2012-01-01

Minimizing The Retransmission Delay And Energy Expense Of ARQ With The Optimal Placement Of Caching Routers In A Video Communication Network

Jesus Enrique Hernandez
University of Texas at El Paso, jesus.e.hernandez.l@gmail.com

MINIMIZING THE RETRANSMISSION DELAY AND ENERGY EXPENSE OF ARQ WITH THE OPTIMAL PLACEMENT OF CACHING ROUTERS IN A VIDEO COMMUNICATION NETWORK

JESUS E. HERNANDEZ

Department of Electrical and Computer Engineering

APPROVED:

Michael P. McGarry, Ph.D., Chair

Patricia Teller, Ph.D.

John Moya, Ph.D.

Benjamin C. Flores, Ph.D.
Interim Dean of the Graduate School

A mis padres, con amor, por su incondicional apoyo.

Su incansable esfuerzo por sacar mi familia adenlante es mi inspiracion.

MINIMIZING THE RETRANSMISSION DELAY AND ENERGY EXPENSE OF ARQ WITH THE OPTIMAL PLACEMENT OF CACHING ROUTERS IN A VIDEO COMMUNICATION NETWORK

BY

JESUS E. HERNANDEZ

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

May 2012

Acknowledgements

I would like to express the deepest appreciation to my research advisor, Dr. Michael McGarry for without his help and guidance this thesis would not have been possible. Not only did he introduce me to the wonderful world of research, but he also offered me the best advice a graduate student could hope for in regard to research, scholarship, and even personal matters. For that I am very grateful to him.

I would also like to extend my gratitude to Dr. Patricia Teller and Dr. John Moya for kindly accepting to be part of my thesis committee, with special thanks to Dr. Moya for acting as my academic advisor during my graduate studies.

I am also very thankful to Dr. Patricia Nava for encouraging me to pursue a master's degree, for helping me with the acceptance in the program, and for giving me the opportunity to be a teaching assistant which, along the opportunity to be a research assistant, allowed me to enjoy the full experience of graduate school.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Motivation

Recent trends show that video communication services, such as internet video to PC or video on demand, are projected to account for the vast majority of consumer traffic in packet-switched communication networks within the next years [1]. A major challenge with video communication services over packet-switched networks is the delivery of low-error video content under the tight delay and bandwidth constraints inherent to video information. Video is particularly vulnerable to errors because in order to transmit video over a communication network the video content has to be compressed to match the transmission rate of the communication channel. The compression and coding techniques introduce vulnerabilities in the video signal such as the possibility of error propagation, due to temporal prediction in the encoding, or the fact that an error in a single bit can result in the damage of a large portion of a video frame [2]. Similarly, video is very vulnerable to errors when transmitted over a packet-switched network because the loss of a packet may result in the loss of a significant part of a video frame, thus, resulting in poor video quality.

Several error control and recovery techniques exist that mitigate the effects of bit errors or packet loss in video communication. However, there is always a trade-off between the gain in video quality and the expense of using the error control and recovery techniques. For instance, one type of error control technique consists of encoding the video signal with redundancy bits so that the decoder at the receiver side can recover damaged sections in the video frame. While this technique provides an effective way of recovering from damaged bits, including redundancy bits in the codewords, it increases the bandwidth required to transmit the same amount of video information in comparison to transmitting without redundancy. Another technique used

to recover from errors in communication is the Automatic Repeat request (ARQ) which consists of retransmitting lost packets repeatedly until successfully received. This technique comes with a trade-off as well; reliable transfer of packets is ensured at the expense of extra delay incurred by the retransmission of lost packets.

One error recovery technique in particular was found to have great potential and the majority of this thesis is focused on improving this technique. Specifically, the reduction of the delay expense of ARQ by means of caching is investigated. In the case of packet loss in a video communication network, retransmission requests of the lost packet can be serviced by intermediary routers with caching capability rather than by the source, thereby reducing the retransmission delay and energy consumption. While other work had already proposed the idea of caching video packets in intermediary routers for robust transmission [3, 4], this thesis proposes a solution to the optimization problem of placing a limited amount of caching routers in a network. In other words, this thesis focuses on the optimal placement of the caching routers in a network rather than the protocol to cache the packets or the optimal selection of video content to be cached.

A mathematical program was formulated with the objective to minimize the retransmission delay and energy consumption of ARQ for video. A dynamic programming solution was found that yields an optimal placement of caching routers [5]. Furthermore, a computer simulation model was developed to validate the results of the analytical model (dynamic programming solution) giving more strength to the conclusions of the analysis presented in chapter 5. Through extensive experimentation with the analytical model and the simulation model, it was found that an optimal placement of cache-capable routers in a network yields a significant reduction in retransmission delay and energy consumption, thus, making ARQ a more enticing error recovery technique for video communication over packet-switched networks.

1.2 Thesis structure

The rest of this thesis is organized as follows. First, an extensive literature review of error control and recovery techniques is presented in Chapter 2. The taxonomy used to describe the literature review can be observed in Figure 2.1. Several error control and recovery techniques of each group are discussed in this chapter along with examples of current implementations of such techniques.

Next, Chapter 3 formulates the research problem investigated in the rest of the thesis. The chapter starts by introducing the network structure used in the analysis to characterize a typical path in a video communication network. Then, the parameters used to describe the behavior of the network, such as the drop probability and the average transmission delay, are described in detail. Finally, a detailed explanation of the metrics used to measure the performance of the proposed solution is presented. These metrics are the average retransmission delay and the average energy consumption. The main purpose of this thesis is to find the optimal placement of caching routers to minimize these two metrics.

The analytical model developed in previous work (of which I am co-author) to find the optimal placement of caching routers is briefly described in Chapter 4; it is followed by a detailed explanation of the computer simulation model. The computer simulation model was developed to validate the results found using the analytical model and to conduct an extensive set of experiments to measure the performance of the optimal placement of caching routers. Finally, at the end of this chapter a brief discussion of how the simulation model was used to extend the unicast analysis to a multicast scenario is presented.

The set of experiments conducted to measure the performance of the optimal placement is described in Chapter 5. The different experiments and their results are presented in this chapter accompanied by visual representation of the data obtained.

Finally, Chapter 6 presents a summary of the thesis as well as the conclusions and proposed

future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Error Control and Recovery Techniques

Extra care must be taken when designing the system levels in a video communication system and the error control and recovery techniques in addition to error concealment techniques have to be included in the design. On the one hand, error control and recovery refers to techniques that achieve the full recovery of lost packets. Specifically, error control refers to techniques that add redundancy to the bit stream in such a way that the decoder can recover damaged bits in a packet using the redundancy bits, this way the damaged packet is fully recovered. Error recovery also refers to techniques that aim at recovering the original video content in a lost packet; but in this case other techniques such as retransmissions are used. On the other hand, error concealment refers to techniques that aim at reconstructing the received signal in order to obtain the closest approximation to the original signal. Furthermore, error resilience refers to encoding techniques that generate robust bit streams to guarantee certain levels of quality or to minimize the decrease in quality upon the occurrence of errors [2].

The four groups are: transport level, encoder level, decoder level and interactive error control. Error-resilient coding techniques must be applied at the source of the video communication system. These error-resilient coding techniques are discussed in the transport level and encoder level error recovery sections. In the decoder level error concealment section, error concealment techniques used when designing the decoder are discussed. Finally, error recovery techniques that involve cooperation between the encoder and the decoder are discussed in the interactive error recovery section. The taxonomy used to describe the research work on error contro and recovery for video communication is shown in Figure 2.1.

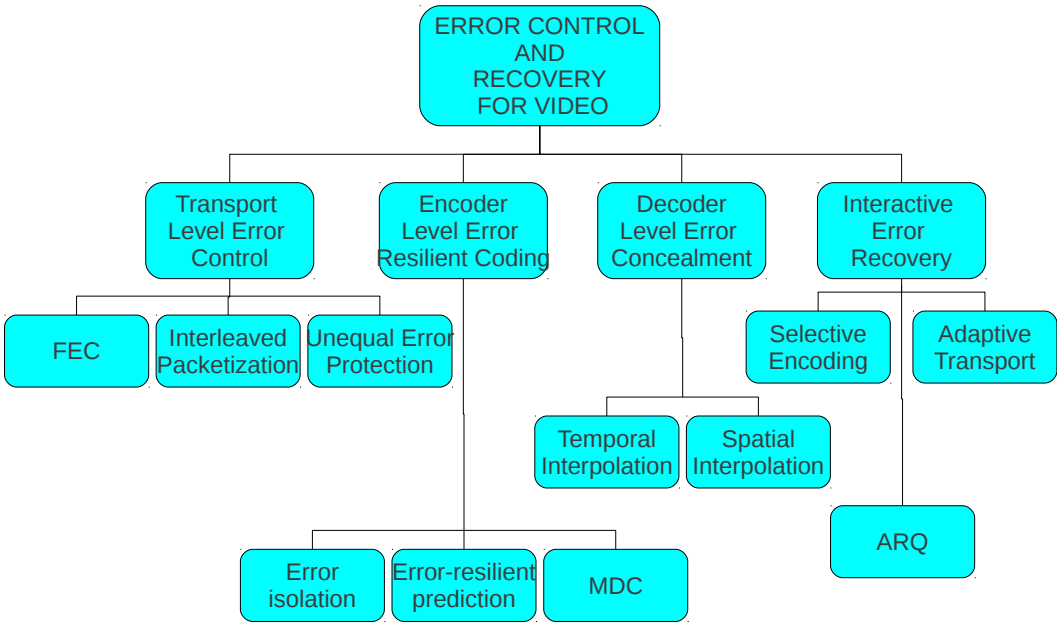


Figure 2.1: Taxonomy of error control techniques

2.1.1 Transport Level Error Control

Error control at the transport level is the most important because it provides the basic quality of service that can be further improved by applying error control and recovery techniques at the encoder or decoder levels [2]. There are many techniques developed to provide transport level error control; in this section three main techniques are described: forward error correction (FEC), interleaved packetization and unequal error protection or layered coding.

2.1.1.1 Forward Error Correction

Forward Error Correction attaches redundant or parity bits to the compressed bit stream prior to transmission in order to increase the data reliability. The encoder inserts redundancy

bits into the bit stream to form codewords. The codewords are, then, transmitted and received by a suitable decoder that can extract the original data from the codeword and possibly correct the data sequence in case of corruption or damage. Since forward error correction increases the number of bits in a single transmission due to redundancy bits, the bandwidth requirements increase with the use of FEC. For that reason FEC must be used with special care when dealing with video signals, since video transmission already has very tight bandwidth requirements. Also, when FEC is used on packet-switched networks additional techniques must be used to support it. For instance, block interleaving can be used so that packet loss only results in the loss of a part of the FEC block and not the entire block. This allows the encoder to be able to recover from packet loss in which many bits are lost [2].

A popular FEC code is the Reed-Solomon error correction code. Reed-Solomon (RS) codes belong to the family of linear block codes. Block codes work on fixed sized blocks or packets, as opposed to arbitrarily long bit streams. An RS (n,k) code encodes a sequence of k bits into a sequence of n encoding elements or codewords [6]. Such encoding enables the detection and correction of bit errors. Another popular FEC code is the Hamming code. The Hamming code adds extra parity bits to the codewords in order to detect errors; however, Hamming codes not only detect errors but can also correct one-bit errors. For instance, the Hamming(7,4) code encodes 4 bits of data into 7 bits of data by adding 3 check bits [7]. Yet another type of FEC code is the turbo code [8].

An implementation of forward error correction for video multicast is presented in [9]. In this technique, FEC is combined with stream replication in order to mitigate packet losses. In this FEC scheme, the server determines the optimal number of FEC packets and replicated streams to meet certain packet loss requirements while maintaining the bandwidth budget under control. The clients, then, autonomously choose based on their local packet loss rate whether to subscribe to the recovery packets or not. According to the Chan, et al. the combination of forward error correction and stream replication yields a reduction in error rate of more than 50% [9].

2.1.1.2 Interleaved Packetization

Another way of controlling errors at the transport level is to packetize the video information in such a way that packets contain independent coded data blocks and the error can be isolated. By using interleaved packetization, loss of contiguous blocks can be avoided. That is, if adjacent blocks are put into separate packets, the loss of a packet will only result in the loss of the block contained in that packet and not in the loss of the adjacent blocks too [2]. An implementation of interleaved packetization is presented in [10]. In this technique, block interleaving is combined with a packetization scheme to suppress the effect of packet loss. For this scheme the authors propose a simple even/odd block interleaving scheme in which successive packets are first filled by even-indexed blocks followed by odd-indexed blocks in the same slice of the macroblock. This way, if an even block is damaged, it will be surrounded by odd-indexed blocks and the corrupted even-indexed block can be recovered. Moreover, the blocks are packetized at the codeword level in the same even/odd interleaving scheme along with the even/odd index information and the address of the first block in the packet. This way, the codewords in each packet can be easily decoded. According to the authors of this technique, this interleaved packetization scheme effectively suppresses the effect of packet loss at the cost of 3.4% of overhead by the added redundancy [10].

