A Dependency-based Strategy for Handling ATM Cell Loss in MPEG-2 Transport Streams

A Thesis

Presented to the

Department of Computer Science

Brigham Young University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Dean L. Scoville

July 1996

This thesis by Dean L. Scoville is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the thesis requirement for the degree of Master of Science.

Mark J. Clement, Committee Chairman

Evan L. Ivie, Committee Member

R. Alan Whitehurst, Committee Member

Date

Scott N. Woodfield, Graduate Coordinator

Acknowledgments

I would like to express my deep appreciation to several people for their friendship and support throughout my graduate school experience. Foremost, I would like to thank Dr. Evan Ivie, who not only encouraged me to come back to school and get a Masters Degree when I expressed interest, but also supported me along the way by giving me jobs as a teaching and research assistant and providing encouragement when it seemed like I would never finish.

I would also like to thank my advisor, Dr. Mark Clement, who got me started on a topic that I have thoroughly enjoyed when he introduced me to MPEG in one of his research meetings. Mark always saw to it that I had the supplies I needed, even when the standards documents were quite expensive! He provided me with direction to my research and gave me one of my greatest incentives when we submitted a paper that was accepted to a research conference in Nashville this past Spring.

I would also like to express appreciation to Dr. Whitehurst for stepping in at the last minute with only two weeks notice when a previous committee member was unable to attend my scheduled defense. Thanks to Dr. Whitehurst I was able to reach my goal of graduating this Summer.

Last, but by no means least, I would like to thank my family and many friends who have made my stay in Provo and Utah a very enjoyable experience in spite of all the work I have had to do at school. I couldn't have done it alone, and express my thanks to all. Thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Network Errors and Error Recovery

Many types of errors can occur when transmitting data across a network. Noise can introduce bit errors into data streams, thereby corrupting data and causing loss of synchronization; congestion can cause buffers in routers, bridges, and switches to overflow and drop data; routing tables can overflow or develop inconsistencies with tables in other routers; links can go down; network devices can fail; contention-based access protocols such as CSMA/CD can produce collisions that require delays and retransmissions; and any number of other errors can occur. These and similar network errors result in lost, corrupted, or delayed data. Some networks are more reliable than others, but whenever data is sent across a network there is a chance the data will not successfully reach its destination.

Networks and applications can provide various levels of error prevention, detection and recovery. In connection-oriented networks, admission schemes often limit the number of allowable connections to help prevent network buffer overflow. Link layer protocols can provide integrity across network links; transport layer protocols can provide in-order end-to-end delivery of data by buffering, re-ordering and retransmitting data. Applications can employ forward error correction, retransmission, or error concealment techniques. Each of these recovery methods has an associated cost, and some schemes are better suited for particular applications and environments than others.

Many communication protocols, such as TCP, address the problem of error recovery by retransmitting lost or corrupted packets. For many applications, including those with real-time constraints, limited feedback, or multiple receivers (such as multicast or broadcast applications), retransmission is often impractical [15]. Even in cases where retransmission is possible, by adding more traffic to near-capacity network queues, retransmission can actually compound the problems it is trying to fix—resulting in increased congestion, more packets being dropped from the network, and even more degraded throughput.

1.2 Multimedia Data and ATM Networks

Video and multimedia data transmissions consume large amounts of network bandwidth and resources. Even compressed video, such as MPEG, require multi-megabit per second bandwidths. When real-time display is involved, stringent time constraints accompany such transmissions. Due to ATM's quality of service guarantees and high bandwidth, ATM networks are considered promising media for multimedia data transmissions. Because optical fiber is typically used for constructing ATM networks, their succeptability to noise-induced bit errors is low. However, congestion and other factors can cause ATM cells to be dropped by the network. Because retransmission is usually impractical for real-time video applications, other error recovery methods must be considered when attempting to recover the data in those lost cells.

This thesis explores one possible solution to the cell loss problem that can potentially increase the effective throughput of MPEG and other applications that contain a high degree of internal data dependencies. MPEG video data, for example, includes small headers and other fields upon which large portions of picture data depend. If the headers are not received correctly, entire pictures may be lost. By taking note of the data stream dependencies and systematically transmitting the more-critical elements of the stream redundantly, the likelihood that the remainder of the stream will be useful should increase.

1.3 Thesis Statement

When transmitting data over unreliable media in environments where constraints preclude the retransmission of data, the adverse effects of lost data can be decreased for certain datastreams containing internal dependencies by redundantly transmitting critical parts of the datastream. This approach is especially applicable to datastreams containing elements so heavily dependent upon other parts of the datastream that they are of no use to the receiver unless the more-critical parts upon which they depend are also received correctly. This concept can be applied to MPEG video streams being carried in MPEG-2 systems transport streams over ATM networks.

1.4 Overview of Thesis Work

This thesis examines the dependencies in MPEG video (see Chapters 2 and 3) and MPEG-2 systems transport streams (see Chapter 4) and uses them as the basis for a data replication scheme for ATM networks. Although other video codecs besides MPEG-2 and other networks besides ATM are in use today, these were chosen because of their wide acceptance and ability to meet the requirements of bandwidth-intensive broadcast-quality multimedia transmissions.

The replication scheme developed in this thesis has been given the name "DART", which stands for Dependency-based Approach for Redundant Transmissions. The basic components of the DART system are shown in Figure 1.1. A preprocessor at the video source evaluates video dependencies and prioritizes individual video elements. Because MPEG video streams contain a lot of internal dependencies, the preprocessor uses the dependency information to select the most important data elements for replication. The preprocessor generates appropriate redundant video streams, multiplexes the normal and redundant video data, and transmits them over the network. A post-processor at the destination attempts data recovery on the received video streams before passing the result along to the normal MPEG decoder. The presence of redundant data in the data stream should improve the quality of the video data passed to the decoder.

As part of the thesis work, two public domain programs have been modified and a third program has been written. The modified programs are the MPEG Systems Group MPEG-2 video decoder and the NIST (National Institute of Science and Technology) ATM network simulator.

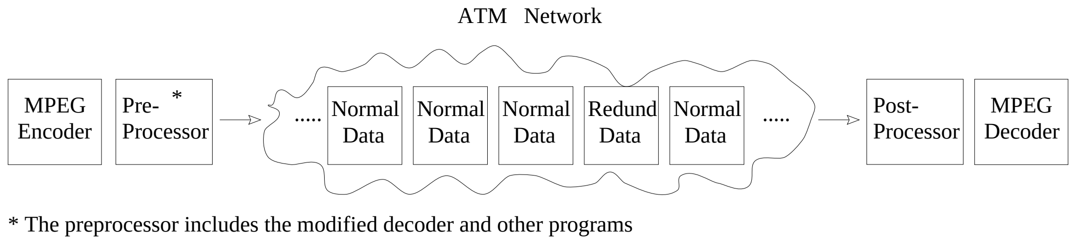

Figure 1.1: *This figure shows the basic components of the DART data replication system. A pre-processor at the video source generates the appropriate redundant data, the normal and redundant data are multiplexed on the transmission line, and a post-processor attempts to recover lost data before passing it along to the decoder.*

The MPEG Systems Group MPEG-2 video decoder has been modified to gather dependency data from MPEG-1 and MPEG-2 videos as part of the preprocessor function. Such data could perhaps more easily be gathered at video encoding time, but MPEG encoders are very CPU-intensive and fast encoders are expensive. Further, the sample clips from actual MPEG movies and commercials available on the Internet provide realistic source material for this work, and can be used as input to the decoder.

The program that was written entirely for this thesis performs many of the other preprocessor and postprocessor functions mentioned earlier in this section. As a result, the program is referred to throughout this thesis as either the "preprocessor" or "postprocessor" program. The preprocessor program packages MPEG-1 or MPEG-2 video data into MPEG-2 systems transport streams, replicates the more-critical data, segments the transport streams into ATM cells, and creates a cell transmission schedule for sending the MPEG data across an ATM network. It determines how many cells are required for the data, reserves space for necessary headers, and keeps track of which cells contain which parts of the normal and redundant data streams. Although it maintains a list of which data goes into each cell and determines when the cells should be sent, the current version of the preprocessor does not actually load the data into the cells. Instead, it produces a cell transmission schedule to be used by the ATM simulator, and a cell data map for analyzing simulation results.

The NIST ATM simulator has been modified to accept the preprocessor program's cell transmission schedules as input. After simulating the transmission of cells across an ATM network, the simulator now produces a list of lost cells at each MPEG destination. The simulator's lost cell list and the preprocessor's cell data map are then fed back as input to the postprocessor program, which gathers statistics about the quality of the resulting video. The statistics from the simulation runs are then analyzed and conclusions drawn.

1.5 Overview of Thesis Document

The MPEG standards define several types of MPEG streams. Chapter 2 gives an overview of the MPEG standards and provides the reader with a basic understanding of what MPEG-2 systems transport streams and MPEG video streams actually are and where they fit into the larger picture.

Because the DART data replication scheme is based on a thorough understanding of the internal dependencies in MPEG-2 video streams, chapter 3 takes an in-depth look at MPEG-2 video dependencies. Chapter 3 includes original analysis of MPEG picture dependencies.

Before MPEG video data can be transmitted across an ATM network, it must first be loaded into MPEG systems stream packets (PES and transport stream packets). MPEG systems stream packets include timing and synchronization information for the videos. Chapter 4 describes the structure of MPEG-2 systems streams, and lists the assumptions made by the DART system when loading MPEG video data into PES and MPEG-2 systems transport stream packets.

As a last step before multiplexing the data for transmission over the ATM network, the preprocessor must segment the MPEG systems stream packets into ATM cells. A broad overview of ATM networks and a discussion of segmentation options are included in chapter 5.

Chapter 7 discusses how the DART system adds redundant video data to the MPEG-2 systems transport streams. An ATM network simulator, rather than an actual ATM network, is used to generate results for this thesis. The simulator is discussed in chapter 7, and the simulation results and conclusions are presented in chapter 8. Finally, chapter 9 discusses related and future research. Simulation data can be found in the appendixes.

Chapter 2

MPEG Overview

2.1 The MPEG Standards

The Moving Pictures Expert Group (MPEG) committee is an ISO (International Standards Organization) and IEC (International Electro-technical Commission) joint technical committee work group charged with establishing standards for the coding of moving pictures and associated audio. The MPEG committee has published or is in the process of creating several standards. Although this thesis deals mainly with the systems and video parts of the MPEG-2 standard, a brief description of the other standards is included here to provide a better understanding of their intended scope.

2.1.1 MPEG-1

The first set of MPEG standards, commonly referred to as the MPEG-1 standards, was adopted in October 1992. MPEG-1 has the official name ISO/IEC-11172 and consists of four main parts. The first part, ISO/IEC-11172-1 systems, defines a standard for multiplexing and synchronizing audio, video and other elementary data streams into a single systems stream. The second part, ISO/IEC-11172-2 video, describes a standard for compressed video streams. The third part, ISO/IEC-11172-3 audio, defines standards for compressing up to two channels of audio (stereo, or two mono channels) into a single compressed audio stream. The latest version of the fourth part, ISO/IEC-11172-4 conformance testing, was adopted in 1995 and contains conformance tests stored on a CD ROM. The MPEG-1 standards are targeted for data rates of up to about 1.5 Mbit/sec, which is roughly the data rate produced by a single-speed CD ROM player.

2.1.2 MPEG-2

The core parts of the second set of MPEG standards, commonly referred to as the MPEG-2 standards, were adopted in November 1994. MPEG-2 has the official name ISO/IEC-13818 and consists of several parts, some of which have already reached international standard status and others of which are still under development or are draft international standards (DIS).

The first part, ISO/IEC-13818-1 systems, defines a standard for multiplexing and synchronizing audio, video and other elementary data streams into a single systems stream. The standard defines two types of systems streams: a program stream (similar to the MPEG-1 systems stream) for synchronizing, multiplexing and transmitting multimedia data in relatively error-free evironments, and a transport stream for sending data over lossy or noisy media such as computer networks.

The second part, ISO/IEC-13818-2 video, describes a standard for compressed video streams. The MPEG-2 video standard is similar to the MPEG-1 video standard, but includes provisions for interlaced video and has profiles and scalability extensions for encoding videos of various sizes at multiple resolutions and levels of detail.

The third part, ISO/IEC-13818-3 audio, defines standards for compressing up to five channels of audio (left, right, center, and two surround channels) into a single compressed audio stream.

The fourth part, ISO/IEC-13818-4 compliance testing, contains compliance tests. The other parts to the MPEG-2 standard will not be enumerated here.

The MPEG-2 standards are designed to be backward-compatible with the MPEG-1 standards. MPEG-2 is designed to facilitate the compression, synchronization and transfer of broadcast-quality video and audio in high-speed applications such as HDTV and video-on-demand at data rates greater than 1.5 Mbit/sec.

MPEG-2 systems transport streams were chosen for study in this thesis because of their ability to handle bandwidth-intensive broadcast-quality video transmissions and because their status as an international standard will likely make them widely-accepted means of transmitting multimedia data in the future. Further, ATM networks' succeptibility to cell losses makes the combination of MPEG-2 over ATM a useful research topic.

2.1.3 MPEG-3

A third MPEG standard, MPEG-3, was originally planned for handling bitrates between 20 and 40 Mbit/sec. When the MPEG-2 standard was found to handle these rates sufficiently well, MPEG-3 was abandoned.

2.1.4 MPEG-4

A fourth MPEG standard, MPEG-4, is currently under development for encoding multimedia data at low bit rates (between 4800 and 64,000 bits/sec). This future standard is anticipated to be useful for such applications as mobile multimedia, video phones, video electronic mail, and sign language captioning.

2.1.5 Other Standards

In addition to the MPEG standards, there are other important standards currently being used to transfer multimedia data. One of these is the ITU (International Telecommunications Union, formerly CCITT) H.261 video codec for Audiovisual Services at multiples of 64 kbit/s data rates. The H.261 standard was completed and approved in December 1990, and is used mainly for video phone and teleconferencing applications.

2.2 MPEG Systems Streams

MPEG streams containing a single thread of homogeneous data, such as video or audio, are called elementary streams. One or more elementary streams can be multiplexed into a systems stream. MPEG systems streams also include timing and synchronization information. In MPEG-1 systems streams and MPEG-2 (systems) program streams, all of the elementary streams are grouped into a single program and are synchronized by a common system clock. In MPEG-2 (systems) transport streams, elementary streams can be grouped into any number of programs, and each program can either have its own program clock or share one with other programs.

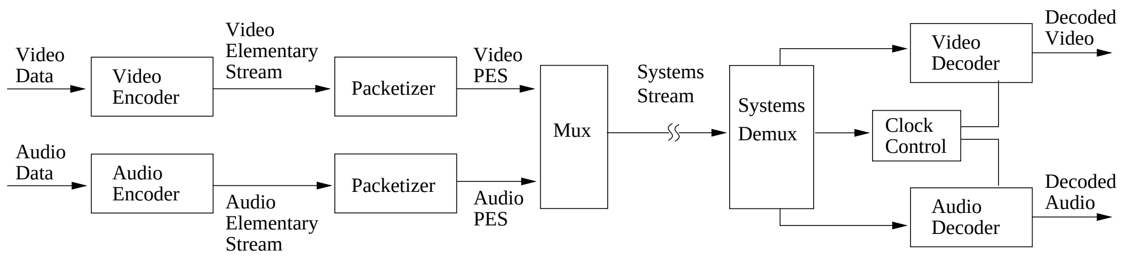

Figure 2.1: *This figure shows the typical processes of encoding, transferring, and decoding MPEG systems streams: raw video and audio data are encoded into elementary streams, which are packetized along with timing information and multiplexed into a single MPEG systems stream. The data is subsequently demultiplexed and the elementary streams decoded to reconstruct the raw data.*

As part of the systems stream building process, individual data elements within each elementary stream are packetized into PES (Packetized Elementary Stream) packets. PES packets include time stamps and other synchronization information. When MPEG video elementary streams are added to MPEG systems streams, for example, each picture in the elementary stream is placed into a PES packet. The PES packet includes a decode time stamp indicating when the picture should be sent to the video decoder. Because the transmission order and the display order of MPEG video pictures are sometimes different (see Section 2.3.1), some PES packets also include a display time stamp, which tells the decoder when the picture should actually be displayed. The overall process is shown in Figure 2.1.

2.3 MPEG Video Elementary Streams

2.3.1 Dependencies between Pictures

An understanding of the dependencies between MPEG pictures is essential to the correct operation of the DART system preprocessor. MPEG video elementary streams include three types of pictures: Intra-coded (I-pictures), Predictive-coded (P-pictures) and Bi-directionally predictive-coded (B-pictures) [12]. I-pictures and P-pictures are referred to as reference pictures because other P-pictures and B-pictures can reference information from them instead of encoding duplicate data.

Of the three types of pictures, I-pictures are typically the largest and least compressed because each encoded I-picture includes enough information to entirely reconstruct the picture without referencing any other pictures in the video. Because I-pictures are entirely self-contained, they establish bounds for error propogation between pictures. In addition, I-pictures help provide random access points into video streams. P-pictures, on the other hand, can achieve greater compression by describing parts of their pictures in terms of information already received from the previous reference picture. B-pictures can achieve even further compression by referencing information from both/either the previous and/or next reference picture. The dependencies between pictures in a sample group of MPEG frame pictures are shown in Figure 2.2. The pictures are shown in display order.

As mentioned earlier, B-pictures typically depend on information from both the

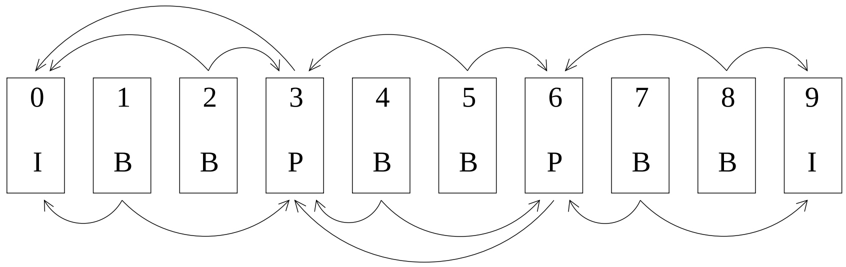

Figure 2.2: *Dependencies between frame pictures in a sample MPEG video elementary stream. Arrows point from P and B pictures to the reference pictures upon which they depend. Pictures are shown in display order.*

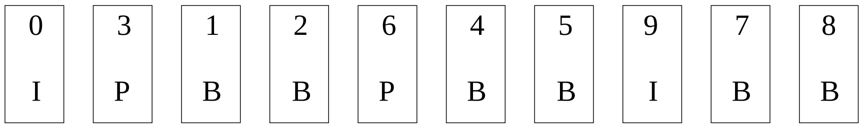

Figure 2.3: *This figure shows the transmission order of the MPEG pictures from Figure 2.2.*

previous and next reference picture. In order to have all of the information required to reconstruct a picture available at the decoder when the picture arrives, MPEG pictures are transmitted in a different order than they are displayed. Reference I-pictures and P-pictures are always transmitted before the pictures that use them. Figure 2.3 shows the transmission order for the frame pictures shown in Figure 2.2.

2.3.2 Hierarchical Structure of MPEG Video Data

The MPEG video codec (encoder/decoder) arranges video data into a hierarchy of layers: the Sequence of Pictures layer, Group of Pictures layer, Picture layer, Slice layer, Macroblock layer and Block layer. Most of layers include headers containing essential decoding and structural information about data in their own and subordinate layers [12]. If any of the higher-layer headers are lost or corrupted, it may be impossible to correctly interpret and decode data in any of the dependent layers.

An encoded video consists of a series of one or more sequences of pictures. Each sequence of pictures is comprised of a sequence header (SEQ) followed by one or more MPEG pictures, followed by a sequence end code. Group of pictures (GOP) headers may optionally precede some of the I-pictures. The upper half of Figure 2.4 shows an encoded MPEG video with at least two sequences of pictures: one of the sequences shown contains groups of pictures and the other does not.

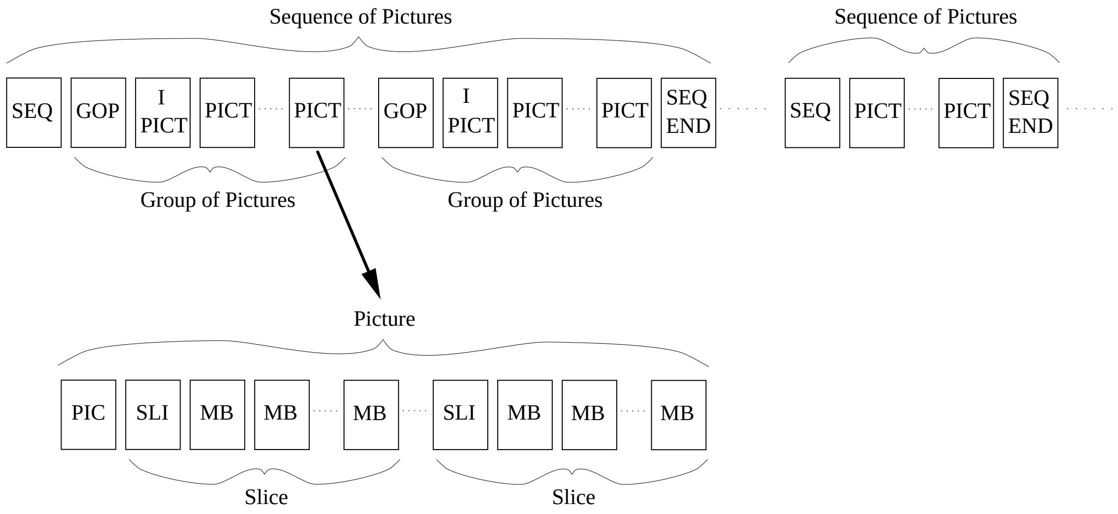

Figure 2.4: *An encoded MPEG video consists of a series of one or more Sequences of Pictures. Each picture is comprised of a picture header, slices, and macroblocks, as shown in the lower half of the figure.*

Sequence of pictures (SEQ) headers contain information common to all the pictures in the sequence, such as horizontal and vertical size, aspect ratio, and frame rate. Group of pictures (GOP) headers are optional and are mainly used to identify good locations for random access into the video. Pictures are numbered consecutively within each group of pictures (or sequence of pictures, if groups of pictures are not being used), the first picture to be displayed always having number zero. Because picture numbers are incremented in display order, the numbers may appear slightly out-of-order in the actual data stream (refer to Section 2.3.1 for a discussion of the difference between display and transmission order).

Much like entire video sequences, individual MPEG pictures are also comprised of a hierarchy of smaller elements. The MPEG codec partitions each picture into a series of 16x16 pixel areas, and encodes the video data for each area in a separate macroblock element. Macroblocks within a picture are numbered sequentially in row-major order, with the first macroblock (number 0) at the upper left-hand corner of the picture and the last (highest-numbered) macroblock at the lower right-hand corner of the picture. Strings of consecutive macroblocks are grouped into slices, with a slice header at the start of each slice. Among other things, the slice header identifies the location of the slice in the picture by specifying the macroblock number of the first macroblock in the slice. An encoded MPEG picture consists of a picture header (PIC) followed by a series of one or more Slices, each comprised of a slice header (SLI) followed by a series of consecutively-numbered[1] macroblocks (MB), as show in the lower half of Figure 2.4. The appearance of a SEQ, GOP, PIC or SEQ END in a data stream signals the end of the current picture's data.

Sequence of Pictures headers, Group of Pictures headers, Picture headers and Slice headers all begin with a unique set of start codes. Lost or corrupted data and encoding errors can cause MPEG decoders to lose synchronization. A decoder restores synchronization by discarding data bits until a start code is encountered in the data stream. Start codes are 4 bytes in length. The the first 3 bytes, called the start code prefix, always have the value 0x000001 hex. The remaining byte identifies the next element in the stream (SEQ, GOP, PIC, SLI, or other element). With few exceptions, MPEG video streams are designed so that the prefix does not appear anywhere in the stream except at the start of synchronizable data elements. Because macroblocks do not begin with start codes, slices are the smallest synchronizable element in the MPEG video hierarchy. Thus, if synchronization is lost while decoding a macroblock,

[1] It is common for slices to contain consecutively-numbered macroblocks, but the standards actually allow macroblocks to be skipped. Thus, although macroblock numbers within slices are always monotonically increasing, they may not always be consecutive.

the remaining macroblocks in that slice are lost and processing resumes at the start of the next slice, picture, or other synchronizable element. See [7] for a more in depth discussion of the layers of the MPEG hierarchy.

2.3.3 Macroblock Structure

As mentioned earlier, each macroblock contains data for a 16x16 pixel area of a picture. Macroblocks can either be intra-coded or include motion vectors. If intra-coded, the macroblock and objects above it in the MPEG hierarchy (the most recent Sequence of Pictures header, Group of Pictures header, Picture header and Slice header are all higher in the hierarchy) contain all of the necessary information for reconstructing the 16x16 pixel area of the picture. If motion vectors are included, however, additional data must be gathered from one or more reference picture macroblocks before the 16x16 pixel area of the picture can be reconstructed.

Macroblocks within I-pictures are always intra-coded; they do not contain normal motion vectors[2]. Macroblocks within P-pictures and B-pictures can either be intra-coded or include motion vectors. (Some macroblocks within a picture may be intra-coded, while others contain one or more motion vectors; the decision is made at encoding time on a macroblock-by-macroblock basis.) Motion vectors in B-picture macroblocks can reference data from an earlier and/or later reference picture; motion vectors in P-picture macroblocks can only reference data from an earlier reference picture. The picture dependencies shown in Figure 2.2 and discussed in Section 2.3.1 are actually produced by the motion vectors inside each picture's macroblocks. I-

[2]The MPEG-2 standard allows intra-coded macroblocks to include "hidden" motion vectors, but their usage is not standardized and will not be considered here.

pictures are independent of all other pictures because their macroblocks never contain normal motion vectors. Macroblock dependencies and motion vectors are discussed in greater detail in Chapter 3.

An understanding of the conversion from RGB to YUV[3] color space, the discrete cosine transform (DCT), coefficient quantization, run-length encoding, and other steps in MPEG macroblock processing are not necessary for the purposes of this thesis and will not be covered here. For a detailed treatment of these subjects, consult any of the following references [2, 6, 12, 13].

2.3.4 Profiles and Levels

The MPEG-1 and MPEG-2 video standards allow videos to be encoded using a wide variety of picture sizes and rates. The MPEG-1 standard allows picture sizes up to 4095 x 4095 pixels and rates up to 60 frames per second; MPEG-2 allows picture sizes up to 16384 x 16384 pixels and rates up to thousands of frames per second. The MPEG-2 standard also allows a multitude of scalability and other options. In order to limit the complexity of decoders, the MPEG-2 video standard defines a set of profiles that support subsets of the allowable syntax, and a set of levels that limit specific parameters within each profile to a range of acceptable values. Similarly the MPEG-1 standard profile, referred to as the constrained parameters bitstream (CPB), specifies a minimum set of values that all MPEG-1-compliant decoders must support.

The MPEG-1 constrained parameters bitstream limits video to 396 macroblocks (101,376 pixels) per frame for frame rates less less than 25 frames per second, and 330 macroblocks (84,480 pixels) per frame for frame rates less than or equal to 30

[3]YUV color space describes color in terms of one luminance and two chrominance components.

frames per second [13]. The maximum allowable CPB bitrate is 1.856 Mbits/sec [2]. MPEG-1 videos are typically encoded in the standard interchange format (SIF), which is 240 lines x 352 pixels per line at 30 frames per second (United States) or 288 lines x 352 pixels per line at 25 frames per second (Europe). Both SIF options satisfy the requirements of the constrained parameters bitstream.

The MPEG-2 standard currently defines five profiles (simple, main, SNR, spatial, and high) and four levels (low, main, high-1440, and high). The main profile at main level allows a maximum picture size of 576 lines x 720 pixels per line, up to 30 frames (60 fields) per second, and a maximum bitrate of 15 Mbit/sec. MPEG-2-compliant decoders should specify the profiles and levels they support.

2.4 MPEG Audio Elementary Streams

The MPEG-1 and MPEG-2 audio codecs each include three compression algorithms for various levels of sound quality [16]. The three algorithms all use psychoacoustic models, but have different levels of complexity and quality. The layer-I algorithms are the simplest and produce the poorest quality sound; the layer-III algorithms are the most complex, but produce the best sound. Although audio elementary streams will likely be included in many real-world MPEG-2 systems streams, MPEG audio compression codec details and error correction techniques are beyond the scope of this thesis.

Assuming a multimedia presentation with MPEG-2 video encoded at the main profile and main level (576 lines x 720 pixels, 30 frames per second, and 15 Mbit/sec), addition of a CD-quality MPEG layer-II audio track (32 - 384 Kbit/sec [13]) would increase the presentation bandwidth by roughly 2 percent. Such a small increase should have negligible effect on this research and will be ignored. The MPEG-2 systems transport streams used in this thesis work will not contain audio elementary streams. For additional information about MPEG audio, see [16].

Chapter 3

MPEG-2 Video Dependencies

3.1 Differences between MPEG-1 and MPEG-2 video Dependencies

The basic structure of MPEG-2 video data is the same as that of MPEG-1 video data, but the MPEG-2 video standard includes additional fields to support new features such as scalability and interlaced video. Because their basic structures are the same, the MPEG-2 video codec is backwards compatible with the MPEG-1 codec and MPEG-2 video decoders should be able to decode both MPEG-1 and MPEG-2 videos. This chapter discusses the dependencies in MPEG-2 video streams with the understanding that MPEG-1 video dependencies are a subset of those discussed.

A correct understanding of MPEG video dependencies is essential to the operation of the DART system preprocessor. This chapter outlines those dependencies and identifies the MPEG encoding options and extensions that are currently supported by the DART system.

3.1.1 Scalability

The MPEG-2 video codec defines scalability extensions that are not part of the MPEG-1 video standard. These scalability extensions allow a single video to be encoded at multiple levels of detail: a base layer provides the basic video information, and one or more enhancement layers provide higher resolution data. The layers are encoded as separate video streams. A decoder can either reconstruct the basic video by decoding the base layer and ignoring the enhancement layers or else integrate the data from the base and enhancement layers into a single higher-quality video. Thus, the base layer could contain a video encoded at the standard television broadcast resolution (NTSC), while an enhancement layer could provide extra data for receivers that support High Definition Television (HDTV) resolutions.

Aside from providing multiple resolutions, the scalability extensions also allow a decoder to gracefully degrade the quality of a video if the base layer is received successfully but data from the enhancement layers is lost or corrupted. There are four types of scalability: spatial scalability, SNR scalability, temporal scalability, and data partitioning. Each of them is described briefly in this section, but scalable video streams are not currently supported by the DART system and are not included in the dependency analysis or thesis simulations. Although future work may combine or compare the effects of the MPEG-2 scalability extensions with those of the DART system, such issues are not treated here.

Spatial Scalability

In spatial scalability, the spatial resolution (number of pixels per picture, or picture size) in the enhancement layer video stream is higher than that of the base layer stream. The base layer data must be "upsampled" to the spatial resolution of the enhancement layer pictures using linear interpolation before the data can be combined into a single high-resolution video.

SNR Scalability

In SNR scalability, both the base and enhancement layer video streams have the same spatial resolution (picture sizes), but the quantization values used to encode the macroblocks are different. The large quantization values used in the base layer produce highly compressed but less-detailed pictures than the smaller quantization values used in the enhancement layers. The enhancement layers do not produce high-quality images by themselves; instead, they contain refinements that must be added to the base layer data. When the refinement data from the enhancement layers is decoded and added to the decoded base layer data, more-detailed (less blocky) video images result.

Temporal Scalability

With temporal scalability, both the base and enhancement layer video streams have the same spatial resolution, but the frame rates (pictures per second) are different. The enhancement layer contains data for pictures to be inserted temporally between those in the base layer. The full frame rate is achieved by multiplexing the base and enhancement layer pictures. Enhancement layer pictures are allowed to reference information from either base or enhancement layer reference pictures. Decoders that ignore the enhancement layer data produce videos at the base picture rate.

