seeds. To vary the transmission rates, we used the coverage ratio parameter (*i.e.*, average ratio of users that are made reliable at the

[6]Note that, since channel popularity for the IPTV service is modeled using Zipf distribution, if the total number of IPTV subscribers for a given WiMAX network is very close to the capacity of the network, then it is possible to have that many or more subscribers connected to the same session.

end of the first transmission phase). We varied the expected coverage ratio from 0.55 to 0.95 [7].

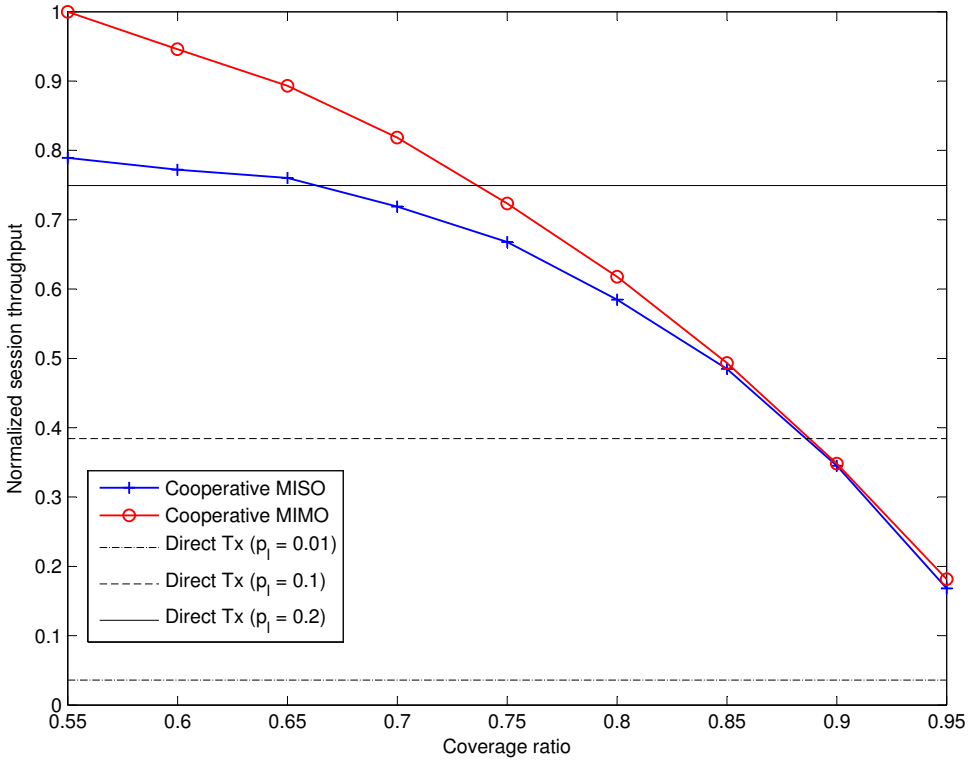


Figure 30: Normalized throughput of a single multicast session.

In Figure 30, we show the throughput results that do not take into account the unicast-based recovery overhead. The throughput results are normalized using the best results we obtained for the cooperative MIMO system. For the direct transmission scenario, we focused on a single-phase transmission, and determined the minimum required rate to deliver at the desired success rates of 0.99, 0.9, and 0.8 [8]. For the cooperative transmission scenarios, we use a success ratio of 0.99. From the results, we observe that the direct transmission scenario requires an additional overhead of at least 20% per user to perform as good as a cooperative MISO system. The best

[7]Note that coverage ratio values that are less than 0.50 usually lead to users that cannot be recovered during the cooperation phase, since the number of potential transmitters would be less than the number of potential receivers.

[8]For the direct transmission scenario, success ratio determines the average overhead required to maintain the desired packet loss characteristics of a typical IPTV system.

results for the proposed system is achieved when the coverage ratio is close to 0.55, which coincides with the results of the considered cooperative MIMO system.

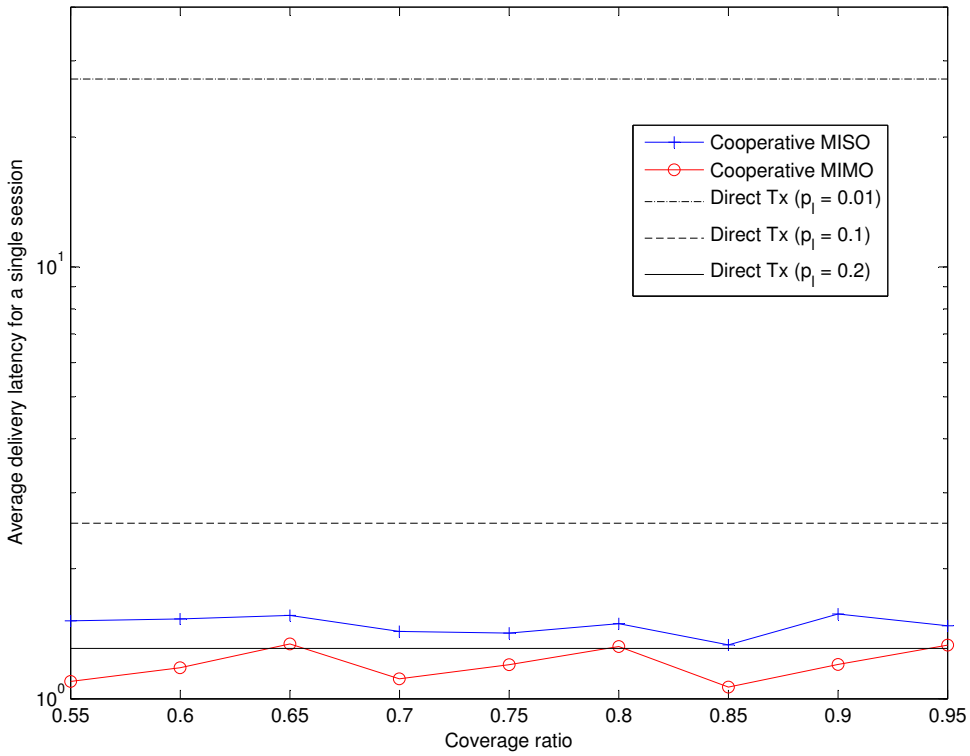


Figure 31: Normalized delivery latency to finalize the IPTV multicast.

We present the latency results in Figure 31. The normalized results are obtained by assuming the transmission of a unit-size data content. Here, for the proposed cooperative multicast framework, delivery latency equals the summation of the duration of multicast and cooperation phases, whereas for the direct transmission case delivery latency equals the duration of the multicast phase. [9] We observe that the aggregate value for the latency does not vary significantly as the coverage ratio changes, with cooperative MIMO performing better than cooperative MISO because of using higher transmission rates during the cooperation phase. Direct transmission scenario can only perform as good as the cooperative multicast approach, if the success threshold is kept at 80% or lower during the multicast. However, because of

[9]Note that, additional latency to deliver unicast-based retransmission data is not included.

the additionally required retransmissions, unless a user allocates a significant portion of its bandwidth for the transmission of proactive repair packets, actual latency for some users would be much higher than the value shown in the figure, which suggests a significant advantage for cooperative multicast scenarios.

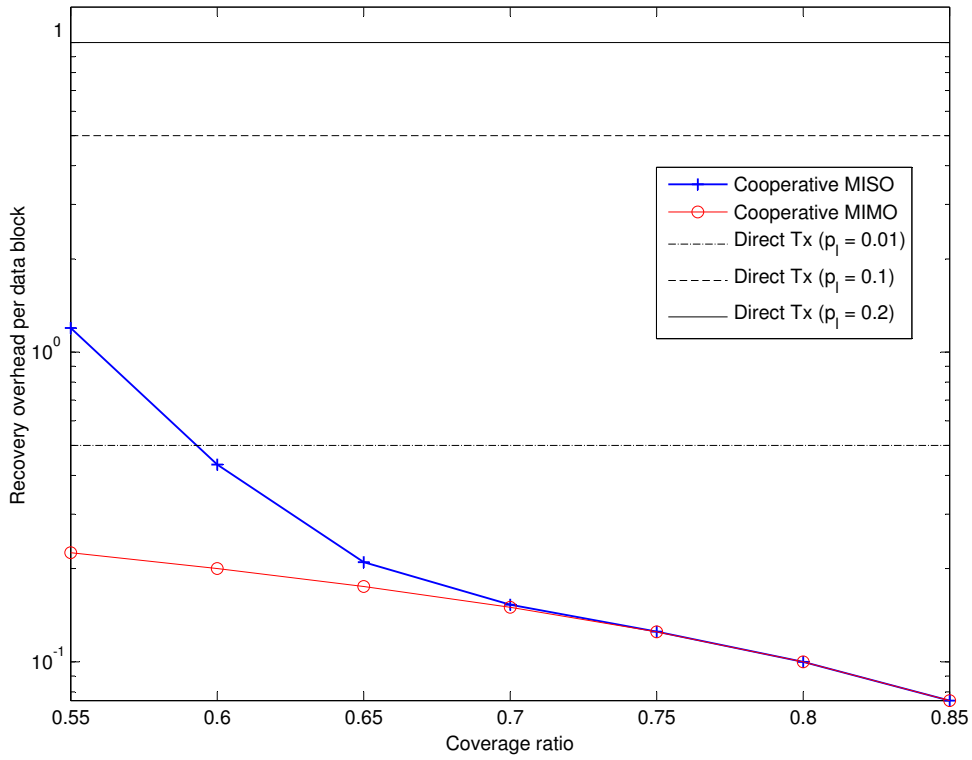


Figure 32: Error recovery overhead comparison between different multicast strategies.

We illustrate the overhead performance in Figure 32, which shows the number of additional retransmissions required for each multicast. The results suggest that with cooperative MISO, at the 0.55 coverage ratio, for each transmission phase, we require one more transmission from the base station to recover from all the losses. For cooperative MIMO, we need one more transmission from the base station for every 4 transmission phases. Direct transmission, on the other hand, can present similar advantages only if a very high value is selected for the success threshold, *i.e.*, 0.99, which significantly lowers the session throughput and increases the delivery latency.

In Figure 33, we compare the energy utilization performance of the proposed MISO

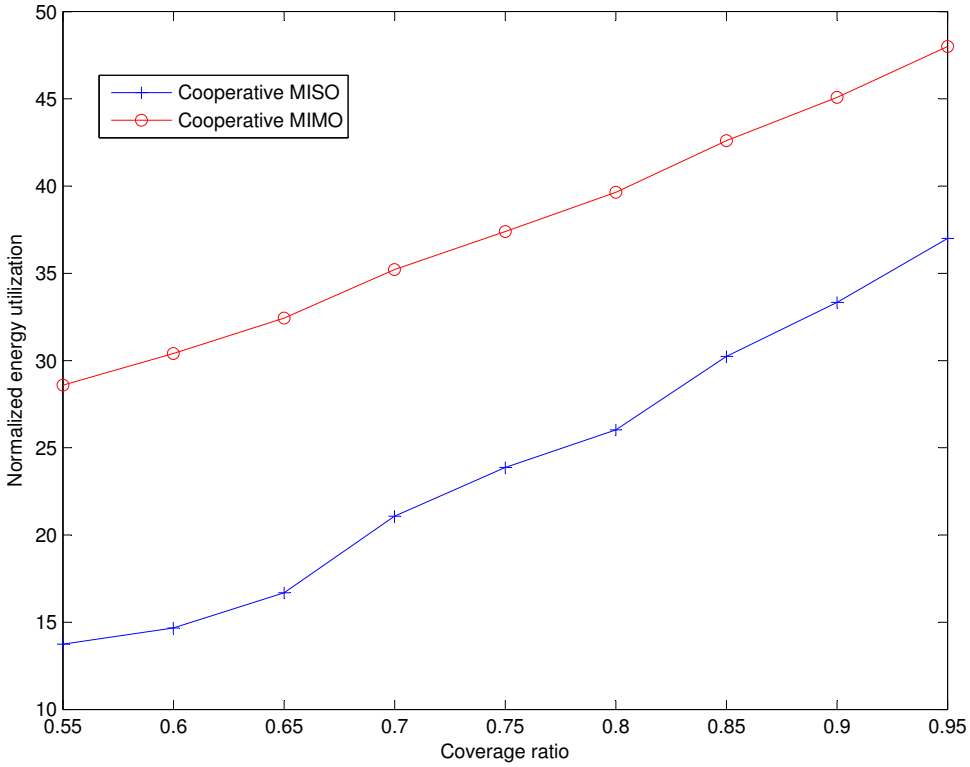


Figure 33: Normalized energy usage comparison between different cooperation strategies.

approach to that of the considered cooperative MIMO approach. The results show the energy utilization during the cooperation phase, and they are normalized with respect to the transmission of unit size data, which takes into account the decrease in the minimum required transmit power to ensure reliable delivery. When the coverage ratio is close to 0.55, which achieves the highest throughput efficiency, we observe that the proposed approach can reduce the energy utilization by $\approx 50\%$. The improvement in energy utilization gradually reduces to $\approx 30\%$ as the coverage ratio increase to 0.95.

5.6 Conclusions

In this chapter, we proposed a resource efficient adaptive rate selection strategy for a two-phase cooperative MISO-based error recovery framework. We presented the analytical framework, which allows us to determine a pseudo-optimal rate selection strategy. We showed that in a two-phase cooperative recovery framework, depending

on the initial size of the cooperation set, selection of the pseudo-optimal transmission rates allowed the proposed cooperative MISO framework to perform very close to a cooperative MIMO framework. Furthermore, we observed that the proposed system presents a good balance between cooperative MIMO and non-cooperative transmission systems with further gains to be achieved with the use of content-adaptive cooperative error control techniques.

CHAPTER VI

ANALYZING PERFORMANCE TRADEOFFS FOR THE DELIVERY OF CONCURRENT CHANNEL CHANGE STREAMS TO ENABLE FAST CHANNEL CHANGE IN IPTV NETWORKS

6.1 Introduction

In IPTV networks, channel change latency is a major concern in bringing the perceived performance of IPTV networks close to that of traditional broadcast networks. To minimize the channel change latency, previous research has mostly focused on modifications at the network level [95], for instance, by utilizing a dedicated server for the delivery of additional channel change streams, or, at the client level, by utilizing concurrently delivered source streams to support the channel change process. In this chapter, our focus is on the latter, *i.e.*, IPTV clients join complementary IPTV streams immediately after making a channel change request. Specifically, we propose an analytical framework to evaluate the performance tradeoffs associated with the use of concurrent stream delivery (CSD) techniques to improve the zap response time in IPTV networks.

The performance tradeoffs associated with the delivery of complementary streams have been previously investigated by Sun *et al.* in [98], for the sequential switching case, and in [108] for the random switching case. Even though the proposed research in [98, 108] presents a good starting point to analyze the performance of CSD techniques, the assumptions and the simplicity of the models utilized by the authors to perform their analysis may prevent the proposed frameworks from capturing the

actual performance tradeoffs associated with the use of CSD-based techniques.

Our research addresses these concerns by creating a more generalized evaluation framework to accurately capture the perceived performance in real-life implementations. Specifically, we propose the following modifications to [98] and [108]:

- *Arrival process for the channel change requests:* In [98, 108], the authors used exponential distribution to approximate the distribution for the interarrival times corresponding to the channel change requests. However, as shown in [81], the actual request arrival process, under a time-invariant assumption, closely resembles to a hyper-exponential (or mixtures of exponentials) distribution. For that reason, in this study, we will utilize the hyper-exponential distribution to approximate the request generation process.

- *Synchronization time:* In [98, 108] the synchronization (or display) time associated with each request is modeled by using a constant-valued parameter (based on either the average statistics or the worst-case scenario). However, in practice, the actual synchronization time may differ from one request to another, and among requests from different sessions. Our analysis and the models we proposed in this study take into account the time-varying nature of the synchronization latency.

- *Sessions to switch to:* In [98, 108], transitions from one session to another is limited to one (going up only) or two cases only (going up and random selection). However, the analysis performed by the authors in [81] suggests a more complex model to approximate the distribution for the channel switch events. Furthermore, more importantly, in [98, 108] the authors do not consider the impact of incorrect session join decisions (which may account for half of the channel change events) on the performance metrics. Channel switching model used in our analysis incorporates all the possible scenarios corresponding to the

distribution of successive channel change events.

In short, our research addresses the above limitations to develop a more realistic model to analyze the performance tradeoffs associated with the use of concurrently delivered channel change streams. Using the proposed framework, we evaluate the level of improvement perceived in the latency performance at the clients and the corresponding bandwidth requirements to achieve the stated latency improvements. We compare both approaches in identical scenarios to illustrate the limitations of the exponential-based frameworks.

6.2 System Model

We assume an IPTV network that consists of N clients and M active IPTV sessions, *i.e.*, the set of active users and active sessions are represented with parameters $\{N\}$ and $\{M\}$. To find the maximum overhead associated with these clients, we focus on the closest common access point shared by these clients. We next give an overview of the modifications proposed for the arrival process and the state transitions associated with the channel switching events.

6.2.1 Arrival Process for the Channel Change Requests

We model the arrival process for the channel change requests using the hyperexponential distribution, which has the following probability density function:

$$f(w; \boldsymbol{\lambda}, \mathbf{p}) = \sum_{i=1}^{\kappa} p_i \times \lambda_i \times e^{-\lambda_i w} \tag{90}$$

where κ represents the order for the hyperexponential distribution, p_i represents the probability of selecting the ith exponential distribution (which has an arrival rate of λ_i) from the given mixture model. [1]

[1] $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_\kappa\}$ and $\mathbf{p} = \{p_1, p_2, \ldots, p_\kappa\}$. Hereafter, presented results will assume the following parameter values: $\kappa = 3$, $\boldsymbol{\lambda} = [6, 1, 0.2]$, and $\mathbf{p} = [0.45, 0.35, 0.2]$, with a mean request arrival rate of 3.09. Note that, the parameter values are chosen based on the mean request arrival rate to reflect the respective frequencies of different zapping activities.

Therefore, the cumulative distribution function for the hyperexponential distribution, $F(w; \boldsymbol{\lambda}, \mathbf{p})$, is given by:

$$F(w; \boldsymbol{\lambda}, \mathbf{p}) = 1 - \sum_{i=1}^{\kappa} p_i \times e^{-\lambda_i w} \tag{91}$$

Since channel change events are generated using the hyperexponential renewal process, to find the distributions associated with the consecutive arrival events, we need to solve for the n-fold convolution of $F(w)$, which is given by:

$$F_n(w) = \int_0^w F(w-x) \times f_{n-1}(x) \times dx \tag{92}$$

where $f_{n-1}(x) = d(F_{n-1}(x))/dx$.