2.1.1.3 Unequal Error Protection

Transport level error control can assign priorities to different layers. Naturally, not all of the bits in a compressed bit stream are equally important. Header and side information, for instance, are more important than block data. Moreover, some types of video frames carry more important information than others. For instance, in a group of pictures, Intra coded frames (I-frames) contain most of the information in a scene and are coded without any prediction. Predicted frames (P-frames) use prediction from one reference in the encoding and contain only the motion information with reference to the previous I-frame or P-frame. Bi-predicted frames (B-frames)

are the most compressed because they contain only the differences between the current frame and the two frames in both directions; that is, the previous frame and the following frame [2]. Hence, the transport controller can assign a higher priority to more important frames (I-frames) or to more important bits in a bit stream such as the header. Besides assigning a higher priority to those bits, extra protection can be provided to make sure the important information gets to its destination without errors [11]. An implementation of unequal error protection in which different levels of FEC are implemented for each bit plane is presented in [12]. In this implementation, more redundancy is added to higher bit planes in order to maximize the quality of the decoded video. Moreover, in this implementation several sub-streams are created and interleaved in such a way that an error in one sub-stream does not affect other sub-streams. Also, different levels of power are used to transmit the sub-streams based on their priority which adds more transport prioritization [12].

2.1.2 Encoder Level Error Control

Source coding techniques are designed to produce a robust bit stream resilient to errors in such a way that the effect on the decoder's ability to extract a good quality signal is minimized. Error-resilient encoding techniques help to obtain good quality decoded video at the expense of introducing overhead in the form of redundancy bits [2]. Three major techniques to use redundancy in order to provide error-resilient encoding are discussed in this section.

2.1.2.1 Error Isolation

Error isolation techniques confine an error within a limited region in order to avoid the propagation of the error and to facilitate error concealment at the decoder level. Two error isolation techniques are described in [2]. The first technique is based on the introduction of synchronization markers. In this technique, the source data is divided into fixed size blocks of data and each block is compressed independently. Easily identifiable synchronization markers are

inserted between each block. The decoder, then, reconstructs the data of each block and expects a synchronization marker at the end of each block. If the decoder does not find a synchronization marker at the end of a block then it knows there was a channel error and that block is corrupted. In that case, the decoder looks for the next synchronization marker and resumes the decoding from there [2].

Another error isolation technique is called data partitioning. Using synchronization markers all the data between two synchronization markers has to be discarded in case of error. However, if the data between two synchronization markers is further divided by smaller synchronization markers to make individual logic units, in case of an error, only the affected logic unit need be discarded and the rest of the units can be correctly decoded. This way, important sections of a bit stream, such as the header or the motion vectors, can be located in individual logic units and be correctly decoded [2].

2.1.2.2 Error-Resilient Prediction

A major issue in error control for video is the propagation of error due to the temporal prediction used at the encoder and decoder. That is, once an error occurs in a frame at the decoding stage, every following frame will be reconstructed erroneously because the previous erroneous frame is used as a reference. Two major techniques to stop error propagation are to insert intra-blocks periodically and to divide the data into independent segments [2]. By using intra-coded blocks periodically the error propagation is effectively controlled because the error can now only propagate in between intra-blocks; once it finds an intra-coded block, the error stops and proper decoding resumes. A similar approach is to split the data into independent segments so that the the encoder/decoder perform temporal prediction only within independent segments and the error does not propagate outside the segment [11].

Weigand, et al. propose a long term memory predictor to avoid error propagation in [13]. In this implementation the number of intra-coded blocks is balanced against the number of

propagating errors because the use of intra-coded blocks is generally less efficient than inter-coded blocks. Also, they use a long-term memory motion compensated predictor that uses several previously decoded frames instead of using only the previous frame in order to stop the possible error that occurred in the previous frame from propagating [13].

2.1.2.3 Multiple Description Coding

Yet another method to avoid error propagation is to generate multiple bit streams, also referred to as descriptions, and transmit each bit stream on a different channel. Then, the decoder will use whichever description arrives without error to reconstruct the video signal. Each description is designed so that the complete video signal can be reconstructed from it with good quality, and extra quality can be obtained by using more descriptions [2]. An issue with the use of multiple descriptions when transmitting video is the assignment of bandwidth to the descriptions. Xia, et al. formulated this issue as an optimization problem and determined the optimal bandwidth assignment for the descriptions given that the number of descriptions to be used does not fall under a predetermined threshold [14].

2.1.3 Decoder Level Error Concealment

These techniques are possible with the help of error-resilient coding techniques, such as the techniques described in the previous section that insert redundancy in the bit stream to be later used to reconstruct the video signal at the decoder. Moreover, the compressed bit stream also contains some statistical redundancy that can be used to reconstruct the video signal. There are three main types of information that need to be recovered from a damaged block: Texture information, motion information and coding mode. Spatial and temporal interpolation are used to recover texture information from damaged blocks due to the inherent low-frequency components of natural scenes that make variation in adjacent pixels very smooth [11]. Two techniques used to recover texture information are described in this section. Motion vectors and

coding mode can also be interpolated from adjacent blocks due to the ineherent temporal and spatial smoothness.

2.1.3.1 Temporal Error Concealment

Temporal error concealment techniques take advantage of temporal redundancy in a sequence and are used for inter-coding pictures since there is motion present. Two techniques are described in [15]. The first technique consists of merely copying the previously decoded macro-block into the damaged macro-block. The second technique is more efficient because it takes into consideration the motion vectors and is combined with spatial interpolation. In this technique the macro-block pointed at by the motion vector in the damaged macro-block is used. Since errors result in a horizontal stripe, the nearest macro-blocks above and below are used for the spatial interpolation [15].

2.1.3.2 Spatial Error Concealment

Spatial error concealment techniques take advantage of spatial redundancy in a picture and are used for intra-coded pictures, that is, frames with no motion information. In spatial error concealment techniques the pixels in a damaged block are interpolated from neighboring blocks or macro-blocks. Two techniques are discussed in [15]. The first technique consists of interpolating every block in a macro-block using the pixels in neighboring blocks. The second technique consists of interpolating every single pixel in a macro-block, instead of interpolating entire blocks, with the pixels in the four neighboring macro-blocks [15].

2.1.4 Interactive Error Recovery

In this section the author describes interactive error recovery for video communication. Error concealment can be greatly improved if the encoder and decoder have some feedback path to

communicate with each other. If such backward channel exists, the encoder and decoder can cooperate by making the encoder aware of damaged blocks so that it can change its encoding parameters (quantization values, reference frames, etc.) accordingly. Furthermore, the decoder can communicate with the transport protocol to control the amount of bandwidth used for forward error correction and for retransmission as well [11]. Two techniques of interactive error recovery are described in this section.

2.1.4.1 Selective Encoding

Selective encoding techniques take advantage of the backward channel from decoder to encoder. When the decoder detects a faulty block, it communicates with the encoder and alerts it of the temporal and spatial location of damaged blocks so that the encoder can treat the affected areas differently; that way the error propagation is reduced. An example of selective encoding is shown in [16]. In this technique, when the decoder detects a damaged or lost packet it informs the encoder of the identity of the damaged packet and proceeds to perform error concealment in the damaged packet. In turn, the encoder determines the affected area from the information received from the decoder and continues the encoding but it does not use the affected area for prediction.

2.1.4.2 Automatic Repeat Request

Automatic repeat request (ARQ) is an error recovery technique that provides reliable data transfer based on retransmissions. Three protocol capabilities are required in ARQ to handle errors: error detection, receiver feedback and retransmissions [18]. Detection of lost or damaged packets in ARQ is achieved by using timers, by receiving negative acknowledgements from the receiver, or by detecting a lost positive acknowledgment. Every time a packet is sent out by the sender, a timer is started. If the timer expires and the sender has not received acknowledgement of the reception of the packet, the sender assumes the packet was lost and retransmits it. Positive

acknowledgments (ACK) are used to notify the sender that a packet or group of packets has been received correctly. On the other hand, the receiver uses a negative acknowledgement (NAK) to notify the sender that a packet was lost or damaged. The receiver feedback is used to send positive or negative acknowledgments between the sender and receiver.

There are two main categories of ARQ protocols: Stop-and-Wait ARQ and Sliding-Window ARQ [17]. In a stop-and-wait ARQ protocol, the sender cannot send a new packet until it is sure that the last packet sent was correctly received, by means of a ACK packet. If the packet was lost, the sender retransmits it and waits for the acknowledgement. This process is repeated until the packet is correctly received and the sender is properly acknowledged. At this point the sender can proceed to send a new packet. In a sliding-window ARQ protocol, the sender may transmit several packets without having to wait for the acknowledgement of each packet. In other words, there might be multiple in-transit unacknowledged packets. The number of unacknowledged packets is bounded by the size of the transmission window. Sequence numbers are used to keep track of the outstanding or unacknowledged packets. This type of ARQ protocols is sometimes referred to as pipelined error recovery [18].

While ARQ is a very widely used error recovery technique for data communication, the use of ARQ in video is restricted because of the extra delay caused by retransmissions. The rest of this thesis is dedicated to reducing the retransmission delay cost of ARQ, as well as the energy consumption cost, to make ARQ a more attractive error recovery technique for video communication.

2.1.4.3 Adaptive Transport

Adaptive transport techniques take advantage of the feedback channel too. However, with these techniques, the transport level protocols are the ones that use the information received from the decoder, not the encoder. The information received from the decoder affects the decisions taken at the transport level, such as retransmissions. As mentioned before, retransmission is

generally not a good idea for real-time video communication. However, when properly controlled, retransmissions can be a great help in the process of error concealment.

An example of an adaptive transport technique is retransmission without waiting which is explained in great detail in [19]. In this technique, when a damaged block is detected the decoder sends a retransmission request to the encoder and then proceeds with error concealment in the affected area. Additionally, the decoder records a trace of the affected area for future use. When the decoder receives the retransmitted block, the affected area, previously recorded, is corrected using the retransmitted data and the recorded trace. In the end, the affected area is reproduced as if no error had occurred [19].

Another example of adaptive transport is shown in [11]. This technique effectively combines unequal error protection and retransmission. With this technique, when a retransmission request is received, multiple copies of the packet are sent to increase the probability of successful arrival and to reduce the number of retransmissions required to correct the damaged area. As opposed to data communications, the number of retransmission trials is bounded by the delay constraints inherent in video communication, which is why multiple copies are retransmitted at once.

The downside of retransmitting multiple copies is the overhead in the output rate of the encoder. In order to satisfy output rate constraints at the encoder, the output rate has to be throttled to accommodate the retransmission traffic. When the output rate at the encoder is reduced, layered coding is used so that low-priority blocks, such as enhancement layers, are partially transmitted or even omitted. In this technique the number of retransmission trials is proportional to the priority of the damaged block; in this way unequal error protection and adaptive transport are effectively combined to optimize the error concealment process [11].

2.2 Caching of Video Packets to Reduce Retransmission Delay

In the remainder of this chapter we discuss previous work related to the caching of video packets in a communication network.

The idea of caching packets containing video was first proposed in [3]. In this work the authors classify packets according to their contribution to the overall quality of the video and select the most important packets, in this case packets containing intra-coded frames, to cache in intermediary routers. Also, the authors developed a model that characterizes the gain in quality obtained by selectively caching packets. In [4] the authors expand their idea of classifying packets and investigate the optimal selection of packets that minimize video distortion to improve the overall video playback quality. The work in this thesis focuses on the optimal placement of caching routers, rather than the optimal selection of packets; therefore the research efforts complement one another.

In [20, 21] the authors discuss the architecture of intermediary nodes capable of caching real-time video packets with the purpose of reducing retransmission paths. They propose a new protocol, called Real-Time Media Engine (RTME), which describes the detail of how packets can be cached inside routers. Even though the authors developed a model for the reduction of retransmission delay and discuss improvements to energy consumption, no mathematical model for the minimization of the average retransmission delay or the average energy consumption is presented in their work. Also, they do not provide a solution to the optimization problem of finding the optimal placement of caching routers.

In [22] the authors propose a peer-assisted protocol that describes how lost packets can be retransmitted in a decentralized manner by having end-user set-top boxes coordinate with an intermediary retransmission server to service the retransmission requests in a multicast IPTV network. The optimal placement of caching devices was not discussed in this work. Since this protocol is designed specifically for IPTV networks, the optimal placement of caching routers de-

scribed in this thesis provides a more general solution and could be employed in conjunction with the retransmission servers and set-top boxes in an IPTV network to reduce the retransmission delay.