Data Partitioning

Data partitioning splits the video data into two layers: a high-priority layer referred to as partition 0, and a low-priority layer referred to as partition 1. A field in the video headers indicates which partitioning algorithm is used and hence, which parts of the video are included in the high-priority partion and which are sent in the other partition. "Sequence of Pictures" headers, "Group of Pictures" headers and "Picture" headers are sent in both partitions to help maintain synchronization between the partitions and to aid in error recovery. The partitions are sent as two separate video elementary streams, which are then merged by the decoder. The assumption is that a higher-quality transmission medium will be used to send partition 0 data, giving the higher-priority data a better chance of being successfully received.

Hybrid Scalability

Any two of the following three scalability extensions can be combined to generate a hybrid form of scalability for a video: spatial scalability, SNR scalability and temporal scalability. Other combinations of scalability options are not currently supported by the MPEG-2 video standard.

3.1.2 Progressive and Interlaced Video

Another new function supported by the MPEG-2 video codec that was not included in MPEG-1 is the ability to encode interlaced video. Any picture can be encoded as either a single frame (progressive picture) or a set of two fields (interlaced picture). The dependencies for a sample sequence of MPEG-1 or progressive MPEG-2 pictures are shown in Figure 3.1. Dependencies between fields in an interlaced MPEG-2 video

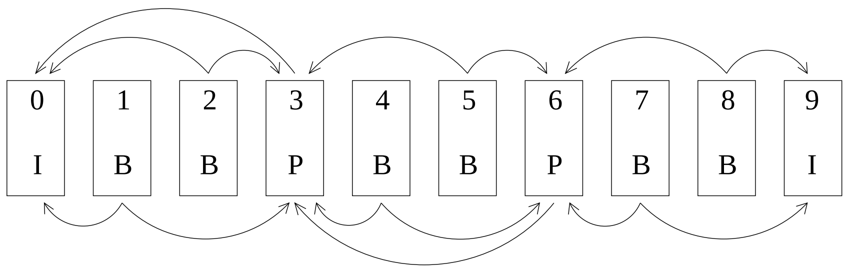

Figure 3.1: *Dependencies between pictures in a sample MPEG-1 or progressive MPEG-2 video elementary stream. Arrows point from P and B pictures to the reference pictures upon which they depend.*

stream are shown in Figure 3.2, and the order of transmission for the fields is shown in Figure 3.3. The MPEG-2 video standard allows field and frame pictures to exist within a single video sequence, as shown in Figure 3.4. The DART system supports all three types of video streams (progressive, interlaced and mixed).

Any picture can be encoded as either a single frame (progressive picture) or a set of two fields (interlaced picture). In either case, the picture is partitioned into regions called macroblocks. Macroblocks are always 16 by 16 pixels in size. The MPEG-2 standard defines a set of levels and profiles that, among other things, establishes standard picture sizes. A main level MPEG-2 video frame is 576 pixels high by 720 pixels wide, which partitions evenly into 1620 macroblocks (arranged in 36 rows, 45 macroblocks per row). Without loss of generality and to simplify the figures, the examples discussed in this chapter will use a square 96 pixel high by 96 pixel wide picture, partitioned into 36 macroblocks (arranged in 6 rows and 6 columns for frame pictures, or 3 rows and 6 columns for each interlaced picture field).

In frame pictures, each macroblock contains data for a 16 by 16 pixel square area of the picture. Macroblocks are numbered in ascending order from left to right and

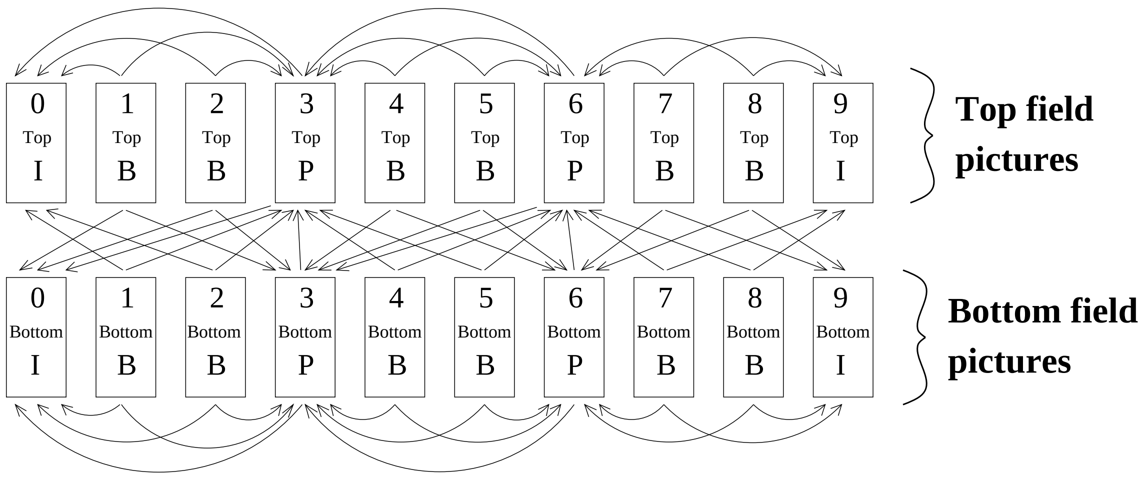

Figure 3.2: *Dependencies between fields in a sample interlaced MPEG-2 video stream. The top fields in this figure correspond to the even-numbered rows of pixels in the interlaced picture; the bottom fields in the figure correspond to odd-numbered rows. Arrows point from P and B fields to the reference fields upon which they depend ("top field first" transmission is assumed). The field pairs are shown in display order (left to right).*

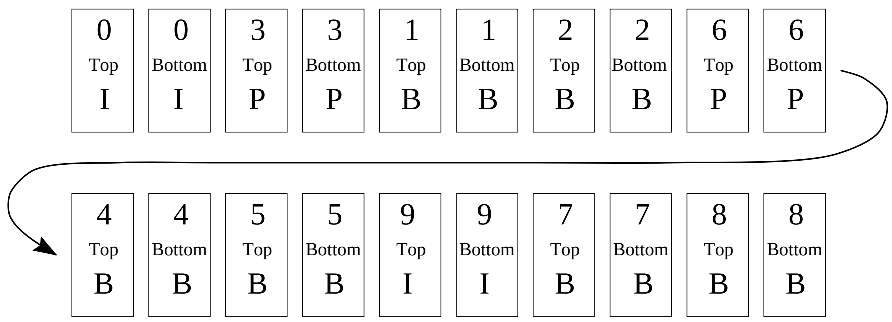

Figure 3.3: *Transmission order for the MPEG fields shown in the previous figure. Reference pictures (fields) are always transmitted before the pictures that reference them. A flag in the data stream indicates whether the top or bottom fields are transmitted first. "Top field first" transmission is assumed in this figure.*

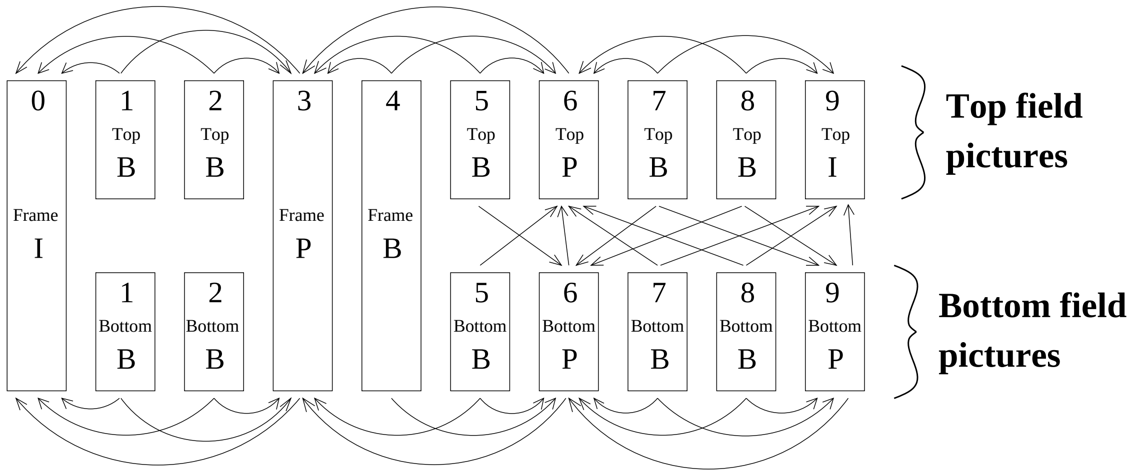

Figure 3.4: *Dependencies between field and frame pictures in a sample MPEG-2 video stream that uses both field and frame encoding. Arrows point from P and B fields and frames to the reference pictures upon which they depend ("top field first" transmission is assumed). If the first field of a picture is an I-field, the second field can either be an I or P-field. In this figure, the top field of picture 9 is an I-field and the bottom field is a P-field.*

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Figure 3.5: *This figure shows the location of macroblocks in a 96 x 96 pixel MPEG-2 frame picture. Each of the 36 macroblocks covers a 16x16 pixel square area of the picture. A more typical 576 x 720 pixel MPEG-2 frame picture would contain 1620 macroblocks.*

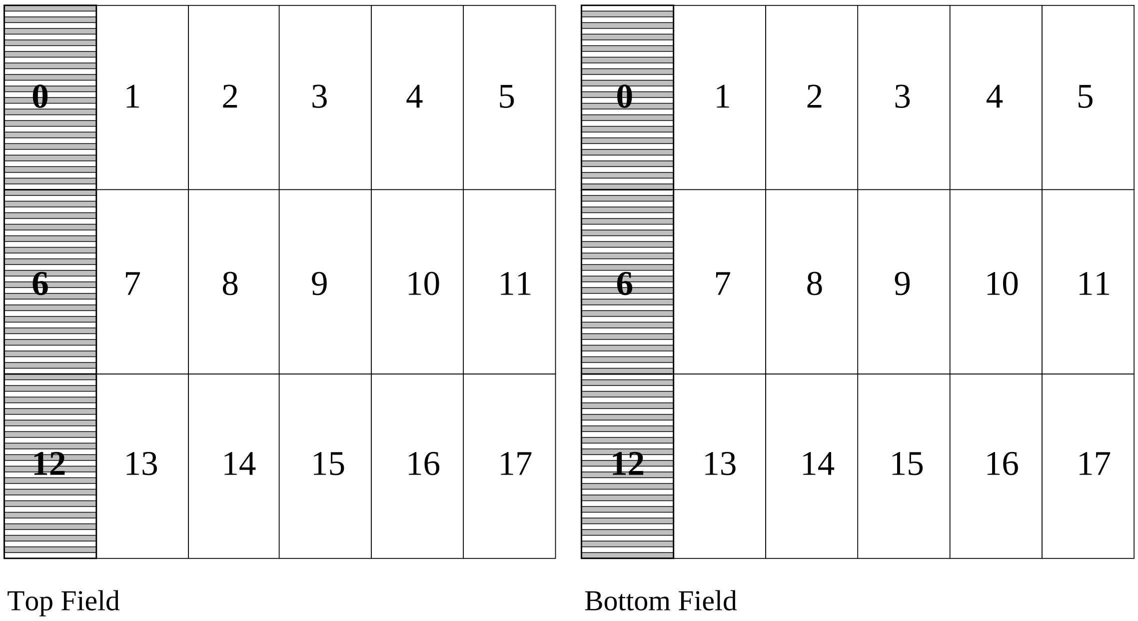

Figure 3.6: *This figure shows the location of pixels included in each macroblock in the top and bottom fields of an interlaced 96 x 96 pixel MPEG-2 picture. The picture contains 36 macroblocks: 18 in each field. Each macroblock is 16x16 pixels in size and covers alternating rows (represented as shaded areas in the figure) of a 16x32 pixel region of the final picture. Assuming that rows of pixels are numbered in ascending order from top to bottom with row 0 at the top, macroblocks in the top field contain data for even-numbered rows and macroblocks in the bottom field contain data for odd-numbered rows of pixels. For simplicity, the alternating (shaded) rows of pixels are shown only for the leftmost macroblock in each row.*

top to bottom, as shown in Figure 3.5. Macroblocks in field pictures are slightly different. Numbering the rows of pixels in ascending order from top to bottom with row 0 at the top, the top field of an interlaced picture contains data from the even-numbered rows of pixels, and the bottom field contains data from the odd-numbered rows. Like frame macroblocks, field macroblocks are 16 by 16 pixels in size. Because their data represent alternating rows of pixels, however, each field macroblock affects pixels within a 16 pixel wide by 32 pixel high area of the final interlaced picture, as shown in Figure 3.6. Each field of an interlaced picture contains roughly half the number of macroblocks as a corresponding frame picture, making the total number of macroblocks in the two fields usually equal to the number of macroblocks in a corresponding frame.

Pictures within MPEG-2 videos need not be displayed in the same format as they were encoded. The MPEG-2 standard distinguishes between encoding and display formats. A picture encoded as a frame picture can be subsequently displayed as a sequence of two interlaced fields on an interlaced device by displaying, in turn, alternating lines from the decoded picture. Similarly, a picture encoded as two fields can be decoded into alternating lines of a common buffer for subsequent display on a non-interlaced device. As shown earlier in Figure 3.4, progressively encoded (frame) pictures and interlaced (field) picture pairs can be interspersed within a single encoded video sequence.

3.2 Dependencies Between Pictures

Due to the temporal dependencies between pictures, the loss of a single reference frame picture can have a rippling affect that makes neighboring pictures useless. The loss of reference picture P6 from Figure 3.1 can make it impossible to reconstruct all or part of four other pictures (B4, B5, B7 and B8) that possibly depend on data encoded in P6. Similarly, the loss of picture P3 could result in the loss of all or part of seven pictures besides itself (B1, B2, B4, B5, P6, B7 and B8), as shown in Figure 3.7. Picture B2, on the other hand, does not affect the decoding of any of its neighboring pictures.

The loss of field pictures has a similar effect. Losing the bottom reference field of picture P6 in Figure 3.2 can make it impossible to reconstruct all or part of eight other fields (top and bottom fields of B4, B5, B7 and B8) that possibly depend on data encoded in the bottom field of P6 (see Figure 3.8). If the top field of P6 were lost instead, nine fields could be affected instead of eight because the bottom field of P6 depends on the top field of P6. Similarly, the loss of the bottom field of picture

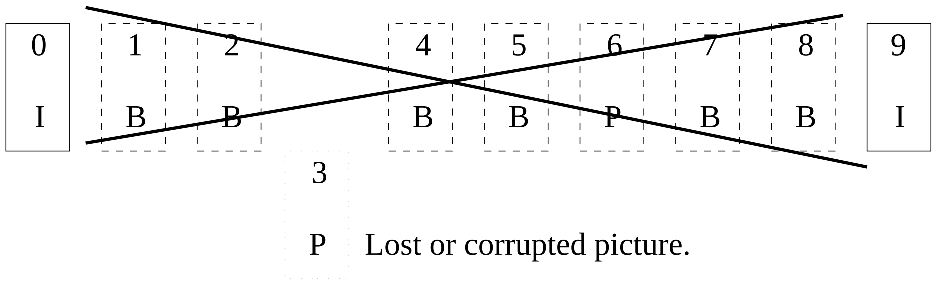

Figure 3.7: *This figure shows the effect the loss of picture P3 has on neighboring pictures. The loss of picture P3 could cause each of the crossed out pictures to become partially or fully undecodable.*

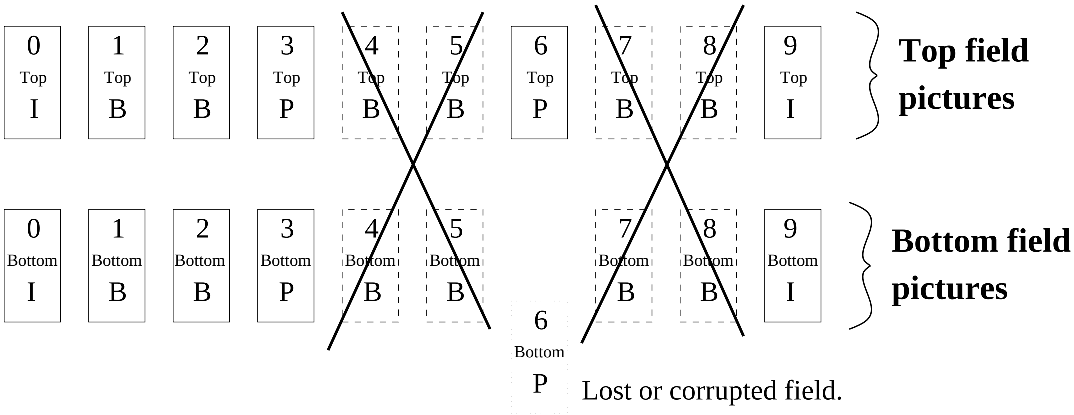

Figure 3.8: *This figure shows the effect that losing the bottom field of picture P6 has on neighboring fields. The loss of the bottom field of P6 could cause any or all of the eight crossed out fields to become partially or fully undecodable.*

P3 could result in the loss of all or part of fourteen fields besides itself (the top and bottom fields of B1, B2, B4, B5, P6, B7 and B8), and the loss of the top field of picture P3 could result in the loss of fifteen fields. Losing the top or bottom field of a B-Picture, on the other hand, does not affect the decoding of any neighboring fields.

3.3 Motion Vectors

Looking at MPEG pictures in more detail, the picture and field dependencies discussed in Section 3.2 and shown in Figures 3.1, 3.2 and 3.4 are actually a result of MPEG's macroblock prediction process. Motion vectors identify relationships between predicted macroblocks and decoded reference pictures. The relationships between predicted macroblocks and the encoded form of their reference pictures, however, is not explicitly discussed in the standards—or to my knowledge, anywhere else in the current literature. The remainder of this chapter examines these relationships in detail, enabling dependencies to be mapped to actual data within encoded video streams.

Macroblock data within P-pictures and B-pictures is predicted (in other words, encoded or decoded) using data from previously decoded reference (I and P) pictures. These inter-picture dependencies (predictions) are encoded as motion vectors in the predicted macroblocks. Motion vectors identify the relative location of a 16x16 pixel reference picture region that is similar to the pixels in the macroblock being predicted (encoded/decoded). To save space, only the differences between the predicted pixels and the reference pixels are encoded in the macroblock. Because motion vectors essentially record the movement of pixel patterns (and hence, larger objects) between pictures, this process is often referred to as "motion compensation."

Macroblocks within I-pictures are always intra-coded; they don't contain normal motion vectors[1]. Macroblocks that are not intra-coded rely on one or more motion vectors to provide relative offsets to data in reference pictures. If a macroblock is skipped or no motion vectors are explicitly encoded in a macroblock, a zero-displacement or other appropriate implicit motion vector is assumed.

Macroblocks within P-pictures and B-pictures can either be intra-coded or include implicit/explicit motion vectors. (Some macroblocks within a picture may be intra-coded, while others contain one or more motion vectors; the decision is made at encoding time on a macroblock-by-macroblock basis.) Motion vectors in P-picture macroblocks specify the relative location of data in a past decoded reference picture. Motion vectors in B-picture macroblocks specify the relative location of data in past

[1]The MPEG-2 standard allows intra-coded macroblocks to include "hidden" motion vectors, but their usage is not standardized and will not be considered here.

and/or future decoded reference pictures. A $(0,0)$ motion vector, for example, references data located at the same screen position in the reference picture as the predicted macroblock occupies in the predicted picture. A motion vector with positive coordinates references an area in the reference picture that appears lower on the screen and to the right of that occupied by the predicted macroblock in the predicted picture; a motion vector with negative coordinates references an area having a screen position higher and to the left of that of the predicted macroblock.

There are two major classes of motion vectors: frame-based and field-based. Frame-based motion vectors can occur only in frame (progressive) pictures; field-based motion vectors can occur in either frame (progressive) or field (interlaced) pictures. A frame-based motion vector specifies a 2-dimensional relative displacement within a reference frame, whereas a field-based motion vector specifies a similar displacement within a reference field— independent of whether the actual reference picture was encoded as a frame or field picture. Because field displacements count only alternating rows of an interlaced picture, field-based motion vectors produce twice the vertical screen displacement as frame-based motion vectors having the same values. A field select value is included in each field-based motion vector to indicate whether the referenced data comes from the top or bottom reference field. MPEG-2 motion vector displacements are always in half-pixel increments[2], and specify an offset relative to the location of the macroblock being predicted. Motion vectors are described in more detail in the next few subsections. The DART system supports all currently-defined types of motion vectors.

[2]Although the MPEG-2 video standard requires motion vector displacements to be specified in half-pixel increments, MPEG-1 videos can use either whole or half-pixel increments.

3.3.1 Locating Data Referenced by Motion Vectors

The DART preprocessor must be able to identify dependencies between predicted macroblocks and reference pictures. Unfortunately, the motion vectors in the predicted macroblocks identify pixel regions in *decoded*, rather than *encoded* reference pictures. As a result, the preprocessor must be able map the decoded regions back to the original reference picture macroblocks that produced them. The remainder of this chapter discusses how the mapping is accomplished.

Consider a hypothetical reference screen buffer large enough to hold data for a frame picture containing r rows by c columns of pixels. Assume the screen buffer holds the decoded pixel data for a reference picture. If the reference picture was originally encoded as two field pictures the even rows of the buffer would contain data from the top field and the odd rows would contain data from the bottom field. Regardless of whether the reference picture was originally encoded as a frame picture or two field pictures, the buffer contains the same amount of data. Further, assume the top left pixel in the reference picture has buffer coordinates $(0, 0)$ and the bottom right pixel has coordinates $(c-1,\ r-1)$.

Data referenced by frame-based motion vectors can come from any row of the reference picture (screen buffer); data referenced by field-based motion vectors must come from either the even or odd-numbered rows (but not both) of the reference picture, depending on the motion vector's field select value.

The width w in macroblocks of a picture containing c columns of pixels is given by Equation 3.1. For $c = 96$, Equation 3.1 yields a macroblock width $w = 6$, which agrees with Figures 3.5 and 3.6.

$$w = \left\lfloor \frac{(c+15)}{16} \right\rfloor \tag{3.1}$$

The height h in macroblocks of a picture containing r rows of pixels is given by Equation 3.2 if the picture is encoded as a single frame; the height h in macroblocks of each field in an interlaced picture of the same size is given by Equation 3.3. For $r=96$, Equations 3.2 and 3.3 yield macroblock heights $h=6$ and $h=3$, respectively, which agree with Figures 3.5 and 3.6.

$$h = \left\lfloor \frac{(r+15)}{16} \right\rfloor \tag{3.2}$$

$$h = \left\lfloor \frac{(r+31)}{32} \right\rfloor \tag{3.3}$$

Each frame or field is partitioned into M macroblocks, as defined by Equation 3.4. Macroblocks are numbered in row-major order, with macroblock 0 at the upper left-hand corner and macroblock $(M-1)$ at the lower right-hand corner of the picture. Valid macroblock numbers m must be in the range specified by Equation 3.5:

$$M = w \times h \tag{3.4}$$

$$0 \leq m \leq M-1 \tag{3.5}$$

In the following discussions of full-pixel and between-pixel displacements, let m be the macroblock number of a predicted macroblock. If macroblock m belongs to a frame or top field picture, let (x, y) be the screen position of the top left pixel in the macroblock; if macroblock m belongs to a bottom field picture, let (x, y) be the top left pixel position of the macroblock having the same number (m) in the corresponding top field picture. Thus, x and y are always even integers. For any macroblock, the screen x-coordinate can be calculated from the macroblock number m and macroblock width w using Equation 3.6:

$$x = 16\left(m\%w\right), \text{ where } \% \text{ is the modulus function} \tag{3.6}$$

The value for the screen y-coordinate can be calculated using Equation 3.7 for frame pictures and Equation 3.8 for field pictures:

$$y = 16\left\lfloor \frac{m}{w} \right\rfloor \tag{3.7}$$

$$y = 32\left\lfloor \frac{m}{w} \right\rfloor \tag{3.8}$$

Thus, the (x, y) coordinates for macroblock 7 are $(16, 32)$ in both the top and bottom fields of Figure 3.6, but are $(16, 16)$ for the frame picture in Figure 3.5.

The source of data for pixels in the reference screen buffer can actually be more complicated than has already been discussed. Consider the bottom field of picture P6 in Figure 3.4 on page 27. The bottom field of picture P6 depends on both reference frame P3 and the top field of reference picture P6. When the top field of picture P6 is being predicted, the reference screen buffer contains the decoded pixel data for frame P3. Because P6 is a reference picture, however, the top field of P6 becomes the new top reference field as soon as it is decoded. Thus, the pixel data for the top field of P6 overlays the even-numbered rows of the P3 pixel data in the reference screen buffer, and the buffer contains data from both P3 and the top field of P6 at the time the bottom field of P6 is predicted. This is not particularly confusing for field-based motion vectors, which contain field select values to indicate whether the prediction data is to be taken from the top field (reference field P6) or bottom field (reference frame P3). In the case of frame-based motion vectors, however, the reference macroblocks are the sum of two calculations: the list of referenced macroblocks from P3 are calculated by treating the reference picture as a frame picture, and the list of macroblocks from P6 are calculated by treating the reference picture as two field pictures.

Let j and k be integers, and let $2j$, $2k$, $2j+1$ and $2k+1$ be legal motion vector displacement values. Because MPEG-2 motion vector displacements are counted in half-pixels, the vertical and horizontal displacements for motion vector $(2j, 2k)$ refer to actual pixels in the reference picture; the displacements for motion vector $(2j{+}1,\ 2k{+}1)$ refer to values midway between pixels.

In the following descriptions of motion vector displacements, assume a reference screen buffer similar to the one described in this section contains the picture data (frame or two fields) referenced by each of the motion vectors:

Full-pixel (even-numbered) Motion Vector Displacements

Assume the macroblock being predicted includes the frame-based motion vector $(2j,\ 2k)$. Reference data for the top left pixel in the predicted macroblock is obtained from position $(x+j,\ y+k)$ of the reference screen buffer. Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x and $(y+y_0)$ for y, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

Assume the macroblock being predicted includes the field-based motion vector $(2j,\ 2k)$. Because field-based motion vectors describe field displacements (they count either even-numbered or odd-numbered rows, but not both), the reference data for the top left pixel in the predicted macroblock is obtained from reference buffer position

$(x+j,\ y+2k)$ if the top field is selected in the motion vector, or from reference buffer position $(x+j,\ y+2k+1)$ if the bottom field is selected. Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x and $(y+2y_0)$ for y, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

Between-pixel (odd-numbered) Horizontal Motion Vector Displacements

Data for between-pixel displacements are obtained by averaging the data from neighboring pixels in the reference frame or field. Assume the macroblock being predicted includes the frame-based motion vector $(2j+1,\ 2k)$. Such a motion vector references locations positioned horizontally midway between pixels in the reference picture. The reference data for the top left pixel in such a predicted macroblock is obtained by averaging the pixel values at coordinates $(x+j,\ y+k)$ and $(x+j+1,\ y+k)$ of the reference screen buffer. Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x and $(y+y_0)$ for y, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

The reference data for the top left pixel in a field-based motion vector $(2j+1,\ 2k)$ with the top field selected is obtained by averaging the pixel values at reference screen buffer coordinates $(x+j,\ y+2k)$ and $(x+j+1,\ y+2k)$; the reference data for a similar motion vector with the bottom field selected is obtained by averaging the values at pixels $(x+j,\ y+2k+1)$ and $(x+j,\ y+2k+1)$. Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x and $(y+2y_0)$ for y, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

Between-pixel (odd-numbered) Vertical Motion Vector Displacements

Assume the macroblock being predicted includes the frame-based motion vector $(2j,\ 2k+1)$. Such a motion vector references locations positioned vertically midway between pixels in the reference picture. The reference data for the top left pixel in such a predicted macroblock is obtained by averaging the pixel values at coordinates $(x+j,\ y+k)$ and $(x+j,\ y+k+1)$ of the reference screen buffer. Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x and $(y+y_0)$ for y, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

Because top field pictures only include even-numbered rows and bottom field pictures only include odd-numbered rows of the screen, the reference data for the top left pixel in a predicted macroblock having a field-based motion vector with the top field selected is obtained by averaging the pixel values at reference buffer coordinates $(x+j,\ y+2k)$ and $(x+j,\ y+2k+2)$; the reference data for a similar predicted macroblock with the bottom field selected is obtained by averaging the values for pixels $(x+j,\ y+2k+1)$ and $(x+j,\ y+2k+3)$ in the reference screen buffer. Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x and $(y+2y_0)$ for y, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

Between-pixel (odd-numbered) Vertical and Horizontal MV Displacements

Reference data for motion vectors such as $(2j+1,\ 2k+1)$, which specify displacements midway between pixels in both the horizontal and vertical directions, are obtained by averaging the four neighboring pixel values. The reference data for the top left pixel in the predicted macroblock is obtained by averaging the values at pixels $(x+j,\ y+k)$, $(x+j,\ y+k+1)$, $(x+j+1,\ y+k)$ and $(x+j+1,\ y+k+1)$ for frame-based motion vectors; $(x+j,\ y+2k)$, $(x+j,\ y+2k+2)$, $(x+j+1,\ y+2k)$ and $(x+j+1,\ y+2k+2)$ for field-based motion vectors with the top field selected; and $(x+j,\ y+2k+1)$, $(x+j,\ y+2k+3)$, $(x+j+1,\ y+2k+1)$ and $(x+j+1,\ y+2k+3)$ for field-based motion vectors with the bottom field selected.

Reference values for other pixels in the predicted macroblock can be found by substituting $(x+x_0)$ for x_0, where x_0 is an integer and $0 \leq x_0 \leq 15$; and by substituting $(y+y_0)$ for y if the macroblock being predicted is in a frame picture, or $(y+2y_0)$ for y if the macroblock being predicted is in a field picture, where x_0 and y_0 are integers and $0 \leq x_0 \leq 15$, $0 \leq y_0 \leq 15$.

Examples of Motion Vector Displacements

Let P be a 96 by 96 pixel MPEG-2 P-picture being predicted (encoded/decoded), and let a screen buffer contain all of the data (one frame or two interlaced fields) for reference picture R. Assume macroblock 7 of picture P contains a motion vector V that references picture R. The actual screen location of the pixel data referenced by V in R depends on P's picture structure. If V is a field-based motion vector the location also depends on V's field select value. The screen coordinates referenced by motion vectors $V = (0,0)$ and $V = (-4,+12)$ are shown in Table 3.1. (For macroblock arrangements within frame and field pictures, refer to Figures 3.5 and 3.6).

In general, Table 3.2 shows the top left reference screen buffer coordinates of the data referenced by any macroblock m having motion vector V. To simplify the equations in the figure, coordinates are given in terms of c and r, where (c, r) are the coordinates (x, y) defined in Equations 3.6 and 3.7 for macroblock number m in frame pictures.