We can then represent the generalized equation for $F_n(w; \boldsymbol{\lambda}, \mathbf{p})$ as follows:

$$F_n(w; \boldsymbol{\lambda}, \mathbf{p}) = \sum_{i=1}^{\kappa} \sum_{j=1}^{n} \left(\frac{a_{ij}}{\lambda_i^j} \right) \times \overline{\Gamma}(j, \lambda_i w) \tag{93}$$

where a_{ij} is a constant whose value depends on $(\boldsymbol{\lambda}, \mathbf{p})$ and $\overline{\Gamma}$ represents the lower incomplete Gamma function, *i.e.*, $\overline{\Gamma}(a, x) = \int_0^x e^{-t} \times t^{a-1} \times dt$.

6.2.2 State transitions

In [81] the analysis performed by the authors suggests the use of Zipf distribution for the most popular IPTV sessions (*i.e.*, top 10%), while using exponential distribution for the less popular sessions. [2] Accordingly, we can state the equations for the steady state probabilities corresponding to each session as follows:

$$\pi_i = \begin{cases} \phi_1 \times i^{-\alpha_{zpf}}/\phi & \text{if } i \leq \lfloor \frac{M}{10} \rfloor, \\ e^{-\alpha_{exp} \times i + \phi_2}/\phi & \text{otherwise.} \end{cases} \tag{94}$$

where i represents the rank of a session, α_{zpf} represents the exponent for the Zipf distribution, α_{exp} represents the rate for the exponential distribution, $\{\phi_1, \phi_2\}$ represents the weights associated with the given distributions, and ϕ is the normalization

[2] These distributions can accurately capture the steady state access rates for the majority of the sessions, *i.e.*, ~98% of the total available sessions.

metric, which is used to ensure that $\sum_{\forall i} \pi_i = 1$. [3]

Next, to approximate the probability of making a transition from one session to another, we use the following relationship:

$$P_{ik} = \begin{cases} 0 & \text{if } k = i, \\ p_u & \text{if } k = (i+1)_M, \\ p_d & \text{if } k = (i-1)_M, \\ \left(\sum_{j \in M'(i)} \pi_j / (\varrho \times \pi_k)\right)^{-1} & \text{if } k \in M'(i). \end{cases} \tag{95}$$

where p_u (or p_d) represents the probability of making a sequential switch to the one higher (or lower) numbered (or ranked) session, ϱ equals $(1 - p_u - p_d)$ and it represents the probability of performing a targeted (or non-sequential) switch, $M'(i)$ equals $\{M\} \backslash M(i)$, where $M(i)$ represents the neighborhood sessions for the ith session (*i.e.*, $M(i) = \{(i-1)_M, i, (i+1)_M\}$), and $(\cdot)_M$ represents the modulo operator. [4]

To determine the n-step transition probabilities, we can make use of the Markovian assumption and utilize the nth order transition probability matrix $\mathbf{P}_n$, for which the equation is given by $\mathbf{P}_n = \prod_{i=1}^{n} \mathbf{P}$, where $\mathbf{P}$ is formed by using (95).

To generate the channel change events, we use the terminating renewal process, which refers to a renewal process that terminates after a random number of renewals [97]. The number of renewals is determined based on the Bernoulli process, for which the probability of continuing the trials (or channel switches) is given by p_h. [5]

6.3 Performance Analysis

In this section, we derive the equations corresponding to the following measures: (*i*) *watching probabilities*, (*ii*) *surfing durations*, (*iii*) *latency distributions*, and (*iv*) *bandwidth allocations*. Our study essentially focuses on the performance perceived

[3] Note that, as long as the weight corresponding to one of these distributions is known, we can determine the value of the other weight using $\phi_2 = \alpha_{exp} \times M/10 - \alpha_{zpf} \times \ln(M/10) + \ln \phi_1$.

[4] In our analysis, we assume the following values for p_u and p_d [81]: $p_u = 0.4032$ and $p_d = 0.1568$.

[5] For the terminating renewal process, the expected number of renewals is equal to $p_h/(1 - p_h)$.

during the surfing periods. Here, a surfing period refers to the duration during which the client makes successive channel change requests until the client makes a surf-termination request and settles on the last selected channel.

6.3.1 Channel Watching Probability

We use $P_\omega(i,t)$ to represent the probability of watching the ith session t time units after the surfing period begins (*i.e.*, $t \geq 0$ and $t = 0$ represents the initialization point). To find the equation for $P_\omega(i,t)$, we condition on the initial channel state as follows:

$$P_\omega(i,t) = \sum_{k=1}^{M} \pi_k \times P_{ki}(t) \tag{96}$$

where $P_{ki}(t)$ represents the probability of making a channel switch from the kth session to the ith session at time t.

To find the equation for $P_{ki}(t)$, we condition on the number of channel change requests that occur within the timeframe $(0,t)$. Using the characteristics of the terminating renewal model, we observe two possible scenarios depending on the last accessed state. Specifically, if n channel change requests are observed within a given timeframe, then the result is triggered by either n renewals with no terminating request or more than $n+1$ renewals with the $(n+1)$th renewal being the terminating request. We can express the equation corresponding to these cases as follows:

$$P_{ki}(t) = \lim_{m^* \to \infty} \sum_{m=0}^{m^*} p_h^m \times \mathbf{P}_{ki}^m \times \Big[F'_m(t) + (1-p_h) \times F_{m+1}(t) \Big] \tag{97}$$

where $F'_m(t)$ is given by $F_m(t) - F_{m+1}(t)$. [6]

[6]Note that, as we increase the value of m, we start to observe a linear increase in the ratio of F_m/F_{m+1}. Using this relationship, we can estimate the required parameters using only a small subset of $F_m(t)$ values. Because of the difficulties involved in finding a closed form expression for (97), the above relationship can help significantly in finding an accurate approximation for the summation as $m^* \to \infty$.

6.3.2 Probability of Concurrent Stream Delivery

For the considered concurrent stream delivery approach, we assume a delivery duration of δ time units for the additional channel change streams after a channel change request is made. If no request is made within this δ-long period, then the client stops receiving the additional streams. If, on the other hand, the client makes another channel change request before the concurrent delivery deadline expires, then the upcoming deadline is extended to expire δ time units after the time of the last made request.

Therefore, to find the probability of delivering extra channel change streams at time t, we need to determine whether or not the client has made any channel change request during the last $\min(\delta, t)$ time units. When $t > \delta$, we use the following equation to find the probability of making a channel change request within $(t-\delta, t)$:

$$P(n_q(t-\delta, t) \geq 1) = \sum_{n=1}^{\infty} p_h^n \times \big(F_n(\delta) - F_{n+1}(\delta)\big) \tag{98}$$

where $n_q(t_1, t_2)$ represents the number of requests that the client makes within (t_1, t_2).

As we mentioned in the previous subsection, we also need to address the restriction imposed on the delivery of extra channel change streams by the terminating renewal process. Specifically, there should be no terminating request that is made within $(0, t-\delta)$. The probability corresponding to this restriction is calculated as follows:

$$P_\tau(t-\delta) = 1 - F_1(t-\delta) + \sum_{m=1}^{\infty} p_h^m \times F'_m(t-\delta) \tag{99}$$

On the other hand, when $t \leq \delta$, we only need to find the probability of making at least one channel change request within $(0, t)$, with also the restriction of not making any terminating request within the timeframe $(0, t)$.

Combining these two scenarios, we can state the general equation for the probability of delivering extra channel change streams as follows:

$$E(t) = \begin{cases} P_\tau(t-\delta) \times P(n_q(t-\delta, t) \geq 1) & \text{if } t > \delta, \\ P(n_q(t-\delta, t) \geq 1) & \text{if } t \leq \delta \end{cases} \tag{100}$$

In Figure 34, we illustrate the impact of varying the value of δ on the probability of delivering concurrent streams. Except for the first few seconds where we observe an initial jump, we observe a gradual decrease in the value of $E(t)$, which, after a while, converges to approximately the same value for all the considered δ values (which is caused by the clients terminating the surfing process).

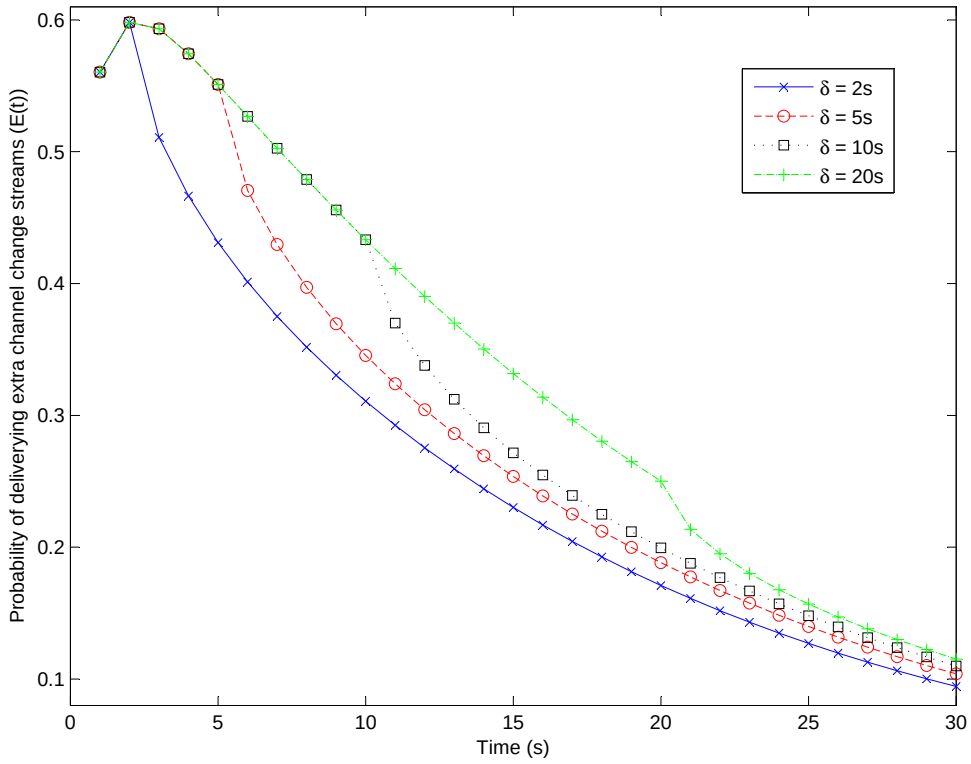


Figure 34: Dependence of $E(t)$ on the δ parameter.

6.3.3 Expected Bandwidth Allocation During the Surfing Period

To find the bandwidth requirements at the access link, we need to determine the steady state probability of delivering each session through the access point to the

clients. There are two possible scenarios for the clients to receive data from the active sessions, *i.e.*, watching the session or requesting it as one of the concurrently delivered extra sessions.

We can therefore state the equation for the probability of receiving the ith session at time t as follows:

$$R(i,t) = P_{\omega}(i,t) + E(t) \times S(i,t) \tag{101}$$

where $S(i,t)$ represents the probability of receiving the ith session as one of the additional channel change streams. The equation for $S(t)$ is given as follows:

$$S(i,t) = \begin{cases} \sum_{j=2}^{M} P_{\omega}(j,t) & \text{if } i = 1, \\ P_{\omega}(1,t) + P_{\omega}(3,t) + P_{\omega}(M,t) & \text{if } i = 2, \\ P_{\omega}(1,t) + P_{\omega}(2,t) + P_{\omega}(4,t) & \text{if } i = 3, \\ P_{\omega}(2,t) + P_{\omega}(3,t) + P_{\omega}(5,t) & \text{if } i = 4, \\ P_{\omega}(i-1,t) + P_{\omega}(i+1,t) & \text{otherwise.} \end{cases} \tag{102}$$

We can then determine the expected bandwidth utilization using the probability of delivering each session to the clients connected to the same access point as follows:

$$E[W(t)] = W_M - \sum_{m=1}^{M} w(m) \times \left[1 - R(m,t)\right]^N \tag{103}$$

where $w(m)$ represents the multicast transmission rate for the mth session, and W_M equals $\sum_{m=1}^{M} w(m)$.

In Figure 35 we illustrate the bandwidth utilization at the access point when δ equals $6s$. In accordance with the results corresponding to the concurrent delivery rates, bandwidth utilization peaks initially as the clients enter the surfing period, after which it starts to decrease as it converges to the steady state values due to the users exiting the surfing period.

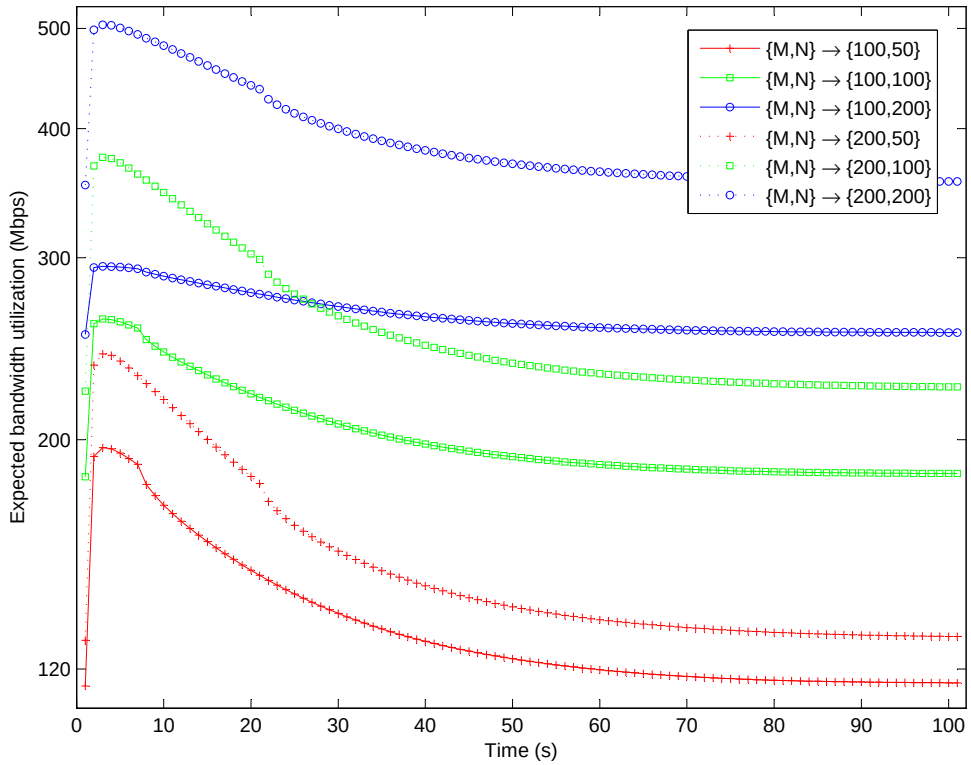


Figure 35: $E[W]$ as we vary $\{N, M\}$, when $\delta = 6s$ and $w(m) = 3Mbps$, $\forall m \in \{M\}$.

6.3.4 Expected Channel Change Latency

In this section, we derive the equations for the channel change latency. Our main focus is on finding the expected value for the accumulative latency during a surfing period and the mean latency per request. We carry out our analysis by initially focusing on the first four requests-to summarize our approach in detail-and then presenting a general formula to represent the rest.

Since the surfing period initiates with no extra channels being delivered, to find the latency for *the first channel change request*, we use the average waiting time for an arbitrarily received request. Let us assume that the distribution for the synchronization (or display) latency is given by $g_i(t)$ for the ith session. Then the expected latency for the first channel change request is given by the following equation:

$$L_{i,1} = \int_{g_{i,\min}}^{g_{i,\max}} t \times g_i(t) \times dt \tag{104}$$

where $g_{i,\min}$ and $g_{i,\max}$ represent the boundary values for the display latency and $g_i(t)$ represent the distribution for the display latency. For the rest of the chapter, we will assume a uniform distribution for the display latency, *i.e.*, $g_i(t) = 1/g_d$, where $g_d = g_{i,\max} - g_{i,\min}$. [7] Based on these assumptions, the expected latency for the first request becomes equal to $g_s/2$, where $g_s = g_{i,\max} + g_{i,\min}$.

To find the latency for the higher order requests, we need to determine the possible scenarios for each of these requests. Specifically, for *the second channel change request*, we observe four different scenarios:

Scenario 2.1: The request arrives within $(0, g_{i,\max})$ of the first request, where $g_{i,\max} < \delta$, and the request is targeting a session that is already being delivered because of the previously received request, *i.e.*, $s_2 \in \{N_1\}$, where s_i represents the session targeted by the ith request ($1 \leq s_i \leq M$) and $\{N_j\}$ represents the extra sessions delivered based on the jth request ($j \geq 1$). In this case, the equation for the expected latency is given as follows:

$$l_{i,2} = \int_{g_{i,\min}}^{g_{i,\max}} \frac{1}{g_d} \int_0^t (t-w) \times f(w) \times dw \times dt \tag{105}$$

Scenario 2.2: The request arrives within $(g_{i,\max}, \delta)$ of the first request, and $s_2 \in \{N_1\}$. In this case, the latency equals 0, *i.e.*, $l_{i,2} = 0$.

Scenario 2.3: The request arrives within (δ, ∞) of the first request. In this case, it does not matter whether or not the current session is delivered based on the previous request, since the system delivers the additional streams for a δ-long period. Hence, for the current case, $l_{i,2} = L_{i,1}$.

Scenario 2.4: The request targets a session that is not included among the concurrently delivered extra sessions, *i.e.*, $s_2 \notin \{N_1\}$. If that is the case, then $l_{i,2} = L_{i,1}$.

Therefore, combining the above scenarios we can find the expected latency during

[7] Note that the distribution selected to represent the synchronization latency does not affect the approach we implement to find the expected latency.

Table 2: Sample values for the ρ_n metric

M	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
100	1	0.7114	0.2673	0.1899	0.0837	0.0635
600	1	0.7089	0.2571	0.1826	0.0778	0.0592

the second channel change as follows:

$$L_{i,2} = \frac{g_s}{2} \times \sum_{j=1}^{\kappa} p_j \times \left(1 + \rho_2 \times e^{-(\lambda_j \times \delta)}\right) - \rho_2 \times \sum_{j=1}^{\kappa} \left(\frac{p_j}{\lambda_j}\right) \times (1 + \frac{g_{i,j}}{g_d}) \qquad (106)$$

where ρ_2 represents the probability of $s_2 \in N_1$ and $g_{i,j}$ equals $(e^{-(\lambda_j \times g_{i,\max})} - e^{-(\lambda_j \times g_{i,\min})})/\lambda_j$.