CHAPTER 3

PROBLEM FORMULATION

The main purpose of this thesis is to reduce the average retransmission delay and the average energy consumption by using intermediary routers that cache video packets to service retransmission requests. If certain routers are designed with the ability to cache video packets for a prescribed period of time and with the ability to service retransmission requests, the average packet retransmission delay can be reduced. Moreover, since the retransmission requests are potentially going to be serviced by intermediary routers rather than the source itself, the distance traveled by the retransmitted packet is shortened, thus, reducing the energy consumed to retransmit a packet. As we discussed in Chapter 2, the idea of caching video packets has been explored before. However, the optimal placement of the caching routers in the network has not been investigated. In this thesis, the effect of the placement of caching routers on the average retransmission delay and average energy consumption is investigated. In order to analyze the optimal placement of caching router, let us first characterize the network and its parameters.

3.1 Network Structure

With the purpose of characterizing the video communication network in the experiments, a typical path in a network is considered. The typical path consists of a video source, several intermediary routers, and a video sink all arranged in a lineal topology for the unicast scenario and in a tree topology for the multicast scenario. To illustrate, consider a typical path with a total of M routers; router 0 is the video source. With the purpose of illustrating the effect of the placement of caching routers on the retransmission delay, the number of routers with caching ability (N) available is limited. In other words, not all of the routers can be empowered with

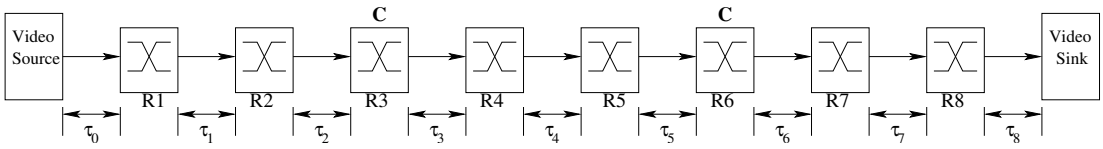


Figure 3.1: Illustration of a network path with video packet caching routers. There are eight routers (i.e., $M = 8$) in the illustrated path. Two routers, 3 and 6, have packet caching abilities (i.e., $N = 2$).

the ability to cache video packets (i.e. $N < M$).

When packets travel through a router in a typical path they suffer from different types of delays such as processing delays, queuing delays, transmission delays, and propagation delays [18]. In this thesis the delay between one router and the next is characterized by a cumulative quantity called average packet transmission delay; it is the summation of all delays from router i to router $i + 1$. A delay vector describes the average delay a packet suffers while traveling through a typical path. Specifically, τ_0 is the total delay from the video source to router 1; τ_1 is the total delay a packet suffers traveling through router 1 and until it arrives at router 2, and so on. The τ vector includes τ_0 to τ_M. Figure 3.1 illustrates a typical path with M=8 routers and N=2 routers with caching ability (routers 3 and 6). Figure 3.2 illustrates a typical multicast network with a total of $M = 6$ routers, 3 video sinks, and $N = 2$ caching routers (routers 3 and 4).

3.2 Packet Drop Probability

In a real video communication network, as opposed to an ideal one, packets can be lost or damaged. For instance, at each router every packet has to be temporarily stored in a buffer before it can be transmitted; moreover, when the packet is ready to be transmitted it has to wait in an output queue until the transmission link is available. Depending on the congestion in the network, these buffers and queues might be filled up with other packets and a packet arriving at a full queue will be dropped. Furthermore, packets might also be damaged in a transmission link

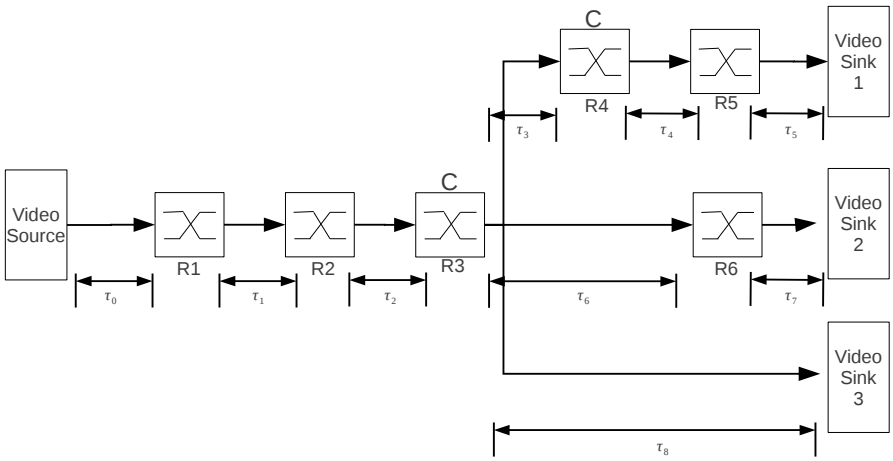


Figure 3.2: Illustration of a multicast network with video packet caching routers. There are six routers (i.e., $M = 6$) in the illustrated network. Two routers, 3 and 4, have packet caching abilities (i.e., $N = 2$).

where bit errors can occur. In other words, in a communication network packets can be dropped or suffer from excessive bit errors; in either case the packet is lost. In this thesis, packet loss is characterized by the probability that a packet is dropped at any given router and is computed as follows:

Let ρ_i be a binary random variable that is 1 when a packet is dropped at router i due to congestion or it being received severely errored and 0 otherwise, Φ_i be the probability that $\rho_i = 1$, $Prob\{\rho_i = 0\} = (1 - \Phi_i)$ and $Prob\{\rho_i = 1\} = \Phi_i$, and γ_i be the probability a packet is dropped at router i. With independent ρ_i random variables,

$$\gamma_i = Prob\left\{\bigcap_{k=1}^{i-1}\{\rho_k = 0\}\bigcap\{\rho_i = 1\}\right\} \tag{3.1}$$

becomes,

$$\gamma_i = \left[\prod_{k=1}^{i-1}(1 - \Phi_k)\right]\Phi_i \tag{3.2}$$

3.3 Average Retransmission Delay

In order to measure the performance of the optimal placement of caching routers, the average retransmission delay has to be characterized because it is the most important metric to compare against other caching router placements. The retransmission delay of a packet depends on the router at which the packet is dropped. For any given packet the retransmission delay is twice the delay from the nearest caching router that contains the packet to the video sink, or twice the delay from the source to the sink in the case that no intermediary caching router contains a copy of the dropped packet. The retransmission delay is twice the delay from one point to the other if we assume symmetry in the typical path and to account for the round trip of an ARQ request. That is, the delay of the ARQ request from the video sink to the caching router or video source, and the delay of the retransmitted packet from the caching router or source back to the video sink. The average retransmission delay depends on the retransmission delay for each router in the typical path and the drop probability associated with each router and is computed as follows:

The analysis of retransmission delay applies to both wired and wireless networks. Let d_i be the retransmission delay for a packet when it is dropped at router i, $\mathbb{E}d$ be the average retransmission delay, η_i be a binary variable that is 1 if router i has caching abilities and 0 if router i does not have caching abilities, and β_i be the upstream caching router that would be able to retransmit a packet that was dropped at router i and is closest to the destination. d_i depends on the round trip delay to the nearest upstream caching router, β_i. A router j has caching capability if $\eta_j = 1$ and is upstream from router i if $j < i$. Therefore,

$$\beta_i = \max\{j : j < i,\ \eta_j = 1\} \tag{3.3}$$

If we assume symmetric packet transmission delays and only consider single retransmissions, then

$$d_i = 2 \cdot \sum_{k=\beta i}^{M} \tau_k \tag{3.4}$$

Using the expectation of a discrete random variable, we obtain

$$\mathbb{E}d = \sum_{i=1}^{M} (d_i \gamma_i) \tag{3.5}$$

3.4 Average Energy Consumption

Besides reducing the retransmission delay, the placement of intermediary caching routers to service retransmission requests potentially reduces the energy consumption of routers in the network as well. In a wireless network, the energy consumed to transmit a packet from one host to another depends on the distance between the hosts. By using intermediary caching routers, the retransmission distance is reduced to twice the distance from the nearest caching router containing a copy of the dropped packet to the sink, instead of twice the distance from the source to the sink. The reduction in the retransmission distance results in the reduction of the energy consumed to retransmit a packet. The average energy consumption depends on the energy consumption at each router in the typical path and the drop probability associated with each router and is computed as follows:

The analysis of energy consumption applies to wireless networks. Let E_i be the energy consumed transmitting one bit from router i to router $i+1$, e_i be the energy consumed for a retransmission of a bit of a packet that was dropped at router i, and $\mathbb{E}e$ be the average energy consumption of a retransmitted bit.

E_i is composed of two parts [23]: e_{circ}, the energy consumed by one bit in the electronic circuitry of a transmitter or receiver and has the units *Joules/bit*, and e_{amp}, the energy consumed by one bit in the amplifier of a transmitter and has the units *Joules/bit/m*^{2}. For our analysis,

both e_{circ}, and e_{amp} can be considered arbitrary constants. For transmission in free space the energy used to transmit a bit at the near end transmitter and receive that bit at the far end receiver is [23]:

$$E_i = 2e_{circ} + x_i^2 e_{amp} \tag{3.6}$$

Accounting for the round trip transmission, e_i is:

$$e_i = 2 \cdot \sum_{k=\beta_i}^{M} E_k \tag{3.7}$$

Using the expectation of a discrete random variable, we obtain

$$\mathbb{E}e = \sum_{i=1}^{M} (e_i \gamma_i) + 0 \left(1 - \sum_{i=1}^{M} \gamma_i\right) \tag{3.8}$$

$$\mathbb{E}e = \sum_{i=1}^{M} (e_i \gamma_i) \tag{3.9}$$

3.5 Analytical Model

Before I describe the simulation model used to perform the experiments and prove that the delay cost of ARQ can be significantly reduced by using an optimal placement of caching routers in a typical path, let us first briefly discuss the analytical model developed in previous work (of which I am co-author) [5]. The analytical model was used to generate preliminary results and used as inspiration to develop the simulation model, later used to validate the results obtained with the analytical model. I wish to make it clear that, even though I am a co-author of the article where the analytical model is presented, the analytical model is not my work. My contribution in that area is that I added functionality to the analytical model by developing

functions that generate other typical placements of caching routers, such as lumped upstream, lumped downstream, spread uniformly, and the average arbitrary placement, to compare against the optimal placement. Also, I developed the functions that calculate the average retransmission delay of the typical placements. The functions I developed for the analytical model were used to conduct an extended numerical analysis to test the performance of the optimal placement against the performance of the typical placements; the results of this numerical analysis are shown in Chapter 5.

For the analytical model, two mathematical programs were developed; one that minimizes the average retransmission delay and the other to minimize the average energy consumption. Both mathematical programs are identical in structure and are solved by the same dynamic programing solution. The main purpose of the mathematical programs is to minimize the average retransmission delay or average energy consumption under the constraints that there are a total of M routers in the typical path, out of those routers only N can have the ability to cache packets, and the decision variables that indicates if a router has the caching ability are binary. The mathematical program for average retransmission delay is shown in Equation 4.1. The mathematical program for the average energy consumption is identical in structure with the delay between routers replaced by the distance between the routers, therefore is omitted.

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{\eta}) = \sum_{i=1}^{M} \gamma_i \sum_{k=\beta i}^{M} \tau_k \\ \text{subject to} \quad & \eta_j \in \{0,1\},\ j = 1, \cdots, M \\ & \textstyle\sum_{j=1}^{M} \eta_j = N \end{aligned} \tag{3.10}$$

The dynamic programming solution that solves both mathematical programs takes as input the aforementioned constraints as well as a delay vector to describe the network, and yields as output the optimal set of caching routers that minimize the average retransmission delay and

energy consumption under the given problem instance. The interested reader is highly encouraged to read the work in [5] for a detailed explanation of the analytical model.

CHAPTER 4

SIMULATION MODEL

We used a simulation model to validate our analytical models presented in Chapter 3. The model represents a network of eight routers arranged in a bus topology. Some of those 8 routers can have the ability to cache packets inside the router to later retransmit them in case packet loss occurs. The main goal of the model is to measure the impact of the placement of intermediate caching routers in the network on the expected retransmission delay. Furthermore, the simulation model was used to extend the analysis to a multicast scenario.

The unicast model consists of four modules developed in C++ using the OMNET++ discrete event simulation library. The four modules are: Source, Sink, Switch, and CacheRouter. Source generates the new packets at a given rate and also services retransmission requests. Sink receives all the packets, requests retransmissions (ARQ), and collects retransmission delay data. Switch is a generic packet-switching device that forwards packets in either direction (upstream or downstream) and keeps track of lost/dropped packets. CacheRouter acts as a packet-switching device as well, but it also caches video packets and services retransmission requests (ARQ).