Predicted Picture					Referenced Picture	
Picture Structure	MB 7 Coordinates (x, y)	(top left)	Motion Vector Type	Field Select	Top Left referenced pixel V=(0, 0)	V=(−4, +12)
Frame	(16,16)	(16,16)	Frame-based	—	(16,16)	(14,22)
			Field-based	Top	(16,16)	(14,28)
				Bottom	(16,17)	(14,29)
Top Field	(16,32)	(16,32)	Field-based	Top	(16,32)	(14,44)
				Bottom	(16,33)	(14,45)
Bottom Field	(16,32)	(16,33)	Field-based	Top	(16,32)	(14,44)
				Bottom	(16,33)	(14,45)

Table 3.1: *Screen coordinates for the top left pixel of macroblock 7 and the areas referenced by motion vector V, for $V = (0, 0)$ and $V = (-4, +12)$. Pictures are assumed to be 96 x 96 pixels in size (comprised of a single frame or two fields), and the top left screen coordinates are $(0, 0)$.*

3.3.2 Motion Vector Types

As stated earlier, there are two major classes of motion vectors: frame-based and field-based. The field-based classification contains three subtypes: simple field-based, 16x8 and dual-prime motion vectors. Simple field-based, 16x8 and dual-prime motion vectors are considered field-based because they describe displacements within

Predicted Picture				Referenced Picture	
Picture Structure	MB m Coordinates (top left)	Motion Vector Type	Field Select	Top Left referenced pixel V= $(0,0)$	V= $(2j+p,\ 2k+q)$, $p, q \epsilon \{0,1\}$
Frame	$(c,\ r)$	Frame-based	—	$(c,\ r)$	$(c+j,\ r+k)$
		Field-based	Top	$(c,\ r)$	$(c+j,\ r+2k)$
			Bottom	$(c,\ r+1)$	$(c+j,\ r+2k+1)$
Top Field	$(c,\ 2r)$	Field-based	Top	$(c,\ 2r)$	$(c+j,\ 2r+2k)$
			Bottom	$(c,\ 2r+1)$	$(c+j,\ 2r+2k+1)$
Bottom Field	$(c,\ 2r+1)$	Field-based	Top	$(c,\ 2r)$	$(c+j,\ 2r+2k)$
			Bottom	$(c,\ 2r+1)$	$(c+j,\ 2r+2k+1)$

Table 3.2: *Screen coordinates for the top left pixel of macroblock m and the areas referenced by motion vector V, for $V = (0,0)$ or any other motion vector with integer coefficients. All coordinates are given in terms of c and r, where (c,r) are the screen coordinates of the top left pixel in macroblock m of a progressive (frame) picture. As elsewhere, the screen origin (top left) is assumed to be $(0,0)$.*

a single field of the reference picture. They accomplish this by including only the even-numbered or odd-numbered rows in their displacement values. Frame-based motion vectors, on the other hand, describe displacements within a reference frame, and therefore include both odd and even-numbered rows in their displacements. The DART system supports all of the currently-defined motion vector types. Because motion vector types are specified only once per macroblock, all motion vectors within a given macroblock have the same type. In the following descriptions, motion vector displacements are assumed to be on full-pixel boundaries; between-pixel displacements would reference one additional row and/or column in each affected reference area.

Frame-based motion vectors

Frame-based motion vectors are found only in frame pictures, and specify the relative offset of a 16x16 pixel region in the reference picture containing data needed to construct the macroblock. Although frame-based motion vectors are allowed only in frame pictures, the reference picture may have been originally encoded as either a frame or two fields. Examples of data dependencies for frame-based motion vectors are shown in Figures 3.9 and 3.10. Figure 3.9 shows a frame-based motion vector that references data in a reference picture originally encoded as a frame picture; Figure 3.10 shows a similar vector that references data in a picture originally encoded as two fields.

Simple field-based motion vectors

Simple field-based motion vectors can be found in both predicted frame and field pictures. They always occur in pairs in frame pictures: the first motion vector specifies the relative offset of a 16x8 pixel field in the reference picture containing data needed to construct the even-numbered rows of the predicted macroblock; the second motion vector specifies the relative offset of a similar field needed to construct the odd-numbered rows of the predicted macroblock. In each case, the motion vector's field select value tells whether the data should come from the top or bottom field of the reference picture. The motion vectors can reference the same or different fields in the reference picture. Each of the motion vectors references 8 alternating rows of a 16x16 area of the reference picture. The reference picture may have been originally encoded as a frame or two fields. Examples of data dependencies for simple field-based motion vectors in predicted frame pictures are shown in Figures 3.11 and 3.12. Figure 3.11 shows a simple field-based motion vector that references data in a reference picture originally encoded as a frame picture; Figure 3.12 shows a similar vector that references data in a picture originally encoded as two fields.

Simple field-based motion vectors in predicted field pictures occur alone rather than in pairs, and specify the relative offset of a 16x16 pixel region (field) containing data needed to construct all the rows of the predicted macroblock in which they occur. As with frame pictures, the motion vector's field select value tells whether the data should come from the top or bottom field of the reference picture. Because fields occupy only the even-numbered or odd-numbered rows of pixels in a picture, each of the 16x16 pixel regions actually occupies alternating rows of a 16x32 area of the reference picture. The reference picture may have been originally encoded as a frame or two fields. Examples of data dependencies for simple field-based motion vectors in predicted field pictures are shown in Figures 3.13 and 3.14. Figure 3.13 shows a simple field-based motion vector that references data in a reference picture originally encoded as a frame picture; Figure 3.14 shows a similar vector that references data in a picture originally encoded as two fields.

16x8 motion vectors

16x8 motion vectors can occur only in predicted field pictures, and always occur in pairs. The first motion vector is used to reconstruct the top half of the predicted macroblock, and specifies the relative offset of a 16x8 pixel region in either the top or bottom reference field; the second motion vector is used to reconstruct the bottom half of the predicted macroblock, and also specifies a 16x8 pixel region in one of the reference fields. In both cases, the motion vector's field select value indicates whether the data should come from the top or bottom field of the reference picture. Each of the 16x8 pixel regions actually occupies alternating rows of a 16x16 area of the reference picture. Examples of data dependencies for 16x8 motion vectors in field pictures are shown in Figures 3.15 and 3.16. Figure 3.15 shows a 16x8 motion vector that references data in a reference picture originally encoded as a frame picture; Figure 3.16 shows a similar vector that references data in a picture originally encoded as two fields.

Dual-prime motion vectors

Although dual-prime motion vectors can occur in both predicted frame and predicted field pictures, there are two restrictions on their use: dual-prime motion vectors can only occur in P-pictures and there cannot be any B-pictures between the predicted P-picture and its reference picture. Each dual-prime motion vector is actually comprised of two vectors: a full-size field-based motion vector followed by a smaller differential motion vector. According to the MPEG-2 video standard, differential motion vector coordinates are limited to the values -1, 0, and $+1$.

The full-size motion vector is scaled, shifted, and added to the differential motion vector to create a second full-size motion vector that is used along with the first motion vector in the macroblock prediction process. The two vectors refer to data in different fields (one top, one bottom) of the reference picture, whether or not the reference picture was originally encoded as a frame picture or two field pictures. The top field has parity 0 and the bottom field has parity 1, as shown in Table 3.4.

In predicted field pictures, the first vector is like a simple field-based motion vector; it provides the displacement for a 16x16 pixel area in the reference field of the same parity as the field being predicted. The second full-size vector is calculated by multiplying the first motion vector by a 0.5 or 1.5 scaling factor to account for temporal differences, adding a vertical shift component of -1 or $+1$ to offset parity differences between the predicted and reference fields, and adding the differential motion vector. Non-integer results are rounded to the nearest integer away from zero. This new vector provides the displacement for a 16x16 pixel area in the reference field of opposite parity to the field being predicted. Each of the 16x16 pixel regions actually

Scaling Factor Values			
Predicted Picture Structure		Field referenced by first motion vector	
		Top Field	Bottom Field
Frame	Top field first	1.5	0.5
	Bottom field first	0.5	1.5
Top Field	–	–	0.5
Bottom Field	–	0.5	–

Table 3.3: *This table shows the scaling factor values (0.5 and 1.5) used to construct the second motion vector in a set of dual-prime motion vectors. The first motion vector in a dual-prime pair is multiplied by the scaling factor as part of the second motion vector calculation.*

occupies alternating rows of a 16x32 pixel area of the reference picture, which may have been originally encoded as a frame or two fields. After the second vector has been calculated, the data at corresponding points in the two reference areas pointed to by the motion vectors are averaged to form a final prediction for each corresponding pixel in the predicted macroblock.

The scaling factor is important because it allows the second motion vector to be constructed by temporal interpolation of the first vector. Scaling factor values depend on the parity of the referenced and predicted fields, as shown in Table 3.3. If the predicted macroblock is part of a frame picture, the scaling factor also depends on the decoding order (i.e., "top field first") of the predicted picture.

Let $V_1 = (j_1, k_1)$ be the first motion vector and $V_d = (j_d, k_d)$, be the differential motion vector in a dual-prime motion vector pair, where $j_d, k_d \epsilon \{-1, 0, +1\}$. Further,

Picture Structure	Parity Value
Frame	1
Top Field	1
Bottom Field	0

Table 3.4: *This table shows the parity value associated with each MPEG-2 picture structure.*

let F_{scale} be the scaling factor as defined by Table 3.3; let P_{parity} and R_{parity} be the respective parities of the predicted and referenced pictures, as defined by Table 3.4; and let F_{vshift} be the vertical shift factor, as defined by Equation 3.11. Finally, let $V_2 = (j_2, k_2)$ be the second full-size motion vector to be constructed from V_1 and V_d. The x-component j_2 of $V_2 = (j_2, k_2)$ is calculated using Equation 3.9 if $j_1 \geq 0$ or Equation 3.10 if $j_1 < 0$. If V_1 references the top field of the reference picture then V_2 must reference data in the bottom reference field and a vertical shift factor of $+1$ is used when calculating the y-component of V_2; otherwise, V_2 must reference data in the top reference field and a vertical shift factor of -1 is used. The y-component k_2 of motion vector $V_2 = (j_2, k_2)$ is calculated using Equation 3.12 if $k_1 \geq 0$ or Equation 3.13 if $k_1 < 0$.

$$j_2 = \lceil (F_{scale} \times j_1 \rceil + j_d, \quad j_1 \geq 0 \tag{3.9}$$

$$j_2 = \lfloor F_{scale} \times j_1 \rfloor + j_d, \quad j_1 < 0 \tag{3.10}$$

$$F_{vshift} = P_{parity} - R_{parity} \tag{3.11}$$

$$k_2 = \lceil F_{scale} \times k_1 \rceil + F_{vshift} + k_d, \quad k_1 \geq 0 \tag{3.12}$$

$$k_2 = \lfloor F_{scale} \times k_1 \rfloor + F_{vshift} + k_d, \quad k_1 < 0 \tag{3.13}$$

Examples of data dependencies for dual-prime motion vectors in field pictures are shown in Figures 3.17 and 3.18. Figure 3.17 shows a set of dual-prime motion vectors that references data in a reference picture originally encoded as a frame picture; Figure 3.18 shows a similar set of vectors that references data in a picture originally encoded as two fields.

In frame pictures, the predicted macroblock is treated like two 16x8 fields (one top, one bottom) of alternating rows. The macroblock includes one full-size motion vector and two smaller differential vectors: one for predicting the top field and another for predicting the bottom field. Each field is predicted as described earlier for field pictures, except that the vectors reference 16x8 pixel areas in the reference fields (alternating rows of 16x16 pixel areas in the reference pictures).

Examples of data dependencies for dual-prime motion vectors in frame pictures are shown in Figures 3.19 and 3.20. Figure 3.19 shows dual-prime motion vectors that reference data in a reference picture originally encoded as a frame picture; Figure 3.20 shows similar vectors that reference data in a picture originally encoded as two fields.

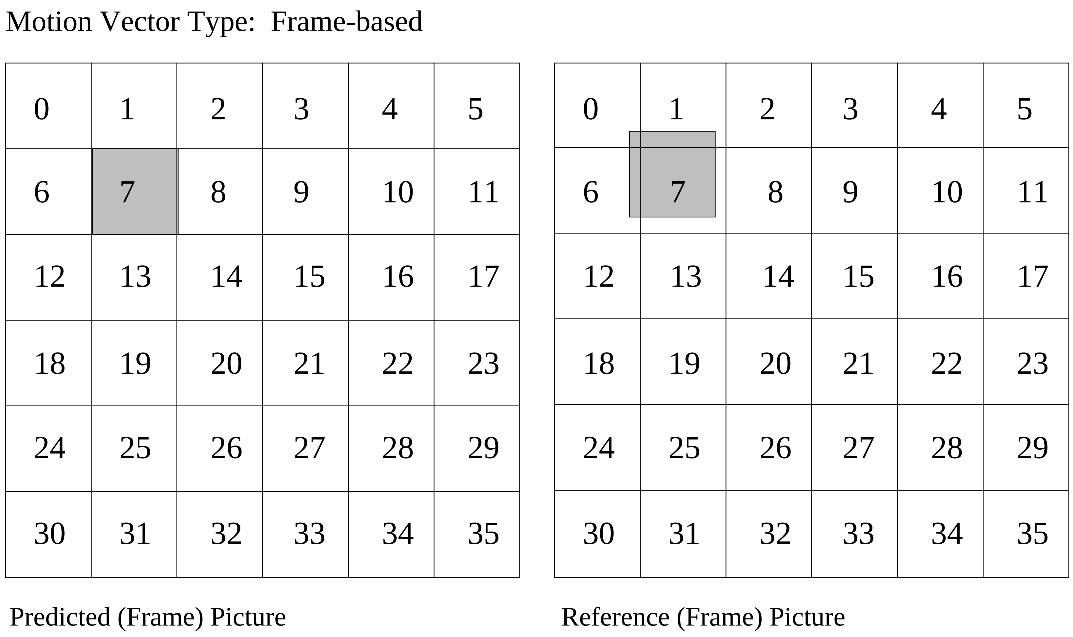

Figure 3.9: *This figure shows an example of the macroblock dependencies produced by a frame motion vector in a predicted frame picture when the reference picture was also encoded as a frame picture. The pictures are shown divided into macroblocks. The shaded rectangle in the reference frame identifies the pixels required to reconstruct the shaded macroblock in the predicted picture. In this example, macroblock 7 in the predicted picture has a frame motion vector* $(-4, -8)$*, and depends on four macroblocks (0, 1, 6 and 7) from the reference frame.*

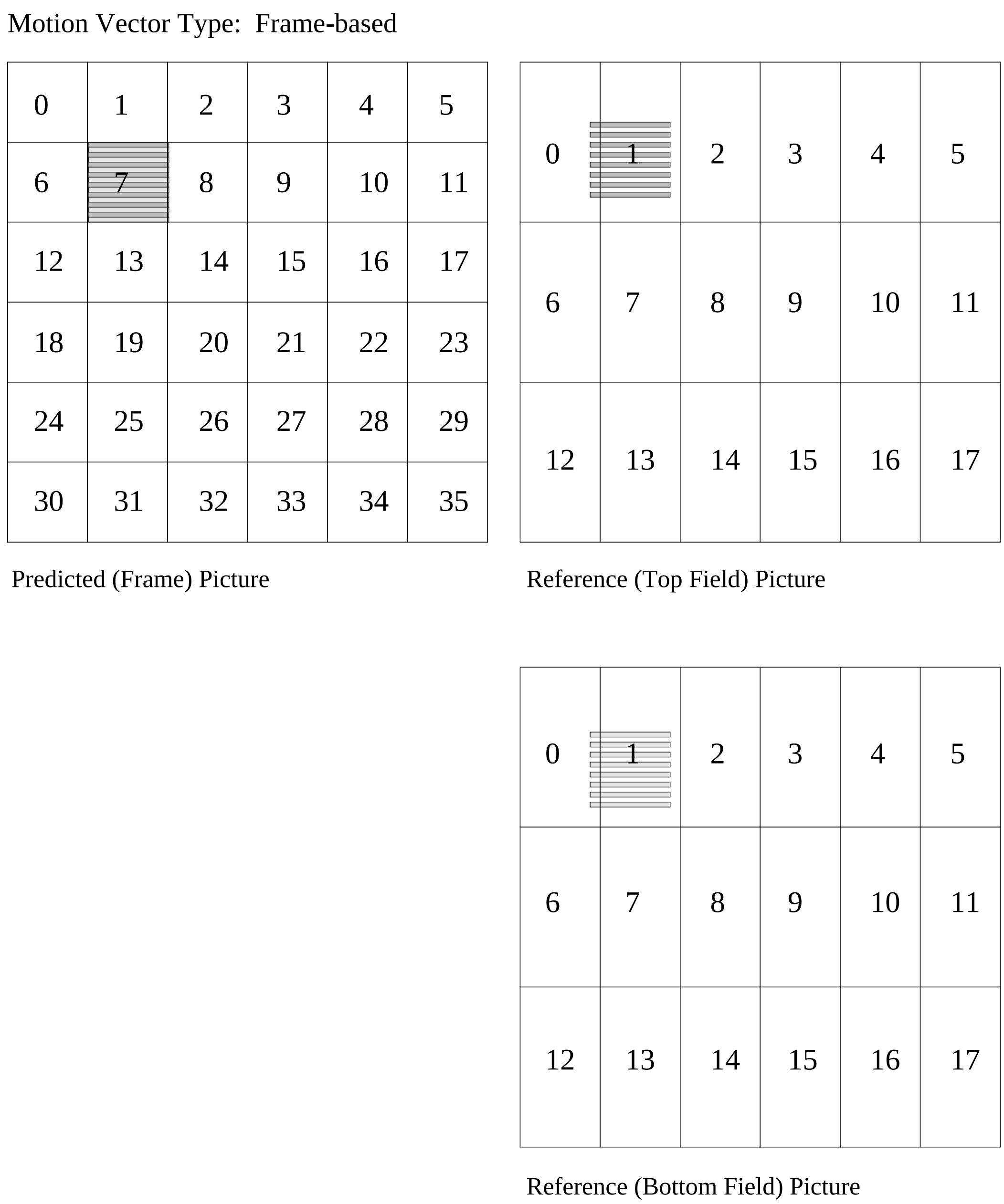

Figure 3.10: *This figure shows an example of the macroblock dependencies produced by a frame motion vector in a predicted frame picture when the reference picture was encoded as two fields. The pictures are shown divided into macroblocks. The shaded rows of pixels in the reference fields identify the pixels required to reconstruct the shaded macroblock in the predicted picture. In this example, macroblock 7 in the predicted picture has a frame motion vector* $(-4, -8)$*, and depends on four reference macroblocks: two macroblocks (0 and 1) from each reference field.*

Motion Vector Type: Simple Field-based

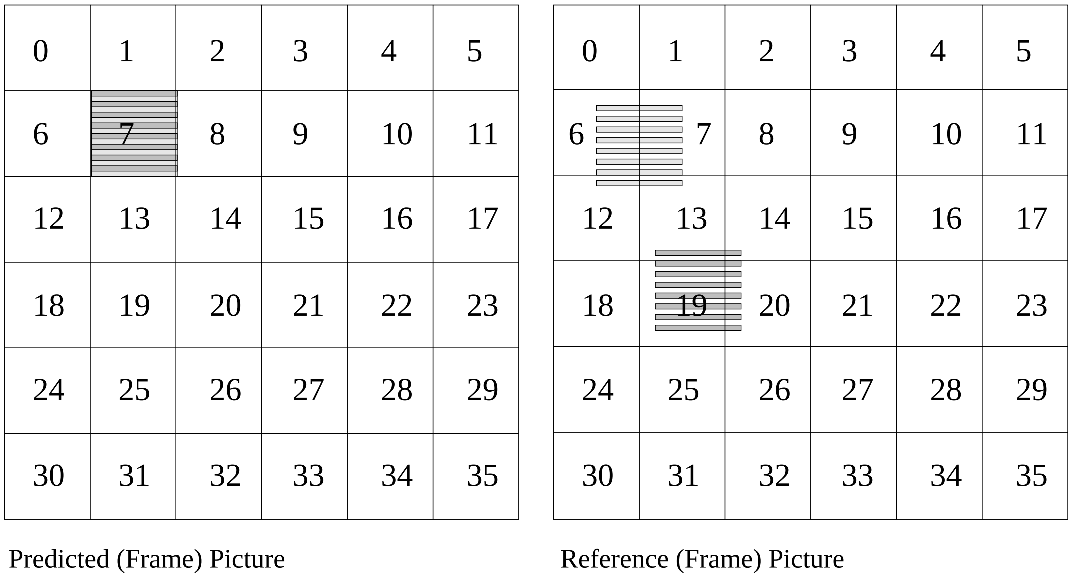

Figure 3.11: *This figure shows an example of the macroblock dependencies produced by a pair of simple field-based motion vectors in a predicted frame picture when the reference picture was also encoded as a frame picture. The pictures are shown divided into macroblocks. Each motion vector typically references 8 alternating rows of a 16 by 16 pixel area of the reference frame. The shaded rows of pixels in the reference frame identify the pixels required to reconstruct the shaded macroblock in the predicted picture. Each motion vector has a field select value indicating whether the alternating rows of data should be taken from the top or bottom field (even or odd-numbered rows) of the reference frame. In this example, macroblock 7 in the predicted picture has two field motion vectors:* $(+6, +30)$ *with a field select value selecting the top reference field, and* $(-16, +2)$ *with a field select value selecting the bottom reference field. Hence, macroblock 7 depends on seven macroblocks (6, 7, 12, 13, 14, 19 and 20) from the reference frame.*

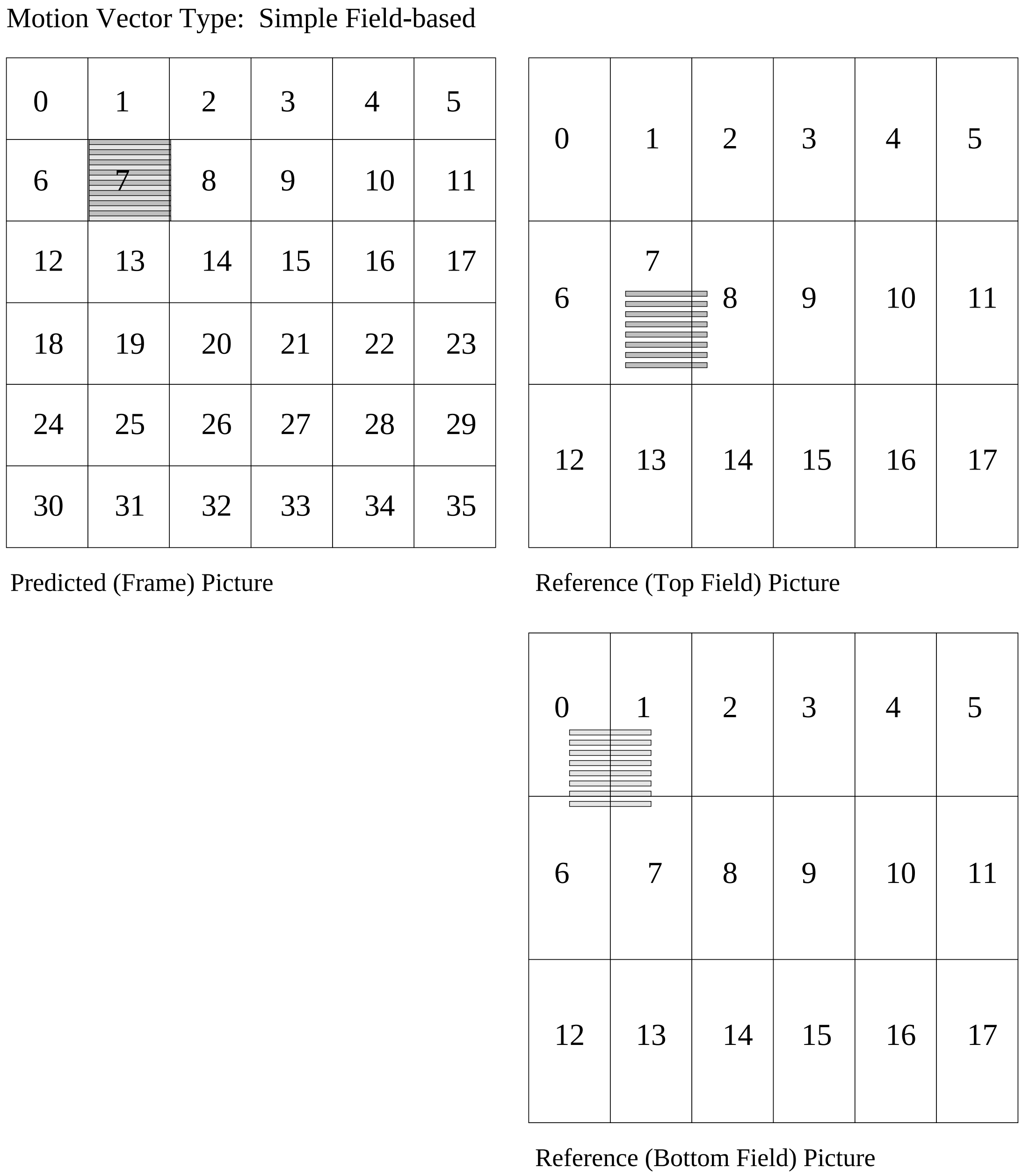

Figure 3.12: *This figure shows an example of the macroblock dependencies produced by a pair of simple field-based motion vectors in a predicted frame picture when the reference picture was encoded as two fields. The pictures are shown divided into macroblocks. Each motion vector typically references 8 alternating rows of a 16 by 16 pixel area of the reference picture. The shaded rows of pixels in each reference field identify pixels required to reconstruct the shaded macroblock in the predicted picture. Each motion vector contains a field select value indicating whether the data should be taken from the top or bottom reference field. In this example, macroblock 7 in the predicted frame has two field motion vectors:* $(+6, +30)$ *with a field select value selecting the top reference field, and* $(-16, +2)$ *with a field select value selecting the bottom reference field. Hence, macroblock 7 depends on six macroblocks: two (7 and 8) from the top reference field and four (0, 1, 6 and 7) from the bottom reference field.*

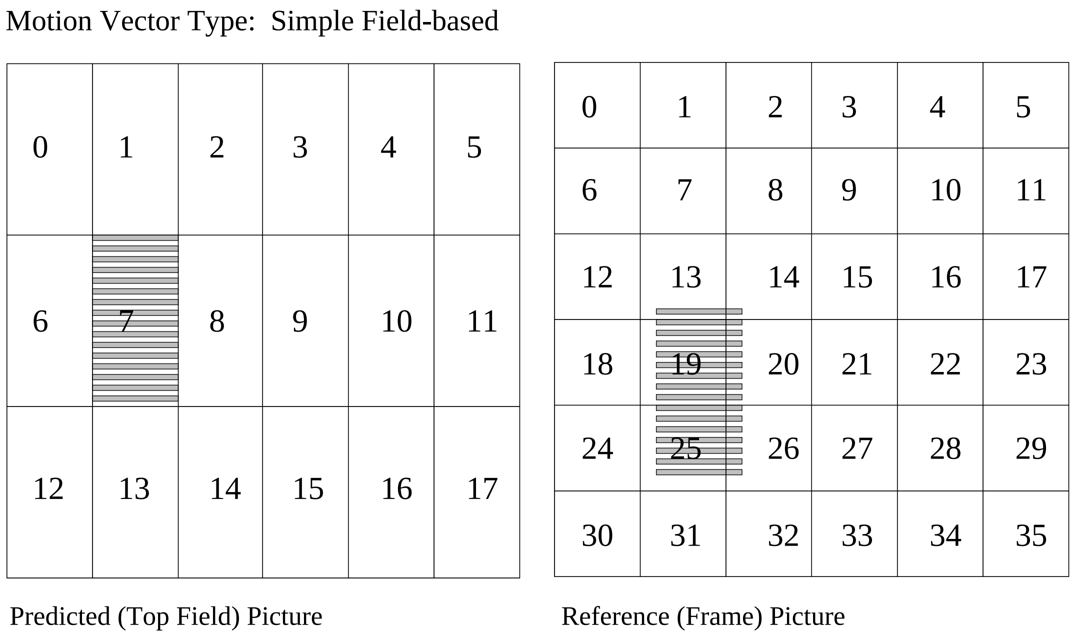

Figure 3.13: *This figure shows an example of the macroblock dependencies produced by a simple field-based motion vector in a predicted field picture when the reference picture was encoded as a frame picture. The pictures are shown divided into macroblocks. A field motion vector typically references 16 alternating rows of a 16 column by 32 row area of the reference picture. The shaded rows of pixels in the reference frame identify the pixels required to reconstruct the shaded macroblock in the predicted field picture. The motion vector's field select value indicates whether the alternating rows of data should be taken from the top or bottom field (even or odd-numbered rows) of the reference frame. In this example, macroblock 7 in the predicted field picture has a field motion vector* $(+6, +14)$ *and a field select value selecting the top reference field. Hence, macroblock 7 depends on six macroblocks (13, 14, 19, 20, 25 and 26) from the reference frame.*

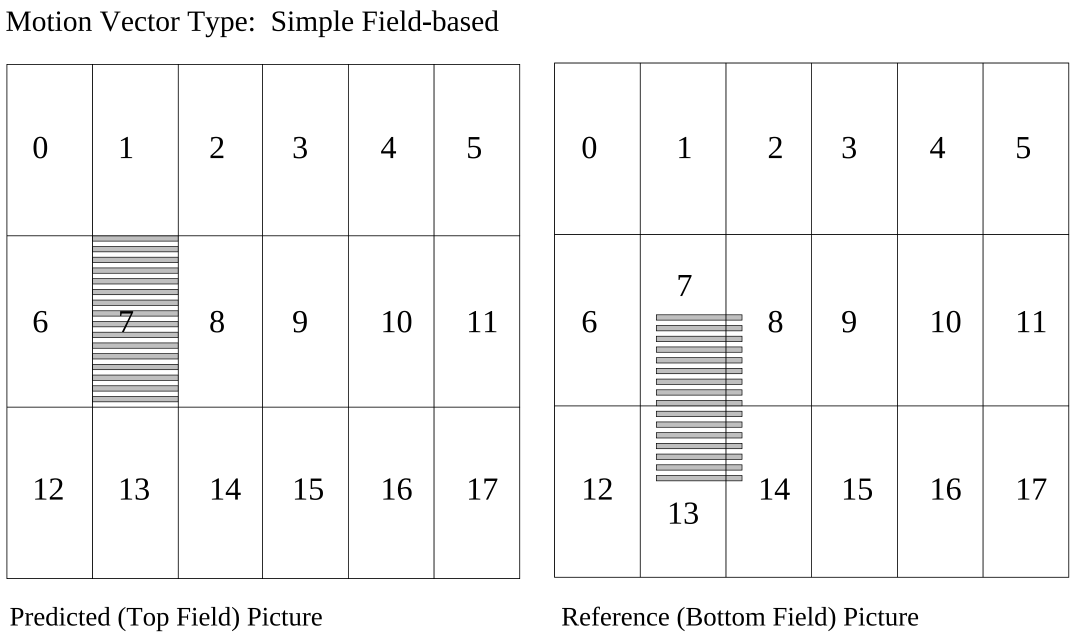

Figure 3.14: *This figure shows an example of the macroblock dependencies produced by a simple field-based motion vector in a predicted field picture when the reference picture was also encoded as two fields. The pictures are shown divided into macroblocks. A field motion vector typically references 16 alternating rows of a 16 column by 32 row area of the reference picture. The shaded rows of pixels in the reference picture identify the pixels required to reconstruct the shaded macroblock in the predicted top field picture. The motion vector's field select value indicates whether the alternating rows of data should be taken from the top or bottom reference field. In this example, macroblock 7 in the predicted top field picture has a field motion vector* $(+6.0, +14)$ *and a field select value selecting the bottom reference field. Hence, macroblock 7 depends on four macroblocks (7, 8, 13 and 14) from the bottom reference field.*