In general, we can determine the value of ρ_n (where $n \geq 2$) using the following equation:

$$\rho_n = \sum_{\forall s_1} \pi^*_{s_1} \sum_{\forall s_2} \cdots \sum_{\forall s_{n-1}} \prod_{j=1}^{n-2} p_{s_j s_{j+1}} \prod_{j=1}^{n-1} \left|Q_{s_j}\right|^* \times \underline{P}_{s_{n-1}} \qquad (107)$$

where $\pi^*_{s_1}$ represents the probability of receiving s_1 with the first channel change request, $|Q_s|^*$ represents a $1 \times M$ row vector for the sth IPTV session which consists of elements that identify the delivery status of the additional channel change streams [8], and $\underline{P}_s$ represents the $M \times 1$ 1-step transition probability vector for the sth IPTV session [9].

The values for the first few ρ_n metrics (when $n \geq 1$) are shown in Table 2. In general, the value of ρ_n does not show significant variations as the number of sessions is varied.

For *the third channel change request*, we combine the possible scenarios based on the latency expectations and determine four major scenarios, which are listed as follows:

Scenario 3.1: The second and the third requests arrive within $(0, g_{i,\max})$ of the first

[8] $Q_s(j) = 1$, if s_j is among the additional channel change streams delivered to the zapping user when the user made a channel change request, which targets session s.

[9] $\underline{P}_s(j)$ equals p_{ss_j}.

request, and $s_3 \in \{N_1, N_2\}$. In this case, the expected latency is given as follows:

$$l_{i,3} = \int_{g_{i,\min}}^{g_{i,\max}} \int_0^t \int_0^{t-w_1} \frac{g_i(t) \times f(w_1) \times f(w_2)}{(t - w_1 - w_2)^{-1}} \times dw_2 \times dw_1 \times dt \quad (108)$$

$$= \frac{g_s}{2} - \sum_{j=1}^{\kappa} \left(\frac{p_j \times g_{ij}}{g_d} \right) \times \left(s + \frac{1}{\lambda_j}\right) + \sum_{j=1}^{\kappa} p_j^2 \times \left(\frac{g_{\min,ij} - g_{\max,ij}}{\lambda_j \times g_d} \right)$$
$$+ \sum_{j=1}^{\kappa} \sum_{\substack{k=1 \\ k \neq j}} \frac{p_k \times p_j \times (g_{ij} - g_{ik})}{g_d \times (\lambda_j - \lambda_k)} - 2 \times \sum_{j=1}^{\kappa} \frac{p_j}{\lambda_j} \quad (109)$$

where $g_{\max,ij}$ equals $e^{-\lambda_j \times g_{i,\max}} \times (g_{i,\max} + 1/\lambda_j)$ and $g_{\min,ij}$ equals $e^{-\lambda_j \times g_{i,\min}} \times (g_{i,\min} + 1/\lambda_j)$.

Scenario 3.2: We have $s_3 \in \{N_2\}$ but $s_3 \notin \{N_1\}$ with no restriction on the arrival times for the second and third requests. In this case, the expected latency equals $L_{i,2}$.

Scenario 3.3: The second request arrives within $(0, g_{i,\max})$ of the first request and the third request arrives δ seconds after the second request, OR $s_3 \notin \{N_2\}$. For these cases, the expected latency equals $L_{i,1}$.

Scenario 3.4: The second and the third requests arrive within $(g_{i,\max}, \delta)$ of the first request, and $s_3 \in \{N_1, N_2\}$, OR, the second request arrives within $(0, g_{i,\max})$ and the 3rd request arrives within $(g_{i,\max}, \delta)$ of the first request, and $s_3 \in \{N_1, N_2\}$. For these cases, the expected latency equals 0.

Consequently, we can express the expected latency for the third channel change request by using the following equation:

$$L_{i,3} = (1 - \rho_2) \times L_{i,1} + (\rho_2 - \rho_3) \times L_{i,2} + \rho_3 \times L^*_{i,3} \quad (110)$$

where

$$L^*_{i,3} = \left(\sum_{j=1}^{\kappa} p_j \times e^{-(\lambda_j \times \delta)} \right) \times \left[L_{i,1} \times \left(1 - \sum_{j=1}^{\kappa} p_j \times e^{-(\lambda_j \times \delta)}\right) + L_{i,2} \right] + l_{i,3} \quad (111)$$

For the higher order requests, except for a small number of scenarios, we can use the equations corresponding to the previously received requests (*i.e.*, the lower order requests) to determine the final equation for the expected latency. To exemplify this

process, we next examine the possible scenarios for *the fourth channel change request* and list them as follows:

Scenario 4.1: If $s_4 \in \{N_2, N_3\}$ but $s_4 \notin \{N_1\}$, then the expected latency equals $L_{i,3}$.

Scenario 4.2: If $s_4 \in \{N_3\}$ but $s_4 \notin \{N_2\}$, then the expected latency equals $L_{i,2}$.

Scenario 4.3: If $s_4 \notin \{N_3\}$, then the expected latency equals $L_{i,1}$.

Scenario 4.4: If $s_4 \in \{N_1, N_2, N_3\}$, then we can determine four major scenarios, for which the categorization can be made based on the interarrival times between the consecutive requests. Specifically, we can list these four cases as follows:

Scenario 4.4.1: If the 2nd request arrives within (δ, ∞), then the expected latency equals $L_{i,3}$.

Scenario 4.4.2: If the 2nd request arrives within $(g_{i,\max}, \delta)$ and if the 3rd request arrives Δt_3 seconds later, where $\Delta t_3 > \delta$, then the expected latency equals $L_{i,2}$.

Scenario 4.4.3: If the 2nd request arrives within $(0, \delta)$ and the 3rd requests arrives Δt_3 seconds later, where $g_{i,\max} < \Delta t_3 \leq \delta$, OR if the 2nd request arrives within $(0, g_{i,\max})$, and $\Delta t_3 \leq g_{i,\max}$, and the 4th requests arrives Δt_4 seconds later, where $\Delta t_4 > \delta$, then the expected latency equals $L_{i,1}$.

Scenario 4.4.4: If all the requests arrive within $(0, g_{i,\max})$ then the expected latency is calculated as follows:

$$\begin{aligned} l_{i,4} &= \int_{g_{i,\min}}^{g_{i,\max}} \int_0^t \int_0^{t-w_1} \frac{g_i(t) \times f_2(w_1) \times f(w_2)}{(t - w_1 - w_2)^{-1}} \times dw_2 \times dw_1 \times dt \\ &= \int_{g_{i,\min}}^{g_{i,\max}} g_i(t) \times \Big(t \times F_3(t) - F_3^*(t) \Big) \times dt \end{aligned} \tag{112}$$

where $F_3^*(t)$ is given by $\int_0^t w \times f_3(w) \times dw$.

We can then calculate the expected latency for the 4th request by using the following equation:

$$L_{i,4} = (1 - \rho_2) \times L_{i,1} + (\rho_2 - \rho_3) \times L_{i,2} + (\rho_3 - \rho_4) \times L_{i,3} + \rho_4 \times L_{i,4}^* \tag{113}$$

where

$$L^*_{i,4} \quad = \quad l_{i,4} + \Delta_\Lambda \times \sum_{j=1}^{3} L_{i,j} \times (1 - \Delta_\Lambda)^{3-j} \tag{114}$$

where Δ_Λ equals $\sum_{j=1}^{\kappa} p_j \times e^{-(\lambda_j \times \delta)}$.

In Figure 36 we illustrate the impact of varying the δ parameter and $g_{\min\,/\,\max}$ values on the expected latency for the second, third, and fourth requests. As expected, increasing the δ value initially decreases the latency, since the probability of a request arriving during a concurrent stream delivery period increases. However, δ is also shown to have a limited impact on the latency, as the expected latency starts to converge as δ is increased beyond a certain limit, which depends on the system parameters. We also observe that the difference among the expected latency values of different requests decreases as we increase the value of δ.

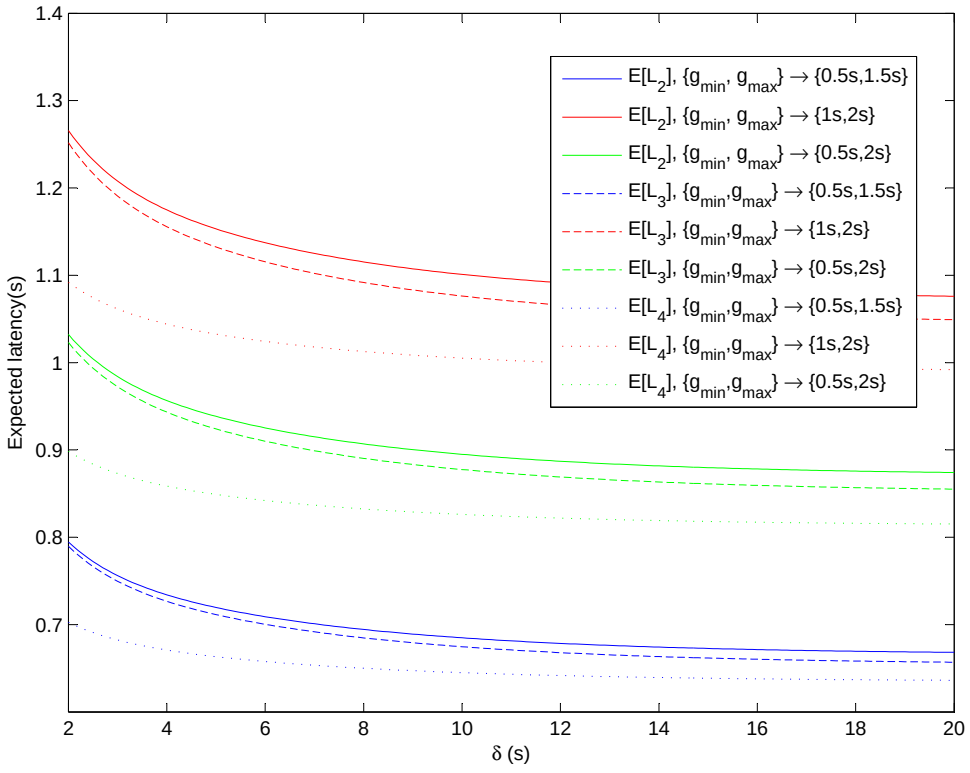


Figure 36: Expected latency for the kth request, when k is selected from $[2, 3, 4]$.

In short, to find the higher order $L_{i,k}$ values, we need to perform two additional

calculations at each step. For instance, for the 4th request, these calculations are performed to determine the values, which correspond to $L^*_{i,4}$ and ρ_4. Consequently, we can determine the total expected channel change latency (during a single surfing period) by using the following equation:

$$E[L_{sp}] = \sum_{i=1}^{M} \pi_i \times E[L_{sp,i}] \tag{115}$$

where

$$E[L_{sp,i}] = \sum_{j=1}^{\infty} \sum_{k=1}^{j} p_h^j \times (1 - p_h) \times L_{i,k} \tag{116}$$

$$= \sum_{j=1}^{\infty} p_h^j \times L_{i,j} \tag{117}$$

We can approximate the equation for $E[L_{sp,i}]$ as follows:

$$E[L_{sp,i}] = \sum_{j=1}^{\infty} L^*_{i,j} \times \Big[\frac{1 + p_h \times (\rho_j - \rho_{j+1})}{\rho_j^{-1} \times p_h^{-j}}\Big] + \rho_j \times (\rho_j - \rho_{j+1})$$
$$\times L^*_{i,j} \times \sum_{k=j+1}^{\infty} p_h^{k+1} \prod_{l=j+1}^{k} (1 + \rho_l - \rho_{l+1}) \tag{118}$$

where $L^*_{i,1}$ equals $L_{i,1}$, $L^*_{i,2}$ equals $\big(L_{i,2} - (1 - \rho_2) \times L_{i,1}\big)/\rho_2$, and $L^*_{i,3}$ is given by:

$$L^*_{i,3} = \frac{L_{i,3} - (1 - \rho_2) \times L_{i,1} - (\rho_2 - \rho_3) \times L_{i,2}}{\rho_3} \tag{119}$$

We can further simplify (118) by approximating the sum of elements corresponding to the higher order latencies (*i.e.*, $j \geq 5$) as follows:

$$E[L_{sp_i}]|_5^{\infty} \approx \frac{L^*_{i,5} \times p_h^5 \times \rho_5}{(1 - p_h)} \times \Big(1 + \frac{p_h \times (\rho_5 - \rho_6)}{(1 - p_h)}\Big) \tag{120}$$

Similarly, we can determine the average latency per request using $E[L] = \sum_{i=1}^{M} \pi_i \times E[L_i]$, where the equation for $E[L_i]$ is given as follows:

$$E[L_i] = \sum_{j=1}^{\infty} L_{i,j} \times (1 - p_h) \times \Big(- ln(1 - p_h) - \sum_{k=1}^{j-1} \frac{p_h^k}{k}\Big) \tag{121}$$

In Figure 37 we illustrate the impact of varying the δ and p_h values on the mean value for the latency per request. [10] Compared to waiting for the next GOP sequence to initiate the decoding process, CSD technique leads to noticeable improvements in the latency performance. However, the improvements are not significant enough to justify the increased bandwidth requirements at the access point as shown in Figure 35.

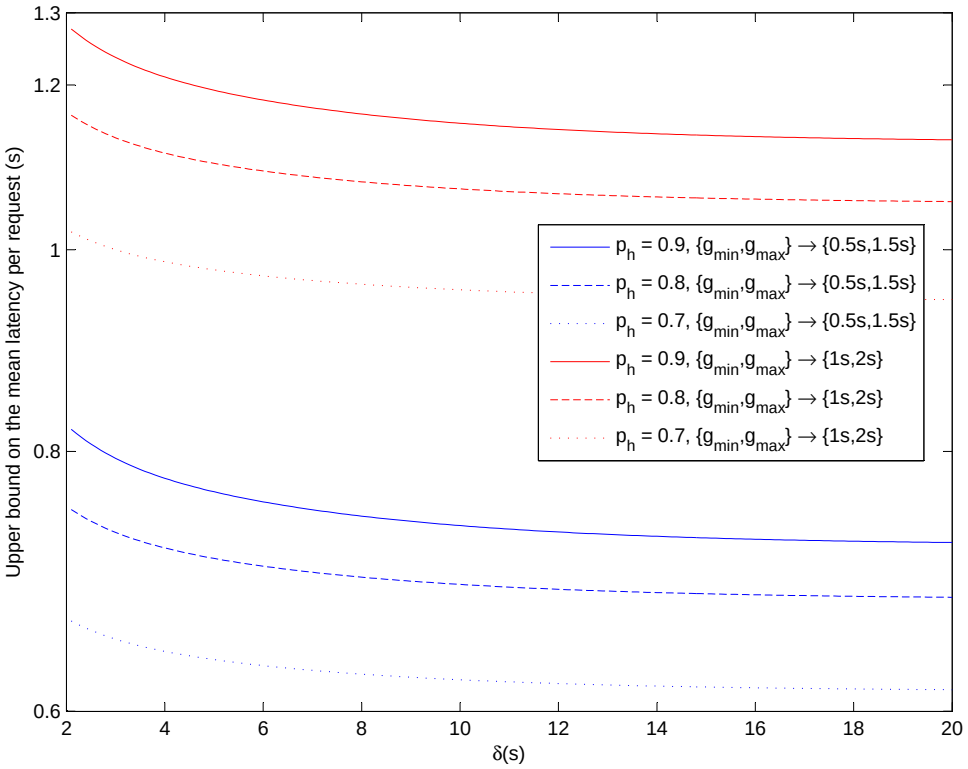


Figure 37: Expected value for the channel change latency per received request.

We can also express the probability of the channel change latency being equal to

[10] Note that in the figure we show the weighted results, which also assumes the terminate request case, which has a latency of 0. We will shortly illustrate the results that excludes the given scenario, which leads to an expected latency of $E[L]/p_h$.

zero as follows (when $g(t) = 1/g_d$):

$$P_{Li1,0} = 0 \tag{122}$$

$$P_{Li2,0} = \rho_2 \times \Big[-\sum_{j=1}^{\kappa} p_j \times e^{-(\lambda_j \times \delta)} - \sum_{j=1}^{\kappa} \frac{p_j \times g_{i,j}}{g_d} \Big] \tag{123}$$

$$\begin{aligned} P_{Li3,0} = {} & (\rho_3 - \rho_2) \times \sum_{j=1}^{\kappa} p_j \times \Big[\frac{g_{i,j}}{g_d} + e^{-(\lambda_j \times \delta)} \Big] \\ & + \rho_3 \times \sum_{j=1}^{\kappa} \frac{(1 - e^{-(\lambda_j \times \delta)})}{g_d / p_j} \times \Big[\sum_{\substack{k=1 \\ k \neq j}}^{\kappa} \frac{p_k \times (g_{i,k} - g_{i,j})}{(\lambda_j - \lambda_k)/\lambda_k} \\ & + p_j \times (g_{\min,ij} - g_{\max,ij}) \Big] \end{aligned} \tag{124}$$

and

$$\begin{aligned} P_{Li4,0} = {} & \widetilde{P}^*_{Li4,0} + P_{Li3,0} \times \big(\rho_3 - \rho_4 + \rho_4 \times \sum_{j=1}^{\kappa} p_j \times e^{-(\lambda_j \times \delta)} \big) + P_{Li2,0} \\ & \times \big(\rho_2 - \rho_3 + \rho_3 \times (1 - \sum_{j=1}^{\kappa} p_j \times e^{-(\lambda_j \times \delta)}) \sum_{j=1}^{\kappa} p_k \times e^{-(\lambda_k \times \delta)} \big) \end{aligned} \tag{125}$$

where

$$\begin{aligned} \widetilde{P}^*_{Li4,0} = {} & \frac{\big(\Delta_\Lambda - 1\big) \times \sum_{j=1}^{\kappa} p_j \times (\delta_j + \frac{g_{ij}}{g_d})}{1 - \Delta_\Lambda + \sum_{j=1}^{\kappa} p_j \times (1 + \frac{g_{ij}}{g_d})} + \sum_{j=1}^{\kappa} \frac{p_j^3 v \times \delta_j \times g_{ij}}{(\frac{g^*_{ij}}{g_{ij}} - 1)^{-1}} \\ & + \sum_{j=1}^{\kappa} \sum_{\substack{k=1 \\ k \neq j}}^{\kappa} \frac{p_j \times p_k}{\lambda^*_{jk}} \Bigg[\frac{g_{ij} - g_{ik}}{\lambda_k^{-1}} \Big(\sum_{\substack{m=1 \\ m \neq j}}^{\kappa} \frac{\delta_j \times \lambda^*_{jm} - \lambda_{jm}}{\lambda^*_{jm} \times p_m^{-1}} \\ & + \frac{\delta_j \times \lambda^*_{jk} + \lambda_j}{\lambda^*_{jk} \times p_k^{-1}} \Big) + \frac{(\delta_j + 1) \times \lambda^*_{jk} + 3 \times \lambda_k}{p_k^{-1} \times (g^*_{ik} - g_{ik})^{-1}} \Bigg] \\ & - \sum_{j=1}^{\kappa} \sum_{k=1}^{\kappa} \sum_{m=1}^{\kappa} p_j \times p_k \times p_m \times \delta_j \times (g_{ik} + g_d) \end{aligned} \tag{126}$$

where δ_j equals $e^{-\lambda_j \delta}$, $g^*_{i,k}$ equals $g_{\min,ik} - g_{\max,ik} + g_{ik}$, λ^*_{jk} equals $\lambda_j - \lambda_k$, and λ_{jk} equals $\lambda_j + \lambda_k$.