The functionality of the model is as follows: the Source generates a certain number of packets at a given rate; the packets propagate through the network passing switches and caching routers along the way and finally arrive at the Sink. In the case of a packet loss/drop, the Switch or CacheRouter where the packet is dropped notifies the Sink of the packet drop and, in turn, the Sink requests a retransmission of the packet which is serviced by the nearest CacheRouter containing the lost packet or by the source in case none of the CacheRouter modules contain the lost packet.

The multicast model represents a network of one source, six routers, and three video sinks. The Network is arranged in a tree topology. In addition to the four modules used in the single-cast

model, two more modules are developed for the multicast model: MulticastRouter, which is the multicast equivalent of the switch module, and MulticastCacheRouter, the multicast equivalent of the CacheRouter.

4.1 Background

An OMNeT++ model consists of hierarchically nested modules, which communicate by passing messages to each other. OMNeT++ models are often referred to as networks. The top level module is the system module. The system module contains sub-modules, which can also contain submodules themselves. Modules that contain submodules are termed compound modules, as opposed simple modules which are at the lowest level of the module hierarchy. Simple modules contain the algorithms in the model. The user implements the simple modules in C++, using the OMNeT++ simulation class library. Both simple and compound modules are instances of module types. While describing the model, the user defines module types; instances of these module types serve as components for more complex module types. Finally, the user creates the system module as an instance of a previously defined module type; all modules of the network are instantiated as submodules and sub-submodules of the system module [24].

Modules communicate by exchanging messages. In an actual simulation, messages can represent frames or packets in a computer network, jobs or customers in a queuing network or other types of mobile entities. Messages can contain arbitrarily complex data structures. Simple modules can send messages either directly to their destination or along a predefined path, through gates and connections. In the OMNET++ simulation class, packets can be simulated using a subclass of cMessage called cPacket which has several attributes such as kind, length and priority. Gates are the input and output interfaces of modules; messages are sent out through output gates and arrive through input gates. Modules can have parameters. Parameters can be assigned either in the NED files or the configuration file omnetpp.ini. Parameters may be used to customize simple module behavior, and for parameterizing the model topology [24].

In this particular simulation model, there are six functional modules that describe the model. The six modules are: Source, Sink, Switch, CacheRouter, MulticastRouter, and MulticastCacheRouter. Source generates the new packets at a given rate and also services retransmission requests. Sink receives all the packets, requests retransmissions (ARQ), and collects retransmission delay data. Switch is a generic packet switching device that forwards packets in either direction (upstream or downstream), and also keeps track of lost/dropped packets. CacheRouter acts as a packet switching device as well but it also caches video packets and services retransmission requests (ARQ). MulticastRouter is the equivalent of the Switch module in the multicast scenario and MulticastCacheRouter is the multicast equivalent of the CacheRouter Module.

4.2 Packet Structure

OMNET++ provides a useful way of extending cPacket or cMessage to add the fields you need for a specific packet structure. In this simulation model the message definition capability was used to create a customized packet structure; the structure is called MyPacket and is described in MyPacket.msg. Since MyPacket is a subclass of cPacket, it contains the same fields as the cPackets (i.e. kind, length, and priority) but also adds four more fields: srcAddress, destAddress, pktNumber, and dropAddress. The MyPacket class has setter and getter methods for each field in the structure. The packet structure is shown in Figure 4.1.

4.3 Modules

The six functional modules of this simulation model (i.e. Source, Sink, Switch, Cacherouter, MulticastRouter, and MulticastCacheRouter) are described in detail in this section.

```
packet MyPacket
{
        int kind;
        int bitLength;
        int byteLength;
        int priority;
        int srcAddress;
        int destAddress;
        int dropAddress;
        int pktNumber;
}
```

Figure 4.1: Packet structure

4.3.1 Source

The source module acts as the video source by creating packets at a given rate; the rate is specified by the parameter SendInterval, which can be modified in the NED file. The source module schedules self-messages (i.e. messages sent to itself), called timer, every SendInterval microseconds to indicate it's time to generate a new packet. When a new packet is created, a unique packet number is assigned to it; moreover, the packet is duplicated and the duplicate is cached in a queue. After copying and caching the packet, the original packet is sent on the output port towards the sink.

Besides generating packets, the Source module acts as a caching device too and services retransmission requests. If a retransmission request (ARQ message) is received, the copy of the dropped packet is found in the queue and a copy of it is retransmitted towards the sink with a time stamp associated with it that will later be used by the video sink to calculate the retransmission delay. To determine what packet to look for in the queue, the module inspects a field in the ARQ message called pktNumber. Every ARQ message contains the number of the packet that needs to be retransmitted as well as the address of the router at which the packet

was dropped. The flow diagram of the Source module is shown in Figure 4.2.

4.3.2 Sink

The sink module acts as the video sink by receiving all the packets and also requests retransmissions. Only two types of packets can arrive at the sink, either a DROP message, or a regular packet. If a DROP message is received, the module inspects the pktNumber field in the DROP message and creates a retransmission request message (ARQ) with the same packet number so that the caching router knows which message to retransmit. On the other hand, if a regular packet is received, there are two options; the packet can be either a retransmission or a regular packet successfully transmitted over the network. If the packet is a retransmission, the module inspects the timestamp as well as the dropAddress field to compute the retransmission delay. If the packet is a regular packet successfully transmitted over the network, then the module only increments the packets-received-counter and deletes the packet. In the end, the sink module computes and displays the average retransmission delay. The flow diagram of the Sink module is shown in Figure 4.3.

4.3.3 Switch

The switch module acts as a generic packet switching device by forwarding packets upstream or downstream depending on the type of packet. If a DROP packet is received, the switch simply forwards the packet upstream towards the video sink. If, instead, the packet is an ARQ message, the switch forwards the packet downstream towards the nearest caching router. Now, if the packet received is a regular packet, the switch first checks if the packet arrived in error; if so, the packet is deleted and a DROP notification with the proper packet number and drop address is sent to the video sink, which in turn requests a retransmission. The packet number of the DROP notifications is set to the same number as the dropped packet. The drop address field is set to the id number of the router sending the DROP notification. If no error is found in the

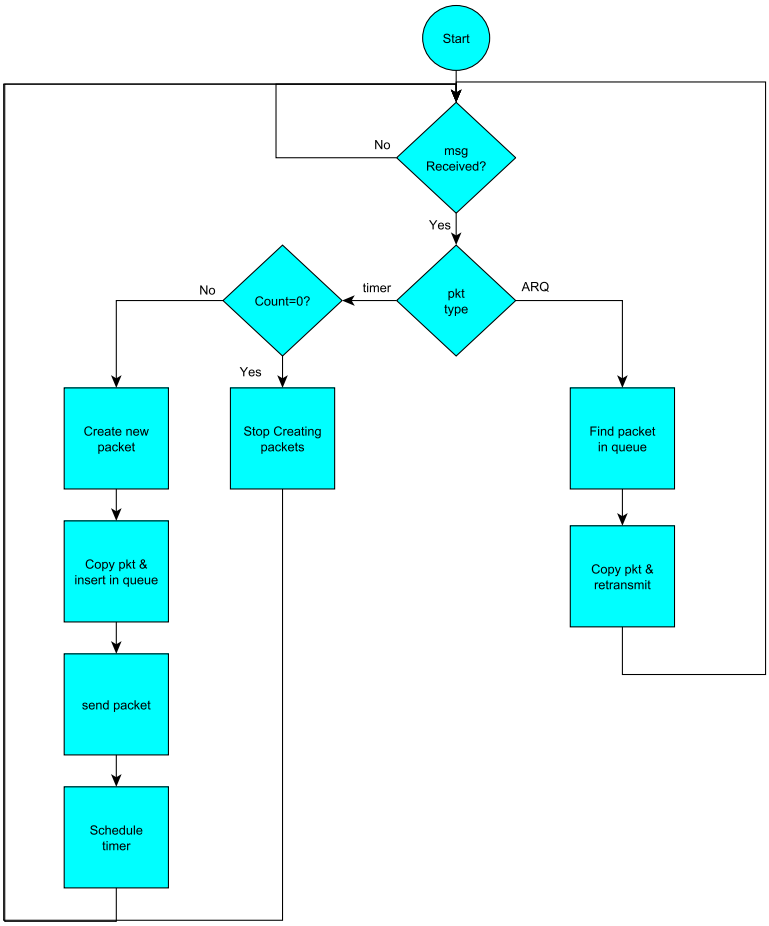


Figure 4.2: Flow diagram of the Source module

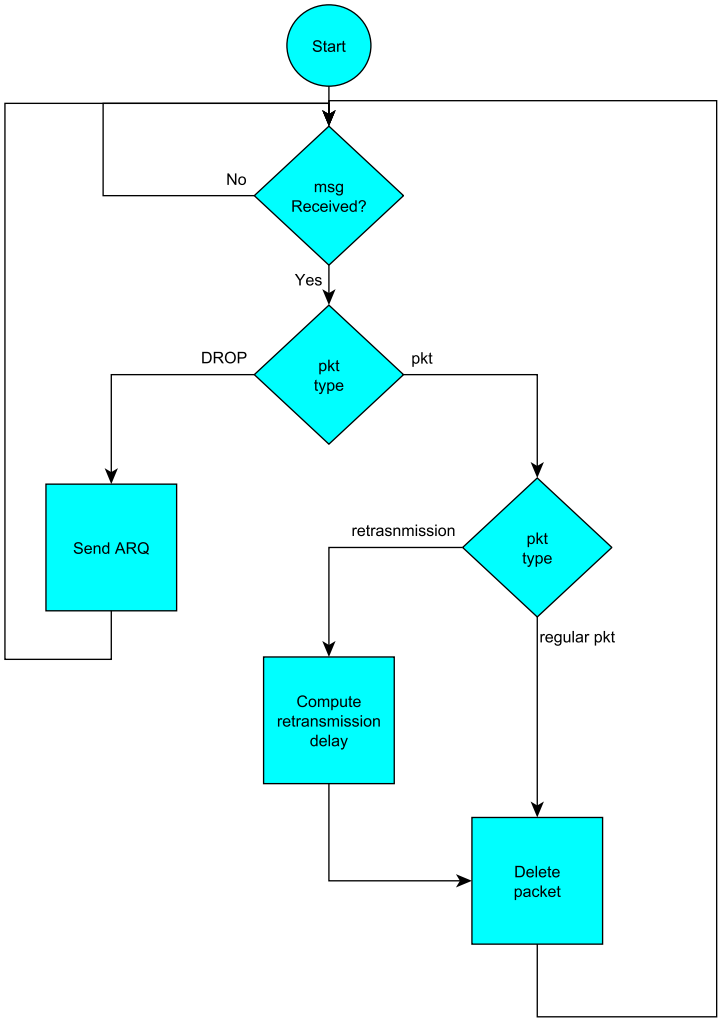


Figure 4.3: Flow diagram of the Sink module

packet, the packet is simply forwarded upstream towards the video sink. The packet errors are simulated using the packet error rate (PER) of the channels connecting the routers. The flow diagram of the Switch module is shown in Figure 4.4.

4.3.4 CacheRouter

The CacheRouter module acts as a simple switching device but also acts as a caching device that services retransmission requests. If a DROP message is received, the CacheRouter module simply forwards the packet upstream towards the video sink. If a retransmission request (ARQ) is received, the module uses the packet number associated with the ARQ to resolve the name of the packet to retransmit and looks for it in its queue. If the packet is not found in the queue, this caching router cannot service the retransmission request and forwards the ARQ packet downstream towards the next caching router. If the packet is found in the queue, a copy of that packet is made; a time stamp is associated to the copy of the packet and then is retransmitted towards the video sink. If, instead, the packet received is a regular packet, the caching router checks for errors. If no error is found and the packet arrives in order, the packet is inserted at the end of the queue; however, if the packet arrives out of order but without errors, the packet is inserted in its corresponding place in the queue. On the other hand, if the packet arrives in error, the packet is deleted and a DROP notification is sent to the video sink. The flow diagram of the CacheRouter module is shown in Figure 4.5.