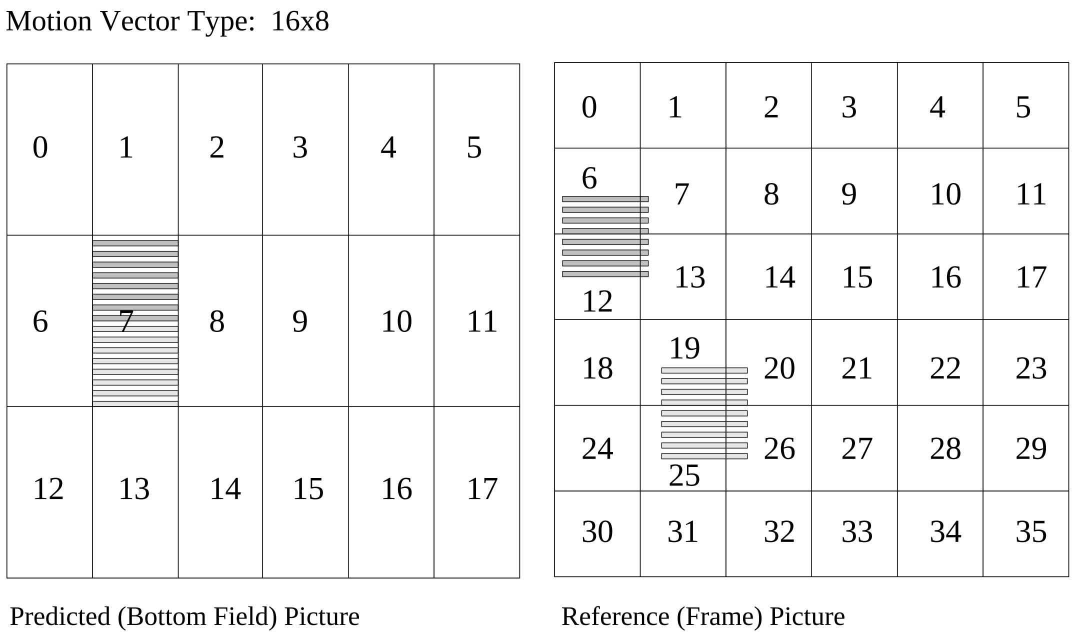

Figure 3.15: *This figure shows an example of the macroblock dependencies produced by 16x8 field motion vectors in a predicted field picture when the reference picture was encoded as a frame picture. The pictures are shown divided into macroblocks. 16x8 field motion vectors always come in pairs, and each motion vector typically references 8 alternating rows of a 16 column by 16 row area of the reference picture. The shaded rows of pixels in the reference frame identify the pixels required to reconstruct the shaded macroblock in the predicted picture. In this example, macroblock 7 has a 16x8 motion vector* $(-28, -8)$ *for predicting the top half of the macroblock, with a field select value selecting the bottom reference field. It also has a 16x8 motion vector* $(+8, +9)$ *for predicting the bottom half of the macroblock, with a field select value that also selects the bottom reference field. Because the second motion vector has an odd-valued* $(+9)$ *vertical displacement value, an extra row of pixels from the reference frame are required to generate between-pixel prediction values for the 8 predicted rows. Macroblock 7 depends on eight macroblocks (6, 7, 12, 13, 19, 20, 25 and 26) from the reference frame.*

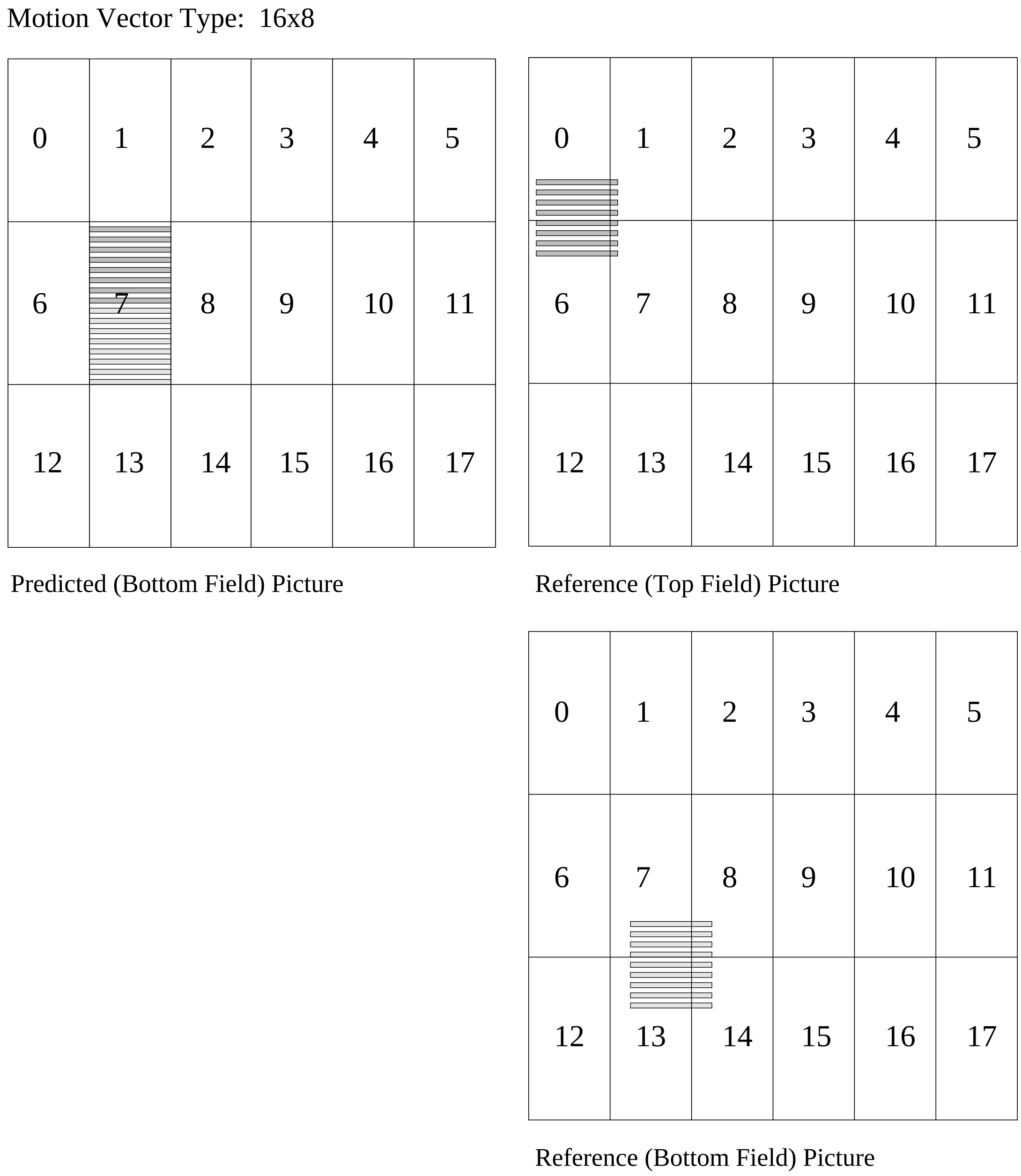

Figure 3.16: *This figure shows an example of the macroblock dependencies produced by 16x8 field motion vectors in a predicted field picture when the reference picture was encoded as two fields. 16x8 field motion vectors always come in pairs, and each motion vector typically references 8 alternating rows of a 16 column by 16 row area of the reference picture. The shaded rows of pixels in the reference fields identify the pixels required to reconstruct the shaded macroblock in the predicted picture. In this example, macroblock 7 in the predicted picture has a 16x8 motion vector* $(-28, -8)$ *for predicting the top half of the macroblock. The motion vector's field select value selects the top reference field. It also has a 16x8 motion vector* $(+8, +9)$ *for predicting the bottom half of the macroblock, with a field select value selecting the bottom reference field. Because the second motion vector has an odd-valued* $(+9)$ *vertical displacement, an extra row of reference field pixels is required to generate between-pixel values for the 8 predicted rows. Macroblock 7 depends on eight macroblocks: four (0, 1, 6 and 7) from the top and four (7, 8, 13 and 14) from the bottom reference field.*

Motion Vector Type: Dual Prime

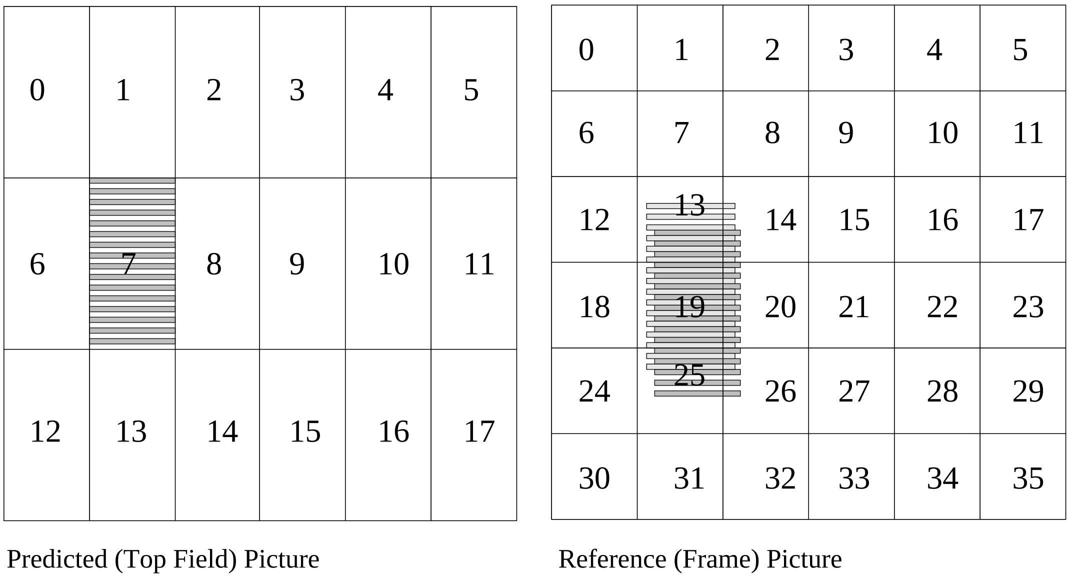

Figure 3.17: *This figure shows an example of the macroblock dependencies produced by dual-prime motion vectors in a predicted field picture when the reference picture was encoded as a frame picture. The pictures are shown divided into macroblocks. The shaded rows of pixels in the reference frame identify the pixels required to reconstruct the shaded macroblock in the predicted picture. In this example, macroblock 7 has a dual-prime motion vector* $(+8, +10)$ *with differential motion vector* $(+1, 0)$*. After scaling, vertical adjustment and adding the differential motion vector, the second motion vector becomes* $(+5, +4)$*. The first motion vector always references data in the reference field having the same parity (top or bottom) as the predicted field; the second motion vector always references data in the opposite parity field. Data referenced by the first motion vector is shaded darker in this figure than data referenced by the second motion vector. Because the second motion vector has an odd-valued* $(+5)$ *horizontal displacement value, it requires an extra column of pixels from the reference field to generate the necessary between-pixel predictions. The predicted values from both motion vectors are averaged for each pixel when forming predictions. Hence, macroblock 7 in the predicted top field picture depends on six macroblocks (13, 14, 19, 20, 25 and 26) from the reference frame.*

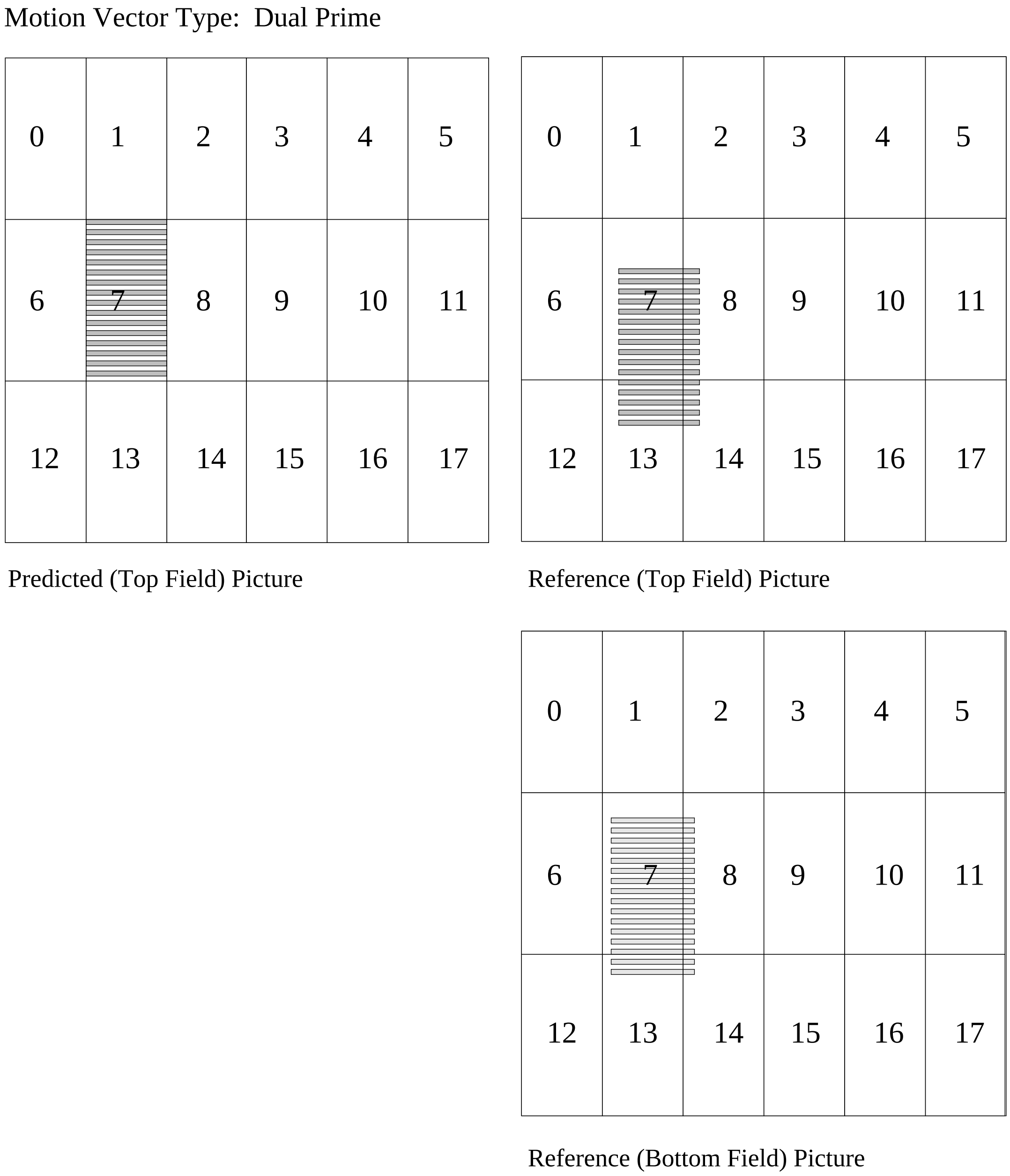

Figure 3.18: *This figure shows an example of the macroblock dependencies produced by a dual-prime motion vector in a predicted field picture when the reference picture was encoded as two fields. The shaded rows of pixels in each of the reference fields identify pixels required to reconstruct the shaded macroblock in the predicted picture. In this example, macroblock 7 has a dual-prime motion vector* $(+8, +10)$ *with differential motion vector* $(+1, 0)$*. After scaling, vertical adjustment and adding the differential motion vector, the second motion vector becomes* $(+5, +4)$*. The first motion vector always references data in the reference field having the same parity (top or bottom) as the predicted field; the second motion vector always references data in the opposite parity field. Data referenced by the first motion vector is shaded lighter in this figure than data referenced by the second motion vector. Because the second motion vector has an odd-valued* $(+5)$ *horizontal displacement, it requires an extra column of reference field pixels to generate the necessary between-pixel predictions. The predicted values from both motion vectors are averaged when predicting each pixel. Hence, macroblock 7 in the predicted top field picture depends on eight macroblocks: four (7, 8, 13 and 14) from the top and four (7, 8, 13 and 14) from the bottom reference field.*

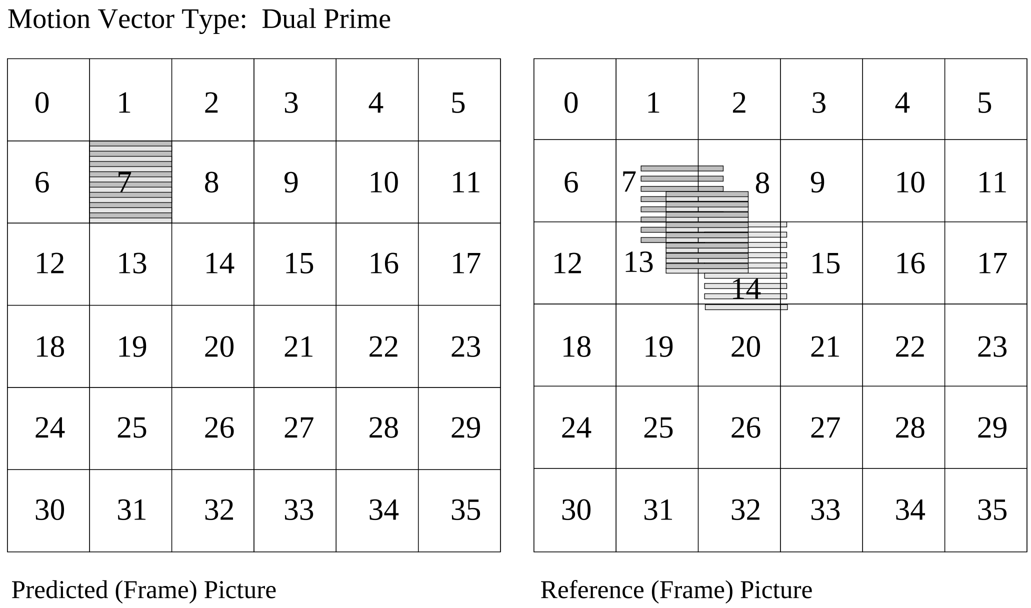

Figure 3.19: *This figure shows an example of the macroblock dependencies produced by dual-prime motion vectors in a predicted frame picture when the reference picture was also encoded as a frame picture. Frame picture macroblock prediction with dual-prime motion vectors requires one full-size motion vector and two differential motion vectors— one for each field of the predicted macroblock. The darkly-shaded rows of pixels in the reference frame identify the pixels required to reconstruct the top field of the shaded macroblock in the predicted picture; the lightly-shaded rows identify pixels required to reconstruct the bottom field. In this example, the top field of macroblock 7 is predicted using vector* $(+24, +10)$ *and differential motion vector* $(0, 0)$*; the bottom field is also predicted using vector* $(+24, +10)$*, but with differential motion vector* $(0, +1)$*. After scaling ("top field first" assumed), vertical adjustment and adding differential motion vectors, the differential vectors for the top and bottom fields are converted to, respectively,* $(+12, +4)$ *and* $(+36, +17)$*. The predicted values from motion vectors* $(+24, +10)$ *in the top reference field and* $(+12, +4)$ *in the bottom reference field are averaged for each pixel to form predictions for the top field of the predicted macroblock; the predicted values from motion vectors* $(+24, +10)$ *in the bottom reference field and* $(+36, +17)$ *in the top reference field are averaged to form predictions for the bottom field. Macroblock 7 in the predicted frame depends on seven macroblocks (7, 8, 13, 14, 15, 20 and 21) from the reference frame.*

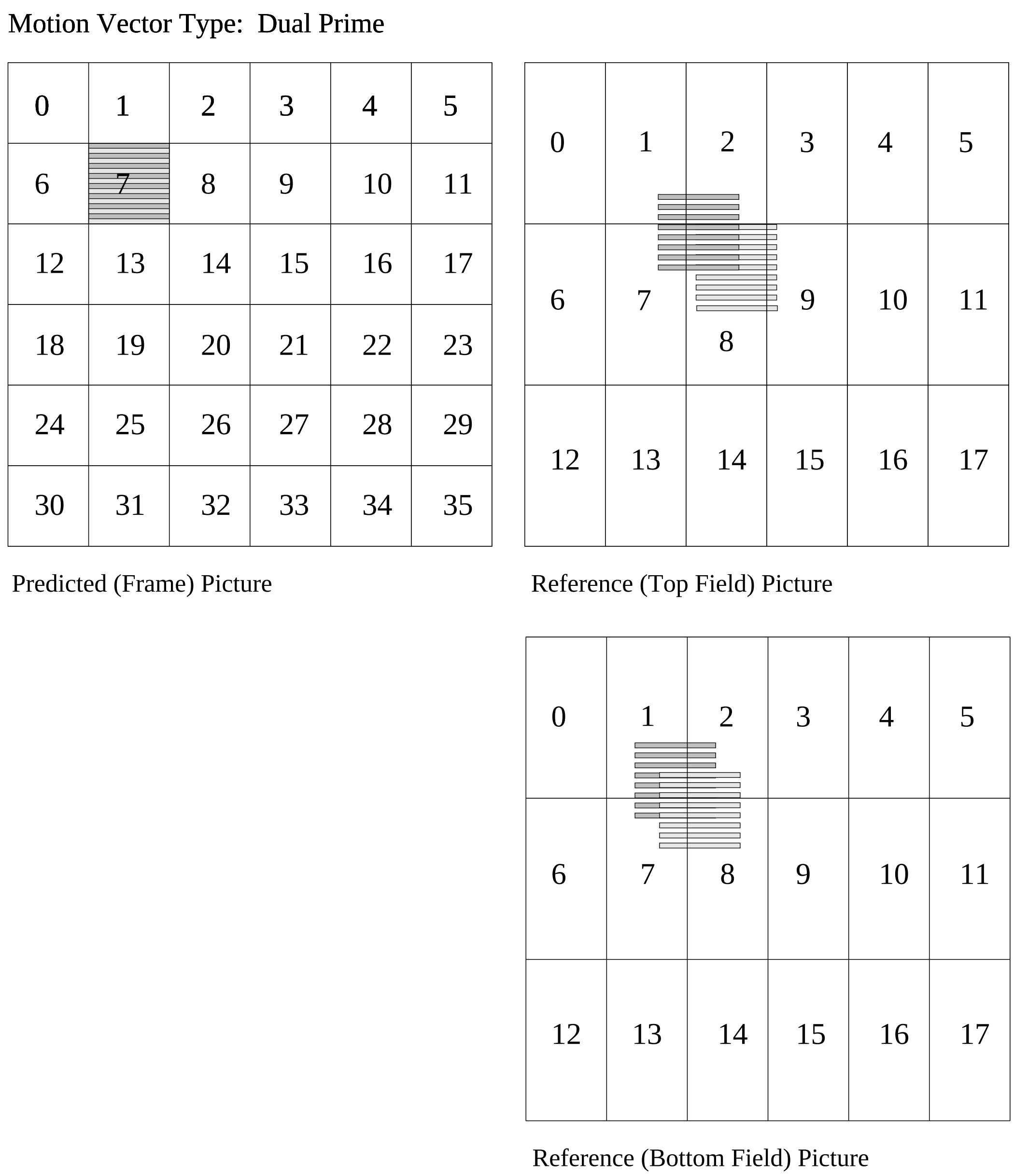

Figure 3.20: *This figure shows an example of the macroblock dependencies produced by dual-prime motion vectors in a predicted frame picture when the reference picture was encoded as two fields. Frame macroblock prediction with dual-prime motion vectors requires one full-size motion vector and two differential motion vectors— one for each field of the predicted macroblock. The darkly-shaded rows of pixels in the reference fields identify the pixels required to reconstruct the top field of the shaded macroblock in the predicted picture; the lightly-shaded rows identify pixels required to reconstruct the bottom field. In this example, the top field of macroblock 7 is predicted using vector* $(+24, +10)$ *and differential motion vector* $(0, 0)$*; the bottom field is also predicted using vector* $(+24, +10)$*, but with differential motion vector* $(0, +1)$*. After scaling ("top field first" assumed), vertical adjustment and adding differential motion vectors, the differential vectors for the top and bottom fields are converted to, respectively,* $(+12, +4)$ *and* $(+36, +17)$*. The predicted values from motion vectors* $(+24, +10)$ *in the top reference field and* $(+12, +4)$ *in the bottom reference field are averaged to predict each pixel in the top field of the macroblock 7; the values from motion vectors* $(+24, +10)$ *in the bottom and* $(+36, +17)$ *in the top reference field are averaged to predict bottom field pixels. Macroblock 7 depends on nine reference macroblocks: five (1, 2, 7, 8 and 9) from the top and four (1, 2, 7 and 8) from the bottom field.*

3.3.3 Summary of Areas Referenced by Motion Vectors

Given the screen coordinates of a predicted macroblock, one of the macroblock's motion vectors, and the referenced and predicted picture structures, it is possible to calculate the screen coordinates of the referenced area in the reference picture. The macroblock dependencies can then be determined by finding the macroblock numbers associated with the pixels in those areas. Table 3.5 shows the areas referenced by any motion vector V in any frame or field picture macroblock m, in terms of reference coordinates (c, r), where (c, r) are the top-left screen coordinates macroblock m would have if it were in a frame picture. Table 3.6 shows the areas referenced by any motion vector V in predicted macroblock m, in terms of the actual top-left screen coordinates (x, y) of the predicted macroblock. Table 3.7 shows the values for a specific example: motion vector $(-4, +12)$ in predicted macroblock 7.

Many predicted macroblocks include more than one motion vector. A macroblock in a predicted frame B-picture that uses data from both a past and a future reference picture, for example, can have four 16x8 motion vectors: one forward pair and one backward pair. The set of macroblocks referenced by any predicted macroblock is the union of the set of macroblocks referenced by each of its motion vectors.

The number of macroblocks that can be referenced by any predicted macroblock is bounded. If the macroblock has motion vectors, the number is dependent on both the motion vector type and the structure of the referenced pictures. The number of macroblocks that can be referenced by the motion vectors in any predicted P-picture macroblock are summarized in Table 3.8 for macroblocks in predicted frames and Table 3.9 for macroblocks in predicted fields. Macroblocks in predicted B-pictures

Area referenced by Macroblock m with motion vector $V = (2j+p,\ 2k+q)$, where $p, q \epsilon \{0,1\}$.					
Predicted Picture				Referenced Picture	
Picture Struct	MB Coord. (top left)	Motion Vector		Referenced Area	
		Motion Vector Type	Field Select	top left Coordinates	bottom right Coordinates
Frame	$(c,\ r)$	Frame-based	–	$(c+j,\ r+k)$	$(c+j+p+15,$ $r+k+q+15)$
		Field-based or 1st Dual-prime	Top	$(c+j,\ r+2k)$	$(c+j+p+15,$ $r+2(k+q)+14)$
		Field-based or 1st Dual-prime	Bot.	$(c+j,\ r+2k+1)$	$(c+j+p+15,$ $r+2(k+q)+15)$
Field		Field-based or 1st Dual-Prime	Top	$(c+j,\ 2r+2k)$	$(c+j+p+15,$ $2r+2(k+q)+30)$
Top or	$(c,\ 2r)$	16x8 (Top)			$(c+j+p+15,$ $2r+2(k+q)+14)$
Bottom	$(c,\ 2r+1)$	16x8 (Bottom)		$(c+j,\ 2r+2k+16)$	$(c+j+p+15,$ $2r+2(k+q)+30)$
		Field-based or 1st Dual-prime	Bot.	$(c+j,\ 2r+2k+1)$	$(c+j+p+15,$ $2r+2(k+q)+31)$
		16x8 (Top)			$(c+j+p+15,$ $2r+2(k+q)+15)$
		16x8 (Bottom)		$(c+j,\ 2r+2k+17)$	$(c+j+p+15,$ $2r+2(k+q)+31)$

Table 3.5: *The area referenced by predicted macroblock m with motion vector $V = (2j+p,\ 2k+q)$, where $p, q \epsilon \{0,1\}$ and the top left screen coordinates are $(0,0)$.*

Predicted Picture		Referenced Picture		
Picture Struct	Motion Vector Type	Referenced Area		
		Fields	top left	bottom right
Frame	Frame-based	2	$(x+j,\ y+F_v+k)$	$(x+j+p+15,$ $y+F_v+k+q+15)$
	Field-based or 1st Dual-prime	1	$(x+j,\ y+F_v+2k)$	$(x+j+p+15,$ $y+F_v+2(k+q)+14)$
Field	Field-based or 1st Dual-Prime	1	$(x+j,\ y+F_v+2k)$	$(x+j+p+15,$ $y+F_v+2(k+q)+30)$
	16x8 (Top)			$(x+j+p+15,$ $y+F_v+2(k+q)+14)$
	16x8 (Bottom)		$(x+j,\ y+F_v+2k+16)$	$(x+j+p+15,$ $y+F_v+2(k+q)+30)$

Table 3.6: *Area referenced by predicted macroblock m with top left pixel at (x, y) and motion vector $V = (2j+p,\ 2k+q)$, where $p, q \epsilon \{0, 1\}$. Vertical field shift factor F_v is the value F_{vshift} defined by Equation 3.11 on page 48, and the top left screen coordinates are $(0, 0)$.*

Referenced Coordinates for Macroblock 7 having motion vector $(-4, +12)$					
Predicted Picture				Referenced Picture	
Picture Structure	MB 7 Coord. (top left)	Motion Vector Type	Field Select	Referenced Area top left	bottom right
Frame	(16,16)	Frame-based	–	(14,22)	(29,37)
		Field-based or 1st Dual-prime	Top	(14,28)	(29,42)
		Field-based or 1st Dual-prime	Bottom	(14,29)	(29,43)
Field (Top or Bottom)	(16,32) (16,33)	Field-based or 1st Dual-Prime	Top	(14,44)	(29,74)
		16x8 (Top)			(29,58)
		16x8 (Bottom)		(14,60)	(29,74)
		Field-based or 1st Dual-prime	Bottom	(14,45)	(29,75)
		16x8 (Top)			(29,59)
		16x8 (Bottom)		(14,61)	(29,75)

Table 3.7: *Screen coordinates for the top left pixel of macroblock 7 and the area referenced by its motion vector* $(-4, +12)$*. Pictures are assumed to have a 96 x 96 pixel (6 x 6 macroblock) frame size, and the top left screen coordinates are* $(0, 0)$*.*

can reference zero, one or two reference pictures. For B-pictures that reference only one reference picture, the values are identical to those for P-pictures (see Tables 3.8 and 3.9); for B-pictures that reference two reference pictures, the values can be obtained by considering each reference picture separately, and summing the results.

Consider a predicted B-picture macroblock with frame-based motion vectors. Assume the macroblock references a previous reference picture that was encoded as two fields and a future reference picture that was encoded as a single frame. According to Table 3.8, the predicted macroblock can reference 2, 4 or 8 macroblocks from the previous reference picture; and 1, 2 or 4 macroblocks from the future reference picture. Summing all possible combinations of the results, we find that the predicted B-picture macroblock can reference 3-6, 8-10 or 12 macroblocks.

By using only a subset of the allowable picture structures, types and ranges of motion vectors in a video, an encoder can limit the number of dependencies produced by the macroblocks it generates. In any case, it is not possible for a predicted macroblock to directly reference more than 16 reference picture macroblocks[3].

3.4 Hierarchical Video Dependencies

As mentioned in Chapter 2, MPEG-1 and MPEG-2 video streams are comprised of a hierarchy of layers: the Sequence of Pictures layer, Group of Pictures layer, Picture layer, Slice layer, Macroblock layer and Block layer. Most of the layers include headers containing essential decoding and structural information about data in their own and

[3] B-pictures cannot have dual-prime motion vectors, so the motion vector type for B-picture macroblocks will always be frame-based, field-based or 16x8.

Number of Macroblocks that can be Referenced by a Predicted P-picture Frame Macroblock			
Motion Vector Type	Reference Picture Structure		
	Frame	2 Fields	Frame & Field
Frame-Based	1,2,4	2,4,8	2-4,6,8
Field-Based	1-8	1-8	1-8
Dual-Prime	1-12	2-16	2-16

Table 3.8: *This table shows the possible values for the number of reference picture macroblocks that can be referenced by a predicted macroblock in a frame P-picture.*

Number of Macroblocks that can be Referenced by a Predicted P-picture Field Macroblock			
Motion Vector Type	Reference Picture Structure		
	Frame	2 Fields	Frame & Field
Field-Based	2,3,4,6	1,2,4	–
16x8	1-8	1-8	–
Dual-Prime	2-12	2-6,8	3-8,10

Table 3.9: *This table shows the possible values for the number of reference picture macroblocks that can be referenced by a predicted macroblock in a P-picture field.*

subordinate layers [12]. If any of the higher-layer headers are lost or corrupted, it may be impossible to correctly interpret and decode data in any of the dependent layers. When prioritizing video data for redundant transmission, the DART system takes into consideration both the motion vector dependencies discussed in Section 3.3 and the hierarchical dependencies discussed in this section. The video layers can be summarized as follows:

- Sequence of Pictures (SEQ) - A header that specifies common information about a sequence of pictures, including horizontal and vertical size, aspect ratio, frame rate, and quantization matrices. MPEG-2 sequence of pictures headers can also provide additional information related to scalability and other features. None of the pictures in the sequence can be successfully decoded without this information.