We compare the impact of δ and $(g_{\min}, g_{\max})$ values on the probability of zero channel change latency in Figure 38. For the most part, we observe better latency results at the higher request counts, especially when the value of δ is kept above a certain limit.

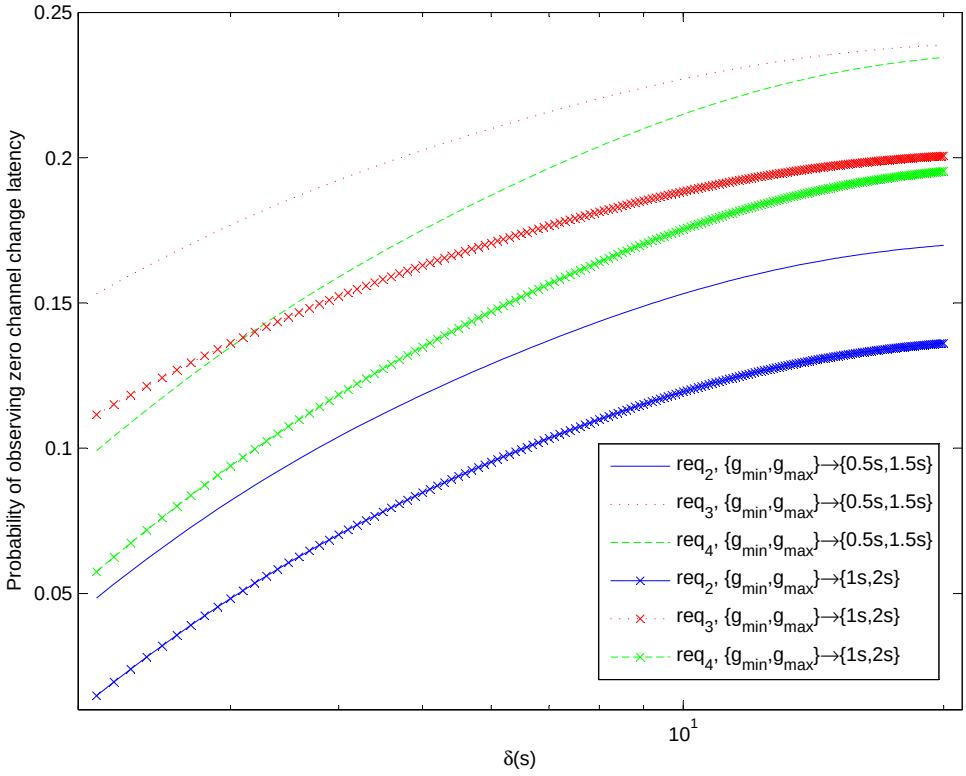


Figure 38: $P_{Lik,0}$ when k is selected from the set $[2, 3, 4]$, and δ (in log-scale) is selected from $(2s, 20s)$.

In Figure 39, we illustrate the impact of increasing the request count on the value of $P^*_{Lik,0}$, where $P^*_{Lik,0}$ is defined as $P_{Lik,0} - \sum_{j=1}^{k-1}(\rho_j - \rho_{j+1}) \times P_{Lij,0}$. Specifically, we focus on the scenario when $s_j \in \{N_1, N_2, \ldots, N_{j-1}\}$. At the limit, increasing the order count has the impact of increasing the probability of observing zero latency, when the last observed session is delivered during the concurrent delivery period of the previously switched sessions.

6.4 Discussions

In this section, we compare the proposed analytical framework, within which the channel change requests are generated based on the mixture exponential distribution, to the referred channel change framework, which assumes exponentially distributed request interarrival times. For both scenarios, we use the same values for the mean

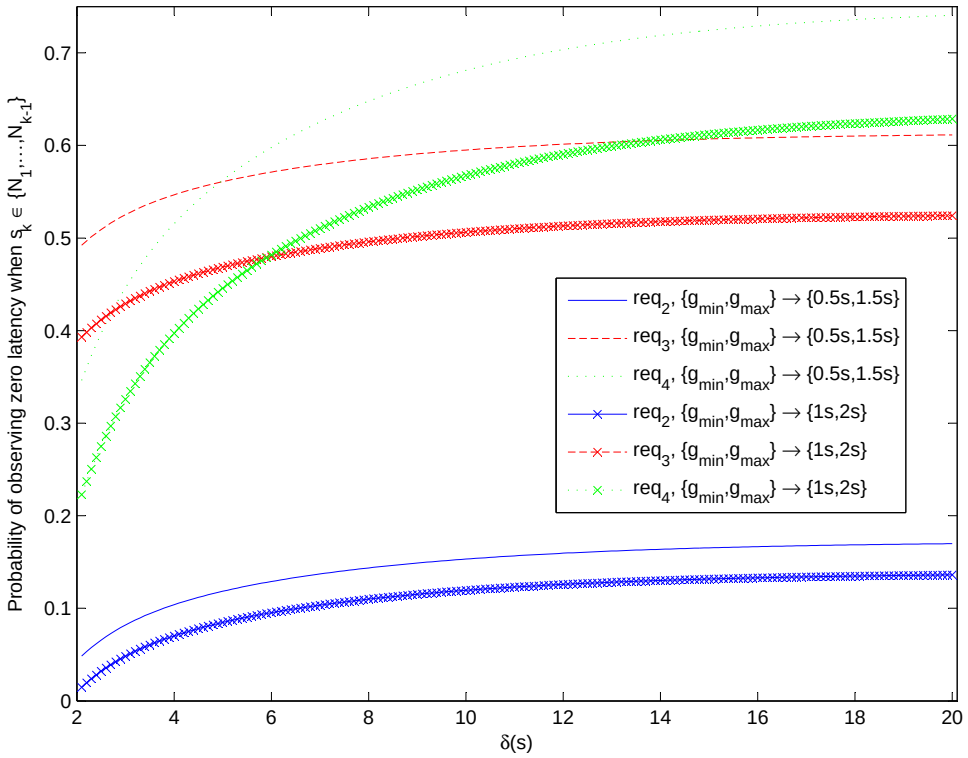


Figure 39: $P^*_{Lik,0}$ when k is selected from the set $[2, 3, 4]$.

request arrival rate, $\bar{\lambda} = 3.09$. We assume the concurrent delivery of three additional channel change streams during a surfing period. To improve the accuracy of our comparisons, for both scenarios, we use the same channel state transition matrix, which assumes the following initial parameter values: $\alpha_{zpf} = 0.513$, $\alpha_{exp} = 0.06$, and $\phi_1 = 12.642$ [81]. [11]

In Figure 40, we illustrate the difference in bandwidth requirements for various M and N values, when δ equals $6s$. In general, we observe that the exponential-based framework overestimates the bandwidth requirements for the early periods (when $t \leq 2\delta$), while underestimating the bandwidth requirements for the later periods. The deviation rate increases as the number of users decreases, or as the duration of the surfing period is increased, and vice versa. Also note that as the duration of the

[11] The values for ϕ_2 and ϕ are selected based on the number of available sessions, *e.g.*, if $M = 100$ then $\phi_2 = 1.4158$ and $\phi = 331.472$, if $M = 200$ then $\phi_2 = 1.12$ and $\phi = 392.249$.

observation period increases, the results for both approaches start to converge, with the ratio becoming very close to 1.

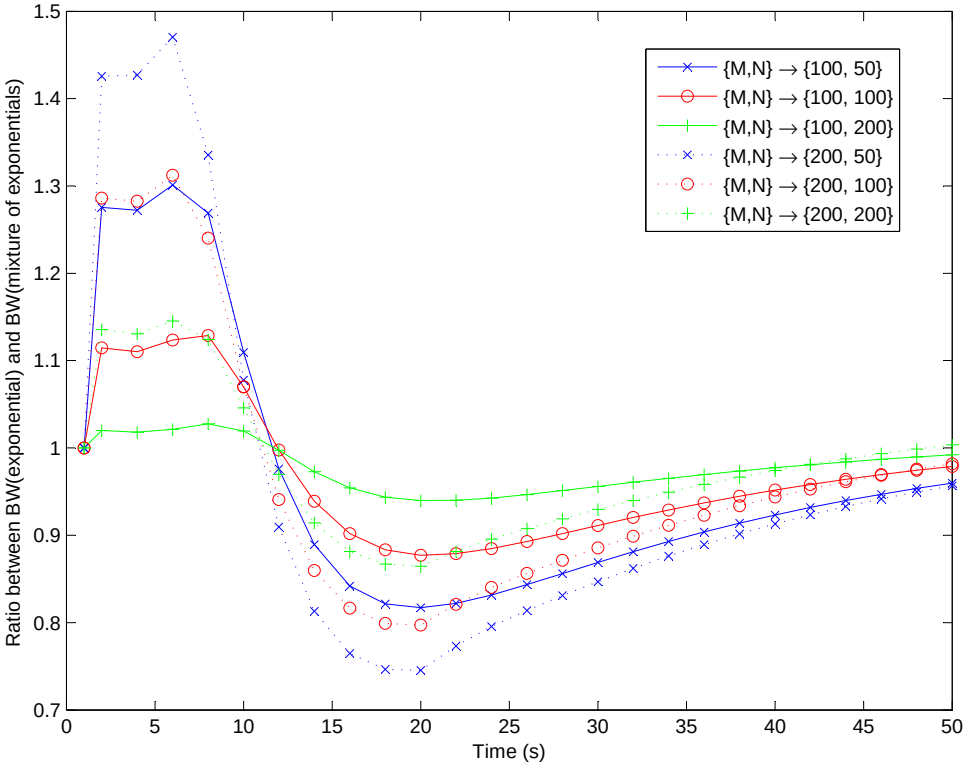


Figure 40: Exponential vs. mixture of exponentials bandwidth comparison as we vary $\{N, M\}$ when $\delta = 6s$.

In Figure 41, we compare the expected latency performance of the two frameworks. Based on these results, we observe two major differences between the two frameworks. The first difference is that the exponential based approach overestimates the improvements in the expected latency performance. The second difference is that the exponential based framework cannot capture the variations in the latency performance at the desired accuracy level. We also consider the following factors to have an effect on the observed differences: exclusion of incorrect channel change decisions from performance evaluations in the referred framework and the limited number of choices implemented to model the state transition events.

In Figure 42, we illustrate the ratio between the expected latency values for the

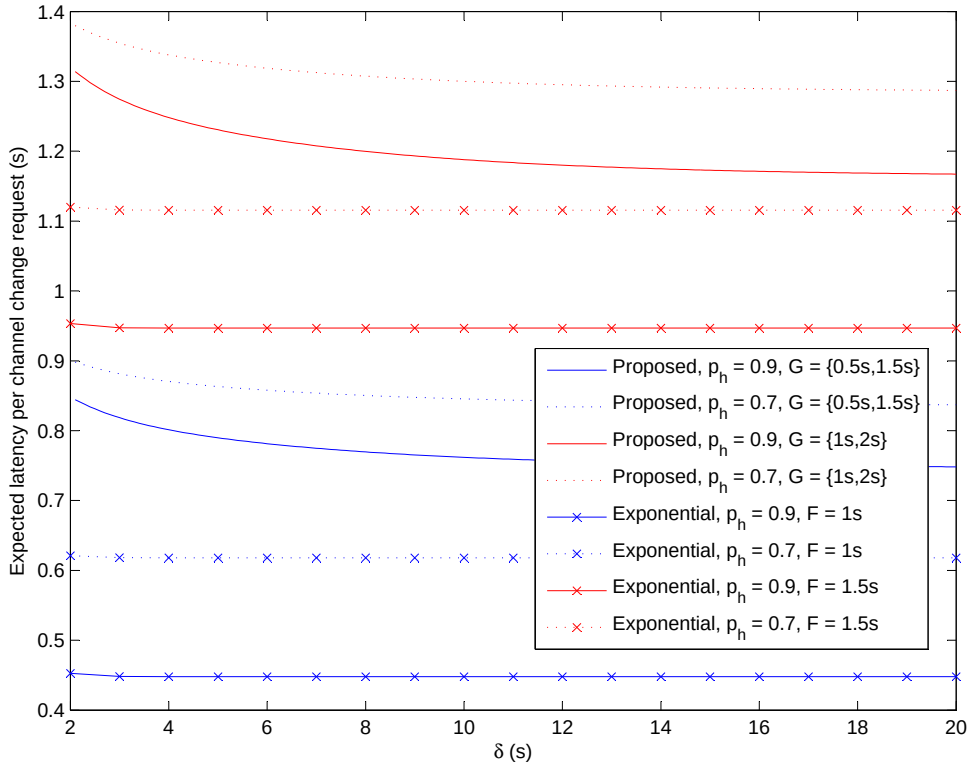


Figure 41: Exponential vs. mixture of exponentials latency per request comparison.

proposed and the referred frameworks as we vary the value of p_h and the mean display latency values. We notice that the percentile difference between the two approaches decreases as the value of p_h decreases, since the number of possible channel change scenarios decreases, or the mean display latency value increases, since the impact of earlier channel change events increases.

6.5 Conclusions

In this chapter, we proposed a novel analytical framework to evaluate the performance of concurrent stream delivery techniques in IPTV networks. By integrating all the major contributing factors within the proposed framework, our model was able to closely approximate the realistic conditions. Exemplary performance evaluations, which showed close to 40% differences in latency and bandwidth results compared to the exponential based framework, emphasized the importance of the proposed

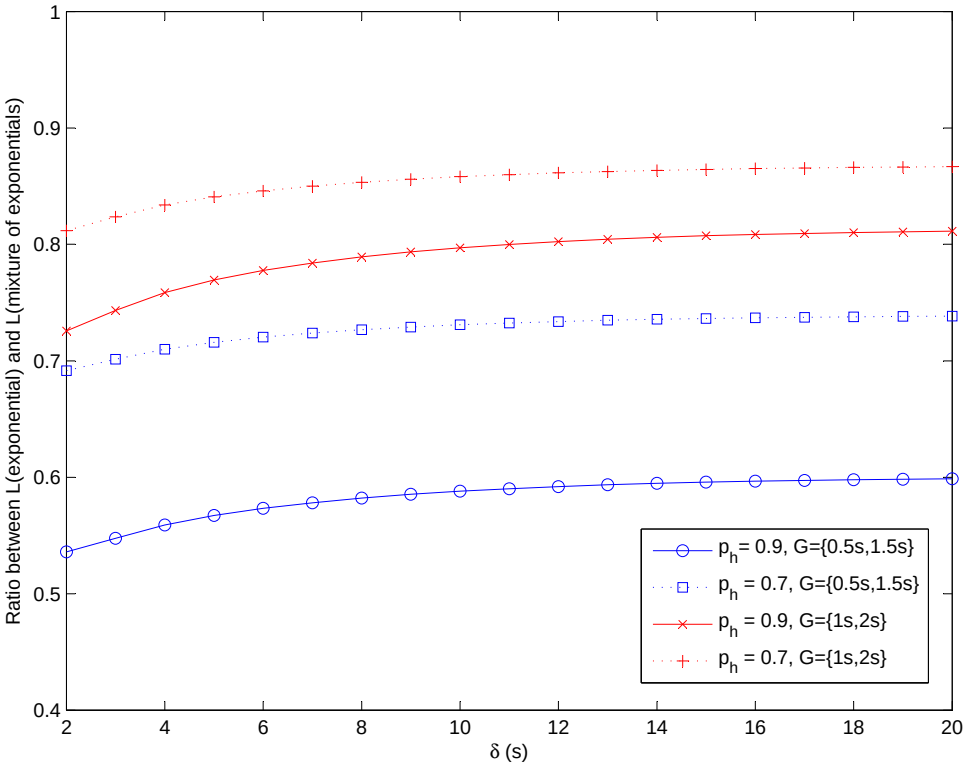


Figure 42: Exponential vs. mixture of exponentials latency per request comparison.

extensions. Our observations also suggest the need of a more exhaustive study on concurrent delivery based channel change techniques, as the perceived improvements in the latency performance were not sufficient to justify the significant increase on bandwidth requirements at the access network.

CHAPTER VII

A PEER-ASSISTED DELIVERY FRAMEWORK TO SUPPORT IPTV CHANNEL CHANGE

7.1 Introduction

Channel change latency refers to the delay from the time a channel change request is made to the time the selected channel is displayed on the user equipment. Traditional broadcast technologies typically offer very small channel change delays, since the users of these technologies can simultaneously receive the content for all the available channels. As a result, for these technologies, channel change delay essentially represents the time to locally make the necessary adjustments to switch to the content received on a different frequency or the time to decode the received content.

On the other hand, IPTV networks utilize the Switched Digital Video (SDV) architecture, which allows the users to have simultaneous access to a limited number of channels. The maximum number of channels an end-user can simultaneously receive depends on a few factors, such as user bandwidth availability and stream bandwidth requirements. To initiate and proceed with the channel change process, IPTV users need to perform session-join and -leave tasks at the network level by communicating directly with the multicast proxy servers. As a consequence of performing channel change operations at the network level, we observe signaling (IGMP join/leave), random access point (RAP) acquisition (*i.e.*, receiving key frames), and buffering delays. To achieve the desired latency performance, previous research has mostly focused on minimizing the RAP acquisition delay. For instance, by using a dedicated server to unicast transmit the channel change data to end-users before the zapping user joins the multicast session [20, 35], by creating separate multicast sessions with different

key frame delivery times [49, 21], by using concurrently delivered streams during surfing periods [98, 108], or by focusing on the characteristics of the broadcast content and the method used to deliver it at the same time (for details, see [95] and references within).

In the previous chapter, we focused on the use of concurrently delivered streams and its impact on the latency performance and the network overhead. We observed significant increase in the required overhead to achieve the desired latency performance. Even though client-based solutions are typically preferred over other schemes as they require the least amount of changes in the network infrastructure, the perceived results in channel change overhead suggested the use of a more network-centric approach to solve the latency problem. Since server-based solutions were shown to achieve the optimal performance in reducing the channel change latency (when compared to all the considered channel change techniques), in this chapter our focus is on the design of a server-based channel change framework.