4.3.5 MulticastRouter

The MulticastRouter module is the multicast equivalent of the Switch module. This module behaves exactly the same as the switch module to the arrival of packets; it merely forwards the packets to the right output depending on the type of packet. The only difference is that the MulticastRouter replicates the packets and sends a copy through each of the output ports when the packets are to be forwarded downstream. There is only one parameter in this module:

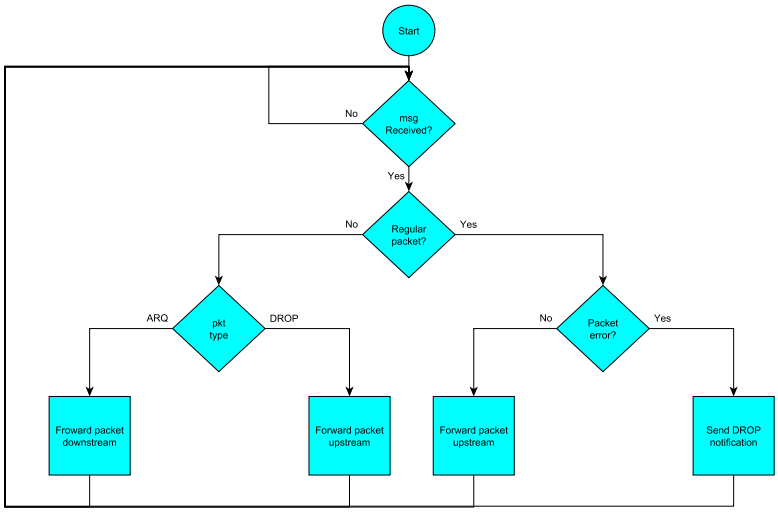


Figure 4.4: Flow diagram of the Switch module

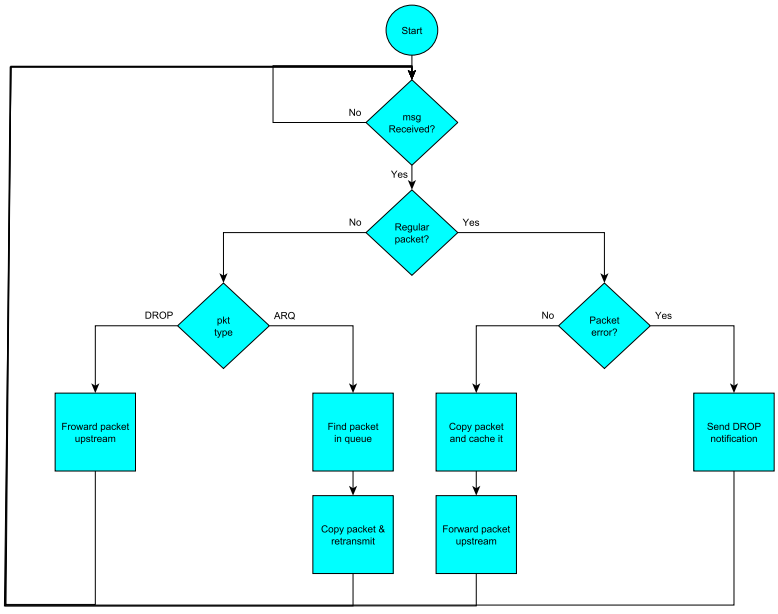


Figure 4.5: Flow diagram of the CacheRouter module

numberOfOuputs, which determines the number of output ports in the multicast router and can be modified in the Ned file. The flow diagram of the MulticastRouter can be observed in Figure 4.6.

4.3.6 MulticastCacheRouter

The MulticastCacheRouter is the multicast equivalent of the CacheRouter. This module acts as a simple packet switching device but also acts as a video packet caching device that services retransmission requests, just like the single-cast CacheRouter. The only difference between the two modules is that the multicast equivalent replicates the packets and forwards them to all of the output ports towards the multiple video sinks. There is only one parameter in this module: numberOfOuputs, which determines the number of output ports in the multicast caching router and can be modified in the Ned file. The flow diagram of the MulticastRouter can be observed in Figure 4.7.

4.4 Network Models

This is a compound module composed of all other simple modules. This module is what represents the actual network to be simulated. The network structures are defined in the NED files Unicast.ned and Multicast.ned and the network diagrams can be observed in Figure 4.8.

The unicast model consists of a typical path with a total of M=8 routers. The number of caching routers and the placement can be varied to simulate different scenarios. The vector can be varied as well. Any router can be empowered with the ability to cache video packets by modifying the NED file.

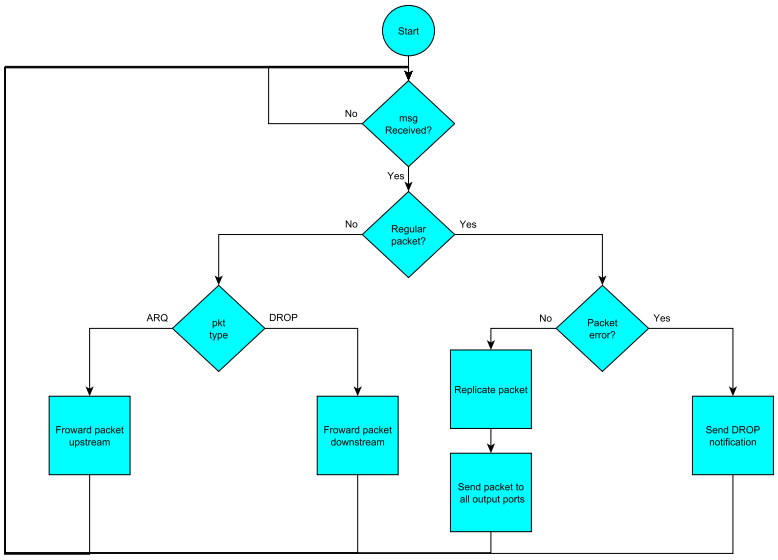


Figure 4.6: Flow diagram of the MulticastRouter module

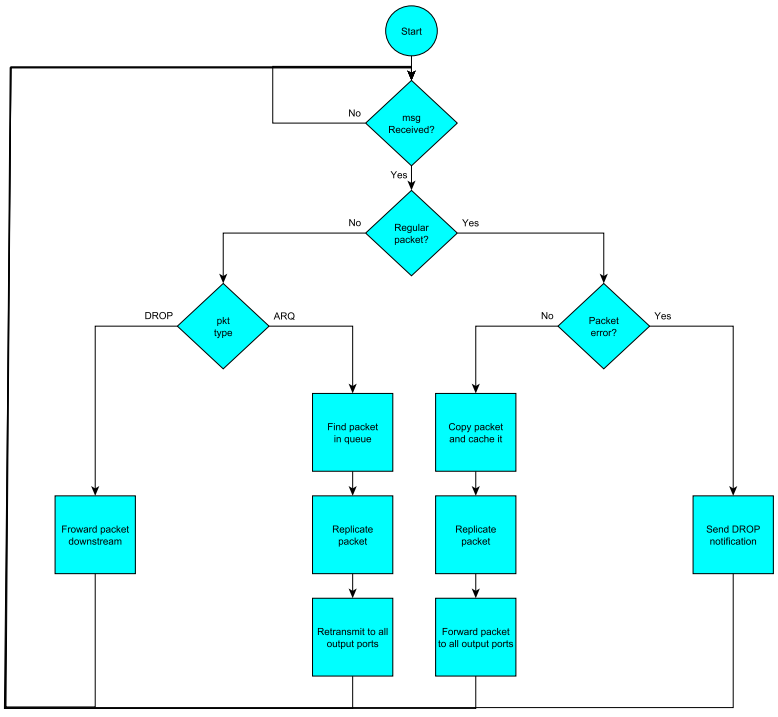


Figure 4.7: Flow diagram of the MulticastCacheRouter module

4.4.1 Multicast Analysis

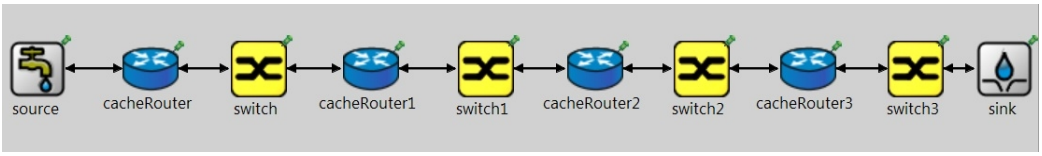
The multicast model consists of a network with a total of M=8 routers arranged in a tree topology with a trunk of three routers and a three branches. The first branch has two routers and a video sink, the second branch has one router and a video sink, and the third branch only has a video sink. Any router can be empowered with the ability to cache video packets by modifying the NED file.

In the multicast analysis the video source generates packets which travel through the trunk and then are replicated in the last router of the trunk, which is a multicast router. Then, a copy of the packet is received at each of the branches video sinks. If a packet is lost in the trunk, the packet is retransmitted from the nearest caching router in the trunk or from the source and is delivered to all of the video sinks. On the other hand, if a packet is dropped at one of the branches and the retransmission request is serviced by a caching router in that branch, the packet is only retransmitted to the video sink of that branch. However, if the retransmission request cannot be serviced by a caching router in that branch, the packet is retransmitted by a caching router in the trunk or by the source and is sent to all video sinks. Duplicate packets received at any video sink are discarded. Each video sink records the retransmission delays of the packets it receives and computes their own average retransmission delay. An avenue for future work is to develop an analytical model of the multicast scenario and compute a global average retransmission delay to find the optimal placement of caching routers.

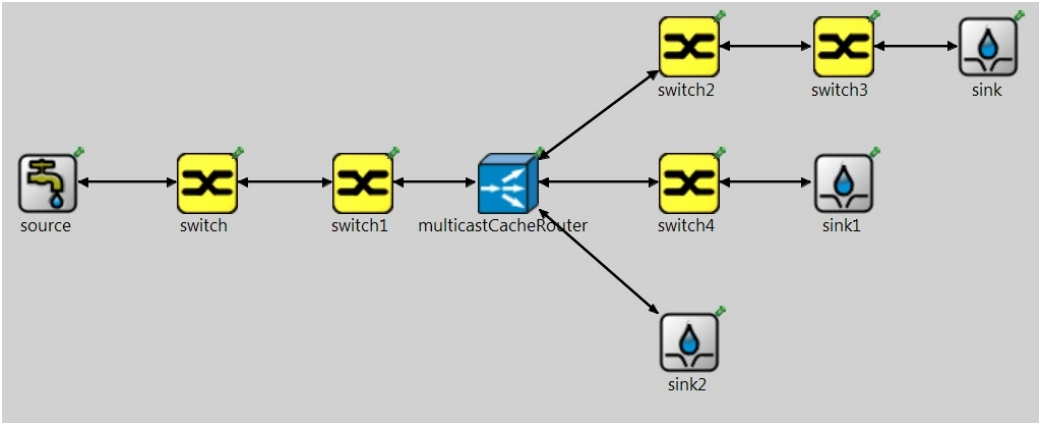
4.4.2 Parameters

There are two important parameters in the simulation model: the delay distribution vector (τ) and the drop probability (Φ). Both parameters are simulated in the channel. The tau distribution can be modified directly in the models network description file (NED), in the connections section, by modifying the delay parameter of the channel. Similarly, the drop probability can be modified directly in the NED file by modifying the packet error rate (PER) parameter of the channel.

Alternatively, the phi value can be varied in the initialization file (omnetpp.ini) to allow the simulation of scenarios with different drop probabilities. To illustrate, if one wants to model a network with drop probability of 10% the initialization file omnetpp.ini would look like Figure 4.9.



a) Unicast network diagram



b) Multicast network diagram

Figure 4.8: Network diagram of the simulation models.

```
[General]
network = NetworkModel
Config [Phi10]
description = "Phi = 10%"
**.Phi = 0.1
**.phi_index = 1
```

Figure 4.9: Sample Initialization File

CHAPTER 5

RESULTS ANALYSIS

A numerical analysis was conducted using both the analytical model and the computer simulation model to test the performance of the optimal placement of caching routers against the performance of the other typical placements. Since the mathematical programs for both energy consumption and retransmission delay are identical in structure and can be solved with the same dynamic programming solution, in this analysis I only mention the average retransmission delay as a metric to test the performance of the optimal solution; however, the reader should keep in mind that the use of caching routers to service retransmission requests results in the reduction of the average energy consumption to retransmit a packet that is directly proportional to the reduction in the average retransmission delay.

The first experiment consisted on using the behavioral model to validate the average retransmission results obtained with the analytical model. For this experiment, I set the total number of routers to M=8, the number of caching routers to N=4, the caching router placement to 10101010 (a 1 represents a caching router and a 0 represents a regular router), and the delay vector to =(44,47,64,67,9,83,21,36,87). The drop probability was varied in the following set =1%, 5%, 10%, 20%, 40%). The expected retransmission delay was measured and compared against the computed results. Table 5.1 shows the measured expected retransmission delays converge to their computed values with little error.

The next experiment consisted on observing the effects of different delay distributions and different drop probabilities on the optimal placement of caching routers. The different delay distributions (τ vectors) used in this experiment are shown in Table 5.2. For this experiment the total number of routers was set to M=8 and the drop probability to 10%. Most of the resulting optimal caching router vectors result directly from the delay pattern. For instance, a

Table 5.1: Average retransmission delay measured vs. calculated

Φ	$\mathbb{E}d$ **meas.**	$\mathbb{E}d$ **calc.**	**Error**
1%	48.02	48.79	1.6%
5%	216.10	218.82	1.2%
10%	379.56	383.86	1.1%
20%	600.06	601.12	0.1%
40%	792.99	794.84	0.2%

network with higher delays in the core (heavy middle), rather than the edges, yields an optimal placement of caching routers with most caching routers in the core. Similarly, a typical path with the heavy upstream delay pattern yields a caching router vector with most caching routers lumped upstream. The exponential delay distribution results in a caching router placement with a tendency to place caching routers towards the video sink, where the delays are larger. Correspondingly, the uniform delay distribution yields a uniform placement of caching routers in the typical path. Other results, however, are not as intuitive as the previously mentioned. For example, one would expect caching routers placed on both edges if the typical path has a delay distribution of higher delays on the edges. On the contrary, the resulting placement with a heavy peripherals delay distribution has a tendency to lump the caching routers upstream with some caching routers placed in the core. This result shows us that the placement is not only affected by the delay vector , but also by the drop probability . The resulting optimal caching router placements can be observed in Table 5.3.