- Group of Pictures (GOP) - A header that contains a time stamp and continuity information about the picture immediately preceding it. This header is mainly used to help with random access into video streams. The temporal reference number is reset to zero at the start of a group of pictures and incremented for each successive picture. Other than that, this header does not have a significant impact on the data stream. Because the change in temporal reference numbering might have an adverse affect some video decoders, however, we consider all of the pictures in a group of pictures to be dependent on the group of pictures header in our data model – although it would not be unreasonable to assume otherwise.

- Picture (PIC) - A header that identifies the picture type (I, P or B) and other information such as motion vector scaling factors that are required when decoding the picture's slices and macroblocks. For MPEG-2 pictures, this header also indicates whether the picture is a field or frame picture.

- Slice (SLI) - A header that identifies the screen position of the macroblocks that follow it. The slice header also contains quantization information needed to decode macroblocks. Motion vector predictors are reset at the start of each slice. None of the macroblocks immediately following the slice header can be successfully decoded without this information. Because slices are the smallest synchronizable units in MPEG video streams, having multiple slices in a picture helps insure against data loss.

- Macroblock (MB) - Each macroblock contains the majority of the coding information for a 16 by 16 pixel area of a picture. Because macroblock information is run-length encoded, coding errors cause loss of synchronization. Because synchronization cannot be restored until the next slice or picture, macroblocks are dependent on all other macroblocks preceding them in the slice.

A decoder can lose synchronization due to encoding errors, lost or corrupted data, or various other reasons. A decoder restores synchronization by discarding data bits until a start code is encountered in the data stream. The Sequence of Pictures layer, Group of Pictures layer, Picture layer and Slice layer of MPEG-1 and MPEG-2 video streams all begin with designated start codes. Because the macroblock and block layers do not begin with start codes, if synchronization is lost while decoding a video block or macroblock, the remaining data is discarded until the start of the next slice, picture, or other synchronizable unit.

The hierarchical dependencies in any MPEG video can be represented as a dependency graph, where the nodes in the graph represent video elements and the directed lines or arcs between nodes represent the dependencies between elements. A sample hierarchical dependency graph for a sequence of pictures is shown in Figure 3.21. The directed lines in Figure 3.21 point from dependent video elements to the elements directly above them in the hierarchy and upon which they depend. Each sequence of pictures in a video has an independent tree structure with a "Sequence of Pictures" header at its root, as shown in Figure 3.22. The loss of any node in the graph results in the loss of all other nodes that directly or indirectly point to it. Thus, the loss of a "Sequence of Pictures" header results in the loss of all data in that sequence; the loss of a "Group of Pictures" header results in the loss of all pictures in the group; the loss of a "Picture" header results in the loss of all slices and macroblocks in the picture; the loss of a "Slice" header results in the loss of all macroblocks in the slice; and the loss of a macroblock results in the loss of all subsequent macroblocks in the slice.

3.5 Combining the Video Dependencies

3.5.1 Dependency Graphs

The macroblock dependencies produced by motion vectors and the hierarchical MPEG layer dependencies can be combined into a single set of dependencies that can be represented in a dependency graph like the one in Figure 3.23. Unlike hierarchical dependency graphs such as the one in Figure 3.21, the sequences of pictures within

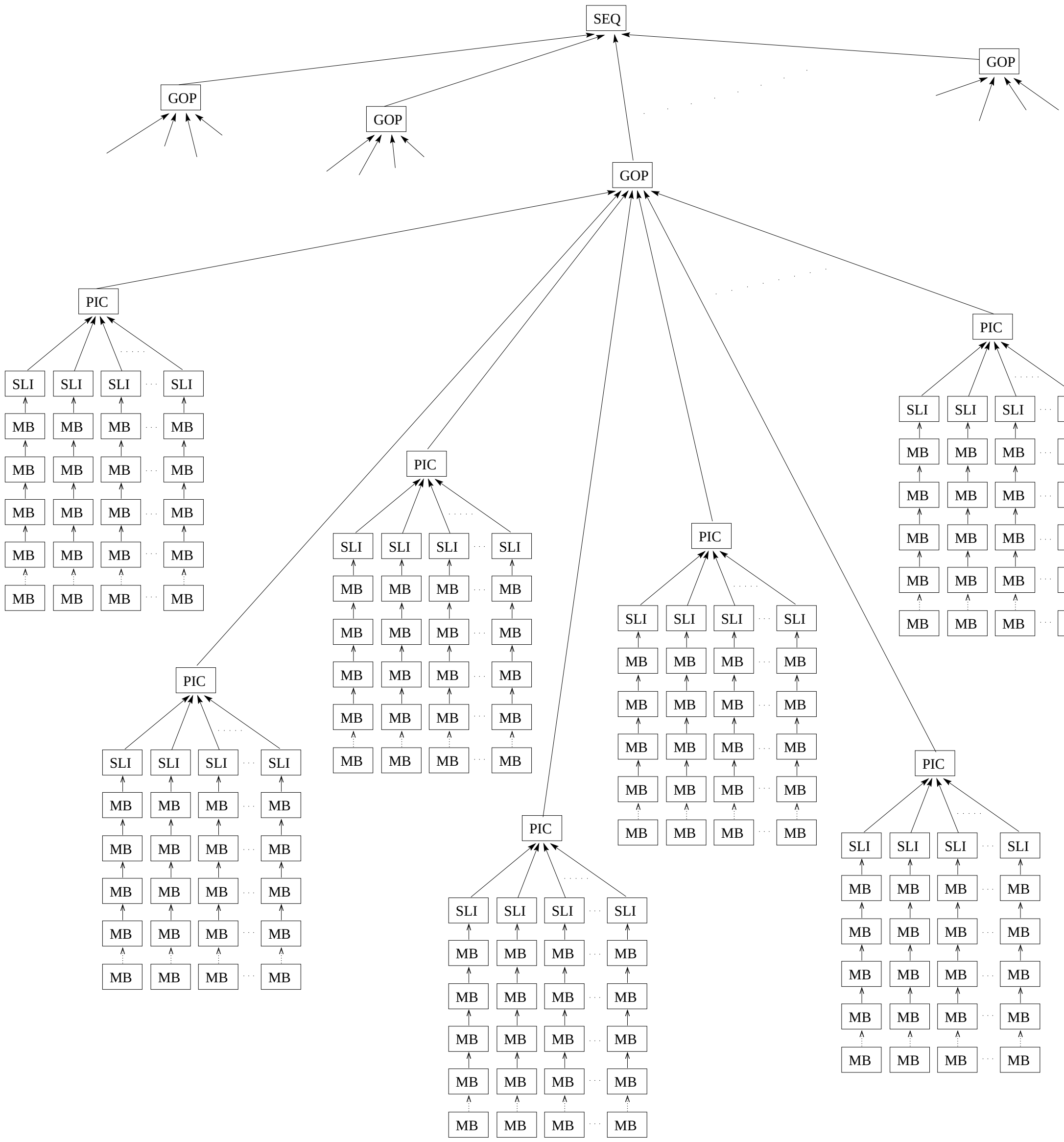

Figure 3.21: *This dependency graph shows the hierarchical dependencies between video elements in a sample MPEG video Sequence of Pictures. Directed lines point from video elements to other elements upon which they depend.*

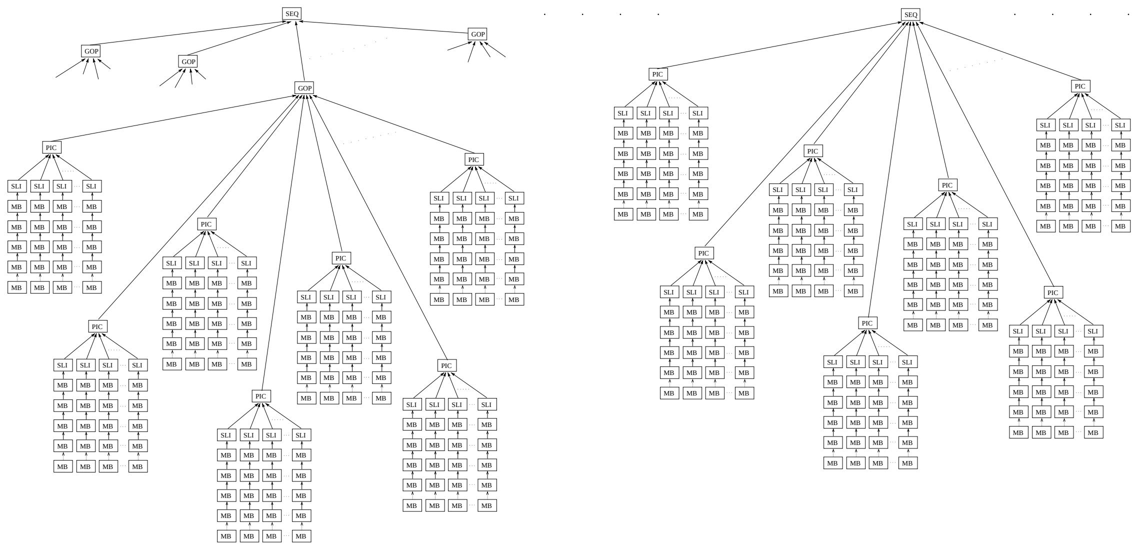

Figure 3.22: *This dependency graph shows the hierarchical dependencies for a video clip that includes several sequences of pictures. Each sequence of pictures has an independent tree structure. The rightmost sequence does not contain any Group of Pictures headers.*

combined dependency graphs do not necessarily have tree structures. In fact, the sequences will not have tree structures unless all of their macroblocks are intra-coded, as occurs for sequences comprised entirely of I-pictures. In such cases the dependency graphs degenerate to hierarchical dependency graphs. Although the combined dependency graphs do not necessarily have a tree structure, they never contain circular dependencies.

The MPEG video dependencies can be examined at different levels of detail. Figure 3.23, for example, shows the dependencies at the macroblock level. The dependencies can also be examined at the slice level by replacing the directed lines between the macroblocks in Figure 3.23 with similarly-directed lines between the slices above them in the MPEG hierarchy. The macroblock nodes can then be removed, resulting in the graph in Figure 3.24. Similarly, the dependencies can be examined at the picture level by eliminating the slices, resulting in the graph in Figure 3.25.

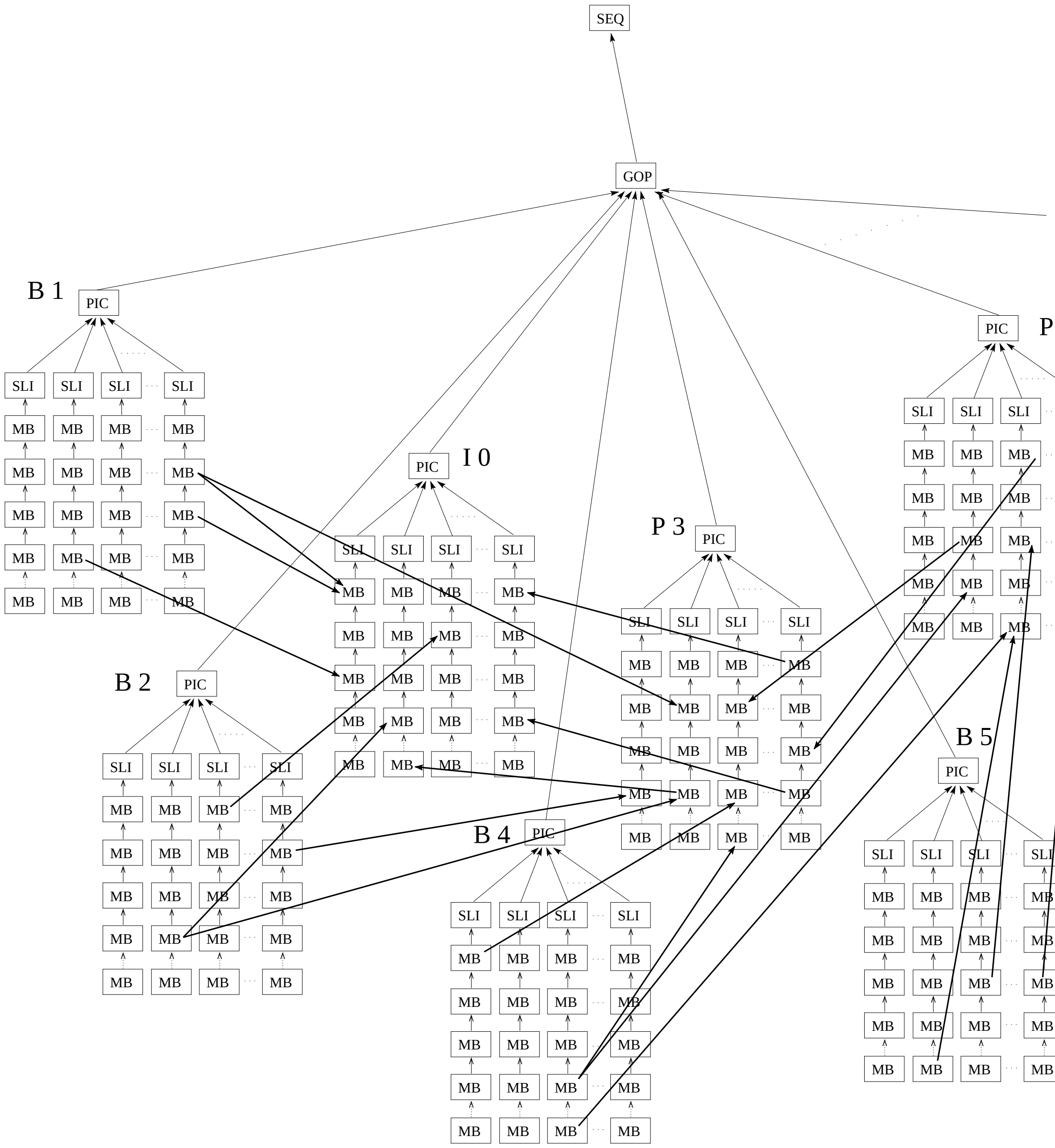

Figure 3.23: *This figure shows a few sample macroblock dependencies for an MPEG video stream comprised of I-B-B-P-B-B-P pictures. In a typical video sequence, each picture would contain more slices, each slice would contain more macroblocks, and there would be many more dependencies between macroblocks.*

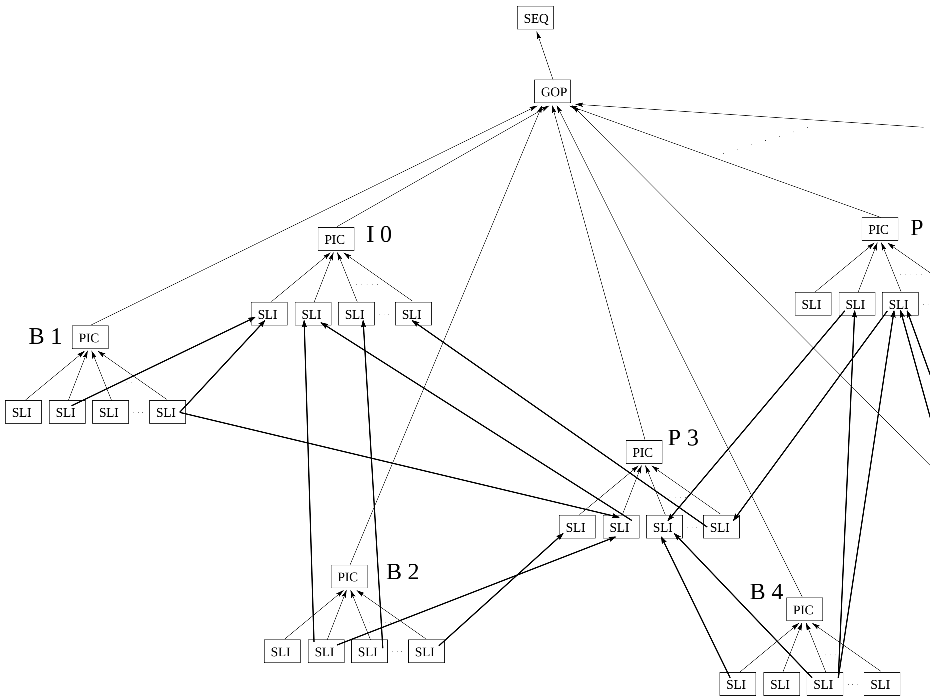

Figure 3.24: *This figure shows the slice dependencies for the sample I-B-B-P-B-B-P MPEG pictures shown in Figure 3.23.*

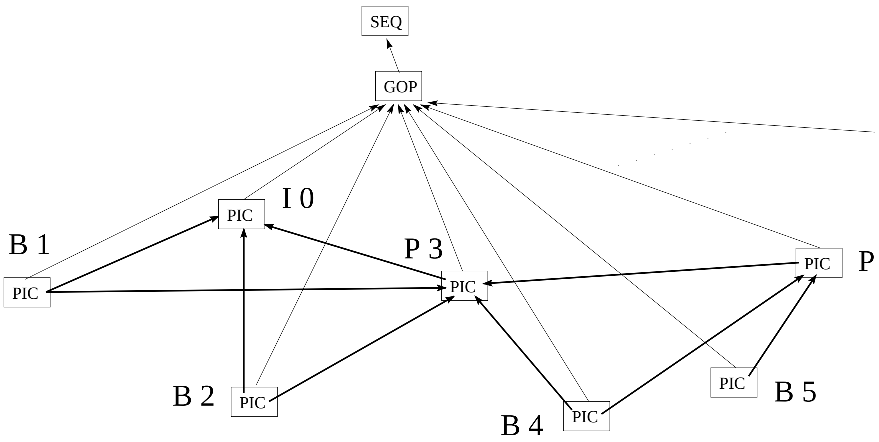

Figure 3.25: *This figure shows the picture dependencies for the sample I-B-B-P-B-B-P MPEG pictures shown in Figures 3.23 and 3.24.*

Note that the I, P and B-picture dependencies discussed in Section 3.2 and shown in Figures 3.1 and 3.2 are entirely the result of motion vectors in the predicted macroblocks. The I, P and B designations do not inherently cause dependencies; instead, they inform MPEG encoders/decoders as to which reference pictures are allowed to be used when forming predictions for the designated picture. Figures 3.23 and 3.25 make this distinction clear. Although B-picture B5 is allowed to rely on data from macroblocks in reference pictures P3 and P6, all of it's macroblocks are either intra-coded or use data from P6. As a result, B5 is not at all dependent on the P3, as would normally be inferred by its B-picture designation.

Due to the fact that slices are the smallest synchronization units in MPEG video streams, it makes sense to treat them as the smallest dependencies when making data replication decisions. If slices are treated as the smallest dependency units, successful decoding of an entire slice depends upon successful decoding of all of the slices (from reference I-pictures and P-pictures) containing macroblocks upon which the slice depends, as shown in Figure 3.24.

3.5.2 Graph Coloring

The introduction of redundant data into MPEG streams can be thought of as a graph coloring problem, where the graphs are as shown in Figures 3.23 to 3.25. Let all of the directed lines originating from the same node (SEQ, GOP, PIC, SLI or MB) be colored with a different color. Let D_{mv} be the maximum number of motion vector dependencies for a given node N, as allowed by Tables 3.8 and 3.9. Because at most one line originating from any given node represents a hierarchical dependency, then at most $D_{mv}+1$ lines can originate from node N, requiring at most $C = D_{mv}+1$

unique colors. Let C_{max} be the maximum number of lines originating from any node in the graph. Then the graph can be colored using exactly C_{max} colors.

Each line can be given a weight $W > 0$ according to its importance in the data stream, where higher weights represent greater importance. Because each line represents a dependency relationship, the weights could be assigned as a function of the number of macroblocks affected by each dependency. For each node N that is the source of less than C_{max} lines, continue adding lines having weight 0 and directed from N to itself, until each node in the graph is the source of exactly C_{max} lines of C_{max} colors. An alternative approach would be to assign a suitable "required color count" C_{req} to each node, rather than adding the 0-weighted lines. If this approach is used, the value of C_{req} will vary from node to node, but each node in the graph will be the source of its own C_{req} lines of C_{req} colors.

Redundant data can be added to the data stream based upon the relative importance of the various nodes, as determined by the weighted lines pointing to them in the graph. Multiple redundant copies could even be added for extremely critical data. Let a new node N_{new} be added to the graph for each new piece of redundant data that is added to the data stream, and let N_{orig} be the original graph node containing the data being duplicated. Further, for each line directed from any node N_{any} to node N_{orig}, let a new line of the same color be drawn from N_{any} to the new node N_{new} and let the weight of the original line be split evenly between the original and new lines. Nodes could continue to be added to the graph, and the corresponding data added to the data stream, until some criterea such as a maximum line weight for the graph is achieved.

After establishing such a graph and transmitting the associated data stream, data losses (lost nodes) can be modeled by removing from the graph all lines that originate or terminate at the lost data nodes. Usable data corresponds to original data nodes having lines of all C_{max} colors (or C_{req} colors, if the alternate approach is being used) directed away from them in the resulting graph.

Although this graph-centered approach determines how many copies of each data element should be replicated, it does not address the issue of where the redundancies should be placed in the data stream. The DART system does not currently use a graph-centered approach for data replication, but such an approach could be considered in future research.

Chapter 4

MPEG-2 Systems Streams

The systems part of the MPEG-2 standard [11] defines two different systems streams: a transport stream for use in environments such as ATM networks where dropped packets and other errors are likely to occur; and a program stream for use in more controlled environments where errors are unlikely. Because this thesis addresses the problem of lost video data in networks, the DART system uses MPEG-2 transport streams to transfer MPEG video data. This chapter includes an overview of MPEG-2 systems streams and a discussion of how the DART system constructs MPEG-2 systems transport streams for video data.

4.1 Systems Transport Stream Packets

MPEG-2 systems transport streams consist of a series of transport stream packets, each 188 bytes in length. Each transport stream packet contains a 4-byte transport stream header and 184 bytes of data. The header includes a sync byte and a 13-bit packet identifier (PID). The PID identifies the stream to which the data in the remainder of the packet belongs. The packet data can be part of an elementary stream (a coded video, audio or other bit stream) or part of a system table that maps sets of elementary streams into programs, lists available programs, or provides network or descrambling information. Data such as MPEG reference pictures that are too large to fit inside a single MPEG-2 systems transport stream packet can be segmented and sent in multiple packets. Packets from different elementary streams can be interleaved in the system stream, as shown in Figure 4.1.

4.2 Hierarchical Dependencies in Systems Transport Streams

Some systems transport stream packets have a more widespread effect on data stream decoding than others. The Program Access Table (PAT), for example, contains a list of programs being carried in the stream and their associated PIDs and timing information. If the transport stream packets containing the PAT table are lost, the remainder of the stream is virtually unusable. Similarly, the PAT identifies a Program Management Table (PMT) for each program in the stream. Each PMT includes a list of elementary streams (and their respective PIDs) associated with the program. The PMT for one program, such as program 1 shown in Figure 4.2, might list 4 elementary streams: a low-resolution MPEG-2 video stream, a high-resolution MPEG-2 video stream, an English audio stream and a Japanese audio stream. The PMT for another program, on the other hand, might list only 1 or 2 elementary streams: an MPEG-1 video stream and/or an audio stream. If the transport stream packets for any of the PMT tables are lost, all of the elementary stream mappings associated

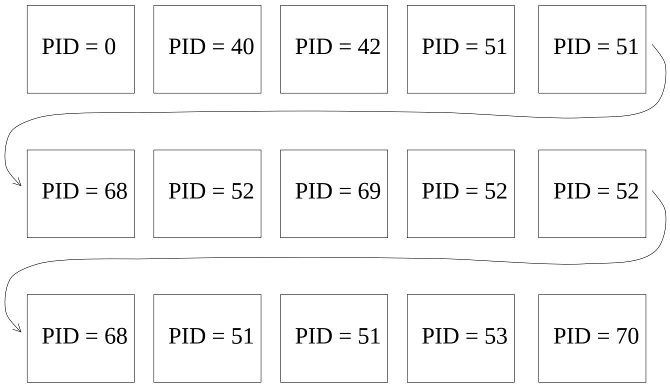

Figure 4.1: *This figure shows a sample transmission order for a set of MPEG-2 systems transport stream packets. Packets from different elementary streams can be interleaved in the system stream.*

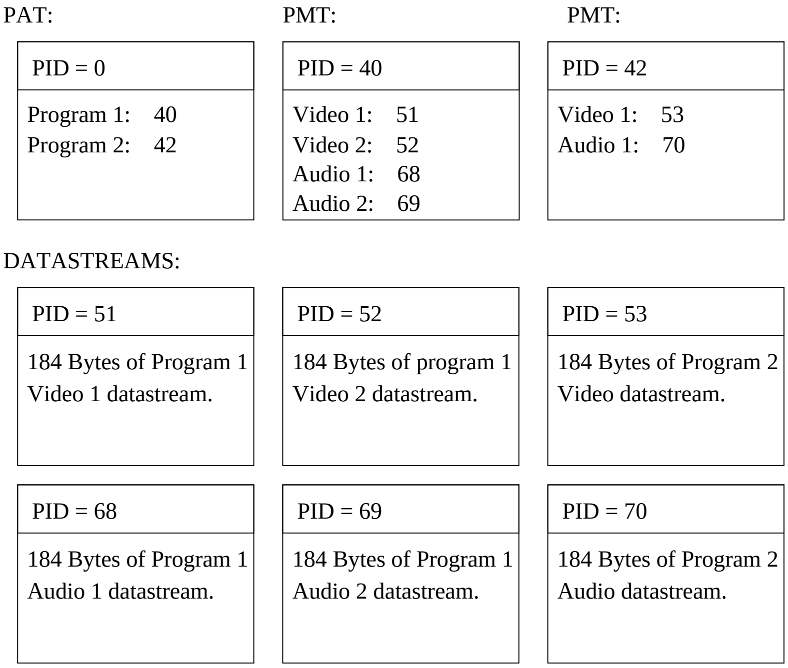

Figure 4.2: *This figure shows sample MPEG-2 systems transport stream packets, with a PID value in each header and a summary of the packet contents in the data portion of each packet. PID 0 is reserved for Program Association Table (PAT) packets, which list all of the programs carried by the systems transport stream and their associated Program Map Table PIDs. Program Map Table (PMT) packets list the PID and stream type for each elementary stream (audio, video, etc.) associated with a given program.*

with that program are lost. Loss of other tables, such as Conditional Access Tables (CATs) containing descrambling information for elementary streams, or the Network Information Table (NIT), can have similarly far-reaching effects. The PAT, PMT, CAT, NIT and other MPEG-2 systems tables are commonly referred to as Program Specific Information (PSI) data.

4.3 PES Packets in Systems Transport Streams

Timing and synchronization information must typically be added to MPEG video and other elementary stream data before they are included in systems transport streams. This is accomplished by packetizing the elementary stream data into Program Elementary Stream (PES) packets.

Each PES packet has a header that includes a packet length and stream id. Most PES packet headers include additional fields such as decode and presentation time stamps for the packet data, elementary stream clock references, and CRCs. A PES packet header with two time stamps, a clock reference, and a CRC, for example, occupies a minimum of 25 bytes.

After creating the PES packets for the elementary stream data, the PES packets must then be placed into the data fields of MPEG-2 systems transport stream packets. Because MPEG-2 systems transport stream packets have a fixed 188-byte size and PES packets are of variable length, this often requires the PES packets to be segmented. If a PES packet is small enough, it is placed directly into the data portion of a single MPEG-2 systems transport stream packet; otherwise, it is segmented so that the header and data are spread across several transport stream packets, as shown

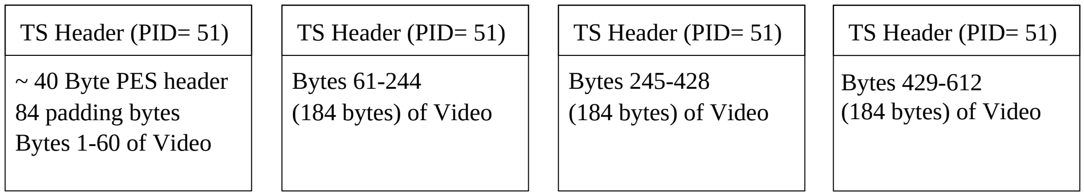

Figure 4.3: *This figure shows a 612-byte video picture segmented into four MPEG-2 systems transport stream packets. When using MPEG-2 systems transport streams, PES packets must always contain a multiple of 184 bytes. Padding bytes can be added at the start of the PES packet to pad the data to a 184-byte boundary.*

in Figure 4.3.

The data portion of MPEG-1 and MPEG-2 video elementary stream PES packets contains the actual MPEG video data. As mentioned in Section 3.4, MPEG video data is arranged in a hierarchy of layers: the Sequence of Pictures layer, Group of Pictures layer, Picture layer, Slice layer, Macroblock layer and Block layer. A sequence of pictures header, group of pictures header, picture header and picture data (slices, macroblocks, etc.) can all be put into the data field of the same PES packet, but a PES packet cannot contain data from more than one picture. PES packets containing picture headers must also include decode time stamps in the PES packet header. Display time stamps are also required for some reference pictures.

MPEG decoders can lose synchronization as a result of encoding errors, lost or corrupted packets, and other errors. Both MPEG systems stream and MPEG video decoders restore synchronization by discarding data bits until an appropriate start code is encountered in the data stream. All PES packets begin with the same 4-byte "PES packet" start code.

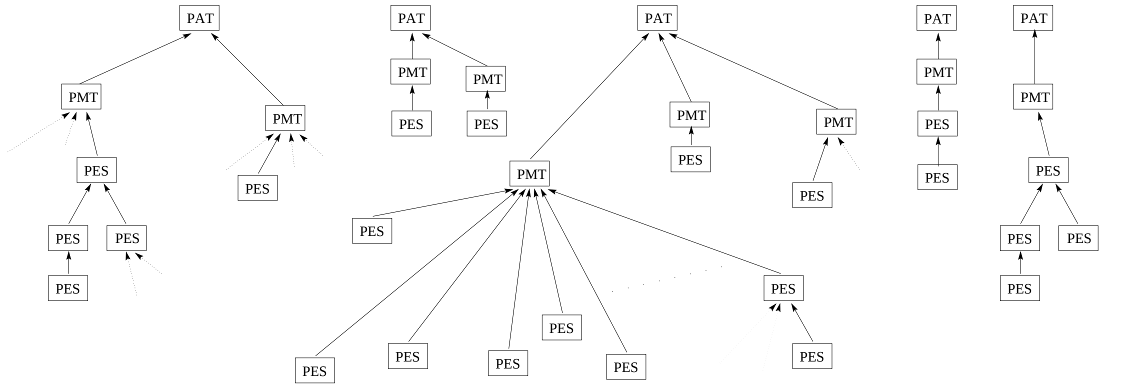

Figure 4.4: *This figure shows a sample hierarchy dependency graph for an MPEG-2 systems transport stream. Systems transport stream graphs are comprised of a series of trees with a Program Access Table (PAT) at each root. One or more Program Map Tables (PMTs) are dependent on each PAT. Each program consists of one or more elementary streams, each comprised of Program Elementary Stream (PES) packets. PES connectivity is determined by the structure of each elementary stream. Dotted lines in the figure represent a subset of locations where additional branches could occur.*

4.4 Implications of Systems Transport Stream Packet Dependencies

The dependencies among PAT, PMT, PES and other MPEG-2 systems transport stream packets can be represented in dependency graphs similar to those used in Section 3.4 for MPEG video streams. A sample hierarchical dependency graph is shown in Figure 4.4. For MPEG-2 systems transport streams carrying MPEG video data, the PES packets contain video data having dependencies such as those shown in Figure 3.21. The data dependencies between video elements residing in different PES packets produce the dependencies among PES packets shown in Figure 4.4.