In server-based channel change solutions, the dedicated servers typically respond to each received request individually, by creating a distinct unicast channel-change stream for each received request. However, these performance improvements come at the expense of an increased server load. As the number of users connected to a channel change server increases, we may start to observe more frequent server overloads. Figure 43 illustrates the relationship between cumulative channel change rate and the probability of a server overload. Notice that, as the channel switching rate increases, server overload probability may reach unacceptable levels. That is especially true, when the IPTV broadcasts use longer group-of-picture (GOP) durations. To keep the probability of server overload within acceptable limits, we may need to implement highly selective request admission policies, which may, as a result, mitigate the advantage of implementing server-based channel change policies.

To maximize the efficiency of a server-based channel change policy, we need to

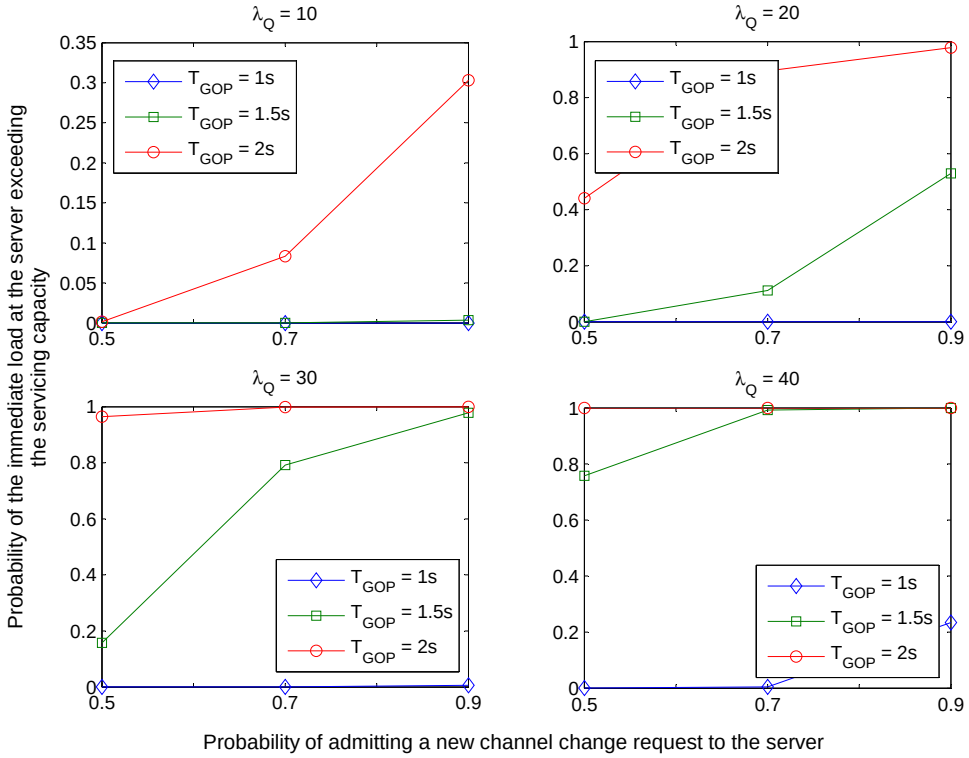


Figure 43: Probability of the server being overloaded as the request arrival rates (λ_Q), request admission rates, and GOP durations (T_{GOP}) are varied.

support it with resource efficient delivery techniques. The proposed research achieves this objective by creating a distributed delivery framework, which makes use of the resource availability at the session peers. To the best of our knowledge, for IPTV networks, the current research offers the most comprehensive discussion on how to combine the strengths of a dedicated server with that of the session peers to achieve a scalable, low latency, and resource efficient channel change framework. In our study, we essentially focus on IPTV sessions with longer GOP durations, since they display the worst synchronization latency characteristics. Through simulations, we showed that our approach is highly effective in reducing the channel change latency while also keeping the overhead at manageable levels.

7.2 *Proposed Channel Change Architecture*

Figure 44 illustrates the main features of the proposed channel change framework, which consists of a group of dedicated servers, referred to as the Channel Switch Coordinators (CSCs), and the session peers. [1] For the given framework, CSC servers are mainly responsible for managing and coordinating the channel change requests received from the session peers.

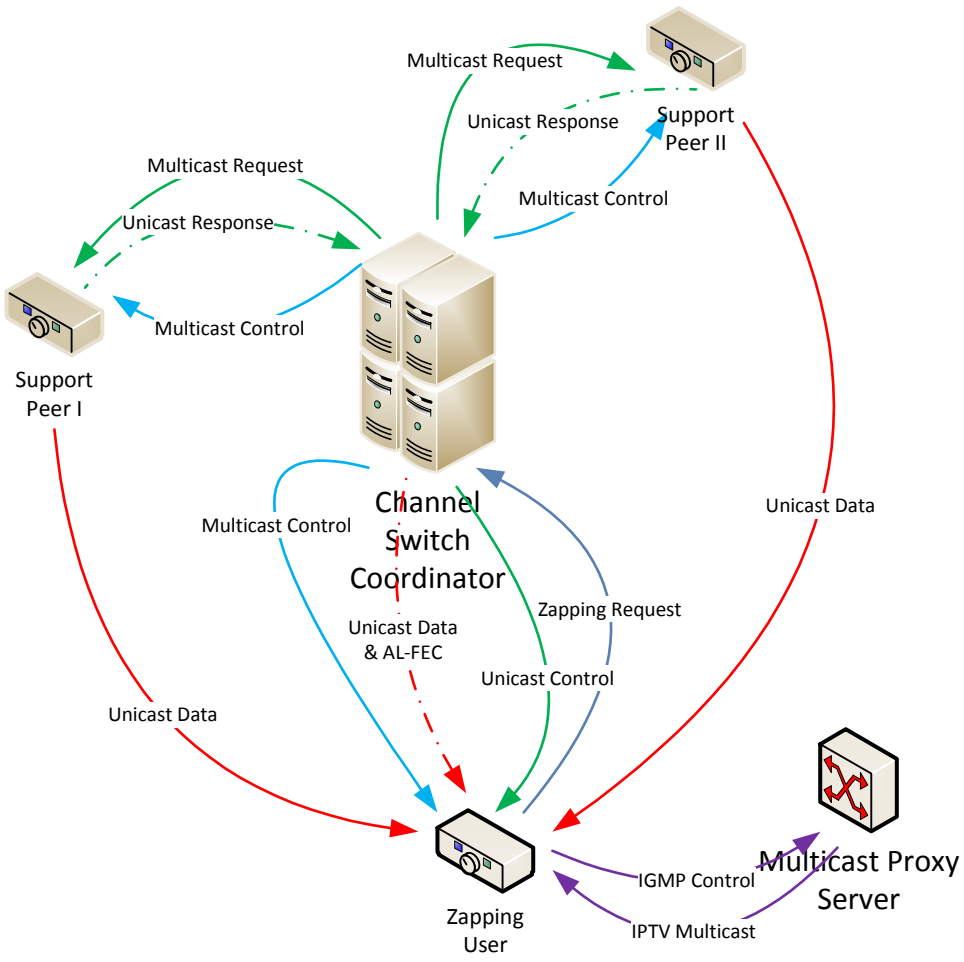


Figure 44: Proposed fast channel change framework.

To initiate the channel change process, zapping user sends a request message, with information on the targeted channel change event, to its corresponding CSC

[1] Hereafter, we refer to active IPTV clients as session peers. In Figure 44, *support peer*s correspond to the session peers that are capable of supporting the channel change process by sending channel change packets to the zapping users.

server. After the CSC server receives the channel change request, it determines the parameters associated with the channel change request (*i.e.*, support peers and the sequence numbers of the packets that will be transmitted by each support peer).

To coordinate the channel change events, CSC server creates a multicast control channel (MCC), through which it delivers the basic information on the κ most popular multicast sessions. [2] Specifically, over MCC, CSC server multicasts the location of the next key frame [3] for each of the κ sessions included in the multicast. Since the number of sessions targeted by each MCC transmission is expected to be small and the key frames are not frequently transmitted (*i.e.*, one frame per group-of-picture (GOP)), control channel overhead at the end-users will also be small. Note that, if the key frame transmission frequencies differ from one session to another, then the CSC server can aggregate multiple MCC update messages in such a way that the end-users can be guaranteed to receive up-to-date information on each of the κ channels T_J time units before the respective delivery times of the key frames. Here the parameter T_J refers to the IGMP signaling latency perceived during a channel change event.

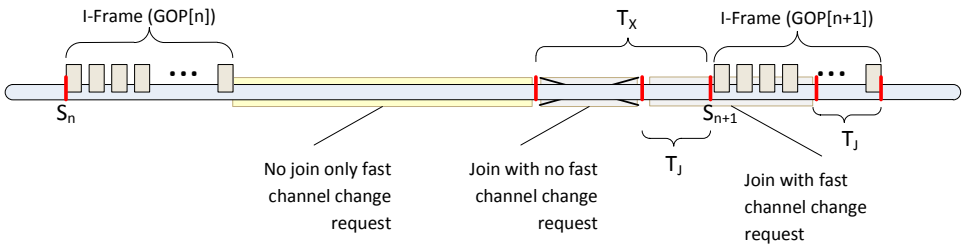


Figure 45: Timing for the session join and channel change operations.

We next summarize the channel change operation for the given framework, from the point of view of an IPTV client, referred to as ν. Assume that ν decides to make a channel switch from session s_C to session s_N. [4] We illustrate the timing process

[2] CSC server acquires this information, as well as information on the framing structure used for the subsequent group of pictures (GOPs), from the head-end server, through head-end server's periodic multicasts to all the active CSC servers.

[3] Hereafter, we will use the terms key-frame and I-frame interchangeably.

[4] The subscripts C and N refer to *Current Channel* and *Next Channel*.

for the channel change operation in Figure 45. Channel change process initiates as ν transmits an IGMP leave message to the Multicast Proxy Server. If the channel change request is made within the interval $J_{I,n+1}$ [5], then ν also sends an IGMP join message to start receiving packets from s_N's source multicast. Furthermore, if the IGMP join message is sent during an interval of $[S_{n+1} - T_J, S_{n+1} + T_I - T_J])$, then ν also sends a channel change update message to the CSC server. If additional channel change packets are required by ν to finalize its channel change process, then the CSC server delivers the requested packets directly to ν, without forwarding the request to the potential support peers within s_N.

If the expected delay to receive the next I-frame from s_N's source multicast is outside the interval of $J_{I,n+1}$, then ν postpones the transmission of its IGMP join message. There are two reasons for this decision. The first reason is to reduce the downlink bandwidth utilization at ν, by limiting the duration for simultaneously receiving the channel change packets and the source multicast. The second reason is to speed up the channel change process. That is because if ν cannot receive the I-frame packets from s_N's source multicast, then joining the multicast session of s_N may significantly reduce the available bandwidth at ν that will be used for the delivery of I-frame packets through the CSC server. Additionally, as ν synchronizes through the CSC server, it can also opt out from receiving the lower-priority packets, such as the B-frame packets, without perceiving noticeable quality degradations [95]. The only exception to the above scenario is when ν can partially receive the I-frame packets from s_N's source multicast. If that is the case, then ν sends the IGMP join message immediately to limit the number of packets delivered through the CSC server. Otherwise, ν waits for an update message from the CSC server with information on

[5] In Figure 45, $J_{I,n+1}$ corresponds to the timeframe of $[S_{n+1} - T_X, S_{n+1} + T_I - T_J]$, where T_I represents the transmission duration for an I-frame, T_X represents the minimum acceptable latency required to successfully initiate the fast channel change process, and S_{n+1} represents the start of delivery time for the next GOP sequence, which has a sequence number of $n + 1$.

when to join s_N's source multicast.

At the CSC server, after the channel change request is received, the first thing that the server checks is whether or not the targeted session utilizes support peers. For that purpose, we use a peer support threshold for each session. Specifically, if the number of users connected to s_N is below the peer support threshold, then channel change packets can only be delivered through the CSC server, *i.e.*, session peers are not probed for any received channel change request targeting s_N. If that is the case, then the CSC server also needs to make a decision on whether or not to accept the request. For instance, if the immediate load at the CSC server is close to a critical level (because of servicing the previously received and admitted channel change requests), then the CSC server rejects ν's request and sends an update message to ν so that the user can join s_N's multicast at the start of s_N's next GOP sequence.

On the other hand, if the received request can be served with the help of the support peers, then the proposed framework considers the use of three possible approaches to help proceed with the channel change process. We can summarize the proposed approaches as follows:

- *Approach* 1 : The first approach uses only the session peers to deliver the channel change packets.

- *Approach* 2 : The second approach uses both the CSC server and the session peers to deliver the channel change packets. Here, CSC server is only used to ensure that the channel change process proceeds in a timely manner. For that purpose, CSC server is only allowed to send channel change packets, until the packets transmitted by the support peers arrive to the access point of the zapping user. In short, the second approach takes advantage of the fast response time through the CSC server by sending an initial burst of channel change packets to the zapping user.

- *Approach* 3 : The third approach uses only the CSC server to deliver the channel change packets.

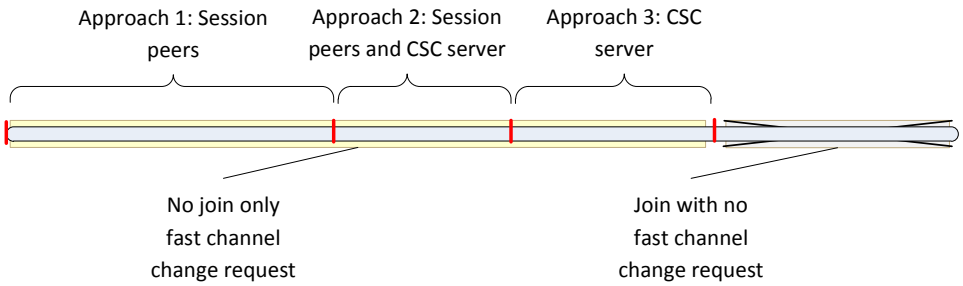


Figure 46: Timing for the three channel change approaches.

We illustrate the timing-based relationship for the considered synchronization approaches in Figure 46. The decision on which synchronization technique to use for any received request depends mainly on the arrival time for the request, expected servicing time, the level of support offered by the session peers and the CSC server, and the GOP duration. Our main objective here is to distribute the channel change overhead to the session peers as much as possible. Hence, as long as a channel change request can be served within the *synchronization latency deadline* [6], the approach that requires the least amount of resources at the CSC server (which suggests that the selected approach is expected to meet the latency deadline the latest) will be selected by the CSC server.

Consequently, by utilizing the CSC server during the channel change process, we can minimize the negative impact of the peer-delivery latency and increase the acceptance rate of requests that can utilize the fast channel change process. When the CSC server decides to proceed with the fast channel change process using mainly the support peers, it uses a fair selection policy to assign the peers to each received

[6] *Synchronization latency deadline* represents the latest point in time for a fast channel change process to stay advantageous over waiting for the next key frame from the session multicast. This point is initially set to the expected start of delivery time for the next key frame through the session multicast.

request. In addition to selecting the peers for the channel change process, CSC server is also responsible for assigning each selected peer with the sequence number of the packets each will be transmitting during the synchronization phase. After the CSC server makes its decision on the channel change parameters, if needed (which is the case for approaches *Approach* 1 and *Approach* 2), it multicasts this information to the session peers to enable the timely delivery of the requested channel change packets. [7].

Finally, to improve the reliability of the channel change process, CSC server is required to transmit a certain number of proactive Application-layer FEC (AL-FEC) packets for each of the received and accepted channel change request. The rate of AL-FEC protection depends on the approach being selected and the average packet loss rate observed on each end-to-end connection. In doing so, even if ν fails to receive some of the channel change packets from the session peers, proactively received AL-FEC packets from the CSC server will allow the user to recover from such losses with no extra delay.

7.3 Analytical Framework

In this section, we present the methodology used to analyze the proposed channel change framework. For the analysis, we focus on the channel change operations at a single CSC server. We assume steady state conditions for the given system, *i.e.*, $N(t) = N$, where $N(t)$ represents the number of IPTV clients connected to the CSC server at time t. The number of IPTV sessions is given by M. To distribute the users to each of the M available sessions, we use the Zipf distribution [81]:

$$\pi_j = \frac{1/j^s}{\sum_{1 \leq m \leq M} 1/m^s} \tag{127}$$

[7]Note that, each user is assigned a unique ID by the CSC server after each distinct join event. In doing so, the amount of information to be carried within a multicast request message can be significantly reduced.

where π_j is the probability of an arbitrary user being connected to the jth most popular multicast session and s represents the shape parameter for the given Zipf distribution. [8]

7.3.1 Modeling the Channel Switch Rates

To analyze the performance of a given IPTV network, we first need a model to approximate the channel switching rates in the network while satisfying the system constraints (*e.g.*, steady state conditions). We use the following approach to approximate the channel switching rates for each session.

The steady state assumption dictates that the probability of switching from s_j to s_i is given by $\pi_i/(1-\pi_j)$. [9] We assume that the arrival process for the channel change events follows the Poisson distribution, and we use λ_j to represent the channel switching rate to s_j. Then, we can determine the entry rate for s_j using the following equation:

$$n_j^+ = \sum_{\forall m \neq j} \frac{N_m \times \lambda_m \times \pi_j}{1-\pi_m} \tag{128}$$

where n_j^+ represents the request rate for the channel change events targeting s_j, and N_j represents the number of users connected to s_j, *i.e.*, $N_j = N \times \pi_j$.

Similarly, we can determine the exit rate from s_j to another session as follows:

$$n_j^- = N_j \times \lambda_j \tag{129}$$

where n_j^- represents the departure rate from s_j.

The steady state assumption allows us to consider the entry and exit rates for each session to be equal, *i.e.*, $n_j^+ = n_j^-$, $\forall s_j$. The reason for that is because, on the average, we expect the size of each session to stay the same throughout the observation period. In doing so, we can preserve the state of each active session as

[8] Here, session ID also represents the popularity rank of a given channel.

[9] Here, the index used for the session ID refers to the popularity rank of the given session, *i.e.*, s_j represents the jth most popular IPTV session.

they were initially determined by the selected Zipf distribution. Consequently, we can use the following equation to represent the relationship between two different sessions's channel switching rates:

$$\lambda_j/(1-\pi_j) = \lambda_i/(1-\pi_i) \tag{130}$$

If the accumulative channel switching rate for sessions connected to the same CSC server is known, we can use its value to determine the channel switching rates for each session:

$$\lambda_j = \frac{\lambda_T \times (1-\pi_j)}{N \times (1-\sum_{\forall i} \pi_i^2)} \tag{131}$$

where λ_T represents the total channel switching rate for all the sessions connected to a given CSC server. .

Figure 47 illustrates the dependence of the channel switching rate on the number of active sessions (M), when $N = 1000$ and $\lambda_T = 10$. [10] The approach presented here to characterize the channel switching events allows us to achieve minimal change in entry rates under different M values.

7.3.2 Resource Allocation at the Session Peers

Since session peers can offer varying levels of support for the channel change process, we need to define the minimum level of peer support required for a session peer to actively participate in the channel change process as a support peer.