Another experiment consisted on observing the effect of the optimal placement of caching routers, compared to the typical placements, on the average retransmission delay. For this experiment I fixed the delay distribution to =(44,47,64,67,9,83,21,36,87) and the drop probability to =10%. The average retransmission delay without any caches is 521.7 sec. Table 5.4 shows the average retransmission delay with N=2,,7) for the different placements. Figure 5.1 shows a graphical representation. Using only 2 caching routers with the best placement yields a reduction of 32.5% in the retransmission delay compared to not using caching routers at all. Similarly, using

Table 5.2: Transmission delay (τ) vectors

Heavy peripherals	80 80 5 5 5 5 5 80 80
Heavy middle	5 5 5 80 80 80 5 5 5
Heavy downstream	5 5 5 5 80 80 80 80 80
Heavy upstream	80 80 80 80 80 5 5 5 5
Interleaved	80 5 80 5 80 5 80 5 80
Inc powers 2	2 4 8 16 32 64 128 256 512
Dec powers 2	512 256 128 64 32 16 8 4 2
Inc powers 10	0.001 0.01 0.1 1 10 100 1000 10000 100000
Dec powers 10	100000 10000 1000 100 10 1 0.1 0.01 0.001
Inc powers 1	10 11 12 13 14 15 16 17 18
Inc powers 1	18 17 16 15 14 13 12 11 10
Exponential	random('expo',100,1,9)
Normal	random('norm',100,1,9)
Poisson	random('poiss',100,1,9)
Uniform	random('unif',20,200,1,9)

Table 5.3: Optimal caching router placement vectors for different transmission delay vector(τ) distributions in a system of 8 routers and drop probability of 10%

tau distribution	**N=2**	**N=3**	**N=4**	**N=5**	**N=6**	**N=7**
Heavy peripherals	11000000	11001000	11010100	11110100	11111100	11111110
Heavy middle	00010100	00011100	01011100	11011100	11111100	11111110
Heavy downstream	00001100	00001110	01001110	10101110	11101110	11111110
Heavy upstream	01001000	10101000	11101000	11111000	11111100	11111110
Interleaved	10001000	10101000	10101010	11101010	11111010	11111110
Inc powers 2	00001010	00010110	00011110	00111110	01111110	11111110
Dec powers 2	10100000	11010000	11101000	11111000	11111100	11111110
Inc powers 10	00000110	00001110	00011110	00111110	01111110	11111110
Dec powers 10	11000000	11100000	11110000	11111000	11111100	11111110
Inc powers 1	01001000	01010100	11010100	11110100	11111100	11111110
Dec powers 1	01001000	01010100	11010100	11110100	11111100	11111110
Exponential	00001010	00011010	10011010	10011110	10111110	11111110
Normal	01001000	01010100	11010100	11110100	11111100	11111110
Poisson	01001000	01010100	11010100	11110100	11111010	11111110
Uniform	00100100	10100100	10101100	10101110	11101110	11111110

the optimal placement of caching routers, as opposed to the worst placement, results in a 24.5% reduction in the retransmission delay. Uniform is the best performing typical placement in this case, but using the best placement still results in a significant reduction in the retransmission delay. Specifically, the optimal placement yields a 6.3% reduction compared to the uniform placement.

Using 4 caching routers and the best placement yields a 42% reduction in retransmission delay compared to not using caching routers at all. In comparison with the worst placement, the optimal placement yields a 17.4% reduction in the retransmission delay. Comparing the optimal placement with the best performing typical placement in this case, which is lumped upstream, yields a 3% reduction in the retransmission delay.

Table 5.4: Average retransmission delay for different placements

N	**Best**	**Worst**	**Average**	**Upstream**	**Downstream**	**Uniform**
2	393.501	489.646	427.133	444.697	489.646	418.631
3	374.922	456.272	404.13	406.485	456.272	401.869
4	365.285	428.991	388.131	376.25	428.991	379.27
5	357.671	401.754	376.382	373.37	399.86	377.261
6	355.662	381.12	367.386	356.608	377.261	362.519
7	354.599	363.93	360.293	354.599	362.519	362.519

The graphical representation in Figure 5.1 shows that as the number of caching routers grows, the effect of the placement of caching routers is less significant and the delays end up converging because there are fewer options where to place the caching routers. In other words, for small number of caching routers the difference in performance between the optimal placement and the rest is very significant; but as N approaches M, the difference is less significant. Still, the optimal placement clearly outperforms the rest of the placements.

To further understand the effect of the placement of caching routers and the impact of , another experiment was conducted. In this experiment the total number of routers M was varied between 8 and 16. The delay vector was randomly selected from the Gaussian distribution

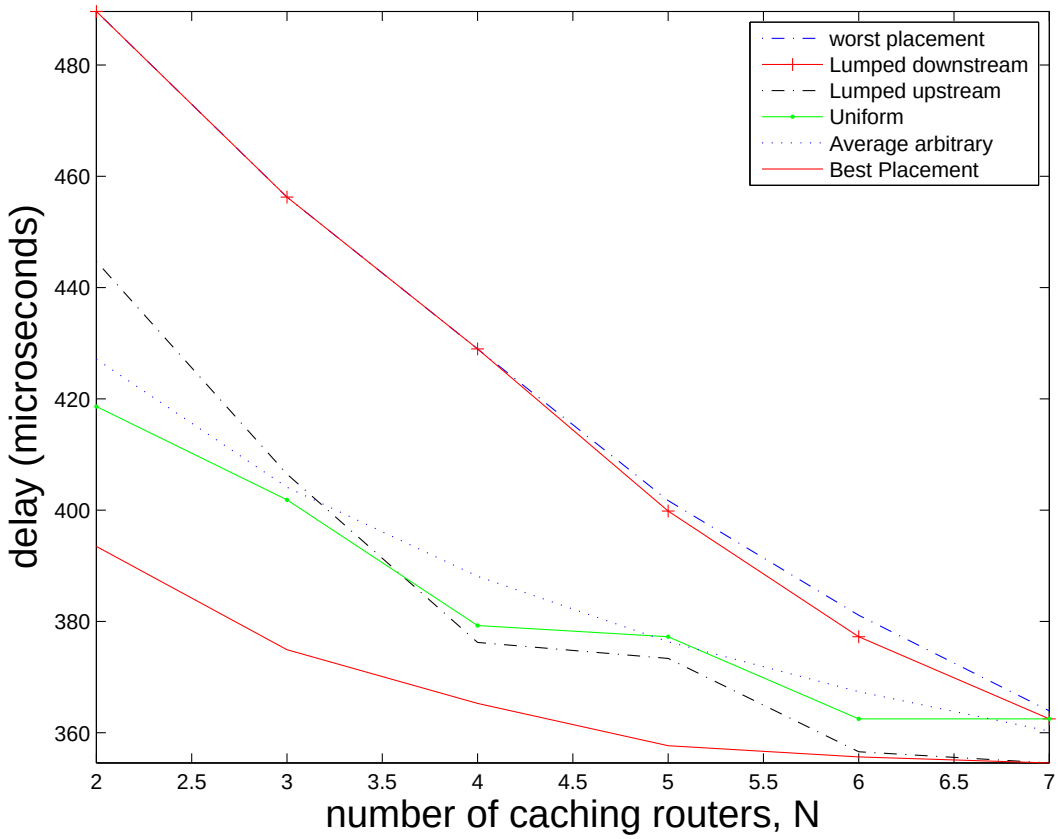
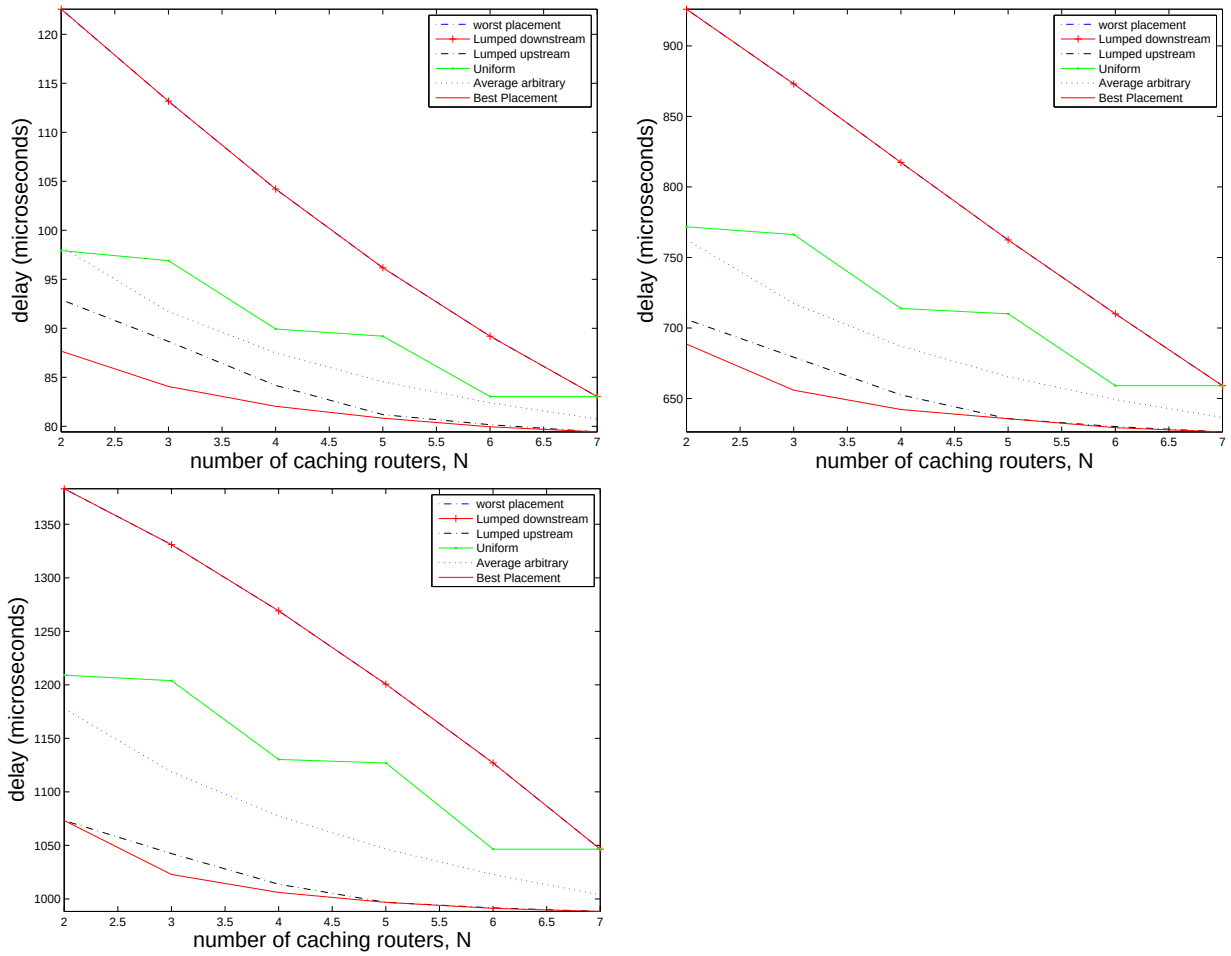


Figure 5.1: Expected total delay for a system of 8 routers using a transmission delay (tau) vector of [44 47 64 67 9 83 21 36 87] and packet drop probability of 10%.

U(20sec,200sec). In order to observe the effect of the drop probability, was varied in the following set 1%,10%,20%. Figure 5.2 shows the number of caching routers versus the expected retransmission delay for all placements in a system with 8 routers. Figure 5.3 shows the number of caching routers versus the expected retransmission delay for all placements in a system with 16 routers. One observation here is that with small number of caching routers, placing all caching routers downstream performs as bad as the worst placement. One can also observe that as increases the average retransmission delay between the optimal placement and the worst placement increases which leads us to conclude that as the drop probability increases the impact of the placement of caching routers is greater.