Because PSI data is so fundamental to the successful decoding of MPEG-2 systems transport streams, the MPEG-2 systems standard allows multiple copies of PAT, PMT and other system tables to be included in transport streams. Annex C of the MPEG-2 systems standard [11] discusses the tradeoffs between the increased overhead and the greater accessability that result from having multiple copies of the tables. Frequent transmission of the PSI data allows new viewers quick access to the video and other elementary stream data, but adds overhead to the systems stream; infrequent transmission of the tables results in a greater initial delay but causes less overhead. Annex C suggests that transmitting PAT and PMT tables 25 times per second allows decoders reasonable access to changes in program tables without introducing unreasonable overhead.

If the 25 times per second suggestion is followed, lost PSI packets will not have a significant impact on MPEG-2 systems transport streams because redundant copies of the tables will be available within less than one picture time (assuming 30 frames per second video). As a result, this thesis makes no attempt to transmit redundant copies of PAT, PMT, or other PSI tables.

4.5 Constructing MPEG-2 Systems Transport Streams from Video Data

Due to the lack of public-domain MPEG-2 systems transport streams, the DART system currently has to construct systems transport streams for each of the videos used in this thesis. It is not a trivial process to encode timing information into PES packet headers. Because the size of the PES packet headers is more important to this study than the actual header data, the current DART implementation merely reserves space in the systems streams for PES packet headers rather than putting actual values into the headers. The PAT and PMT tables included in the transport streams are also merely placeholders, and do not contain actual data.

The video data for the streams was taken from MPEG-1 and MPEG-2 streams found on the Internet. Some of the MPEG-1 video clips were taken from MPEG-1 systems streams, in which cases the elementary stream video data was extracted for inclusion in the MPEG-2 systems transport streams. Because public domain MPEG-1 videos are more readily available than MPEG-2 videos, most of the videos used for this thesis are MPEG-1.

The following assumptions are made by the DART system when constructing MPEG-2 systems transport streams:

1. All video clips are encoded at 30 frames per second (60 fields per second for interlaced video).

2. A Program Association Table (PAT) is sent in the first transport stream packet. Thereafter, PATs are sent approximately 12 times per second. Each PAT requires a single transport stream packet.

3. A single Program Map Table (PMT) follows each PAT, and also requires a single transport stream packet.

4. Each video frame (or field) is encapsulated in a separate PES packet with a 40-byte header. Although reference pictures sometimes include an extra Presentation Time stamp in addition to the normal Decode Time stamp, the 40-byte header should be sufficient to handle both situations. 40 bytes is also sufficient for CRC checking, if desired. PES packets that don't require a 40-byte header are assumed to be padded to 40 bytes. All slice and macroblock data associated with the picture are included in the PES packet.

5. After sending the first byte of a PES packet containing an encoded picture, the next PES packet cannot be scheduled for the next 1/30 second if the picture was a frame picture, or 1/60 second if the picture was a field picture.

6. Sequence of Pictures and Group of Pictures data bytes are included in the same PES packet as the picture immediately following them in the data stream.

7. End of Sequence data bytes are included in the same PES packet as the picture immediately preceding them in the data stream.

Although MPEG-2 systems transport streams can carry several audio and video elementary streams each, the transport streams created for this thesis are each comprised of a single video elementary stream. However, sending multiple systems streams over the same network is roughly equivalent to sending a single systems stream containing multiple video elementary streams. The only difference between the two data streams is that the former contains more PAT and PMT tables. The PAT and PMT tables occupy such a small fraction of the data stream that their effect is negligible.

Chapter 5

MPEG-2 over ATM

5.1 ATM Cell Structure

Asynchronous Transfer Mode (ATM) is a connection-oriented protocol that uses fixed-size cells to transmit data [5]. Each ATM cell consists of a 5-byte header and a 48-byte payload, for a total of 53 bytes (see Figure 5.1). Bandwidth and other quality-of-service parameters are negotiated at connection-establishment time. Connections are identified by virtual path and virtual channel identifier (VPI and VCI) pairs. Although ATM networks guarantee in-order delivery of cells across connections (virtual channels), network congestion and other factors can result in lost cells. The Cell

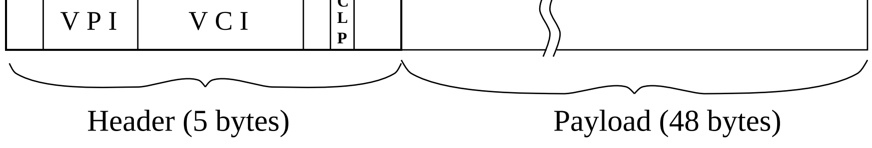

Figure 5.1: *The basic ATM cell format consists of a 5 byte header and a 48 byte payload.*

Loss Priority bit (CLP) in each ATM cell header provides a coarse prioritization of cells [10]. If a party attempts to send more cells than allowed by the negotiated quality-of-service or if the network experiences congestion, low-priority cells (those with the CLP bit set) should be dropped by the network before any higher-priority cells. Because the DART system must segment MPEG data into ATM cells, a basic understanding of ATM is central to this thesis.

5.2 Quality of Service

ATM quality of service (QOS) parameters are negotiated at virtual connection establishment time. The ATM Forum defines the following five service classes: "best effort" unspecified bit rate (UBR), constant bit rate (CBR), variable bit rate (VBR), connection-oriented data, and connectionless data [14]. No traffic parameters are negotiated for UBR service. Traffic contract negotiations for the other service classes can include such parameters as peak cell rate, sustainable cell rate, maximum burst size, cell loss ratio, cell error rate, average delay, cell delay variation, loss on CLP=0 ATM cells, loss on CLP=1 ATM cells, and similar quantities [14, 5].

The MPEG-2 video scalability extensions, discussed in Section 3.1.1, can be tied directly to the QOS concept. Because each of the scalability layers are encoded as a separate elementary stream, it is possible to establish individual virtual connections for each layer– each with its own negotiated quality of service. Thus, the base layer could have a negotiated QOS that guarantees fewer errors than the higher, less critital layers. The emphasis of this thesis, however, is to look at video transmissions over a single virtual connection with a single negotiated QOS. The DART system assumes all data to be sent using a single negotiated quality of service.

5.3 ATM Adaptation Layers (AALs)

ATM Adaptation Layers (AALs), architecturally located above the ATM layer, are protocols for packaging data into the payload field of ATM cells. AALs address such issues as segmentation and reassembly, multiplexing of data from several applications, and CRC protection of payload data. At the present time, three standard AALs have been defined: AAL1 for constant bit-rate applications, and AAL3/4 and AAL5 for data applications. AAL5 is often referred to as the "Simple and Efficient Adaptation Layer" (SEAL) because it has much lower protocol overhead than AAL3/4 [4]. AAL2 is a place-holder AAL for variable bit-rate applications requiring delay bounds, and has not been defined yet. A single 188-byte MPEG-2 systems transport stream packet can be segmented into exactly 4 ATM cells when using AAL1 (see Figure 5.2 and Equation 5.1), but requires 5 ATM cells when using AAL5 (see Figure 5.3 and Equation 5.2).

The AAL5 overhead can be reduced by sending multiple MPEG-2 systems trans-

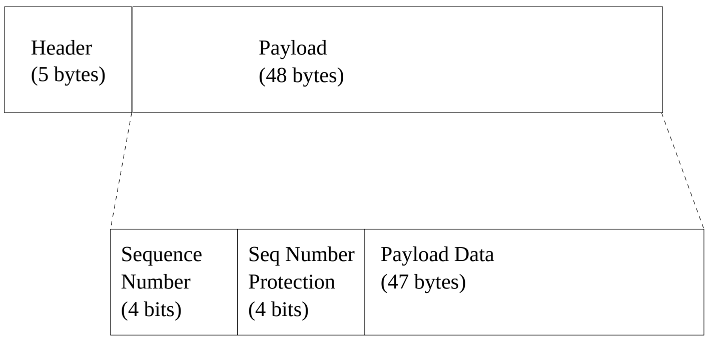

Figure 5.2: *AAL1 ATM cell format. AAL1 allows 47 bytes of user data per ATM cell.*

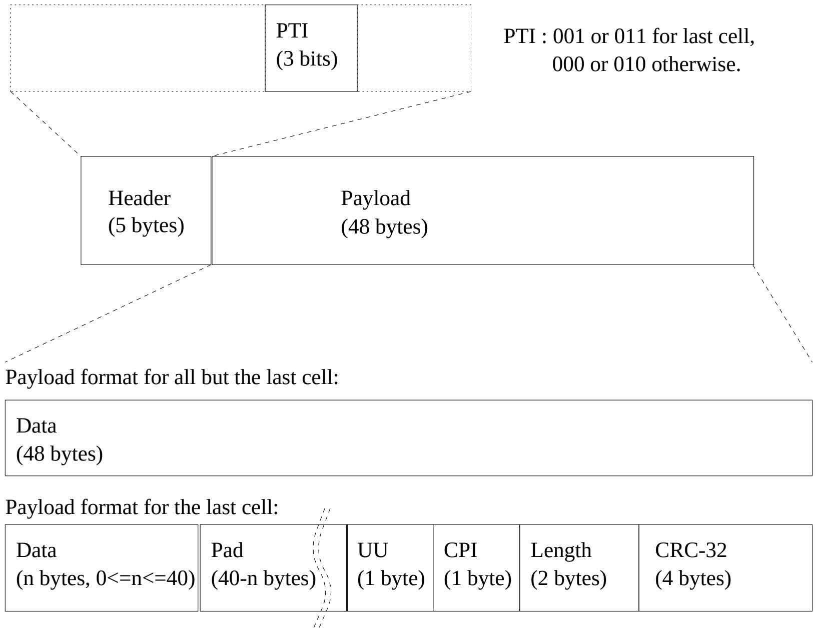

Figure 5.3: *AAL5 ATM cell formats. AAL5 allows 40-48 bytes of user data per ATM cell.*

TS Packets per AAL5	*ATM Cells*	*Unused Bytes*	*% Overhead*
1	5	44	21.7
2	8	0	2.1
3	12	4	2.1
4	16	8	2.1

Table 5.1: *Multiple MPEG-2 systems transport stream packets can be sent in a single AAL5 packet. The number of ATM cells required to send N transport stream packets in a single AAL5 packet, the number of unused bits, and the percent overhead due to AAL5 headers and unused bytes are shown in the above table for values $N = 1$ to 4.*

port stream packets in a single AAL5 packet (see Table 5.1 and Equation 5.2), but care should be taken not to introduce unnecessary delays by holding onto partially-filled AAL5 packets. One drawback to having multiple MPEG-2 systems transport streams packets in the same AAL5 packet is that the entire data field is protected by a single CRC; if any of the ATM cells are lost or corrupted then the entire AAL5 packet is discarded by the adaptation layer.

N 188-Byte Transport Stream packets require:

$$\frac{188N}{47} = 4N \quad \text{ATM cells when using AAL1.} \tag{5.1}$$

$$\left\lceil \frac{188N+8}{48} \right\rceil = \left\lceil \frac{47N+2}{12} \right\rceil \quad \text{ATM cells when using AAL5.} \tag{5.2}$$

5.4 VBR vs CBR Traffic

Variable bitrate (VBR) ATM service is designed to handle bursty traffic efficiently; constant bitrate (CBR) service, on the other hand, requires that data always be sent at a constant rate. In order to send bursty video traffic over CBR connections, the peak bandwidth must always be reserved. Using a CBR connection with a bursty data source is expensive and consumes unnecessary network resources, but these factors must be weighed against the simplicity of CBR service. AAL1 is the simplest AAL defined for CBR service; AAL5 is the simplest defined for VBR service.

5.5 The DART System's Use of ATM Cells

Because MPEG video data is bursty, the DART system uses the variable bit rate AAL5 protocol rather than the constant bit rate AAL1 protocol. As a compromise between protocol overhead and the risk of data loss, two MPEG-2 systems transport stream (TS) packets are sent in each AAL5 packet whenever possible. As shown in Table 5.1, packaging two MPEG-2 systems transport stream packets per AAL5 packet reduces the AAL5 overhead to 10 percent of that produced by one per packet. This reduced overhead comes at the cost of losing both TS packets (instead of just one) if any of the ATM cells in the AAL5 packet are lost or corrupted. Other researchers have looked at other packing algorithms such as packing only integral numbers of macroblocks in each AAL5 packet [8], but the marginal improvements in quality are not worth the added complexity and wasted bandwidth.

The DART system assumes PAT and PMT tables to occupy one MPEG-2 systems transport stream packet each. Therefore, the PAT and PMT are sent together in a single AAL5 packet. PES packets, which are the other major component of MPEG-2 systems transport streams (see Section 4.3), are of variable length and can each require multiple systems transport stream packets. For PES packets requiring an odd number of MPEG-2 systems transport stream packets, the last TS packet is sent in its own AAL5 packet so that the PES data is not unnecessarily delayed; otherwise, PES data is always packaged two TS packets per AAL5 packet.

As mentioned in Section 5.1, the ATM cell header's CLP bit can be used to give higher priority to critical data, thus helping reduce its loss. The DART system does not currently make use of the CLP bit, although such support may be considered in future research. Similarly, the use of multiple virtual circuits and multiple qualities of service (QOS) for video data is also reserved for future research.

Chapter 6

Sending Redundant video data

6.1 Replication Scheme Requirements

The preceding chapters discussed data dependencies in MPEG video streams and MPEG-2 systems transport streams. This chapter introduces DART's data replication scheme for MPEG video elementary streams, and describes how it can be applied to MPEG-2 systems transport streams carrying MPEG videos over ATM networks.

To be effective, a data replication scheme based on data dependencies should be able to identify data dependencies, prioritize individual data elements, determine excess bandwidth available for transmission of redundant data, package and transmit the redundant data in a way that the original and redundant data are distinguishable, use the redundancies to recover missing data, and provide criteria for evaluating the results. The remainder of this chapter will discuss the DART data replication scheme in terms of these requirements.

6.2 Identifying Data Dependencies

As discussed in Chapters 3 and 4, both MPEG video streams and MPEG-2 systems transport streams contain internal dependencies. Because it is already standard practice (see [11], Appendix A) to periodically transmit redundant copies of the MPEG-2 systems transport stream tables (PAT, PMT, etc.) within systems transport streams, no further replication of these tables is necessary. Instead, the DART scheme only attempts to replicate the most-critical elements of the video elementary stream data carried in MPEG-2 systems transport stream PES packets. The MPEG Software Simulation Group's public domain MPEG-2 and MPEG-1 video decoder [9] was modified to identify and prioritize video data for the DART system. The modified decoder accomplishes this by creating a list of video stream elements and their sizes, mapping dependencies to reference picture macroblocks, and keeping track of the number of macroblocks dependent upon each video element.

6.2.1 Hierarchical Dependencies

MPEG video encoders/decoders have access to all of the video dependency information during the encoding/decoding process. By keeping track of the location and size of four video headers (SEQ, GOP, PIC and SLI) and N macroblocks, where N is the maximum number of macroblocks per slice, the hierarchical dependencies for any video element are readily available at encoding/decoding time. If dependencies are to be evaluated on a slice rather than macroblock basis, the requirements are even less stringent: the size of the current slice is needed instead of the slice header size, and the number of macroblocks in the slice is needed instead of the size and location of each macroblock in the slice.

6.2.2 Motion Vector Dependencies

Information about the interpicture macroblock dependencies produced by motion vectors are also available at encoding/decoding time, although additional steps are required to map the reference picture pixels to their slice or macroblock data. A video encoder can be modified to perform the required mappings at encoding time for new videos, or a video decoder can be modified to generate mappings for previously encoded videos.

As part of the normal encoding process for each macroblock, the reference pictures are searched for similarities to the macroblock being encoded. If suitable similarities are found, motion vectors are encoded into the predicted macroblock. At the time of encoding, the referenced pixel locations in the reference pictures are known to the encoder. If the structure of the reference pictures (frame, top field or bottom field) is known, the referenced regions can be mapped to a set of reference picture macroblocks (or slices) using the equations and tables in Chapter 3. Thus, the dependencies for each motion vector, and ultimately each predicted macroblock, can be determined.

For previously encoded videos, the motion vectors are already encoded in the macroblocks. MPEG video decoders map the motion vectors to pixel regions in the reference pictures as part of the normal macroblock decoding process. In order to determine the dependencies between macroblocks, however, the referenced regions must again be mapped to a set of reference picture macroblocks (or slices) for each motion vector, and ultimately for each predicted macroblock.

Whether the mappings are generated by a modified (MPEG) video encoder or decoder, the mapping program must keep track of the video stream locations (and sizes) of all slices in the currently accessable reference pictures in order to enable quick replication of all of the dependency data. (If dependencies are to be evaluated on a macroblock basis, the locations and sizes of reference picture macroblocks must also be maintained.) Because slices can include an arbitrary number of macroblocks, the program must also have a way of mapping macroblock numbers to slice numbers in the reference pictures.

6.2.3 Required Lifetime of Dependency Information

If the dependency data associated with video elements (pictures, slices, macroblocks, etc.) need to be accessable only when the elements are being encoded/decoded, the required lifetime of the dependency information is bounded. Information about the most recent SEQ, GOP, PIC, and slice headers should be maintained at all times, and replaced as new elements are encountered in the video stream. If dependencies are being evaluated on a macroblock basis, information about the current slice and preceding macroblock should be maintained instead of the slice header information. The useful lifetime of this hierarchical dependency information does not extend beyond the replacement points.

The information described in Sections 6.2.1 and 6.2.2 must be maintained for each of the two active reference pictures (or four active reference fields, if applicable), and for the current picture if it is a reference picture. This can easily be accomplished using a list of pointers to current and reference picture data. Whenever a new reference picture is processed, the reference data is shifted from the current pointer to the younger reference (field or frame) pointer to the older reference (field or frame) pointer, and the data originally pointed to by the older reference pointer is discarded. Thus, the lifetime of the reference picture dependency information need not extend beyond the next two reference pictures (or four reference fields).

6.3 Prioritizing Data Elements

After identifying the dependencies associated with each video element, the DART system assigns a priority to each synchronizable element (SEQ, GOP and PIC headers; and slices) in the video stream. The priority is a function of the element size and the number of macroblocks that depend upon it, as shown by Equation 6.1:

$$\text{Priority} = (\text{Number dependent MBs})/(\text{Element size}) \tag{6.1}$$

When deciding which data to send redundantly, elements having a higher priority are sent before elements having lower priorities. Future work could investigate the use of remote feedback to dynamically readjust priorities based on the receipt/loss of earlier data.

6.4 Allocating Bandwidth for Redundant Data

Although it might be beneficial to adjust the bandwidth allocated to redundant data based on network conditions and the quality of the reconstructible video at the receiver, such considerations are reserved for future work. The DART system assigns a target bandwidth to each "epoch" (see Section 6.4.1) in an MPEG-2 systems transport stream based on the epoch's original mean bit rate. Future work could allow other factors such as the burstiness of the bit stream to also be used in the target bandwidth calculation. Any excess bandwidth not used by the original stream is filled with redundant data. This has the effect of decreasing the burstiness of the MPEG stream, making it easier for network providers to meet their quality-of-service commitments.

6.4.1 Epochs

For the purposes of calculating bandwidth usage and sending redundant data, the DART system partitions each data stream into epochs. As currently implemented, an epoch consists of a series of contiguous reference pictures and a series of contiguous B-pictures, and spans at most 10 field times (5 frame times— approximately 1/6 of a second). Epochs also include any PAT and PMT packets immediately preceding the epoch pictures. For the sake of explanation, consider all reference pictures be of "positive"polarity and all B-pictures be of "negative" polarity. An epoch is the longest series of contiguous pictures having less than two polarity changes and a duration of no more than 10 field times (see Figure 6.1).

Large epochs should be avoided because the entire epoch must be buffered at the receiver before its first picture can be displayed. Otherwise, redundant data sent late in the epoch might arrive after the associated pictures have already been displayed. Thus, a short buffering delay is required before the initial pictures in a video are displayed. This delay is only required for the first epoch because data for subsequent epochs can be received while the previous epoch is being displayed. As long as epoch sizes remain small, this initial delay should not be noticeable.

Small epochs should also be avoided. If epochs are too small, the redundant

| I P B B | P B B | P B B | P B B | I B B | P...

| I P B | P B | P B | P B | P B | P B | P B | P...

| I P B B | I B B | P I B B | P B B B B | I ...

Figure 6.1: *Each of the three rows in this figure contains the beginning picture sequences (I, P and B-pictures) taken from an actual MPEG video. In each case, vertical lines separate the epochs as defined by the redundant data transmission scheme.*

data associated with large reference pictures can overload the network. The epoch definition currently employed by the DART system attempts to balance the delay and buffering demands of large epochs with the need to have sufficient bandwidth for redundant reference picture information. Large epochs also allow more flexibility in positioning redundant data away from peaks in bandwidth consumption. Further study would be needed to determine whether other epoch definitions might produce better results.

6.4.2 Bandwidth Allocation within Epochs

Originally, a single target bandwidth was assigned to the entire video stream based on its mean bitrate, as shown in Figure 6.2. This target was used when allocating redundant data in each epoch. Although this original method of allocating bandwidth reduced the burstiness of video traffic (See Figures 6.4 and 6.5), epochs with large

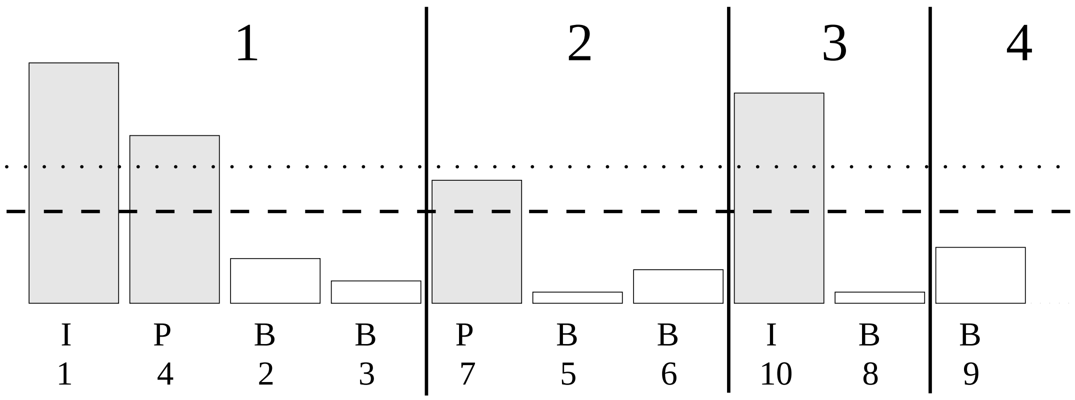

Figure 6.2: *This figure shows the former method of calculating target bit rates. The pictured video clip is divided into four epochs. The rectangle heights represent the I, P and B-picture sizes before adding redundancies. The dashed line represents the mean bitrate for the entire video, and the dotted line represents the target bit rate to be used when adding redundant data to each epoch.*

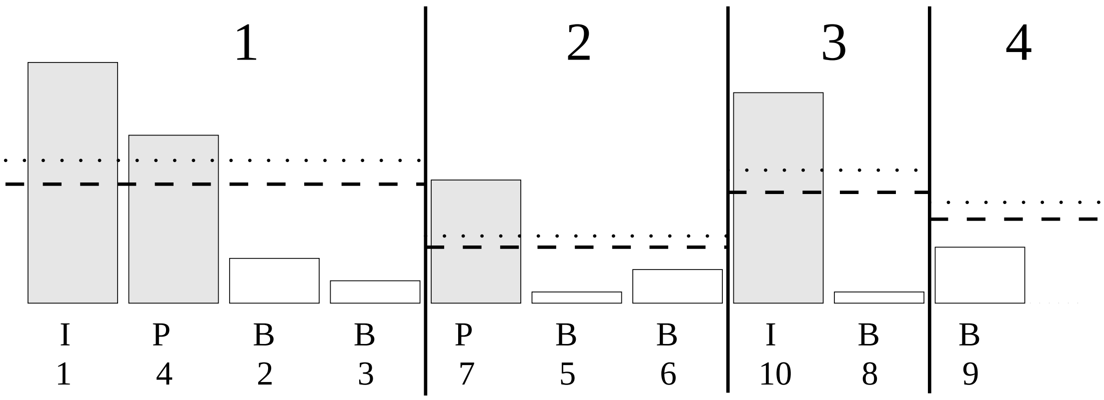

Figure 6.3: *This figure shows the current method of calculating target bit rates. The pictured video clip is divided into four epochs. The rectangle heights represent the I, P and B-picture sizes before adding redundancies. The dashed lines represent the mean bit rate for each epoch, and the dotted lines represent the target bit rates to be used when adding redundant data to each epoch.*

reference pictures and few B-pictures often exceeded their target mean bandwidth before any redundant data had been added. Such epochs were particularly bothersome because large reference pictures tend to include lots of data needed by other pictures. The large picture sizes also increase the likelihood of network losses.

To eliminate the lack of bandwidth in such situations, the DART system currently assigns target bandwidths on an epoch-by-epoch basis. Targets are based upon the mean bitrate of each epoch, as shown in Figure 6.3. Thus, epochs containing an abundance of large reference pictures reserve a correspondingly large amount of bandwidth for redundant data.

Bursty traffic can be expensive for networks to handle because it requires the allocation of extra network resources, such as buffers and queues, to handle fluxuations in traffic. Two common methods of measuring burstiness are shown in Equations 6.2 and 6.3 [18].

$$Burstiness = \frac{\text{Std Dev Rate}}{\text{Mean Rate}} \tag{6.2}$$

$$Burstiness = \frac{\text{Peak Rate}}{\text{Mean Rate}} \tag{6.3}$$

The redundant data streams are not as bursty as the original streams (compare Figures 6.4 and 6.6), but are burstier than they would have been using the old bandwidth allocation scheme, as shown by Figures 6.5 and 6.6.

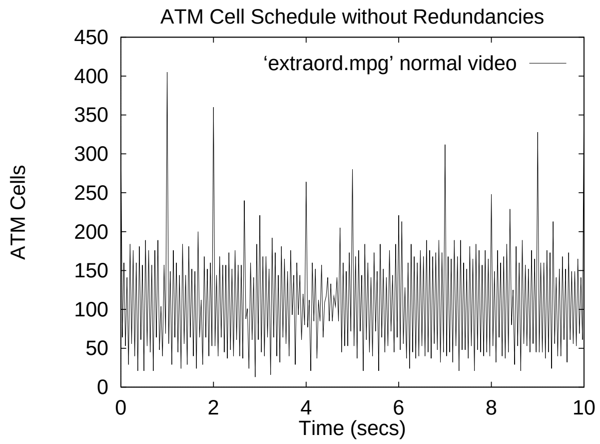

Figure 6.4: *This graph shows bandwidth utilization for the "extraord.mpg" video without any redundant data. (Std Dev/Mean) Burstiness = 0.632*

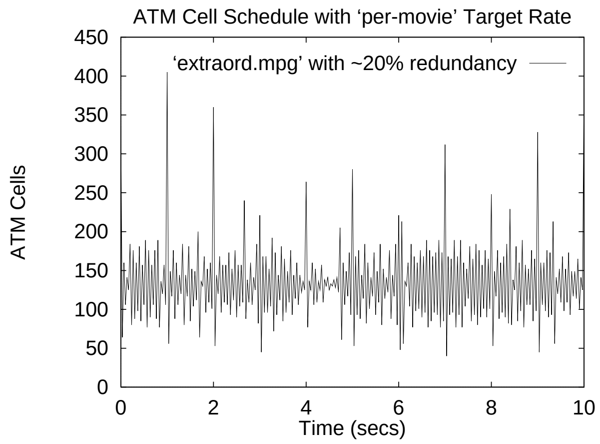

Figure 6.5: *This graph shows bandwidth utilization for the "extraord.mpg" video with about 20 percent redundant data added using the original "per-movie" target bandwidth assignment method. (Std Dev/Mean) Burstiness = 0.376*

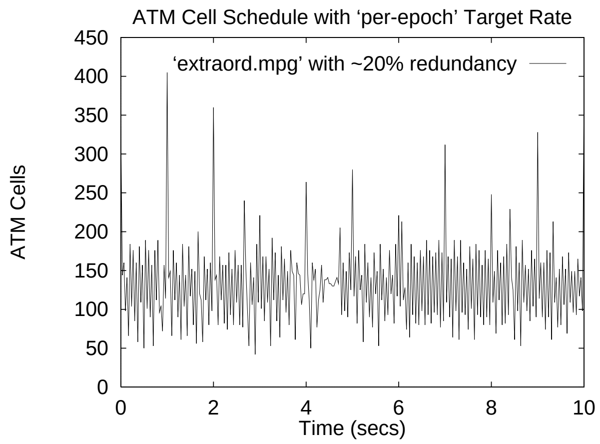

Figure 6.6: *This graph shows bandwidth utilization for the "extraord.mpg" video with about 20 percent redundant data added using the new "per-epoch" target bandwidth assignment method. (Std Dev/Mean) Burstiness = 0.396*

6.4.3 Target Bandwidth Calculation

The video data within PES packets in each epoch are prioritized as described in Section 6.3, and transport stream packets (TS packets) containing redundant data are added to the redundant video elementary stream. In order to reduce the burstiness of the systems stream, the DART system attempts to add the redundant data packets at points of low bandwidth utilization. This is accomplished by assigning an allotment of TS packets to each picture in the normal video elementary stream, and multiplexing the redundant video packets into the systems stream at points where the normal video elementary stream pictures use less than their allotted number of TS packets. At such points, DART inserts redundant packets into the systems stream until the allotment is filled. This reduces the burstiness of systems streams containing redundant data. The algorithm for adding redundant data is described more formally in the remainder of this section.

Let R_{norm} be the normal video bit rate for an epoch containing N fields and/or frames, and let α be the target percentage of redundant data for the video. Then the overall target bit rate R_{target} for the epoch is given by Equation 6.4. Let T_{norm} be the number of MPEG-2 systems transport stream (TS) packets required to send all of the epoch's normal video elementary stream data (without redundancies) and all of its PAT and PMT tables. Then T_{target}, the target number of TS packets with redundancies for the epoch, is defined by Equation 6.5. Let T_j be the number of TS packets required to send the jth picture (without redundancies) and any tables immediately preceding it in the epoch, and let F_j be the number of field times required to display the jth picture in the epoch. (Note: $F_j = 1$ for field pictures and $F_j = 2$ for frame pictures.) The total number of field times F_{total} in the epoch can be calculated using Equation 6.6.

$$R_{target} = \frac{(100 + \alpha)\, R_{norm}}{100} \tag{6.4}$$

$$T_{target} = \left\lfloor \frac{R_{target}\, T_{norm}}{R_{norm}} \right\rfloor + 1 \tag{6.5}$$

$$F_{total} = \sum_{j=1}^{N} F_j \tag{6.6}$$

If the transmission stream packets could be allocated evenly among the fields and frames according to their required amount of display time, then I_j, the ideal number of TS packets allocated to the jth picture in an epoch containing redundant data, could be calculated using Equation 6.7. However, some pictures in the epoch may already exceed their ideal allotment of TS packets before any redundant data is added. To minimize burstiness, redundant video packets should only be added to the systems stream at points where the normal pictures do not use their entire allotment of packets. Therefore, let T_{under} be the total number of TS packets for pictures not exceeding their ideal allotment, as defined by Equation 6.8; and let F_{under} be the number of field times in those pictures, as defined by Equation 6.9. Then A_j, the TS packet allotment actually assigned to pictures that do not exceed their ideal allotment, is defined for the jth picture in the epoch by Equation 6.10. No redundant packets are added to the stream at points where the ideal TS packet allotments have already been exceeded.

$$I_j = T_{target} \left(\frac{F_j}{F_{total}} \right) \tag{6.7}$$

$$T_{under} = \sum_{j=1}^{N} T_j \; : \; T_j < I_j \tag{6.8}$$

$$F_{under} = \sum_{j=1}^{N} F_j \; : \; T_j < I_j \tag{6.9}$$

$$A_j = T_j + (T_{target} - T_{norm}) \left[\frac{F_j}{F_{under}} \right], \quad \forall\, j \; : \; T_j < I_j \tag{6.10}$$

Immediately after all but the last video elementary stream picture that falls short of its ideal allotment I_j, the DART system adds redundant video TS packets to the systems stream until the TS packet allotment specified by A_j in Equation 6.10 has been sent. After the last smaller-than-ideal picture, the DART system sends enough redundant TS packets to bring the TS packet total for the epoch as close as practical to T_{target} (instead of using Equation 6.10 to determine the number of TS packets to send).