We use the parameter $\omega_{u,\min}$ to represent the minimum level of peer support required for the channel change process, and calculate its value as follows:

$$\omega_{u,\min} = \frac{l_P}{\tau_S} \tag{132}$$

where l_P represents the packet size and τ_S represents the upperbound on the packet transmission time at the session peers. Here, $\omega_{u,\min}$ is essentially used to find the

[10] The given values suggest a total of 36 zapping events per hour per user.

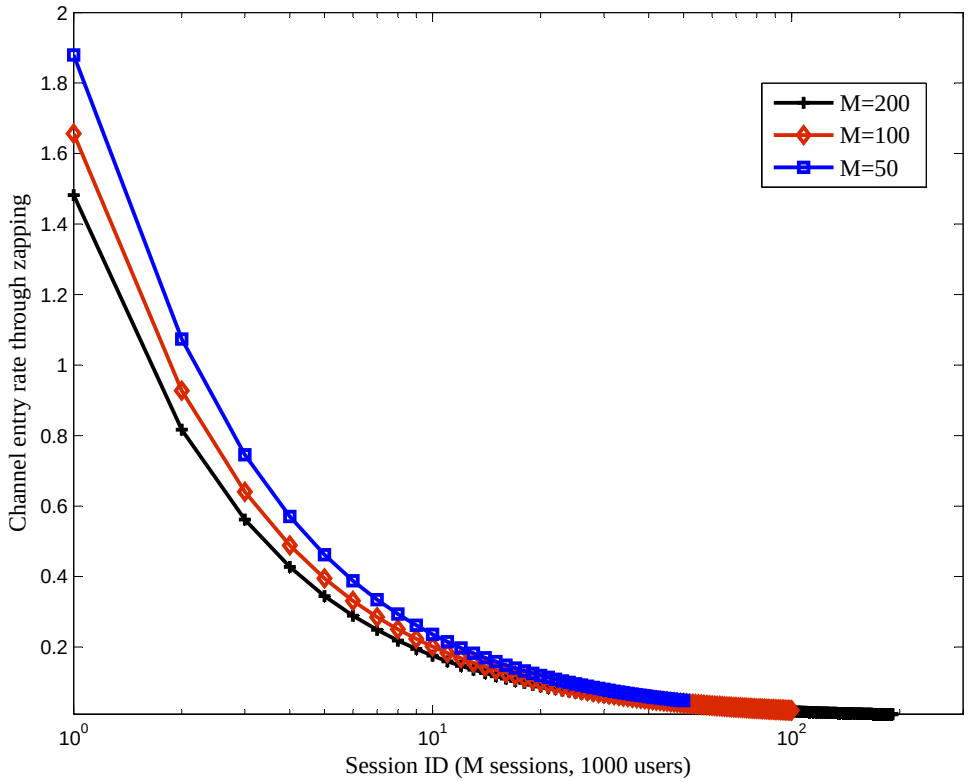


Figure 47: Channel switching rate for each active session when $\lambda_T = 10$.

instantaneous channel-change support offered by the session peers. [11] By varying the value of τ_S, we can vary the number of connections available at each session peer for the channel change process.

Next, we need to find the total peer support available for each active session's channel change process. To do that, we need to find the individual support offered by each session peer using the following equation:

$$\varphi_\nu = \frac{W_u^{(\nu)}}{\omega_{u,\min}} \tag{133}$$

where $W_u^{(\nu)}$ represents the uplink bandwidth availability at ν. [12]

[11] Instantaneous support refers to the number of simultaneous connections available at a given session peer, at any given point in time.

[12] Since the available bandwidth varies in time, φ_ν is essentially a time-dependent parameter. In this study, we assume the value of $W_u^{(\nu)}$ to stay the same during a given observation period.

To find the total peer support for session s_j, we use the following equation:

$$\varphi(j) = \sum_{\forall \nu \in s_j} \lfloor \varphi_\nu \rfloor \tag{134}$$

where $\varphi(j)$ represents the maximum (or idle-state) peer support available for s_j.

Note that the instantaneous peer support depends on the previously received active requests, hence, its value varies in time. We use the parameter $\Phi_j(t)$ to represent the instantaneous peer support offered by s_j at time t. [13] Therefore, any channel change request that is received at time t for s_j and requires more than $\Phi_j(t)$ packet transmissions will require support from the CSC server. The level of support offered by the CSC server is expected to change in time depending on the service completion time of the past requests and the arrival time of the future requests. However, to avoid the additional overhead of going through the session peers after each update, initially selected channel change parameters will be preserved during the synchronization phase.

To find the parameters for the channel change process (*i.e.*, available session peers, delivery rate, and the transmission order), we start by determining, at each session peer, for each incoming request, the amount of unreserved resources during the next τ_S period. Let us initially assume that, at support peer ν, there are μ_ν active requests, each of which has a completion time of $e_{\nu,l}$, where $l \leq \mu_\nu$. Assume that the last request is expected to be received by ν at time $T(\mu_\nu + 1)$. We can then calculate the unreserved resources that the latest received request can utilize at ν as follows:

$$\varphi_\nu(T_{\mu_\nu+1}) = \varphi_\nu - \sum_{l=1}^{\mu_\nu} \frac{e_{\nu,l} - T(\mu_\nu + 1)}{\tau_S} \tag{135}$$

[13] Note that t normally represents the delivery time of the request to the session peers. However, since the decision for selecting the session peers is made at the CSC server, to keep the convention the same for all the session peers, we use t to represent the arrival time of the request to the CSC server.

Consequently, we can use the following equation to find the instantaneous peer-based resource availability for s_k at t:

$$\Phi_k(t) = \sum_{\forall \nu \in s_k} \varphi_\nu(t) \tag{136}$$

After the value of $\Phi_k(t)$ is determined, the requested resources are then distributed to the available support peers using a weighted round robin based approach. Specifically, for each active session with available peer support, CSC server creates a predesignated assignment order (*e.g.*, using peer IDs, which are updated after each join/leave event), which is then combined with the selected weights to allocate the resources. The selected weights represent the idle resource availability at each support peer. Also note that, if $\Phi_k(t)$ is less than the minimum required resources for the received channel change request, then the portion of the requested resources that cannot be delivered by the support peers will be provided by the CSC server.

7.3.3 Approximating the Peer-Support Threshold

For the peer-assisted channel change process to be effective, the number of clients connected to a given session needs to be higher than a certain threshold. That is because, if the synchronization packets are requested from a small set of support peers, then the uplink bandwidth limitations at these users may cause a significant increase in the synchronization latency. As a result of these limitations, CSC server may be asked to deliver more packets than required, had the server been selected as the prior choice for the delivery of the synchronization packets. Since support peers are essentially utilized to reduce the instantaneous load at the CSC server, using support peers for sessions that do not meet the peer threshold criteria is not a preferable approach.

To find the optimal threshold, we can make use of two different approaches. The first approach focuses on long-term resource utilization at the CSC server. For that purpose, we reserve a small number of simultaneously active unicast-based streams at

the CSC server to deliver the channel change packets. The second approach, on the other hand, focuses on short-term resource utilization at the CSC server. Specifically, we increase the instantaneous bandwidth availability at the CSC server for the given request to speed up the synchronization process. Compared to the first approach, the second approach may lead to less overhead at the CSC server while also causing smaller latency values. For that reason, our analysis focuses on the second approach.

Assume that the parameter $\Delta_{rqs}^{(s)}$ represents the average request size at the CSC server, when the server is the prior choice for delivering the synchronization packets. Similarly, assume that the parameter $\Delta_{rqs}^{(p)}(k)$ represents the average request size at the CSC server, when a k-sized subset of the available support peers act as the prior choice for delivering the synchronization packets. Then, the minimum number of synchronization streams that are required to satisfy the bandwidth requirements at the CSC server can be found as follows:

$$\Delta_{rqs}^{(p)}(\delta_p) \leq \Delta_{rqs}^{(s)} \leq \Delta_{rqs}^{(p)}(\delta_p + 1) \tag{137}$$

where δ_p represents the resource optimal peer-support threshold for the proposed channel change framework. [14]

To find an accurate estimate for δ_p, we need information on the following parameters: bandwidth availability at the CSC server (W_S) and the session peers (W_U), mean GOP duration for the given session (T_{GOP}), minimum required uplink bandwidth availability at the session peers (*i.e.*, l_p/τ_S), end-to-end delays (uplink or downlink propagation delays, d_u and d_d), and maximum allowed synchronization latency ($T_{L,\max}$). Furthermore, we also need to utilize a recursive update procedure to find the actual delivery time of each channel change packet. However, to limit the additional complexity involved in finding the actual value for the δ_p metric, we simplify

[14]Depending on the uplink bandwidth availability at the session peers, it is possible for some peers to support multiple (or zero) synchronization streams (SSs). Therefore, the case of having δ_p SSs can be considered as having δ_p session peers with each supporting a single SS.

the estimation process by removing the recursive update part. Instead, we assume the synchronization packets to be available at the session peers when they receive the multicast-transmitted request message from the CSC server. [15]

We illustrate the timing for the synchronization phase in Figure 48. To find the peer-threshold, we focus on two distinct intervals, which are shown as $I1$ and $I2$ in Figure 48. First, we find the maximum number of packets that can be delivered to the zapping user during $I2$. We refer to this parameter as n_J^+. Then, we find the minimum number of synchronization streams, which can guarantee that at most n_J^+ number of packets are queued at the zapping user's access point by the end of $I1$.

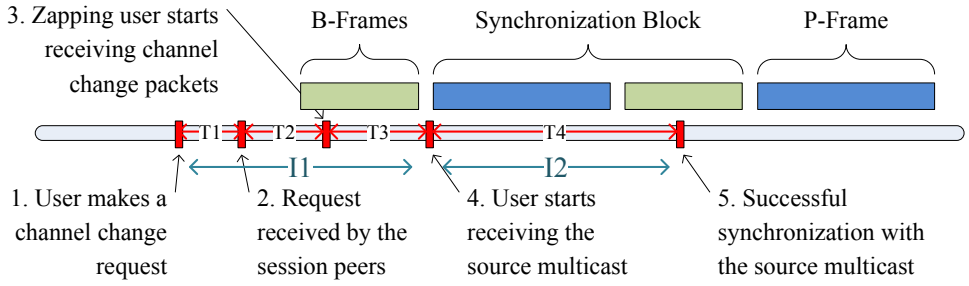


Figure 48: Timing for the synchronization phase.

To find the value of n_J^+, we use the information on the interframe spacing between the two P-frames, which we refer to as $T_{P,P}$ (or $T_{P,I}$ for the last P-frame, which represents the interframe spacing between the last P-frame of the current GOP sequence and the I-frame of the next GOP sequence). The maximum number of packets that can be delivered after the user joins the source multicast is found as follows:

$$n_J^+ = \frac{T_{P,P} \times (W_U - W_M)}{l_p} \tag{138}$$

where W_M represents the delivery rate for the IPTV multicast.

To successfully complete the synchronization phase during the selected synchronization block, the number of packets, which are queued at the zapping user's access

[15] Note that, because of the limited bandwidth availability at the user side and the use of sequential transmissions for the channel change packets, we expect our results to be close to the actual results.

point by the end of $I1$ (which is referred to as $n_{q,J}$), needs to be smaller than or equal to n_J^+. To approximate the value of $n_{q,J}$, we use the following equation:

$$n_{q,J} = n_{req} - \min\{\gamma_u(\tau_S) \times \delta_p, n_{d,\max}\} \tag{139}$$

where $\gamma_u(\tau_S)$ represents the maximum number of packets that can be delivered to the zapping user with a single channel change stream, which transmits data at a rate of τ_S^{-1}, and $n_{d,\max}$ represents the maximum number of packets that can be received over zapping user's downlink connection.

To find the value of $\gamma_u(\tau_S)$, we use the following equation:

$$\gamma_u(\tau_S) = \frac{(T_P^* - T_{req}) - (2d_u + d_d)}{\tau_S} \tag{140}$$

where T_{req} represents the time when the channel change request is made, and T_P^* represents the time to start the delivery of the targeted P-frame (from the source multicast) to the zapping user's access point. For the given equation, we make two assumptions: (i) session peers observe the same uplink and downlink propagation delays, and (ii) end-to-end delay between the two access points is negligible compared to the propagation delay observed along the last mile.

To find the value of $n_{d,\max}$, we use the following equation:

$$n_{d,\max} = \frac{(T_P^* - T_{req}) - (2d_u + d_d + \tau_S)}{l_p/W_M} \tag{141}$$

To approximate the value of T_P^* we use the following equation, which uses the assumption that the jitter for the delivered packets is negligible:

$$T_P^* = T_{S,GOP} + (\eta_I + (\kappa - 1)(\eta_P + \eta_B)) \times l_p/vW_M \tag{142}$$

where $T_{S,GOP}$ represents the transmission start time for the current GOP sequence, κ represents the order of the P-frame (which the zapping user receives first through the source multicast), and the parameters $\{\eta_I, \eta_P, \eta_B\}$ represent the sizes of I-, P-, and B-frames.

Table 3: Approximate Peer Threshold Values

τ_S	50ms	100ms	150ms
$T_{GOP} = 1s$	$\delta_{p,\max} = 3$ $E[\delta_p] = 1.87$	$\delta_{p,\max} = 4$ $E[\delta_p] = 2.58$	$\delta_{p,\max} = 5$ $E[\delta_p] = 3.02$
$T_{GOP} = 2s$	$\delta_{p,\max} = 5$ $E[\delta_p] = 1.92$	$\delta_{p,\max} = 5$ $E[\delta_p] = 2.63$	$\delta_{p,\max} = 5$ $E[\delta_p] = 3.06$

Consequently, the solution to the following inequality is used to find the peer-support threshold:

$$\min(\gamma_u(\tau_S) \times \delta_p, n_{d,\max}) \geq \eta_I + \eta_P \times (\kappa - 1) - n_J^+ \tag{143}$$

which leads to the following equation for δ_p: [16]:

$$\delta_p = \frac{\eta_I + \eta_p \times (\kappa - 1) - n_{join}^+}{\gamma_u(\tau_S)} \tag{144}$$

Table 3 shows the minimum required peer-threshold values (*i.e.*, worst-case and mean values), when the following parameters are used for the delivery of the source and channel change packets: $W_M = 3Mbps$, $W_U = 4.5Mbps$, $T_J = 100ms$, $d_u = d_d = 20ms$, and $0s$ delay between access points and the CSC server. In our simulations, we used two different GOP durations ($1s$ and $2s$) and three different τ_S values ($50ms$, $100ms$, and $150ms$). For the considered set of system parameters, optimal results can be attained when the peer-threshold value is selected from $\{2, 4\}$. Also note that, changing the GOP duration only affects the worst-case threshold values, *i.e.*, the mean peer-threshold values, which are obtained by taking the average of the minimum required peer-threshold values over the duration of a single GOP, stayed almost the same.

7.4 Performance Analysis

In this section we present the simulation results for the proposed channel change framework. For our simulations, we used the CSIM discrete event simulation software.

[16]Note that, if $\delta_p > \delta_{p,\max} = n_{d,\max}/\gamma_u$, then the synchronization phase is considered to be unsuccessful.

Table 4: Simulation Parameters

Total number of users, N	1000
Total number of sessions, M	100
Exponent for the Zipf distribution, s	1
Delivery rate for the IPTV multicast, W_M	3Mbps
Downlink bandwidth availability at the users, W_U	$[3.6, 5.4]$ Mbps
Uplink delay for the access network, d_u	20ms
Downlink delay for the access network, d_d	20ms
Group-of-picture (GOP) duration, T_{GOP}	$\{1s, 2s\}$
Session join delay, T_J	100ms
Transmission delay for the synchronization phase, τ_S	50ms
Packet length for the MPEG-2 Transport Stream, l_P	1356Bytes
Ratio of I-frame length to P-frame length, ρ_I/ρ_P	3
Ratio of P-frame length to B-frame length, ρ_P/ρ_B	2.5
Peer threshold, δ_p	2

The parameters that are used in our simulations are shown in Table 4. We used $30fps$ as the frame generation rate. The frames within a GOP are sequenced as $IBBPBB\cdots$, which also assumed a constant ratio among the I-, P-, and B-frame sizes. We performed 10 simulation runs each time starting with a different random seed. Each simulation run corresponds to an 8-hour long IPTV broadcast. The reported results represent the average of the results from each of these simulation runs.

In our simulations, to measure the perceived channel change latency, we mostly focused on the synchronization latency and set the boundary conditions (*i.e.*, deadline for accepting a request) according to its value. Figure 49 illustrates how the synchronization latency values relate to the display latency values, when $W_U = 4.5Mbps$. On the average, we observed a difference of 250ms between the two values. Therefore, to achieve the desired display latency performance, we can allocate the resources assuming a less strict limit on the synchronization time-frame.

Figure 50 shows the cumulative distribution function corresponding to the synchronization latency values for the $T_{GOP} = 1s$ scenario. Similarly, Figure 51 shows

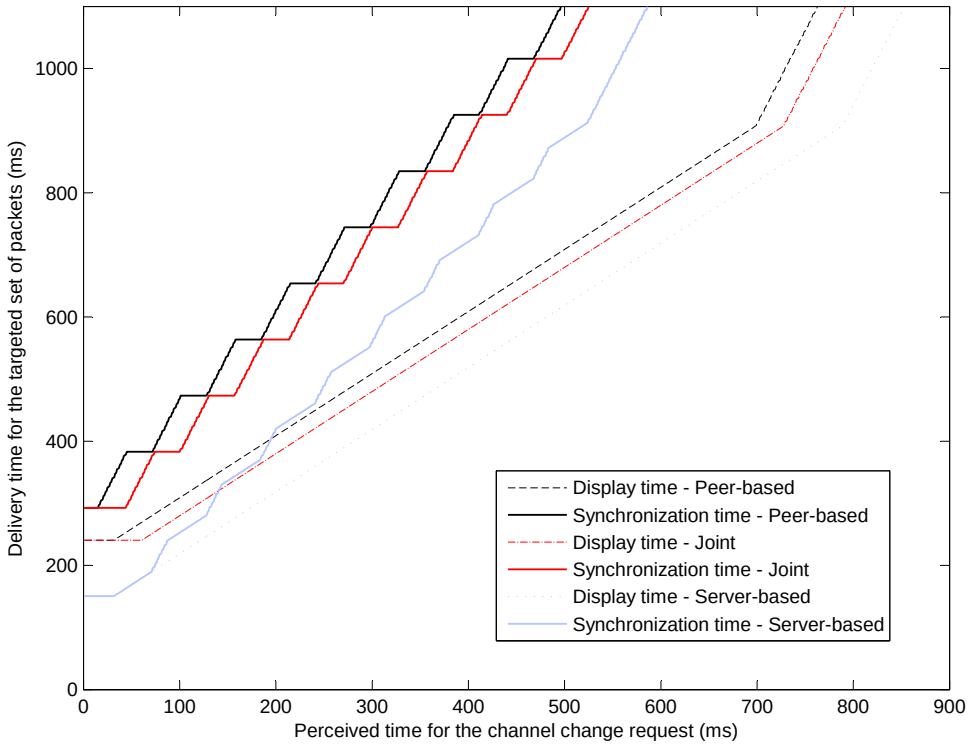


Figure 49: Comparison between the stream synchronization time and the user-perceived content display time.

the same set of results for the $T_{GOP} = 2s$ scenario. For each of these scenarios, we varied the downlink bandwidth availability at the client side. We then compared their results to that of the $Wait$ approach, which required the users to wait for the next I-frame transmission from the targeted session's source multicast. When compared to the $Wait$ approach, the proposed channel change framework achieved 40%-to-60% performance improvements in the synchronization latency. Table 5 also presents the results corresponding to the average display latency for the proposed channel change framework. Notice that, even at presumably low user bandwidth availability, we were able to keep the average display latency below $500ms$. Also note that the rate of increase in the perceived latency values is much smaller than the rate of increase in the GOP duration, showing the efficiency of utilizing a distributed framework.