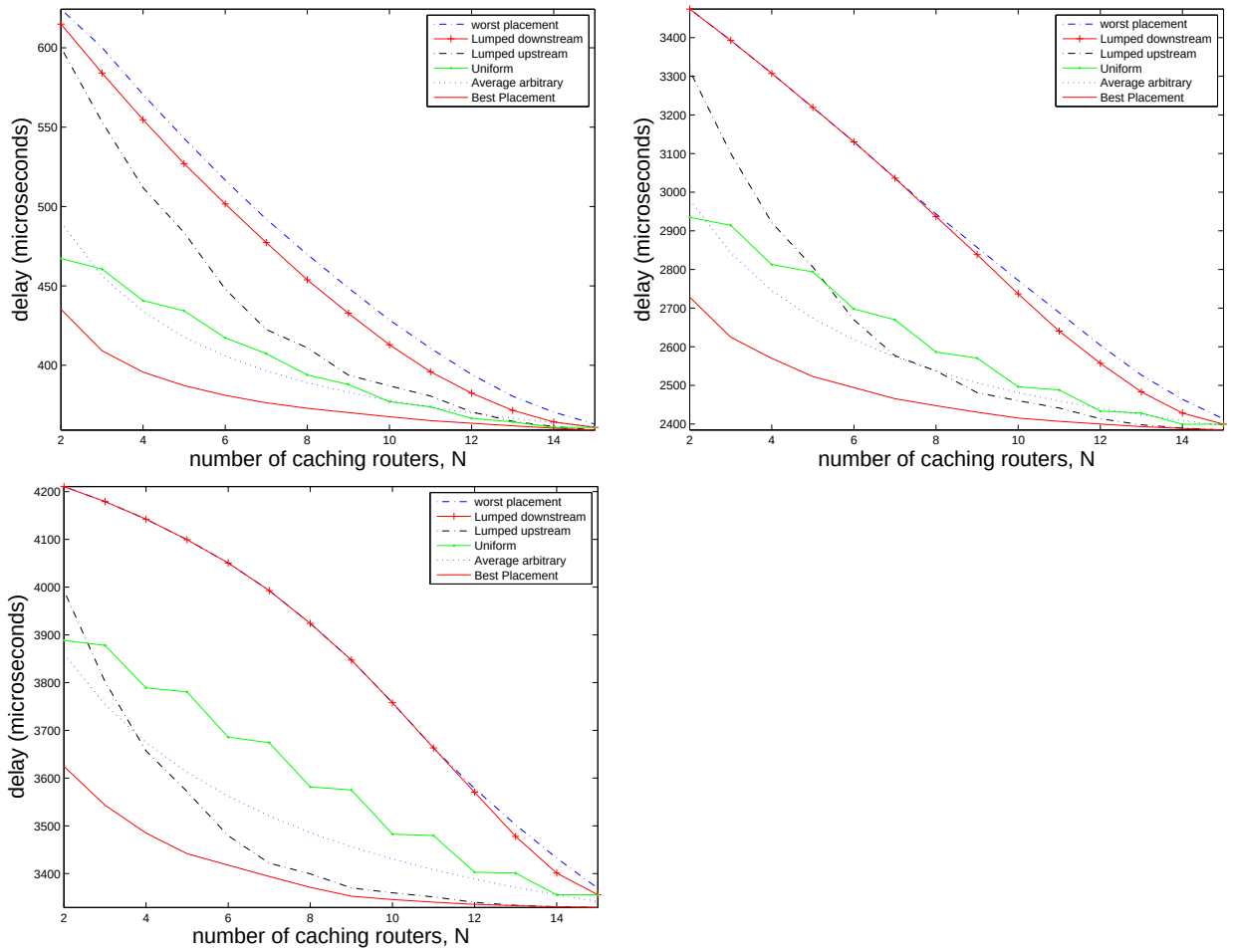


a) Ed using Phi=1%

b) Ed using Phi=10%

c) Ed using Phi=20%

Figure 5.2: Expected total delay for a system of 8 routers using a transmission delay (τ) distribution caracterized by U(20us, 200us) and packet drop probabilities of 1%, 10% and 20%.



a) Ed using Phi=1%

b) Ed using Phi=10%

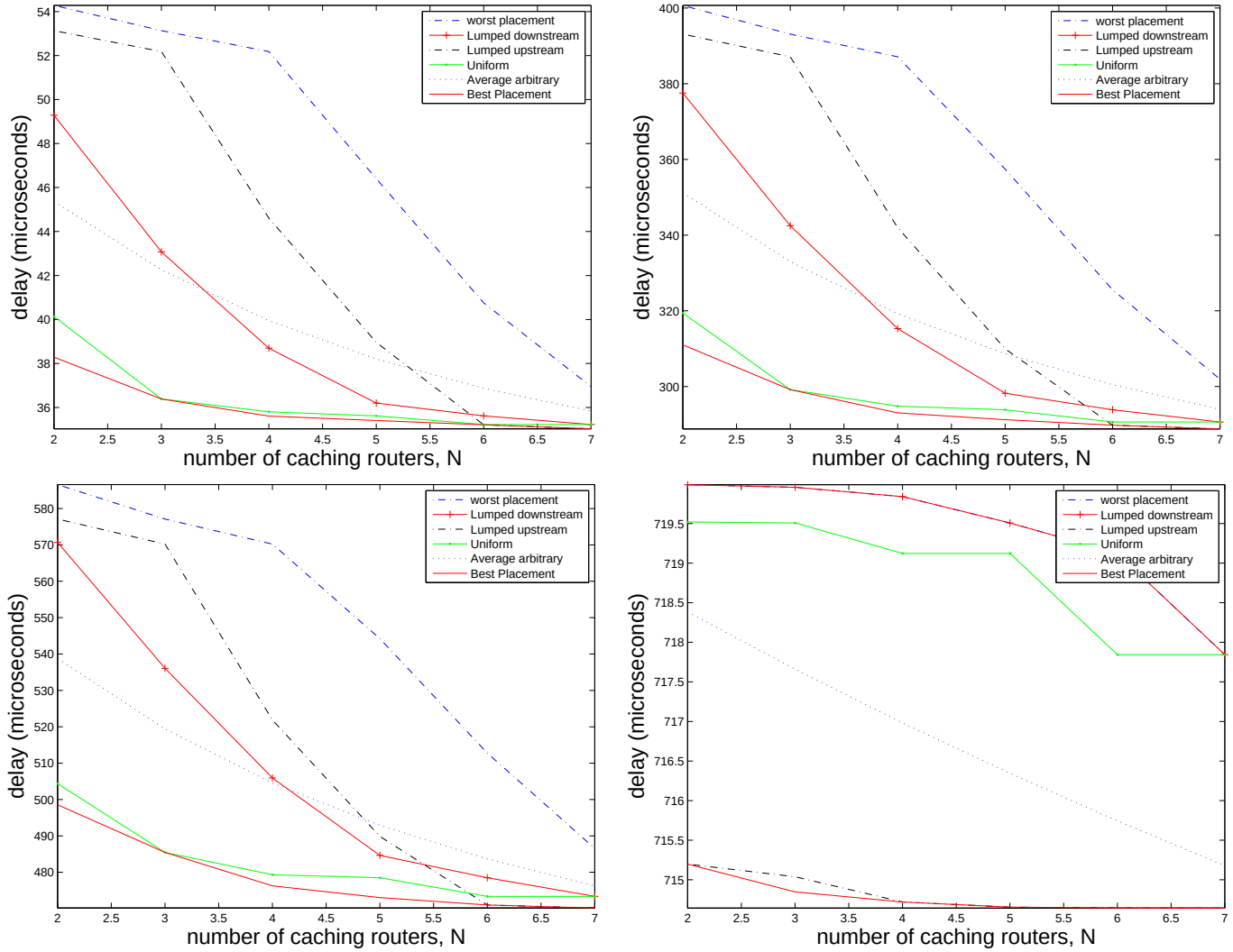
c) Ed using Phi=20%

Figure 5.3: Expected total delay for a system of 16 routers using a transmission delay (τ) distribution caracterized by U(20us, 200us) and packet drop probabilities of 1%, 10% and 20%.

Another important conclusion is that the optimal placement of caching routers can greatly reduce the cost of a typical path because using the optimal placement compared to the other placements results in a dramatic reduction in the number of caching routers required to achieve a retransmission delay below a specific value. To illustrate, one can observe in Figure 5.2 b) that if one wants to achieve a retransmission delay below 680 sec only 2 caching routers are needed if the optimal placement is used; on the contrary, 7 caching routers would be needed if the worst placement is used. Even when compared to the best performing typical placement in this case, the optimal placement results in a smaller number of caching routers. That is, 3 routers are needed to achieve a delay below 680 sec using the lumped upstream placement, while only 2 are required with the optimal placement. The difference in the number of caching routers required to achieve a certain retransmission delay becomes even greater with larger number of total routers (M) in the typical path. For instance, if an average retransmission delay below 2700 sec is desired, 11 caching routers are required with the worst placement, 6 with the lumped upstream placement and only 2 with the optimal placement.

Yet another experiment was conducted. In this case to test the performance of the optimal placement against that of the typical placements in a network that resembles a typical path on the internet. The delay distribution was set so that there are small hops on the edges and considerably larger hops in the core of the network. The total number of routers was varied between 8 and 16 and the drop probability was varied in the set 1%,10%,80%. Figure 5.4 shows the retransmission delay for all the placements. In this experiment we can observe that with very high drop probability, 80% in this case, the lumped upstream placement performs just as good as the optimal placement. This tells us that with large packet drop probabilities, packets are more likely to be dropped at the first routers and not make it to the last routers; therefore, the lumped upstream placement will be in fact the optimal solution. Another important observation in this experiment is that clearly the optimal placement outperforms all of the typical placements and the difference is accentuated with higher number of routers, with the exception of lumped upstream at very high drop probabilities in which case both perform the same. To

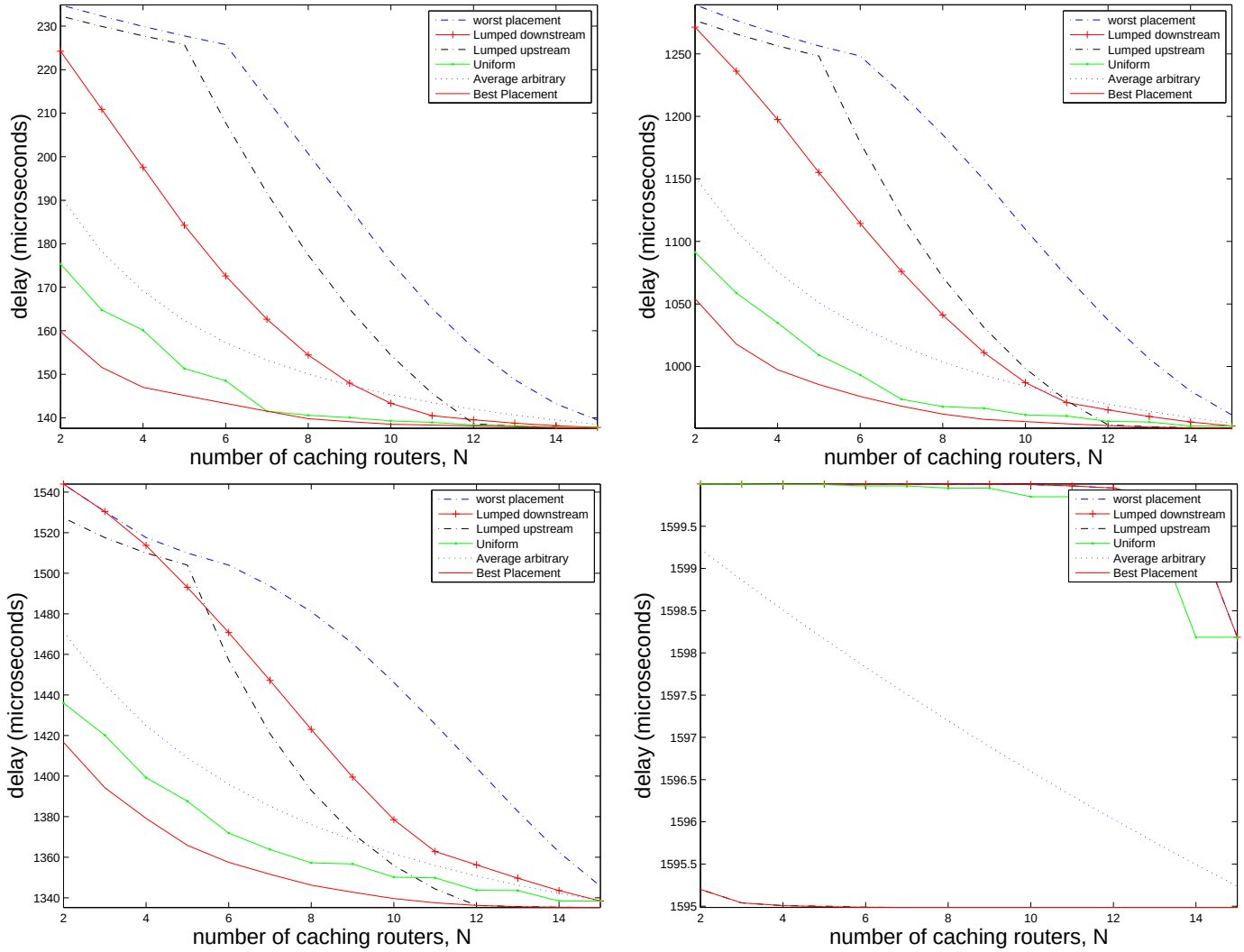
illustrate, Figure 5.5 shows that with a drop probability of 1% the best placement results in a 50% reduction in average retransmission delay compared to the worst placement. Moreover, the best placement yields a 10% reduction in the retransmission delay when comparing it to the best performing typical placement, in this case the uniform placement. Even though the uniform placement performs admirably in this experiment that simulates a typical path on the internet, our best placement still performs significantly better than the uniform placement and the rest of the typical placements.



a) $\Phi = 0.01$ b) $\Phi = 0.1$

c) $\Phi = 0.2$ d) $\Phi = 0.8$

Figure 5.4: Average retransmission delay for $M = 8$ with a delay pattern of 10 times larger delay in the core than the edge.



a) $\Phi = 0.01$ b) $\Phi = 0.1$

c) $\Phi = 0.2$ d) $\Phi = 0.8$

Figure 5.5: Average retransmission delay for $M = 16$ with a delay pattern of 10 times larger delay in the core than the edge.

CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 Summary

Several error control and recovery techniques and their implications were presented in this thesis in the spirit of finding a suitable error recovery technique for video communication. The techniques are divided into four groups based on the system level at which the error control and recovery is applied. All of the techniques have their own advantages and disadvantages as well. However, a general rule to choose a proper error control and recovery technique is to take the system configuration into consideration. For instance, if the system presents long transmission error bursts, error-resilient coding techniques may not be the most appropriate; in this case controlled retransmission might do the trick. Similarly, if the system has a backward channel, clearly the best choice is an interactive error recovery technique. Moreover, transport level error control techniques are the most important because they offer the basic QoS levels. However, used on their own these techniques are not enough to ensure the delivery of high quality video. Instead, error resilient encoding techniques as well as error recovery techniques must be used in conjunction to improve the overall quality of video communication.

Automatic Repeat request (ARQ) is the error recovery technique chosen for investigation in this thesis. This technique ensures the proper delivery of each packet by retransmitting packets that were dropped in the network or packets that arrive with damaged bits. ARQ is very widely used for data communication over packet switched networks; however, for real-time video communications ARQ might not be the best choice because retransmitting lost video packets incurs an extra delay. An improvement to the ARQ technique is proposed in this thesis to make Automatic Repeat request a more attractive error recovery technique for video communication.

Generally, when ARQ is used, the network sink requests retransmission of dropped or damaged packets to the source; so, the retransmission delay is twice the distance between the source and the sink. In this thesis, the use of intermediary caching routers to service retransmission requests is investigated; the retransmission delay is effectively reduced to twice the distance from the sink to the nearest caching router containing a copy of the lost packet. The main goal is to reduce the expected retransmission delay in a video communication network where the capacity for caching routers is constrained. In order to reduce the retransmission delay, the effect of the placement of routers with caching ability is investigated. Moreover, as a consequence of reducing the retransmission distance, the energy spent on retransmitting video packets is reduced as well. In order to characterize the effect of the placement of caching routers, the average retransmission delay and the average energy consumption were modeled with mathematical programs that select the optimal placement of caching routers to minimize the expected retransmission delay and the expected energy consumption respectively.

Both mathematical programs are identical in structure and a solution method is developed to solve both programs. The solution method is a dynamic program that takes as input the total number of routers in the network, the number of caching routers available, the packet drop probability, and the average packet transmission delay between each router and its neighbor; and yields the optimal placement of the caching routers which minimizes the average retransmission delay or the average energy consumption. An extensive numerical analysis was conducted to measure the performance of the optimal placement by comparing it with the worst, the average of all placements, lumping all caching packets upstream or downstream, and spreading the caching routers uniformly across the network.

Furthermore, a computer simulation model was developed to validate the results obtained in the numerical analysis conducted on the analytical model. The computer simulation model was developed with the C++ programming language aided by the discrete event simulation library OMNET++. The results found in the analysis conducted on the simulation model demonstrate

the accuracy of the previous results obtained with the analytical model and strengthen the conclusions. Additionally, the computer simulation model was used to extend the analysis performed with the analytical model to a multicast scenario. An avenue for future work is to use the information obtained with the simulation model and develop an analytical model for the multicast scenario to conduct a numerical analysis.

6.2 Conclusions

The use of intermediary caching routers to service retransmission requests in a video communication network with the purpose of reducing the expected retransmission delay as well as the expected energy consumption was investigated in this thesis. Both an mathematical program and a simulation model, that characterize the optimization problem that yields the optimal set of routers to empower with the ability to cache video packets to minimize the average retransmission delay and the average energy consumption, were developed. A dynamic programming solution was used to conduct an extensive numerical analysis to compare the performance of the optimal placement of caching routers to the worst, average, and typical plcements of cachign routers.

The model parameters (M, N, Φ, τ), where M is the total number of routers in the typical path, N is the number of caching routers available, Φ is the packet drop probability, and τ is the transmission delay vector, were varied to conduct several different experiments. The different experiments were used to explore the impact of the optimal placement of caching routers on the expected retransmission delay. Moreover, the optimal placement was compared against other typical placements to get a sense of how beneficial it would be to use the proposed optimal placement of caching routers in a typical path of a video communication network.

The numerical analysis shows that the optimal placement of caching routers has a significant impact on the expected retransmission delay. For instance, using 4 caching routers and the best

placement on a typical path yields a 42% reduction in retransmission delay compared to not using caching routers at all. In comparison with the worst placement, the best placement yields a 17.4% reduction in the retransmission delay. Also, in a typical path resembling a path on the internet, the optimal placement yields a 10% reduction in the average retransmission delay compared to the best performing typical placement, the uniform placement. Clearly the optimal placement of caching routers outperforms the other typical placements.

Furthermore, the analysis shows that the optimal placement can significantly reduce the number of caching routers required to keep the average retransmission delay below a desired value. For example, if an average retransmission delay below 2700 sec is desired on a typical path in a network with 16 routers, 11 caching routers are required with the worst placement, 6 with the lumped upstream placement and only 2 caching routers are required with the optimal placement. These results show that the optimal placement can significantly reduce network cost by reducing the number of caching routers required to meet a desired average retransmission delay performance.

The aforementioned conclusions and the rest of the numerical analysis clearly indicate that the optimal placement of caching routers significantly reduces the average retransmission delay and the average energy consumption for retransmission, making Automatic Repeat requests (ARQ) a more enticing error recovery technique for video communication over a packet switched network.

6.3 Future Work

An avenue for future research is to use the simulation model of the multicast scenario to obtain preliminary results and use insights from those results to develop an analytical model and a subsequent solution method. Right now we are able to compute the average retransmission delay for each of the video paths in the multicast network; however, in the future a joint optimization of multiple paths can be developed. Another possible avenue for future work is to investigate

the feasibility of applying probabilistic caching to our proposed solution.

References

[1] Cisco, "Approaching the zettabyte era," *Cisco Visual Networking Index*, 2008.

[2] Y. Wang, J. Ostermann, and Y.Q. Zhang, *Video Processing and Communications*, Prentice Hall, 2001.

[3] I. Bouazizi and M. Gunes, "Selective proxy caching for robust video transmission over lossy networks," in *International Conference on Information Technology: Research and Education (ITRE)*, August 2003, pp. 69–73.

[4] I. Bouazizi, "Size-distortion optimized proxy caching for robust transmission of mpeg-4 video," in *International Workshop on Multimedia Interactive Protocols and Systems*, November 2003, number 2899, pp. 131–142.

[5] M. McGarry, J. E. Hernandez, R. Ferzli, and V. Syrotiuk "Minimizing Video Retransmission Delay and Energy Consumption with Caching Routers," in *IEEE International Conference on Multimedia and Expo (ICME)*, July 2012.

[6] J. Lacan, V. Roca, J. Peltotalo, S. Peltotalo "Reed-Solomon Forward Error Correction (FEC) Schemes," in *IETF, Network Working Group, RFC5510*, April 2009. [Online]. Available at:
http://tools.ietf.org/html/rfc5510

[7] Tood K. Moon *Error Correction Coding*, New Jersey: John Wiley & Sons, 2005.

[8] A. Glavieux, P. Titimajshima "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1 ," *IEEE International Conference on Communications (ICC) 1993*, vol. 2, pp. 1064–1070, April 2006.

[9] S. H. Chan, X. Zheng, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Video Loss Recovery with FEC and Stream Replication," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 370–381, April 2006.

[10] Q. Zhu, Y. Wang, and L. Shaw, "Coding and Cell-Loss Recovery in DCT-Based Packet Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 3, June 1993.

[11] Yao Wang and Qin-Fan Zhu, "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.

[12] J. Kim, R. Mersereau, and Y. Altunbasak, "Error-Resilient Image and Video Transmission Over the Internet Using Unequal Error Protection," *IEEE Transactions on Image Processing*, vol. 12, no. 2, February 2003

[13] T. Wiegand, N. Farber, K. Stuhlmuller, and B. Girod, "Error-Resilient Video Transmission Using Long-Term Memory Motion-Compensated Prediction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, June 2000.

[14] P. Xia, G. Chan, and X. Jin, "Optimal Bandwidth Assignment for Multiple-Description-Coded Video," *IEEE Transactions on Multimedia*, vol. 13, no. 2, April 2011.

[15] S. Aign, and K. Fazel, "Temporal & Spatial Error Concealment Techniques for Hierarchical MPEG-2 Video Codec," *Proceedings of the 1995 IEEE International Conference on Communications*, vol. 3, pp. 1778-83, June 1995.

[16] W. Wada, "Selective recovery of video packet loss using error concealment," *IEEE Journal on Selected Areas of Communication*, vol. 7, pp. 807814, June 1989.

[17] G. Fairhurst, L. Wood "Advice to link designers on link Automatic Repeat reQuest (ARQ)," in *IETF, Network Working Group, RFC3366*, August 2002. [Online]. Available at: *http://www.ietf.org/rfc/rfc3366.txt*

[18] J. Kurose, K. Ross *Computer Networking*, Addison-Wesley, 2010.

[19] Q. F. Zhu, "Device and method of signal loss recovery for real-time and/or interactive communications" *U.S. Patent 5 550 847*,Aug. 1996.

[20] T.P. Van, "Proactive adhoc nodes for real-time video," in *IEEE Singapore International Conference on Communication Systems (ICCS)*, October 2006, pp. 1–5.

[21] T.P. Van, "Efficient relaying of video packets over wireless ad hoc devices," in *IEEE Annual Wireless and Microwave Technology Conference (WAMICON)*, December 2006, pp. 1–5.

[22] Z. Li, X. Zhu, A.C. Begen, and B. Girod, "IPTV Multicast With Peer-Assisted Lossy Error Control," in *IEEE Transactions on Circuits and Systems for Video Technology*, March 2012, pp. 434–449.

[23] Xiaowei Zhang and N. F. Maxemchuk, "A generalized energy consumption analysis in multihop wireless networks," *2004 IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 3, no. 1, pp. 1476–1481, March 2004.

[24] "OMNET++ User Manual," Available at: *http://www.omnetpp.org/doc/omnetpp33/manual/usman.html*.

CURRICULUM VITAE

Jesus Enrique Hernandez was born in Juarez, Chihuahua, Mexico. The first son of Luis Hernandez and and Ana Leon, he obtained a bachelor's degree in electrical engineering at The university of Texas at El Paso. While pursuing a bachelor's degree, he received the Benito Juarez scholarship as well as the Jimmy and Yolanda Janacek scholarship fund. In the Fall of 2010, he entered the Graduate School at the same institution to pursue a Master's degree in computer engineering. While pursuing a master's degree, he worked as a teaching assistant for the Digital Systems Design Laboratory. He also worked as a research assistant in the Communication Networks Laboratory. During his time at the communication networks research lab, he contributed to the publication of *Minimizing Video Retransmission Delay and Energy Consumption with Caching Routers* in the IEEE International Conference on Multimedia and Expo (ICME 2012) and *Accuracy of Video Frame size Forecasting* in the 2012 IEEE International Conference on Electro/Information Technology (EIT 2012). Also, he presented the article *Least Sensitive (Most Robust) Fuzzy Exclusive OR Operations* in the 2011 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS) and won the Outstanding Student Paper Award.

Permanent Address: Calle Del Lago 7417, Fuentes del Valle
Cd Juarez, Chihuahua, Mexico. CP. 32500