6.5 Packaging and Transmitting Redundant Data

6.5.1 Overview

As mentioned in Section 4.3, video elementary stream data is usually comprised of PES packets containing video data. Section 4.5 discussed how the DART system constructs PES packets and MPEG-2 systems transport streams for video data.

MPEG-2 systems transport streams have the capability of carrying multiple elementary streams of various types, grouped into programs. Typical programs might, for example, contain video and audio elementary streams. The DART system defines a new elementary stream type: redundant video elementary stream. Each redundant video elementary stream is assigned a unique PID that is carried in the header of the systems transport stream packets containing the redundant data. The PID also appears in the Program Map Table (PMT) associated with the program, making the redundant stream independent of the original video stream to the extent that decoders will not be confused.

Each redundant video elementary stream contains redundant data associated with a separate video elementary stream. Because the normal and redundant data are sent in different elementary streams, decoders that do not support DART's redundant transmission scheme can decode the video elementary streams without paying attention to the accompanying redundant video streams.

Figure 6.7 shows the tables associated with an MPEG-2 systems transport stream carrying two programs: the first program contains video and audio streams, and the second program contains video, redundant video, and audio elementary streams. Because there is no redundant data associated with the video stream in the first program, video packets lost by the network cannot be recovered at a receiving decoder. The second program, on the other hand, includes a redundant video stream. As a result, a decoder has the option of either decoding the normal video stream without

PAT:

PID = 0	
Program 1:	40
Program 2:	42

PMT:

PID = 40	
Video 1:	51
Audio 1:	68

PMT:

PID = 42	
Video 1:	52
Redund. video 1:	70
Audio 1:	69

Figure 6.7: *This figure contains sample PAT and PMT tables for a MPEG-2 systems transport stream carrying two programs. One of the programs includes a redundant video elementary stream and the other does not.*

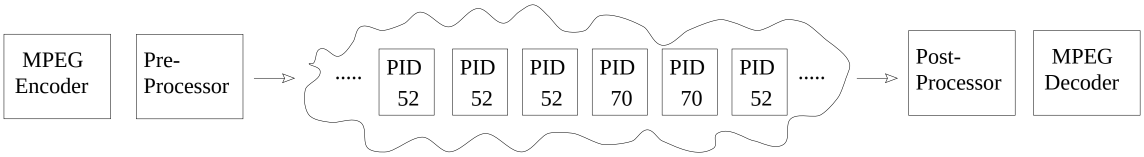

Figure 6.8: *This figure shows the basic components of the redundant transmission scheme. A pre-processor at the video source generates the appropriate redundant video stream, the normal and redundant data are multiplexed on the transmission line, and a post-processor attempts to recover lost data before passing it along to the decoder.*

any help, or else using data provided in the redundant video stream to help recover data that might be missing from the original video stream.

A preprocessor is required at the video source to evaluate the video dependencies and generate an appropriate redundant video stream, as shown in Figure 6.8. A post-processor is also required at the destination to attempt data recovery on the received video streams before passing the result along to the normal MPEG decoder. The PES packets from the elementary streams are multiplexed over the transmission line, as shown in Figure 6.8.

6.5.2 Format of Redundant Data PES Packets

Like normal video elementary stream data, redundant video elementary stream data is sent in PES packets. Unless there is no more redundant data to be sent in the epoch, a PES packet containing redundant data is sent after each normal video PES packet that doesn't exceed its allotted bandwidth. A PES packet is considered to exceed its allotted bandwidth if the entire packet cannot be sent within one picture time (1/30 second for frame pictures; 1/60 second for picture fields) without exceeding the target bandwidth.

Because normal and redundant video are both sent in PES packets, they both require PES packet headers. Each redundant video PES packet includes a 36-byte PES header followed by a 4-byte number identifying the epoch to which it belongs. The DART system consecutively loads redundant video elements from the current epoch, highest priority first, into a redundant PES packet until the target bandwidth for the current picture is reached. The target bandwidth for the current picture is considered to be reached when the redundant PES packet and the normal video PES packet immediately preceding it cannot both be sent within one picture time (1/30 second, if the normal video picture is a frame; 1/60 second, if the normal video picture is a field) without exceeding the target bandwidth.

As mentioned in Section 4.3, PES packet headers include time stamps. For normal video data, the decode time stamp identifies the point in time when the video data (picture) should be decoded. The DART system assigns each redundant PES packet header's Decode Time stamp according to the value of the earliest Decode Time stamp associated with data in the packet; the Display Time stamp in the header is assigned the value of the latest Decode time stamp associated with data in the packet. By comparing time stamps in the normal and redundant video stream packets, the decoder can decide when to process the redundant data.

Because only synchronizable data units are sent redundantly, each redundant data element begins with a start code. When loading data elements into redundant PES packets, a 5-byte Decode or Presentation time stamp is placed into the packet immediately after each start code. The 5-byte time time stamp is copied from the PES packet in which the original data is being sent, and is used by the post-processor to identify the picture to which the redundant data belongs. The time stamp is followed by the actual video element data.

PAT and PMT tables do not change very often and are mainly used at program startup time to identify the elementary stream PIDs associated with each program. The PAT and PMT tables are sent frequently[1] to insure new receivers a short lag time when they begin decoding the systems streams. Because the tables are already sent on such a frequent basis, the DART system does not include them in its redundant streams.

6.6 Recovering Missing Data

Because every picture in a normal video elementary stream has a unique Display Time Stamp associated with it, the time stamps packaged with each video element in the redundant video PES packets should be sufficient to identify where the redundant data belongs in the original data stream. Thus, the time stamps packaged with redundant SEQ and GOP headers identify the display times of the pictures they should have preceded in the normal data stream, and time stamps packaged with redundant picture headers and slices identify the PES packets that should have contained the original data. Because slice start codes include the slice's screen position, the post-processor should be able to successfully merge redundant slice and original picture data.

The details of the reconstruction process are, however, beyond the scope of this thesis. Having allocated sufficient space in the redundant data streams to identify

[1]Appendix A of the MPEG-2 systems standard suggests sending the tables 25 times per second; the DART system considers this excessive, and includes only 12 copies per second in its systems transport streams.

the correct placement of redundant elements in the original video streams, the DART system currently assumes that the post-processor will be able to correctly merge the redundant and original video data.

To avoid implementing a fully functional post-processor, the current implementation of the DART system saves information about the normal and redundant video streams in a file that the postprocessor can use to merge the streams and gather video reconstruction statistics. In a true implementation where the pre-processors and post-processors are fully functional, such external files would be unnecessary. If the findings of this thesis indicate that redundant transmission is worth further exploration, fully-functional pre- and post-processors may be included in future implementations of the DART system, and future research may more fully examine the challenges and problems involved in data reconstruction.

6.7 Evaluating the Results

When transmitting MPEG video data, the major objective is to maximize the picture quality at the receiver. This is accomplished by maximizing the number of reconstructible macroblocks at the receiving end. For every macroblock that cannot be reconstructed, a 16 by 16 pixel area of the picture being decoded is lost. Let S be the number of macroblocks sent in a particular MPEG-2 systems transport stream, and let R be the number of reconstructible macroblocks received at the decoder. The quality Q of the video transmission can be expressed using Equation 6.11.

$$Q = \frac{R}{S} \tag{6.11}$$

The DART system measures video quality interms of Q. The quality measurement Q, however, does not take into account bandwidth costs. A more appropriate measurement would be the percent increase in quality per percent increase in bandwidth. The DART system calls this measurement the "effectiveness" of the redundant video transmission. Let Q_r be the quality of a video transmission that includes redundant data and let B_r be the mean bit rate for that transmission. Further, let Q_n be the quality of a similar video transmission that does not include redundant data and let B_n be the mean bit rate for that transmission. The effectiveness of transmitting redundant data at the designated rate can be expressed using the following equation:

$$\text{Effectiveness of Redundant Data} = \frac{\frac{Q_r - Q_n}{Q_n}}{\frac{B_r - B_n}{B_n}} = \frac{B_n\,(Q_r - Q_n)}{Q_n\,(B_r - B_n)} \tag{6.12}$$

Future work could employ other quality measurements such as subjective visual quality of the received video, or a signal-to-noise comparision of pixel values at the output of an MPEG decoder with the original video data. Such additional measurements, however, are not currently implemented by DART.

Chapter 7

Simulation

7.1 The Simulator Source Code

The simulator used for this research was developed at the National Institute of Standards and Technology (NIST), and provides for event-driven simulation of ATM networks. The public-domain source code is available over the Internet. The source code had to be modified to support MPEG traffic.

The NIST simulator allows a network configuration file to be loaded at the start of each simulation run. Among other things, the network configuration file can define the network topology, specify network queue sizes and link rates, and establish a set of virtual connections between BTEs (Broadband Terminal Equipment). BTEs represent physical devices along the perimeter of ATM networks, and the connections they support appear as leaves or endpoints in the configuration file's network topology. Configuration files associate each of the connection endpoints with one of a set of pre-defined connection types. The connection types affect the overall behavior of the connections by controlling the introduction of data into the network at data sources, defining the classes of connections through the network (VBR, CBR, ABR, etc.), and controlling the receipt of data at the destination nodes.

7.2 Modifications to the Simulator

As part of the changes made to support MPEG data flows, a new "MPEG connection" connection type had to be defined for the simulator. The MPEG connection endpoint can include the name of an MPEG video source file in its configuration. If a source file is specified, the connection endpoint is considered an MPEG video source; otherwise it is considered an MPEG video destination.

Rather than containing actual video data, MPEG video source files contain cell transmission schedules that simulate video traffic patterns. The ATM simulator has a clock that measures the elapsed simulation time. Each line of a video source file includes two fields: a start time and a number of cells to be sent. When the simulator's elapsed time is greater than or equal to the time specified on the first line of the input file, it starts sending cells over the MPEG connection at the defined link rate. The simulator continues sending cells until the number of cells specified on the first line of the input file has been sent. At that point, the simulator reads the next line from the input file. If the time specified on that line is greater than or equal to the elapsed simulation time, the designated number of cells are transmitted; otherwise, the simulator waits until the simulation clock reaches the specified start time before sending the cells. This process is repeated for each line of the video source file. If the simulator runs long enough to read all of the lines in the video source file, the current simulation time is added to each of the start times in the file and the process repeats indefinitely until the simulator is stopped. Thus, the simulated video traffic is endlessly repeated as long as the simulator continues running.

Because the cell transmission schedules do not include actual video data, the cells sent by MPEG connection sources are empty except for a "cell count" value that the connection source puts into each cell. MPEG connection destinations examine the "cell count" field in each received cell. Because ATM networks guarantee in-order delivery of cell, any out-of-sequence cell counts indicate lost cells. The MPEG destinations write a list of lost cells to a log file.

Other modifications were also made to the simulator to allow random cells to be dropped by network switches, to allow simulation runs to exceed 2 seconds of simulated time, and to fix minor bugs and make other improvements to the simulator.

7.3 Creating Cell Transmission Schedules

The cell transmission schedules used in the thesis simulations are based on size and dependency information taken from actual MPEG-1 and MPEG-2 videos. Because the original MPEG videos were not pre-packaged in MPEG-2 systems transport streams, however, the sizes and locations of video elements within each video had to be adjusted to reflect the positions they would occupy in systems transport streams. They also had to be adjusted to reflect the addition of redundant data. Three programs were used to accomplish these tasks:

- First, for videos that were already embedded in MPEG-1 systems streams, the MPEG-1 video data had to be extracted from the MPEG-1 systems stream. This was accomplished using a public domain MPEG-1 systems stream demultiplexer program.

- Second, the size, location and dependency information associated with each element in the video had to be gathered before redundant video data could be added to the data stream. The MPEG video decoder discussed in Section 6.2, which was modified to generate dependency information for this thesis, was used to accomplish this task.

- Finally, a model MPEG-2 systems transport stream had to be constructed for carrying the video and redundant data, and an ATM cell transmission schedule produced for the resulting systems stream. These tasks were accomplished using the preprocessor program developed for this thesis. When redundant data was needed in the simulation stream, the preprocessor was invoked with a special option that caused redundant video elementary stream cells to be included in the cell transmission schedule.

In order to develop an ATM cell transmission schedule for a video, the preprocessor must determine how the video should be segmented into PES packets, how the PES packets should be segmented into MPEG-2 systems transport stream packets, how the MPEG-2 systems transport stream packets should be loaded into AAL-5 packets, how many ATM cells are needed to send the AAL-5 packets, and when each of the ATM cells should be sent. The schedule is developed using the assumptions outlined in Section 4.5.

If redundant data is to be included in the data stream, the preprocessor must also divide the data stream into epochs, decide which video elements in each epoch should be duplicated, determine how the redundant data should be segmented into PES packets, how the redundant PES packets should be segmented into PMEG-2 systems transport stream packets, how the MPEG-2 transport stream packets should be loaded into AAL-5 packets, how many ATM cells are needed to send the AAL-5 packets, and where each of the ATM cells should be added to the existing cell transmission schedule. These decisions are made using the procedures outlined in Section 4.5 and Chapter 6.

The resulting ATM cell transmission schedule produced by the preprocessor contains an entry for each field and frame picture in the video. The entry specifies the start time for sending the picture, and the total number of ATM cells required to transmit the video data, PES headers, systems transport stream packet headers, AAL headers, and redundant video elementary stream data scheduled to be sent during that picture's time slot.

In addition to generating an ATM cell transmission schedule, the preprocessor also generates a systems transport stream map file. The map file lists all of the video elements and their dependencies, all of the redundant video elements, all of the MPEG-2 systems PAT and PMT table packets, and identifies the ATM cells associated with each item in the systems transport stream. After each simulation run, a postprocessor program uses information from the map file to gather statistics about the quality of the received video transmission.

7.4 The Simulator Configuration

The ATM simulator handles multiple simultaneous streams of data, each with its own cell schedule and associated virtual connection. A network configuration with

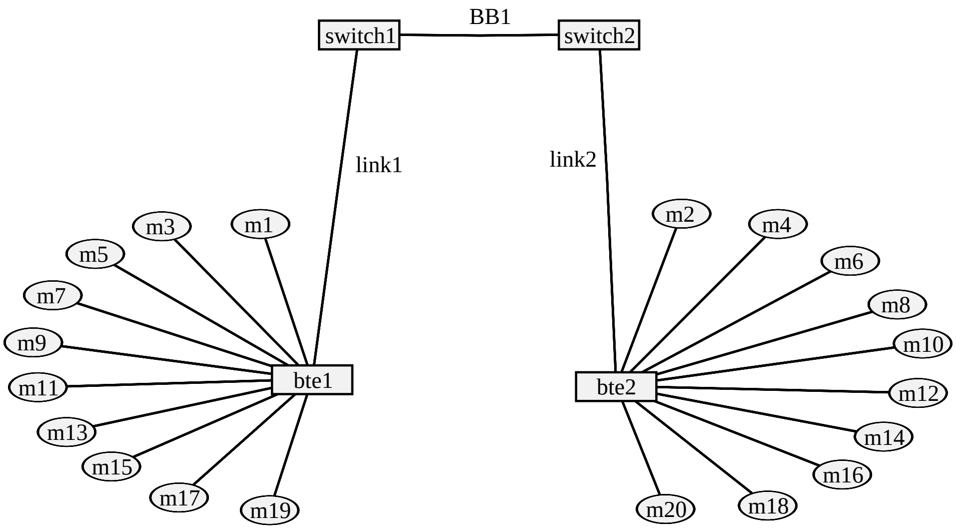

Figure 7.1: *This figure shows the topology of the ATM network used in the MPEG simulations. The 10 MPEG-2 systems transport stream video sources are located at the lower left of the figure, and the 10 receivers are at the lower right. The link rate and ATM switch queue sizes at the top of the figure are set to values that cause occasional congestion and lost cells.*

10 MPEG video sources sending data over a series of links to receiver applications at the other end of the network (see Figure 7.1) was used for the thesis simulations. The link rate on one of the intermediate links (BB1) and queue size on an intermediate switch (switch 1) were set to values that caused the network to experience occasional congestion and drop ATM cells. The start times for the 10 MPEG streams were staggered so that a new video was started every 20 milliseconds until all 10 videos were running.

The amount of redundant data carried over the virtual connections, the maximum link rates, network queue sizes, and other factors were varied between simulation runs. Three minutes worth of video transmissions were simulated in each simulation run.

The simulation results are discussed in Chapter 8.

7.5 Accounting for Lost Cells

As mentioned in Section 7.2, the simulator checks the sequence numbers in the ATM cells at the receiving end of each MPEG video connection. Whenever out-of-sequence cells are received, the receiver's id and the missing cell numbers are written to a special MPEG log file.

After each simulation run, a postprocessor program is run for each MPEG connection. The postprocessor uses information from the preprocessor's map file (see the end of Section 7.3) to identify which normal and redundant video elements were lost during the transmission. The postprocessor then determines which video elements can be reconstructed from the successfully received data, and generates statistics about the quality of the received video.

The map file would not be necessary in a true implementation of the redundant transmission scheme. The map file is required in the simulations because the simulated ATM cells do not include any actual video data. In addition, the map file helps in the evaluation process by providing information about the source video.

7.6 Other Considerations

7.6.1 Producing Cell Losses

Having developed realistic data streams, identified the internal data dependencies within those streams, and provided a means of evaluating the quality of the received data, the major remaining issue to be addressed is how to get the simulator to generate reasonable patterns of ATM cell losses.

The quality-of-service parameters for each virtual connection in a simulated network configuration are implicitly defined as a function of the link rates, switch queue sizes, and other configuration values in the switches and other components lying along the virtual path between the connection endpoints. Thus, rather than explicitly supplying quality-of-service values to the simulator, the configuration must be altered until the desired behavior is achieved over the designated virtual circuit. For some quality-of-service parameters, such as peak bit rate, this is not a problem; others, such as mean burst error size and mean error rate are much more difficult to achieve.

Because internal network congestion is one of the most frequent causes of cell loss in ATM networks, the simulated networks were configured such that all cell losses would be produced by a switch and backbone link common to all of the video connections. To insure that the backbone link and switch were the limiting factors in the virtual connections, the paths between the video sources and BTEs were each configured to handle the peak cell rate of their associated systems stream. Further, links 1 and 3 (see Figure 7.1) were configured with link rates exceeding the mean bit rate of the combined videos. Link "BB1," on the other hand, was configured to slightly below the combined videos' mean bit rate, so as to cause congestion at switch 1. Because the output queues in switch 1 are finite, the resulting congestion caused occasional bursts of lost cells.

Network topoligies are usually much more complex than those used in our simulations. In order to account for cell losses caused by other virtual connections that might share network resources, the simulator code was modified to include a "percent external cell loss" parameter in each switch. Switches configured with a non-zero "percent external cell loss" randomly drop a specified percentage of the cells they receive. The "percent external cell loss" parameter for switch 1 was set to a non-zero value in the thesis simulations.

7.6.2 Evaluating the Effectiveness of Redundant Data

Because of the difficulty of generating consistent "quality of service" values for virtual connections, due to their "implicit" nature, it becomes difficult to compare the results of redundant and non-redundant video transmissions. Two approaches could be used to obtain fair comparison data: link rates and queue sizes could be adjusted until similar error rates are obtained for the normal and redundant simulation runs; or a single simulation run that includes redundant data could be used, and its video quality could be compared against the quality that would result if the redundant data were ignored. The second approach assumes the frequency and distribution of losses in the normal and redundant video data are similar. Both approaches are used in Section 8.4.

Chapter 8

Results and Conclusions

8.1 Overview

Eighteen ATM simulations were run— each simulating three minutes of traffic on an ATM network carrying ten MPEG-2 systems transport streams. Thus, simulation data was collected for 180 three-minute video transmissions, each consisting of approximately 5,400 pictures (30 frames-per-second for 180 seconds). The network topology shown in Figure 7.1 and described in Section 7.4 was used. Each of the MPEG-2 systems transport streams included either an MPEG-1 or MPEG-2 video elementary stream. The video start times for each simulation were staggered so that all ten videos did not start generating traffic at the same time, but once started, all of the videos continued producing ATM cells until the simulation's three minute time limit was reached.

Size and dependency data from 15 MPEG video clips were used to generate the simulated systems transport stream traffic. The video clips were downloaded from various sites on the Internet. Because public domain MPEG-1 videos are more readily available than MPEG-2 videos, most of the videos are MPEG-1. The MPEG-2 videos are all test sequences designed to test the functionality of MPEG-2 decoders, whereas most of the MPEG-1 videos are entertainment-quality videos. A brief description of each of the videos, their picture sizes and encoding types (MPEG-1 or MPEG-2) are shown in Table 8.2; encoding details and bit rates are shown in Table 8.3.

Of the 18 ATM simulations, six used MPEG-1 video data, six used MPEG-2 video data, and six used a mixture of MPEG-1 and MPEG-2 video data. The MPEG-1 simulations carried traffic for the ten MPEG-1 videos listed at the top of Table 8.2; the MPEG-2 simulations carried traffic for two copies each of the five MPEG-2 videos listed at the bottom of the table; and the remaining simulations carried a combination of eight MPEG-1 and two MPEG-2 videos.

The ATM simulator was configured to produce different patterns of cell loss for each of the three traffic types: MPEG-1, MPEG-2 and mixed video traffic. The simulator dropped 0.1% of the MPEG-2 video traffic cells, 1% of the MPEG-1 video traffic cells and 4% of the mixed video traffic cells. The original intention was to simulate all three types of traffic using the same sets of cell loss patterns and to compare the results. Such a thorough comparison, however, is reserved for future work. The current results are intended to show the effects different cell loss rates have on DART system performance, rather than show the characteristics of different types of video traffic. In all of the simulations, switch-1 was configured to randomly drop a designated percentage of ATM cells to simulate the effects of congestion and other network errors. Additionally, for all but the mixed video simulations, switch-1's output queue was configured so that longer bursts of cells would also occasionally be dropped to simulate network queue overflows. Thus, a combination of both short and

Simulated Traffic Type	% Randomly Dropped Cells	% Cells dropped by overflowed queues	Total % Dropped Cells
MPEG-1 video	0.5%	0.5%	1%
MPEG-2 video	0.1%	0%	0.1%
Mixed video	1%	3%	4%

Table 8.1: *This table shows the ATM cell loss rates configured for switch-1 in the simulated network. To examine the effects of cell loss on the DART system, the simulations carrying MPEG-1, MPEG-2, and mixed (MPEG-1 and MPEG-2) video traffic were each configured to have different cell loss rates.*

long bursts of cell losses were simulated. The cell loss characteristics for each of the simulated traffic types are summarized in Table 8.1.

Each of the six simulations for a given traffic type (MPEG-1, MPEG-2 or mixed) carried a different amount of redundant data: one simulation for each traffic type carried normal traffic without any redundancies, and the remaining five simulations carried traffic that had been padded with redundancies to a target mean bit rate 10%, 20%, 30%, 40%, or 50% above the normal traffic rate.

This chapter examines the characteristics of the redundant data sent in the simulations and the effects of redundant data on video quality, effectiveness and burstiness. The structure of the simulated videos is also briefly discussed. Finally, some general conclusions are drawn at the end of the chapter.

Video Name	Content of Video	Picture Size (Row x Col)	Video Type
aspen3.mpg	Advertisement for Aspen, Colorado	160 x 120	MPEG-1
blossom.mpg	Opening dance scene from a TV sit com	176 x 144	MPEG-1
casino02.mpg	Opening scend from “Casino” movie	352 x 240	MPEG-1
casinopr.mpg	Press release for “Casino” movie	352 x 240	MPEG-1
Ext.lg.mpg	Action video: skiing, biking, rock climbing	240 x 160	MPEG-1
extraord.mpg	“Something Extraordinary” movie preview	352 x 240	MPEG-1
icecream.mpg	Scenes from “Icecream” movie	352 x 240	MPEG-1
kiss.mpg	Low action scene from “Lounge People”	352 x 240	MPEG-1
loubrock.mpg	Television commercial for Cardinals baseball	352 x 240	MPEG-1
neptune.mpg	Underwater scene with divers and sea turtle	304 x 228	MPEG-1
boat.m2v	Panning shot of ships in a harbor	720 x 576	MPEG-2
flwr_080.m2v	Test video clip: flower garden	704 x 480	MPEG-2
np.es	Panning shots of a swimming pool area	720 x 576	MPEG-2
susi_080.m2v	Test video clip: woman using telephone	704 x 480	MPEG-2
tens_080.m2v	Test video clip: table tennis game	704 x 480	MPEG-2

Table 8.2: *This table lists basic information about all of the video clips downloaded from the Internet for use in the simulations.*

Video Name	Most Common Pattern of Encoded Pictures	Slices /Frame	Mblocks /Slice	Bit Rate (Mbps)		
				Mean	Peak	Std Dev
aspen3.mpg	IBBPBB	8	10	0.65	1.49	0.360
blossom.mpg	IBBPBBPBBPBB	9	11	0.44	1.49	0.291
casino02.mpg	IBBPBBPBBPBBPBB	15	22	1.42	4.64	0.694
casinopr.mpg	IBBPBBPBBPBBPBB	15	22	2.61	7.59	1.344
Ext.lg.mpg	IBBBPBBBPBBBPBBB	10	15	4.58	4.58	0.566
extraord.mpg	IBPBPBPBPBPBPBP BPBPBPBPBPBPBPB	15	22	1.41	5.15	0.890
icecream.mpg	IBPBPBPBPBPBPBP BPBPBPBPBPBPBPB	15	22	1.41	5.50	0.927
kiss.mpg	IBBPBBPBBPBBPBB	1	330	1.41	6.41	1.021
loubrock.mpg	IBBPBBPBBPBBPBB	1	330	1.41	5.80	0.821
neptune.mpg	IBBBPBBBPBBB	15	19	1.33	3.56	0.734
boat.m2v	IBBPBBPBBPBB	36	45	7.13	25.77	3.828
flwr_080.m2v	IBBPBBPBBPBB	30	44	9.33	32.16	7.195
np.es	IBPBPBPBPBPB	36	45	8.42	27.88	4.375
susi_080.m2v	IBBPBBPBBPBB	30	44	9.33	32.77	6.840
tens_080.m2v	IBBPBBPBBPBB	36	44	11.15	36.29	8.110

Table 8.3: *This table lists encoding information and bit rates for all of the MPEG videos used in the simulations.*

8.2 Characteristics of Redundant Data

As a direct result of the data prioritization scheme described in Section 6.3, SEQ, GOP, and Picture headers are given high priority due to their small size and the large number of macroblocks that depend upon them. Even in simulations where the redundant data occupied as little as 10% of the bandwidth, redundant copies of all of the videos' SEQ, GOP, and picture headers were included in the redundant video streams. The only times when the simulation results did not show 100% redundancies for these headers were when the simulation ended in the middle of an epoch—before the elements' counterparts had a chance to be sent. The "% Sent" column in Tables A.1 to A.3 in Appendix A show this for the MPEG-1 simulations. DART's prioritization scheme could be simplified in the future to always include these headers, rather assigning them priorities like the remaining data. The "% Sent" column in Table A.4 shows that at 10% redundancy, less than half of the slices were included in the same redundant video streams that included 100% of the previously-mentioned headers.

Slices are the main video stream elements that require prioritization. Table 8.4 shows the percentage of slices included in the redundant data stream for target bit stream rates between 10% and 50% above the normal bit rate. The size of the slices (number of macroblocks per slice) has an affect on the percentage of slices appearing in the redundant data stream. The "kiss.mpg" and "loubrock.mpg" videos are of particular interest because Table 8.3 shows that those videos have only one slice per frame. Thus, for each redundant slice that is sent, an entire picture must be replicated. That explains the low percentages in Table 8.4 and the "% Sent" column

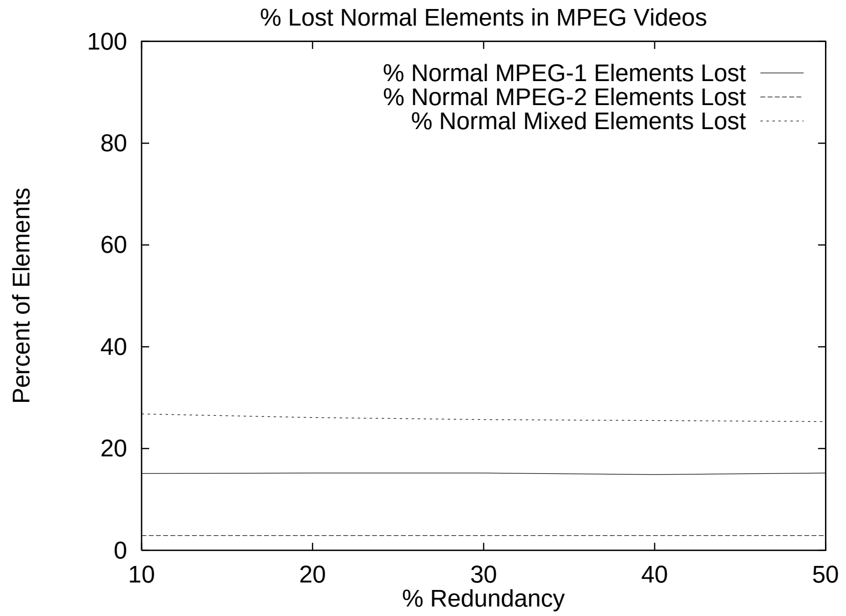

Figure 8.1: *This graph shows the percentage of lost normal video elements, when the target redundancy level is 10-50% of the normal video stream bit rate.*

of Table A.4 for those two videos.

Whether or not the redundant data is actually useful depends on several factors. One of these factors is the ATM cell loss rate: redundant data serves no useful purpose unless there is missing data to replace. The cell losses generated by the simulator (see Table 8.1) produce corresponding video stream element losses. The percentage of normal video stream elements that were lost during the simulations are shown in Figure 8.1. Although the cell losses between traffic types are quite different (due to simulator configuration—see Section 8.1), the rates remain relatively constant across the six simulations within each traffic type. This allows the qualities of the videos at various levels of redundancy within each group to be compared with each other.