In Figure 52, we show the results corresponding to the channel change overhead

Table 5: Perceived Display Latency Values

W_U/W_M	1.2	1.5	1.8
$T_{GOP} = 1s$	$329ms$	$270ms$	$238ms$
$T_{GOP} = 2s$	$451ms$	$354ms$	$297ms$

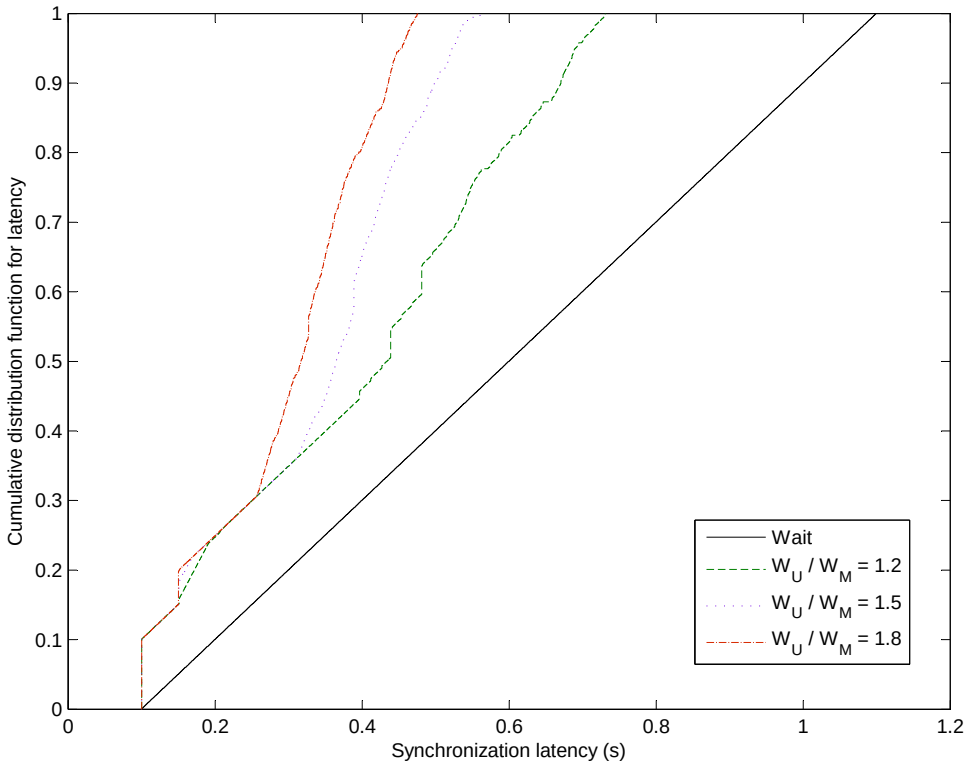


Figure 50: Cumulative distribution function for the synchronization time, when GOP duration equals 1s.

at the session peers and the CSC server, as we vary the channel change rates. For the simulations, we assumed that the session peers utilize a single channel change stream to deliver the requested packets. At the session peers, we observed an average uplink bandwidth utilization that is less than $80Kbps$, which is considered as an acceptable overhead for the end users. As the zapping rate reached 100, uplink bandwidth utilization at each session peer reached its limit. That is to say, as the zapping rate is further increased, all the received requests were mostly serviced by the CSC server. As a result, bandwidth requirements at the CSC server increased at a rate faster than the rate of increase for the channel change rates.

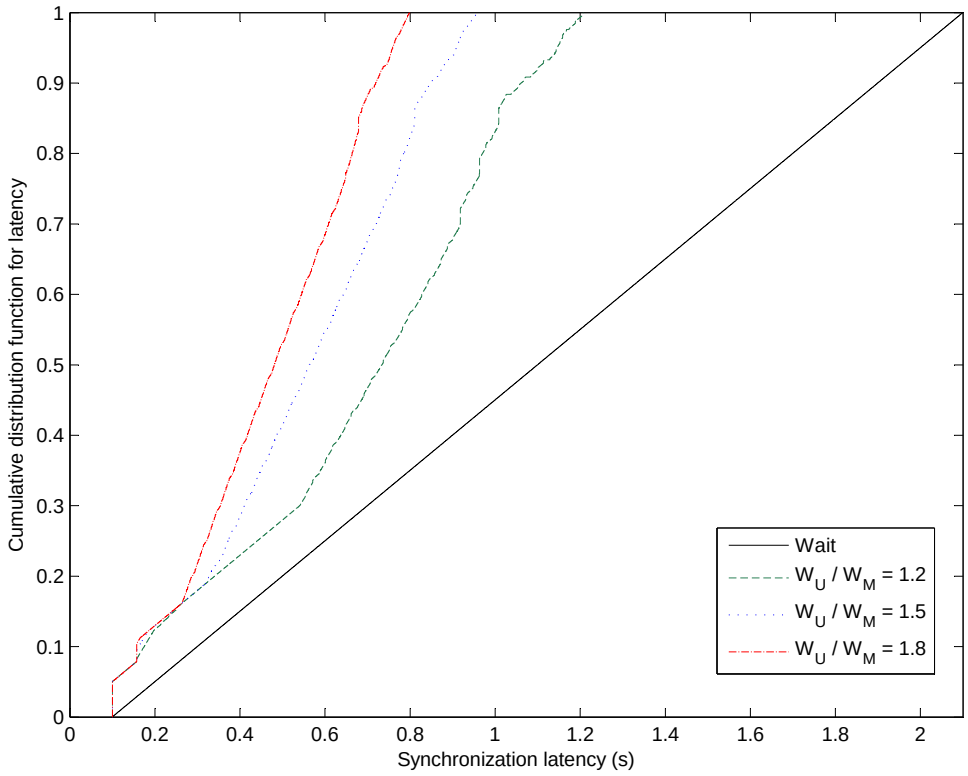


Figure 51: Cumulative distribution function for the synchronization time, when GOP duration equals 2s.

Note that, if the CSC server would have a downlink capacity of $100Mbps$, then, for the selected system parameters, the proposed framework reaches a critical state when the zapping rate reaches 250. If that happens, to stabilize the system, we need to increase the number of available streams at the session peers. We can also implement a dynamic request admission policy at the CSC server to maximize the acceptance rate for the received requests without significantly affecting the average latency performance.

In Figure 52 we also compare the overhead performance of the proposed framework to a server-based-only channel change policy, which requires the zapping user to immediately send an IGMP join message for the targeted session's multicast. [17] For

[17]Note that by immediately sending the IGMP join message, we can minimize the overhead at the channel change server.

the given policy, if the earliest display time is expected to be higher than the earliest display time for the *Wait* approach, then the channel change request is not accepted. Results shown in Figure 52 validate our expectations. Compared to a server-based-only channel change policy, distributing the channel change overhead to the session peers significantly improved the servicing capacity of the network.

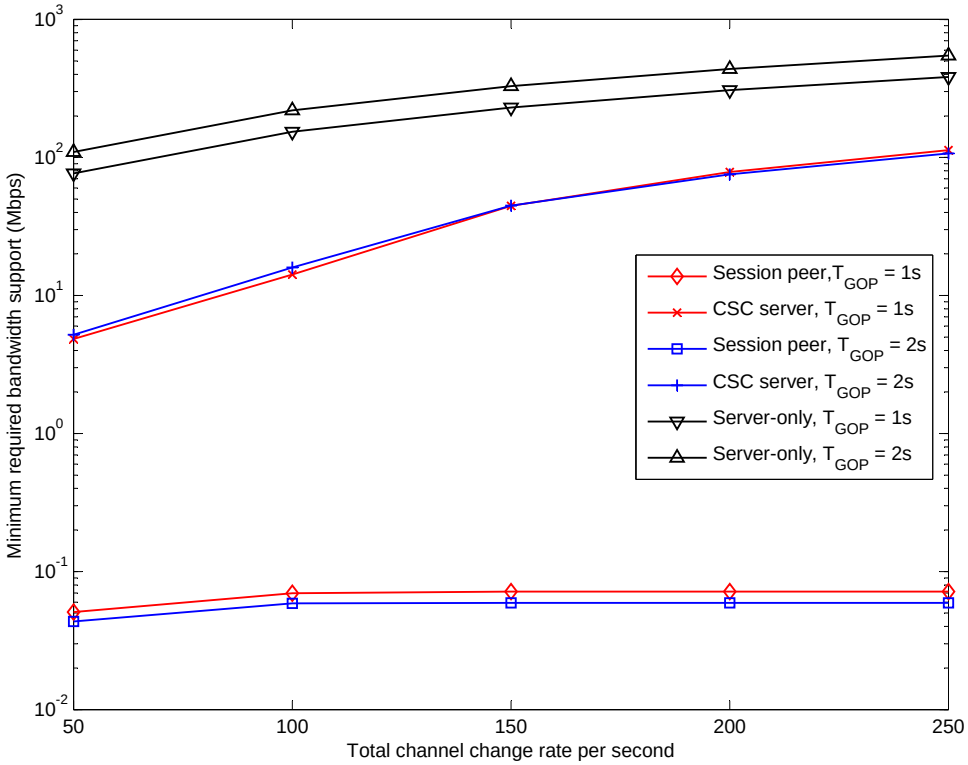


Figure 52: Channel change overhead at the session peers and the CSC server.

Finally, we also observed that increasing the GOP duration had a limited impact on the channel change overhead. The proposed framework allowed more efficient use of the waiting period to deliver the channel change packets. Also, since the ratio of the peer-serviceable portion of a GOP sequence increases as the duration of the GOP increases, session peers can more effectively contribute to servicing the received requests, leading to a reduced bandwidth demand at the CSC server.

7.5 Conclusions

In this chapter, we proposed a semi-distributed channel change framework for the IPTV networks. The proposed approach combines the capabilities of a dedicated server, referred to as the Channel Switch Coordinator (CSC), with that of the session peers to create a resource efficient, scalable, and reliable channel change framework for the IPTV networks. By making use of the uplink bandwidth availability at the session peers, we distributed the channel change overhead to a large number of session peers. In doing so, we were able to utilize the CSC server more efficiently. As a result, we were able to significantly speed up the channel change process for the IPTV sessions. For all the considered scenarios, the proposed framework was able to keep the channel change latency below the critical latency threshold of 500 milliseconds. Even as we increased the GOP duration to 2 seconds, the average latency continued to stay within the acceptable boundaries. Finally, when compared to a pure centralized channel change policy, we observed significant improvements in terms of bandwidth utilization and system scalability.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1 Thesis Contributions

In this thesis, we addressed the challenges in delivering broadcast quality content over IPTV networks. Our thesis focused on two specific problems in regards to achieving the desired quality of experience levels in IPTV networks, namely the error recovery problem and the channel change latency problem.

The first part of the thesis focused on the error recovery problem in IPTV networks. We started by proposing a generalized error recovery framework, first, to investigate the tradeoffs associated with the delivery of proactive and reactive repair packets in IPTV networks, and second, to develop a practical error recovery protocol to achieve resource-optimal recovery in IPTV networks. We next focused on the impact of correlated packet losses in IPTV error recovery. For that purpose, we first developed three group loss correlation models that are applicable to IPTV networks. We then developed a server-assisted error recovery framework to exploit the clients' correlated losses and used the earlier developed loss correlation models to validate the performance improvements in IPTV error recovery. Next, to minimize the impact of scalability problems that arise with the use of server-based error recovery frameworks, we proposed a distributed error recovery framework that utilized the support of session peers during the error recovery phase. The results suggested remarkable improvements in the scalability performance without introducing significant overhead at the session peers. We lastly focused on the delivery of IPTV content over wide-area wireless access networks, and especially over the WiMAX networks, and addressed the reliable delivery concerns over these networks. For that purpose,

we proposed a cooperative-diversity driven delivery framework to support quick and resource-efficient error recovery in WiMAX-based IPTV networks. The proposed approach is shown to achieve great results in error recovery with limited use of additional energy resources.

The second part of the thesis focused on the channel change latency problem in IPTV networks. We started our study by focusing on the analysis of generalized client-based solutions that utilize concurrently delivered support streams to reduce the channel change latency. Client-based solution framework is chosen initially due to the practical advantages it presents, such as higher accessibility rate for the clients and a wider adoption possibility. We presented a realistic theoretical framework to analyze in detail the performance of channel change protocols that relies on the use of concurrently delivered support streams. The results suggested significant limitations on the use of client-based solutions due to the overhead they introduce near the access network, and the insufficient gains in the channel change latency performance to validate the additional overhead. Combining these results with the results from the first part of our thesis, we focused on the design of a semi-distributed channel change framework that is capable of supporting the channel change process using session peers whenever possible and/or needed. The results suggested remarkable improvements in the latency performance without introducing significant overhead in the network, while satisfying the requirements for a scalable implementation.

8.2 Future Research Directions

In this thesis, we focused on the error recovery and the channel change latency problems separately. Even though it is possible to implement these approaches independently, in some scenarios (*i.e.*, depending on the location of processing/support), achieving the optimal performance may necessitate jointly evaluating the performance

of these frameworks. For instance, if we use the session peers to simultaneously support the error recovery process and the channel change process, then the reciprocal impact of each of these approaches needs to be carefully investigated. In doing so, we can allocate the resources more efficiently and without overloading certain nodes in the network.

Additionally, for the IPTV channel change, we strictly limited our study and discussions on client-based and/or peer-assisted solutions. However, since IPTV is considered as a service provider-driven technology, in some scenarios, it may be desirable (or required) to implement the channel change procedure strictly at the network infrastructure through the help of dedicated servers. Therefore, we need to also investigate scalable server-driven solutions to achieve that objective. Current server-driven studies typically focus on developing either multicast-based techniques or unicast-based techniques. However, to optimize the resource usage efficiency in the network, any server-driven framework should be capable of simultaneously supporting and utilizing both of these approaches. For that purpose, we need to evaluate the performance tradeoffs associated with the use of these approaches at any given point in time, and based on the attained results, we need to decide on a schedule to select the approach that achieves the optimal resource utilization, and allocate the resources accordingly to each of these approaches.

REFERENCES

[1] "Digital video broadcasting (DVB)– framing structure, channel coding, and modulation for cable systems," ETSI EN 300 429 v1.2.1 (1998-04).

[2] "Information technology – generic coding of moving pictures and associated audio – part 1: Systems," ISO/IEC 13818-1, ITU-T Rec H.222.0:1996.

[3] "MPLS traffic engineering fast reroute: Link protection," Cisco Systems.

[4] "A proposed media delivery index (MDI)," RFC 4445.

[5] "RTP: A transport protocol for real-time applications," RFC 3550.

[6] "Pro-MPEG Code of Practice 3 release 2," Pro-MPEG forum, 2004.

[7] "Assuring quality of experience for IPTV," *Alcatel Heavy Reading White Paper*, 2006.

[8] "ATIS IPTV packet loss issue report," ATIS-0800005, Dec 2006.

[9] "Triple-play services quality of experience (QoE) requirements," tech. rep., DSL Forum, 2006.

[10] "DVB application layer FEC evaluations," DVB Bluebook A117, 2007.

[11] "Fast channel changing in rtp," tech. rep., Internet Streaming Media Alliance, Dec 2007.

[12] Alouini, M. and Goldsmith, A., "Capacity of rayleigh fading channels under different adaptive transmission and diversity-combining techniques," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 44, pp. 1165–1181, 1999.

[13] Asghar, J., Hood, I., and Faucheur, F. L., "Preserving video quality in IPTV networks," *IEEE Transactions on Broadcasting*, vol. 55, pp. 386–395, Jun 2009.

[14] Azarian, K., El Gamal, H., and Schniter, P., "On the achievable diversity-multiplexing tradeoff in half-duplex cooperative channels," *Information Theory, IEEE Transactions on*, vol. 51, pp. 4152 –4172, dec. 2005.

[15] Azgin, A., Altunbasak, Y., and Al-Regib, G., "Cooperative mac and routing protocols for wireless ad hoc networks," in *IEEE Global Communications Conference, GLOBECOM'05*, 2005.

[16] Banodkar, D., Ramakrishnan, K., Kalyanaraman, S., Gerber, A., and Spatscheck, O., "Multicast instant channel change in IPTV systems," in *IEEE International Conference on Systems Software and Middleware*, 2008.

[17] Basso, A., Dalgic, I., Tobagi, F., and Lambrecht, C., "Study of MPEG-2 coding performance based on perceptual quality metric," in *Picture Coding Symposium (PCS)*, 1996.

[18] Begen, A. C., "RTP payload format for 1-D interleaved parity FEC," Internet-draft on the FEC framework.

[19] Begen, A. C., "Error control for IPTV over xDSL networks," in *5th IEEE Consumer Communications and Networking Conference*, pp. 632 –637, 2008.

[20] Begen, A. C., Glazebrook, N., and Steeg, W. V., "A unified approach for repairing packet loss and accelerating channel changes in multicast IPTV," in *6th IEEE Consumer Communications and Networking Conference*, pp. 1–6, 2009.

[21] Bejerano, Y. and Koppol, P., "Improving zap response time for IPTV," in *INFOCOM 2009, IEEE*, pp. 1971 –1979, april 2009.

[22] Bletsas, A., Khisti, A., Reed, D., and Lippman, A., "A simple cooperative diversity method based on network path selection," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 659 – 672, march 2006.

[23] Boyce, J. M. and Tourapis, A. M., "Fast efficient channel change [set-top box applications]," in *IEEE International Conference on Consumer Electronics*, 2005.

[24] Boyer, J., Falconer, D., and Yanikomeroglu, H., "Multihop diversity in wireless relaying channels," *Communications, IEEE Transactions on*, vol. 52, pp. 1820 – 1830, oct. 2004.

[25] Cho, C., Han, I., Jun, Y., and Lee, H., "Improvement of channel zapping time in IPTV services using the adjacent groups join-leave method," in *IEEE International Conference on Advanced Communication Technology '04*, 2004.

[26] Choi, S. G., Park, H.-J., Lee, J. M., and Choi, J. K., "Adaptive hybrid transmission mechanism for on-demand mobile IPTV over WiMAX," *IEEE Transactions on Boradcasting*, vol. 55, no. 2, pp. 468 –477, 2009.

[27] Cui, S., Goldsmith, A., and Bahai, A., "Energy-efficiency of mimo and cooperative mimo techniques in sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 22, pp. 1089 – 1098, aug. 2004.

[28] Deb, S., Jaiswal, S., and Nagaraj, K., "Real-time video multicast in WiMAX networks," in *The 27th International Conference on Computer Communications, INFOCOM'08*, pp. 1579 –1587, 2008.

[29] Degrande, N., de Vleeschauwer, D., and Laevens, K., "Protecting IPTV against packet loss: Techniques and trade-offs," *Bell Labs Technical Journal*, vol. 13, no. 1, pp. 35–51, 2008.

[30] Degrande, N., Laevens, K., Vleeschauwer, D. D., and Sharpe, R., "Increasing the user perceived quality for IPTV services," *IEEE Communications Magazine*, pp. 94–100, Feb 2008.

[31] Doverspike, R., Guangzhi, L., and Oikonomou, K. N., "Designing a reliable IPTV network," *IEEE Internet Computing*, vol. 13, no. 3, pp. 15–22, 2009.

[32] Ebert, J.-P. and Willig, A., "A Gilbert-Elliot bit error model and the efficient use in packet level simulation," *TKN Technical Report, TKN-99-002*, 1999.

[33] Elliott, E. ., "Estimates of error rates for codes on burst-noise channels," *Bell System Technical Journal*, vol. 42, pp. 1977–1997, 1963.

[34] Fisher, W. and Meier-Hellstern, K. S., "The Markov-modulated Poisson process (MMPP) cookbook," *Performance Evaluation*, vol. 18, pp. 149–171, 1992.

[35] Fuchs, H. and Farber, N., "Optimizing channel change time in IPTV applications," in *Proc. IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pp. 1–8, March 31 2008–April 2 2008.

[36] Gilbert, E. N., "Capacity of burst-noise channels," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, 1960.

[37] Greengrass, J., Evans, J., and Begen, A. C., "Not all packets are equal, Part 2," *IEEE Internet Computing*, pp. 74–82, 2009.

[38] Hasslinger, G. and Hohlfeld, O., "The Gilbert-Elliott model for packet loss in real time services on the Internet," in *Proc. of the 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB)*, pp. 269–283, March 2008.

[39] Heffes, H. and Lucantoni, D. M., "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *IEEE Journal on Selected Areas in Communications*, vol. 4, no. 6, pp. 856–868, 1986.

[40] Henkel, W. and Kessler, T., "An impulse-noise model - a proposal for SDSL," ETSI WG TM6 TD45, 1999.

[41] Heyman, D. P. and Lucantoni, D., "Modeling multiple IP traffic streams with rate limits," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 948–958, 2003.

[42] Himsoon, T., Siriwongpairat, W. P., Han, Z., and Liu, K. J. R., "Lifetime maximization via cooperative nodes and relay deployment in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 306–217, Feb 2007.

[43] Hou, F., Cai, L. X., She, J., Ho, P.-H., Shen, X., and Zhang, J., "Cooperative multicast scheduling scheme for IPTV service over IEEE 802.16 networks," in *IEEE International Conference on Communications, ICC'08*, 2008.

[44] Hou, F., Cai, L., Ho, P.-H., Shen, X., and Zhang, J., "A cooperative multicast scheduling scheme for multimedia services in IEEE 802.16 networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1508–1518, 2009.

[45] Jennehag, U., Dohla, S., Fuchs, H., and Thoma, H., "Gradual tune-in pictures for fast channel change," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pp. 766 –770, jan. 2011.

[46] Jennehag, U. and Pettersson, S., "On synchronization frames for channel switching in a GOP-based IPTV environment," in *IEEE Consumer Communications and Networking Conference '08*, 2008.

[47] Jennehag, U. and Zhang, T., "Increasing bandwidth utilization in next-generation IPTV networks," in *IEEE International Conference on Image Processing '04*, 2004.

[48] Jennehag, U., Zhang, T., and Pettersson, S., "Improving transmission efficiency in H.264 based IPTV systems," *IEEE Transactions on Broadcasting*, vol. 53, pp. 69–77, Mar 2007.

[49] Joo, H., Song, H., Lee, D.-B., and Lee, I., "An effective IPTV channel control algorithm considering channel zapping time and network utilization," *IEEE Transactions on Broadcasting*, vol. 54, pp. 208–216, June 2008.

[50] Karan, A., Filsfils, C., and Farinacci, D., "Multicast-only fast reroute," IETF draft, Cisco Systems.

[51] Kerpez, K., Waring, D., Lapiotis, G., Lyles, J., and Vaidyanathan, R., "IPTV service assurance," *IEEE Communications Magazine*, vol. 44, no. 9, pp. 166–172, 2006.

[52] Kim, Y., Park, J. K., Choi, H. J., Lee, S., Park, H., Kim, J., Lee, Z., and Ko, K., "Reducing IPTV channel zapping time based on viewer's surfing behavior and preference," in *Broadband Multimedia Systems and Broadcasting, 2008 IEEE International Symposium on*, pp. 1 –6, 31 2008-april 2 2008.

[53] Kooji, R., Ahmed, K., and Brunnstrom, K., "Perceived quality of channel zapping," in *5th IASTED International Conference on Communication Systems, Networks, and Applications*, 2006.

[54] Kuo, W.-H., Liu, T., and Liao, W., "Utility-based resource allocation for layer-encoded IPTV multicast in IEEE 802.16 (WiMAX) wireless networks," in *IEEE International Conference on Communications, ICC'07*, pp. 1754 – 1759, 2007.

[55] Lambert, P., Debevere, P., Cock, J. D., Macq, J.-F., Degrande, N., Vleeschauwer, D. D., and Walle, R. V. D., "Real-time error concealing bitstream adaptation methods for SVC in IPTV streams," *Journal of Real-Time Image Processing*, vol. 4, no. 1, pp. 79–90, 2008.

[56] Laneman, J., Tse, D., and Wornell, G., "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *Information Theory, IEEE Transactions on*, vol. 50, pp. 3062 – 3080, dec. 2004.

[57] Le, L. and Hossain, E., "Cross-layer optimization framework for multihop wireless networks using cooperative diversity," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 2592–2602, Jul 2008.

[58] Lee, C. Y., Hong, C. K., and Lee, K. Y., "Reducing channel zapping time in IPTV based on user's channel selection behaviors," *Broadcasting, IEEE Transactions on*, vol. 56, pp. 321 –330, sept. 2010.

[59] Lee, E., Whang, J., Oh, U., Koh, K., and Bahn, H., "Popular channel concentration schemes for efficient channel navigation in internet protocol televisions," *Consumer Electronics, IEEE Transactions on*, vol. 55, pp. 1945 –1949, november 2009.

[60] Lee, J., Lee, G., Seok, S., and Chung, B., "Advanced scheme to reduce IPTV channel zapping time," in *Managing Next Generation Networks and Services*, vol. 4773 of *Lecture Notes in Computer Science*, pp. 235–243, 2007.

[61] Lee, Y., Lee, J., Kim, I., and Shin, H., "Reducing IPTV channel switching time using H.264 scalable video coding," in *Consumer Electronics, 2008. ICCE 2008. Digest of Technical Papers. International Conference on*, pp. 1 –2, jan. 2008.

[62] Lee, Y., Lee, J., Kim, I., and Shin, H., "Reducing IPTV channel switching time using H.264 scalable video coding," *Consumer Electronics, IEEE Transactions on*, vol. 54, pp. 912 –919, may 2008.

[63] Li, Z. and Herfet, T., "MAC layer multicast error control for IPTV in wireless LANs," *IEEE Transactions on Broadcasting*, vol. 55, pp. 353–362, Jun 2009.

[64] Li, Z., Zhu, X., Begen, A. C., and Girod, B., "Peer-assisted packet loss repair for IPTV video multicast," in *ACM Multimedia*, pp. 401–409, 2009.

[65] Liu, P., Tao, Z., Lin, Z., Erkip, E., and Panwar, S., "Cooperative wireless communications: a cross-layer approach," *Wireless Communications, IEEE*, vol. 13, pp. 84 –92, aug. 2006.

[66] Liu, P., Tao, Z., Narayanan, S., Korakis, T., and Panwar, S. S., "Coopmac: A cooperative mac for wireless lans," *Selected Areas in Communications, IEEE Journal on*, vol. 25, pp. 340 –354, february 2007.

[67] Lu, M.-T., Nien, H., Wu, J.-C., Peng, K.-J., Huang, P., Yao, J. J., Lai, C.-C., and Chen, H. H., "A scalable peer-to-peer IPTV system," in *4th IEEE Consumer Communications and Networking Conference*, pp. 313–317, 2007.

[68] Lu, M.-T., Wu, J.-C., Peng, K.-J., Huang, P., Yao, J. J., and Chen, H. H., "Design and evaluation of a P2P IPTV system for heterogeneous networks," *IEEE Transactions on Multimedia*, vol. 9, pp. 1568–1579, Dec 2007.

[69] Luby, M., Stockhammer, T., and Watson, M., "Application layer FEC in IPTV services," *IEEE Communications Magazine*, pp. 94–101, May 2008.

[70] Luebben, E., Li, G., Wang, D., Doverspike, R., and Fu, X., "Fast rerouting for IP multicast in managed IPTV networks," in *17th International Workshop on Quality of Service*, pp. 1–5, 2009.

[71] Madan, R., Mehta, N., Molisch, A., and Zhang, J., "Energy-efficient cooperative relaying over fading channels with simple relay selection," *Wireless Communications, IEEE Transactions on*, vol. 7, pp. 3013 –3025, august 2008.

[72] Maisonneuve, J., Deschanel, M., Heiles, J., Li, W., Liu, H., Sharpe, R., and Wu, Y., "An overview of IPTV standards development," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 315–328, 2009.

[73] Mamun-Or-Rashid, M., Kim, D. S., and Hong, C. S., "A channel management framework to construct user preferred fast channel change stream in IPTV," in *Proceedings of the 11th Asia-Pacific Symposium on Network Operations and Management: Challenges for Next Generation Network Operations and Service Management*, APNOMS '08, (Berlin, Heidelberg), pp. 462–465, Springer-Verlag, 2008.

[74] Mann, I. and McLaughlin, S., "Appropriate models to represent impulsive noise inter-arrival times, duration, and amplitude statistics," ETSI WG TM6 TD14, 2000.

[75] Mann, I., McLaughlin, S., Henkel, W., Kirkby, R., and Kessler, T., "Impulse generation with appropriate amplitude, length, interarrival, and spectral characteristics," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 901–912, Jun 2002.

[76] Mingardi, C. and Brunner, M., "IPTV quality of service management in home networks," in *IEEE International Conference on Communications, ICC'09*, pp. 1938–1883, 2009.

[77] Nedev, N. H., *Analysis of the Impact of Impulse Noise in Digital Subscriber Line Systems*. PhD thesis, The University of Edinburg, 2003.

[78] Nosratinia, A., Hunter, T., and Hedayat, A., "Cooperative communication in wireless networks," *Communications Magazine, IEEE*, vol. 42, pp. 74 – 80, oct. 2004.

[79] Oh, U., Lim, S., and Bahn, H., "Channel reordering and prefetching schemes for efficient IPTV channel navigation," *Consumer Electronics, IEEE Transactions on*, vol. 56, pp. 483 –487, may 2010.

[80] Prins, M. J., Brunner, M., Karagiannis, G., Lundqvist, H., and Nunzi, G., "Fast RTP retransmission for IPTV - implementation and evaluation," in *IEEE Global Communications Conference, GLOBECOM'08*, 2008.

[81] Qiu, T., Ge, Z., Lee, S., Wang, J., Zhao, Q., and Xu, J., "Modeling channel popularity dynamics in a large IPTV system," in *ACM SIGGMETRICS/Performance'09*, 2009.

[82] Reed, I. S. and Chen, X., *Error-control Coding for Data Networks*. Kluwer Academic Publishers, 1999.

[83] Retnasothie, F. E., Ozdemir, M. K., Yucek, T., Celebi, H., Zhang, J., and Muththaiah, R., "Wireless IPTV over WiMAX: Challenges and applications," in *IEEE Annual Wireless and Microwave Technology Conference, WAMICON'06*, pp. 1 –5, 2006.

[84] Robins, M., "Delivering optimal quality of experience (QoE) for IPTV success," whitepaper, Spirent Communications, 2006.

[85] Sarni, M., Hilt, B., and Lorenz, P., "A novel scheme for a fast channel change in multicast IPTV system," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1 –6, june 2011.

[86] Sasaki, C., Tagami, A., Hasegawa, T., and Ano, S., "Rapid channel zapping for IPTV broadcasting with additional multicast stream," in *IEEE International Conference on Communications '08*, 2008.

[87] Scaglione, A. and Hong, Y.-W., "Opportunistic large arrays: cooperative transmission in wireless multihop ad hoc networks to reach far distances," *Signal Processing, IEEE Transactions on*, vol. 51, pp. 2082 – 2092, aug. 2003.

[88] Schwartz, M., "Enabling a quality IPTV customer experience," whitepaper, Telcordia, 2005.

[89] Sendonaris, A., Erkip, E., and Aazhang, B., "User cooperative diversity Part I and Part II," *IEEE Transactions on Communications*, vol. 51, pp. 1927–1948, Nov 2003.

[90] Shang, Y., *Global Search Methods for solving nonlinear optimization problems.* PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[91] She, J., Hou, F., Ho, P.-H., and Xie, L.-L., "IPTV over WiMAX: Key success factors, challenges, and solutions," *IEEE Communications Magazine*, vol. 45, pp. 87–93, Aug 2007.

[92] She, J., Yu, X., Ho, P.-H., and Yang, E.-H., "A cross-layer design framework for robust IPTV services over IEEE 802.16 networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 235–245, Feb 2009.

[93] Shihab, E., Wan, F., Cai, L., Gulliver, A., and Tin, N., "Performance analysis of IPTV traffic in home networks," in *IEEE Global Telecommunications Conference, GLOBECOM'07.*, pp. 5341–5345, 2007.

[94] Shokrollahi, A., "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, Jun 2006.

[95] Siebert, P., Caenegem, T. N. M. V., and Wagner, M., "Analysis and improvements of zapping times in IPTV systems," *IEEE Transactions on Broadcasting*, vol. 55, pp. 407–418, Jun 2009.

[96] Singh, H., Kvvon, C. Y., Kim, S. S., and Ngo, C., "IPTV over wireless-LAN: Promises and challenges," in *5th IEEE Consumer Communications and Networking Conference, CCNC'08*, pp. 626–631, 2008.

[97] Smith, D. E., "IP TV bandwidth demand: multicast and channel surfing," in *IEEE International Conference on Computer Communications, INFOCOM'07*, pp. 2546–2550, 2007.

[98] Sun, W., Lin, K., and Guan, Y., "Performance analysis of a finite duration multichannel delivery method in IPTV," *IEEE Transactions on Broadcasting*, vol. 54, pp. 419–429, Sep 2008.

[99] Uilecan, I. V., Zhou, C., and Atkin, G. E., "Framework for delivering IPTV services over WiMAX wireless networks," in *IEEE International Conference on Electro/Information Technology*, pp. 470 – 475, 2007.

[100] Vandalore, B., Feng, W.-C., Jain, R., and Fahmy, S., "A survey of application layer techniques for adaptive streaming of multimedia," *Real Time Imaging*, vol. 7, pp. 221–235, 2001.

[101] Wah, B. W. and Wu, Z., "The theory of discrete lagrange multipliers for nonlinear discrete optimization," *Principles and Practice of Constraint Programming*, pp. 28–42, Oct. 1999.

[102] Wang, Y. and Zhu, Q.-F., "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, 1998.

[103] Waring, D. L., "The asymmetrical digital subscriber line (ADSL): A new transport technology for delivering wideband capabilities to the residence," in *IEEE Global Communications Conference, GLOBECOM'91*, 1991.

[104] Wu, M., Makharia, S., Liu, H., Li, D., and Mathur, S., "IPTV multicast over wireless LAN using merged hybrid ARQ with staggered adaptive FEC," *IEEE Transactions on Broadcasting*, vol. 55, pp. 363–374, Jun 2009.

[105] Wu, Z., "The discrete lagrangian theory and its application to solve nonlinear discrete constrained optimization problems," Master's thesis, University of Illinois at Urbana-Champaign, 1998.

[106] Xiao, Y., Du, X., Zhang, J., Hu, F., and Guizani, S., "Internet protocol television (IPTV): The killer application for the next-generation Internet," *IEEE Communications Magazine*, pp. 126–134, Nov 2007.

[107] Xiao, Y., Du, X., Zhang, J., Hu, F., and Guizani, S., "Internet protocol television (IPTV): The killer application for the next-generation Internet," *IEEE Communications Magazine*, vol. 45, pp. 126–134, November 2007.

[108] Xun, B., Sun, W., Jin, Y., Guo, W., Hu, W., and Lin, K., "Performances of random IPTV channel change with finite duration multi-channel delivery," in *Communications (ICC), 2010 IEEE International Conference on*, pp. 1 –5, may 2010.

[109] Yajnik, M., Kurose, J., and Towsley, D., "Packet loss correlation in the MBone multicast network," in *IEEE Global Telecommunications Conference, GLOBECOM '96*, pp. 94 – 99, 1996.

[110] Yuksel, M., Ramakrishnan, K. K., and Doverspike, R. D., "Cross-layer failure restoration techniques to provide a robust IPTV service," in *IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, 2008.

[111] Zeadally, S., Moustafa, H., and Siddiqui, F., "Internet protocol television (IPTV): Architecture, trends, and challenges," *Systems Journal, IEEE*, vol. 5, pp. 518 –527, dec. 2011.

[112] Zhang, J., Wang, Y., and Rong, B., "QoS/QoE techniques for IPTV transmissions," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB'09*, pp. 1 – 6, 2009.

VITA

Aytac Azgin received his B.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, and the M.S degree in electrical and computer engineering from the University of Arizona, Tucson, AZ. He is currently a Ph.D. candidate in School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta, GA. His research interests include multimedia networking, IPTV QoE, and next-generation wireless networks.