The redundant video streams contained copies of some of the video elements that were lost from the normal video streams during simulations. The percentage of lost normal video elements that had a corresponding element in the redundant video stream (whether or not that redundant copy was also lost during the simulation) are

Video Name	Percent of Redundant Slices (Target = 10 to 50% redundant data)				
Target	10 %	20 %	30 %	40 %	50 %
aspen3.mpg	27	44	59	68	76
blossom.mpg	30	47	60	70	77
casino02.mpg	21	37	50	60	69
casinopr.mpg	20	37	51	62	71
Ext.lg.mpg	19	32	45	55	65
extraord.mpg	39	55	65	72	79
icecream.mpg	41	58	67	74	79
kiss.mpg	2	23	42	53	67
loubrock.mpg	1	16	35	47	63
neptune.mpg	21	38	52	64	73
boat.m2v	21	35	49	59	69
flwr_080.m2v	27	45	59	69	76
np.es.m2v	23	40	51	60	68
susi_080.m2v	20	38	54	65	73
tens_080.m2v	22	40	54	65	72
OVERALL	24	41	54	64	72

Table 8.4: *This table shows the percentage of slices present in the redundant video stream when the target bit rate is 10 to 50 percent above the normal data rate.*

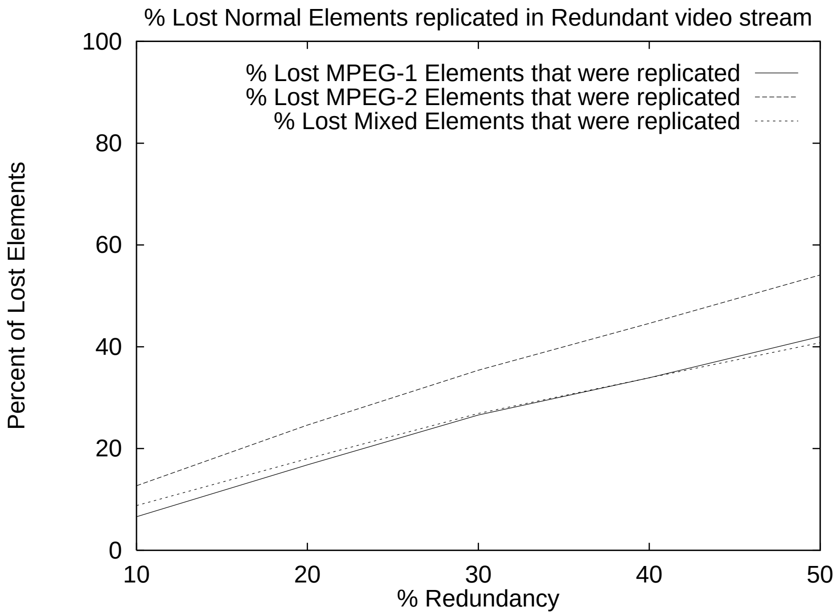

Figure 8.2: *This graph shows the percentage of lost normal video elements that were replicated in the redundant video stream.*

shown in Figure 8.2. As expected, the percentage of redundant elements associated with lost normal video elements increased with increasing data stream redundancy.

Figure 8.3 shows the percentage of normal video elements that were replicated in the redundant video streams. This graph differs from Figure 8.2, which shows the percentage of *lost* normal video elements that were replicated. The percentage of replicated video elements are virtually the same for all of the simulated streams. Because SEQ, GOP and picture headers are small and typically have highest priority, the percent of redundant elements is always significantly higher than the percent of redundancy.

Unless the original data is severely corrupted, the receiver only needs a small percentage of the redundant data to reconstruct the video stream. Figure 8.4 shows the percent of redundant video elements that were actually needed by the receivers during simulation— in other words, the percent of redundant video stream elements that replicated *lost* data from the original video stream. As expected, the percentages

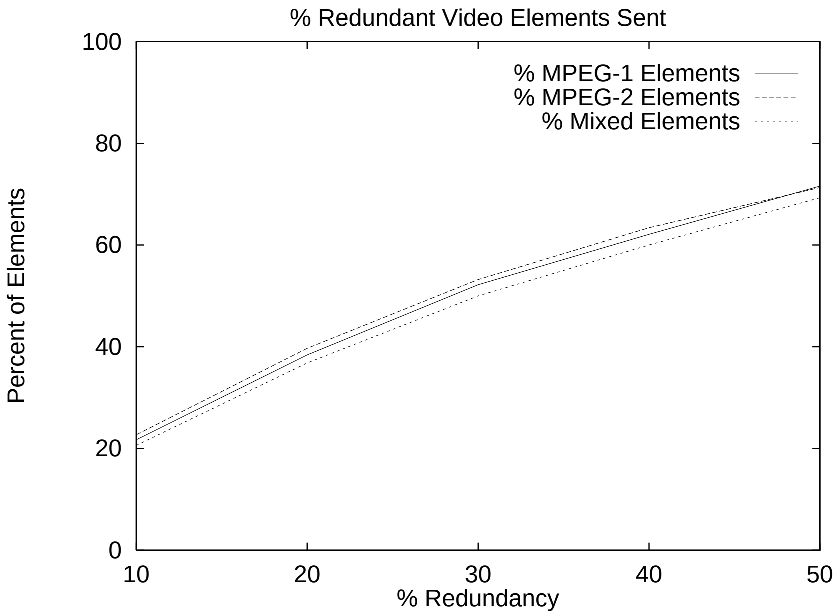

Figure 8.3: *This graph shows the percentage of normal video elements that were sent in the redundant video stream during simulation.*

are lowest for the streams with the lowest cell error rates.

Figure 8.5 shows the percentages of *redundant* video elements that were lost during simulation. These percentages are slightly lower than the percentages for normal elements (compare Figure 8.1), but increase as the amount of redundancy increases. This makes sense because DART's prioritization scheme favors redundant copies of small video elements, such as headers; and for a given cell loss rate, large elements have a greater chance of being lost than small elements because the loss of any one of their cells results in the loss of the entire element. Thus, it makes sense that the percentage of lost elements would be smaller for redundant than for normal data, and would increase as additional (larger) redundant data elements are added. The slightly lower-than-average losses for redundant data also show that DART doesn't place the redundant data in unnecessarily high-risk sections of the data stream.

Figure 8.6 shows the percent of redundant elements that were "usable" during simulation. Usable redundant elements are elements of the redundant video stream

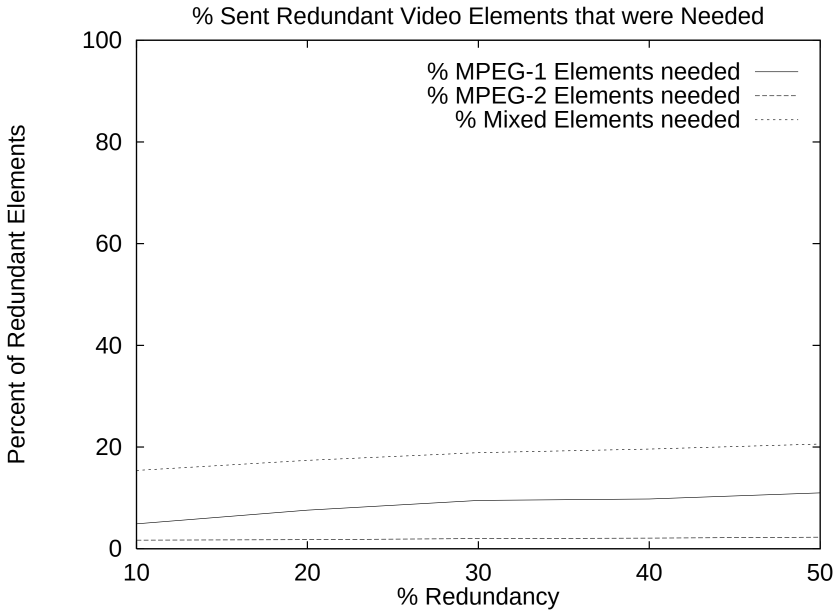

Figure 8.4: *This graph shows the percentage of the redundant video elements that were actually needed (replicated lost data).*

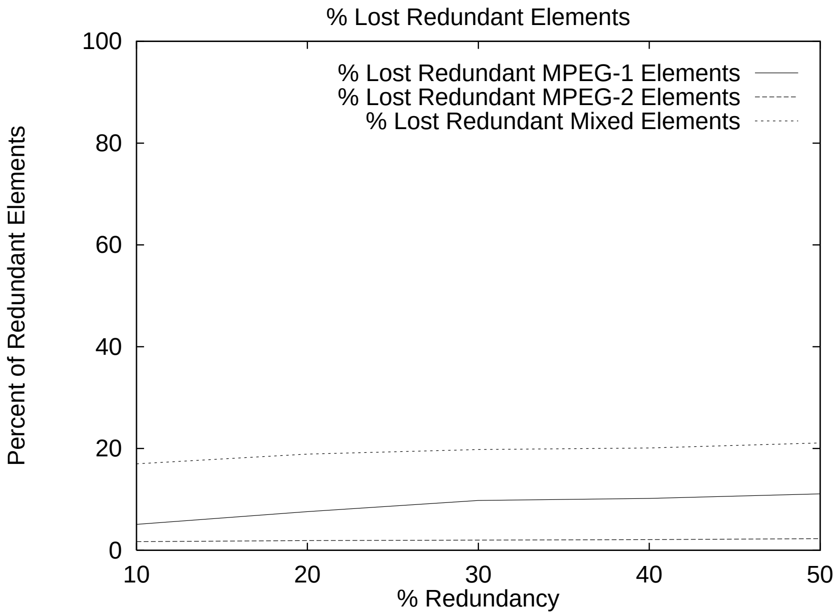

Figure 8.5: *This graph shows the percentage of lost redundant elements.*

that successfully arrive at their destination, but whose counterparts in the normal video stream are lost in the network. Not all "usable" redundant elements can actually be used by the receiver. "Usable" elements cannot actually be used unless all of the data stream elements upon which they depend are also "usable." Figure 8.7 shows the percentage of "usable" redundant video elements that were actually used by the receivers during the simulations. Some of the lines in the graph have negative slopes. Negative-sloped line segments are an indication that as the number of usable redundant elements increased, other elements upon which the new elements depended were missing. Hence, the percentage of "usable" redundant units that could actually be used decreased! As more and more redundant units are introduced into a data stream, however, this trend seems to be reversed as redundant copies of the other elements become available. The fact that some of the lines in Figure 8.7 have negative slopes is an indication that some critical data was either lost or given too low of a priority, and changing either the prioritization or epoch-generation scheme might improve the quality of the received videos.

8.3 Redundant Data and Video Quality

As demonstrated by Equation 6.11 in Section 6.7, the quality of a video can be represented as the ratio of the number of macroblocks reconstructible (R) by the receiver to the total number of macroblocks sent (S). Quality values (R/S) for the each of the simulated video transmissions are shown in Figures 8.8 to 8.10 and Tables 8.5 to 8.7. As expected, the quality of the received videos generally increases as the amount of redundancy in the data stream is increased.

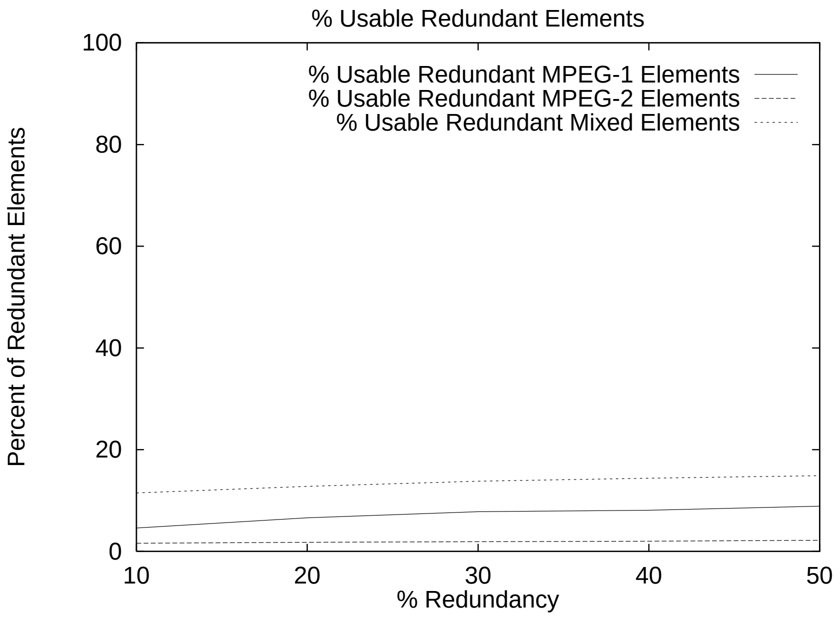

Figure 8.6: *This graph shows the percentage of redundant elements that were "usable." Usable elements are elements that were not lost in the redundant video stream, but the data they replicate was lost in the normal video stream.*

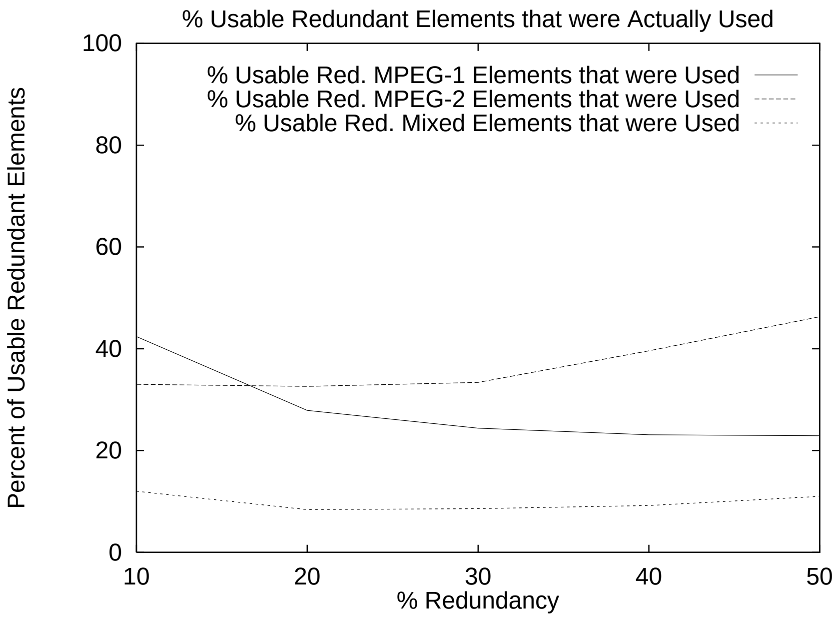

Figure 8.7: *This graph shows the percentage of "usable" redundant elements that were actually used. Elements can only be used if all of the other elements upon which they depend are also usable.*

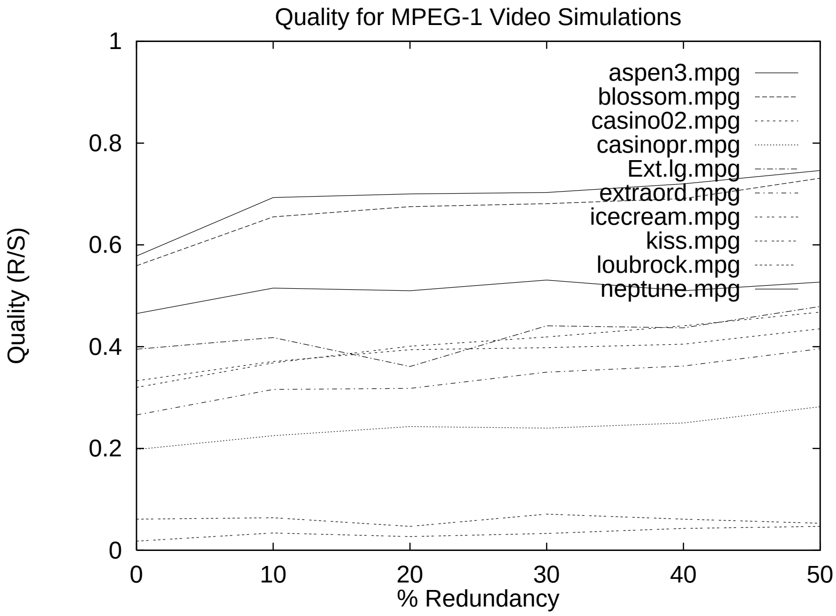

Figure 8.8: *This graph shows the quality of received videos from simulations containing only MPEG-1 videos.*

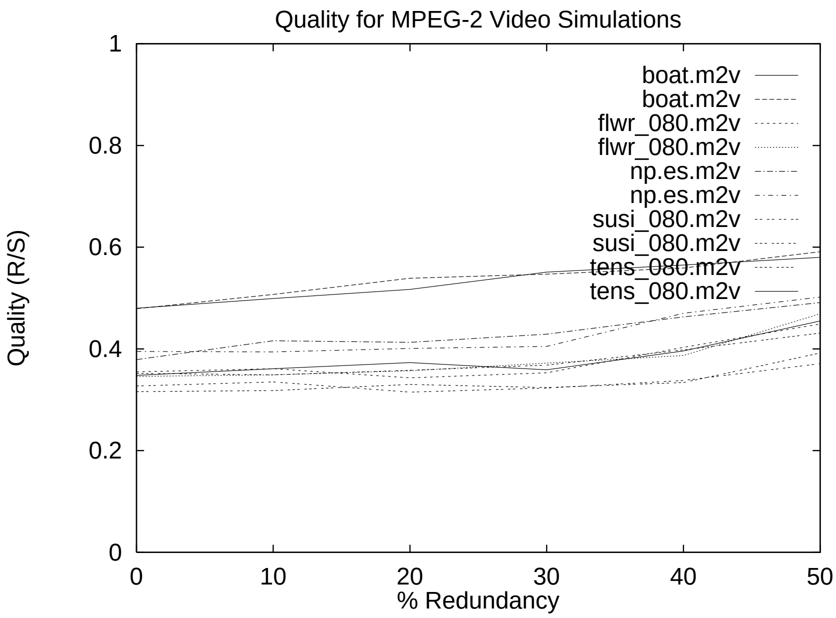

Figure 8.9: *This graph shows the quality of received videos from simulations containing only MPEG-2 videos.*

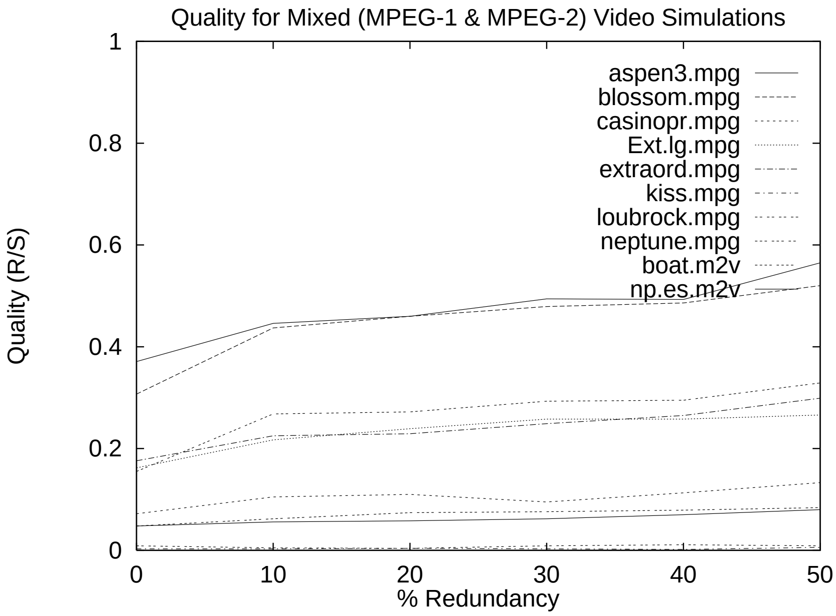

Figure 8.10: *This graph shows the quality of received videos from simulations containing both MPEG-1 and MPEG-2 videos.*

The "aspen3.mpg" and "blossom.mpg" videos have particularly high quality values in Figure 8.8; conversely, the "loubrock.mpg" and"kiss.mpg" videos have particularly low quality values in the same figure. This can easily be explained when one considers the number of macroblocks per slice listed for each of the videos, as shown in Table 8.3. The "aspen3.mpg" and "blossom.mpg" videos have a small number of macroblocks per slice, whereas the opposite is true for "loubrock.mpg" and "kiss.mpg." Because this thesis evaluates dependencies on a slice basis, the loss of any macroblock in a slice is assumed to prevent the entire slice from being reconstructed. Similarly, the loss of any macroblock in a reference picture precludes the reconstruction of any slices in other pictures that reference the lost slice. This underscores the error recovery advantages of using small slice sizes when encoding MPEG videos. The effects of slice size on error recovery are discussed in [17].

Video Name	Quality of Received MPEG-1 Video Data (where Target = 0 to 50% redundant data)					
Target	0 %	10 %	20 %	30 %	40 %	50 %
aspen3.mpg	0.578	0.693	0.700	0.703	0.720	0.746
blossom.mpg	0.559	0.655	0.675	0.681	0.691	0.731
casino02.mpg	0.333	0.371	0.394	0.398	0.405	0.435
casinopr.mpg	0.198	0.225	0.243	0.240	0.250	0.282
Ext.lg.mpg	0.395	0.418	0.361	0.441	0.437	0.479
extraord.mpg	0.266	0.316	0.318	0.350	0.362	0.396
icecream.mpg	0.320	0.368	0.401	0.419	0.441	0.468
kiss.mpg	0.018	0.034	0.027	0.033	0.043	0.047
loubrock.mpg	0.061	0.064	0.047	0.071	0.061	0.053
neptune.mpg	0.465	0.515	0.510	0.531	0.510	0.527

Table 8.5: *This table shows the quality of received video data from simulations containing MPEG-1 video traffic. Quality values are shown from six simulation runs (having 0 to 50 percent redundant data).*

Video Name		Quality of Received MPEG-2 Video Data (where Target = 0 to 50% redundant data)					
	Target	0 %	10 %	20 %	30 %	40 %	50 %
boat.m2v		0.480	0.499	0.517	0.551	0.565	0.580
		0.479	0.507	0.539	0.547	0.559	0.591
flwr_080.m2v		0.355	0.361	0.343	0.353	0.403	0.449
		0.346	0.349	0.357	0.372	0.387	0.469
np.es.m2v		0.379	0.416	0.413	0.429	0.463	0.491
		0.395	0.394	0.401	0.405	0.470	0.502
susi_080.m2v		0.327	0.335	0.315	0.323	0.338	0.371
		0.316	0.318	0.330	0.324	0.334	0.392
tens_080.m2v		0.352	0.349	0.358	0.368	0.398	0.431
		0.348	0.361	0.373	0.359	0.396	0.455

Table 8.6: *This table shows the quality of received video data from simulations containing MPEG-2 video traffic. Quality values are shown from six simulation runs (having 0 to 50 percent redundant data).*

Video Name	Quality of Received MPEG-1 & 2 Video Data (where Target = 0 to 50% redundant data)					
Target	0 %	10 %	20 %	30 %	40 %	50 %
aspen3.mpg	0.371	0.446	0.460	0.494	0.493	0.565
blossom.mpg	0.307	0.437	0.460	0.479	0.486	0.520
casinopr.mpg	0.072	0.105	0.110	0.095	0.113	0.133
Ext.lg.mpg	0.162	0.217	0.239	0.258	0.258	0.266
extraord.mpg	0.176	0.225	0.229	0.249	0.265	0.299
kiss.mpg	0.003	0.003	0.004	0.003	0.002	0.006
loubrock.mpg	0.009	0.005	0.004	0.009	0.011	0.009
neptune.mpg	0.155	0.268	0.272	0.293	0.295	0.329
boat.m2v	0.048	0.062	0.074	0.076	0.079	0.084
np.es.m2v	0.048	0.056	0.058	0.062	0.070	0.080

Table 8.7: *This table shows the quality of received video data from simulations containing a mixture of MPEG-1 and MPEG-2 video traffic. Quality values are shown from six simulation runs (having 0 to 50 percent redundant data).*

8.4 Redundant Data and Effectiveness

As defined by Equation 6.12 in Section 6.7, the effectiveness of a redundant video transmission is a measure of the percent increase in quality per the percent increase in bandwidth caused by transmission of the redundant data. The "effectiveness" measure attempts to weigh the benefits of sending more data against the associated cost in bandwidth consumption. Effectiveness values greater than "one" indicate that the quality improves at a faster rate than would normally be expected for the corresponding increase in bandwidth.

Because the effectiveness measurement compares the quality and bit rate of a video transmission that includes redundant data against the quality and bit rate of a reference transmission that doesn't include any redundancies, a reference transmission must be chosen before comparisons can be made. For this thesis, two sets of reference transmissions were chosen for evaluating each redundant transmission:

- Values for the first set of reference transmissions were obtained directly from the redundant simulations. In each case, the quality of the reference transmission was calculated by ignoring the redundant data stream and observing what percentage of the original macroblocks could be reconstructed without the redundant data. Similarly, the reference bit rate was also calculated by ignoring the redundant data stream.

- Values for the second set of reference transmissions were obtained by configuring the simulator to produce cell losses similar to those produced for the redundant transmissions, and then sending video data without any redundancies and observing the resulting video quality and required bit rate.

Both methods of calculating effectiveness values produced similar results, as shown by Figures 8.11 and 8.12. The effectiveness values were highest for simulations with high error rates, which means that the DART system is probably most effective when the video data is being sent over error-prone networks. The MPEG-1 and mixed simulations proved particularly effective when there was about 10% redundant data. Future work can use data streams with between 0 and 10% redundancy to more accurately locate the point of greatest efficiency. The effectiveness of higher redundancy levels was somewhat less than expected, but the quality improvements might be more noticable during actual viewing of the reconstructed videos, rather than looking only at the mathematical quality and efficiency calculations, which don't currently take into account granularity below the slice level.

Although the effectiveness calculation can help weigh the advantages against the costs of sending redundant data, it should not be given undue emphasis. Transmissions that already have low error rates, such as the simulations carrying MPEG-2 data, won't generally benefit much from redundant transmission (as evidenced by the low effectiveness values in Figures 8.11 and 8.12), but redundant transmissions may still be useful as insurance against occasional SEQ, GOP and Picture header losses.

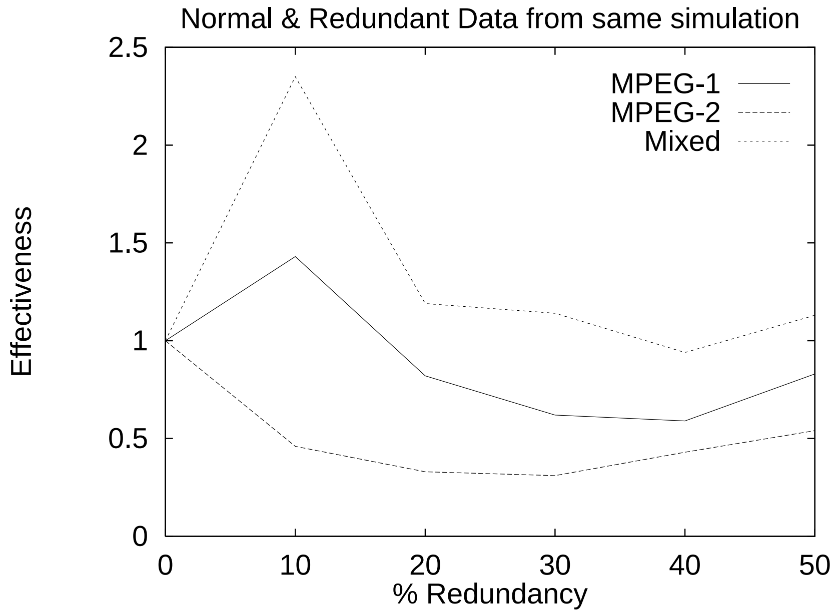

Figure 8.11: *This graph shows the effectiveness of redundant data. The effectiveness at each point was calculated using normal and redundant data values that were calculated from a single simulation: for the normal data values, the redundant data was ignored; for the redundant data values, the redundant data was included in the bit rate and quality calculations.*

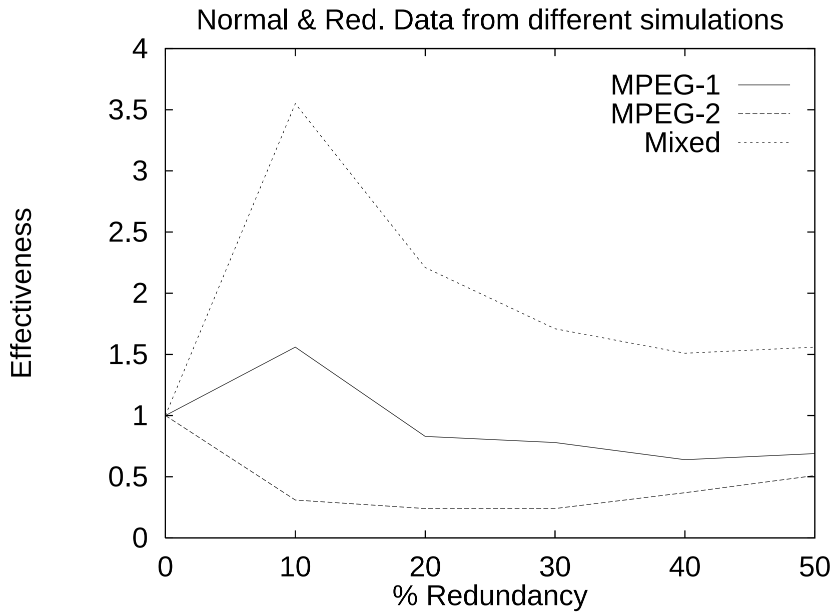

Figure 8.12: *This graph shows the effectiveness of redundant data. The effectiveness was calculated using normal and redundant data from different simulations.*

8.5 Redundant Data and Burstiness

Bursty traffic can be expensive for networks because it requires them to reserve additonal network resources, such as buffers and queues, to handle fluxuations in traffic. If several applications produce bursts at the same time, network queues can overflow and lose data. Such losses can be very detrimental to the quality of the received videos [20]. In addition, bursty traffic often causes network providers to limit the number of network connections or otherwise limit capacity in order to insure their contracted quality of service guarantees can be met.

For these and other reasons, it is desirable to reduce the burstiness of video data. Burstiness can be measured in several ways. The DART system uses Equations 6.2 and 6.3 to measure burstiness. The equations are repeated here for convenience:

$$Burstiness = \frac{\text{Std Dev Rate}}{\text{Mean Rate}} \tag{8.1}$$

$$Burstiness = \frac{\text{Peak Rate}}{\text{Mean Rate}} \tag{8.2}$$

The DART system not only reduces the burstiness of individual MPEG-2 video streams, it also reduces the burstiness of aggregate data arriving at network switches. Table 8.8 shows the burstiness of individual MPEG-2 systems transport streams containing each of the 15 simulation video clips padded with 0–50% redundancies. Figures 8.13 and 8.14 show graphical representations of the same data. In every case, the burstiness of the MPEG data decreases as the amount of redundant data is increased.

Tables 8.9 and 8.10 show the aggregate burstiness of ATM cells from 10 different video sources arriving at switch-1 during simulations. The tables show the

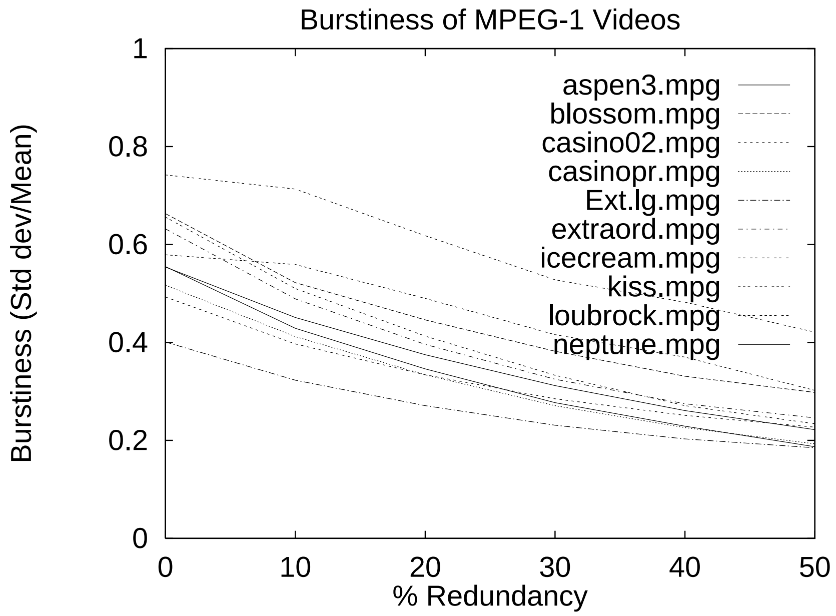

Figure 8.13: *This graph shows the burstiness of redundant data for MPEG-1 videos.*

burstiness of aggregate traffic from MPEG-1 and MPEG-2 videos, respectively. Both the "Peak/Mean" and "Standard deviation/Mean" measurements of burstiness are shown in the tables— the higher the values the burstier the traffic. Figure 8.15 shows a graphical representation of the same data. As with the individual MPEG-2 systems transport streams, the burstiness of the aggregate streams (cells arriving at switch-1, see Figure 7.1) decreases as the amount of redundant data is increased. The actual number of cells arriving at switch-1 are plotted against time in Figures 8.16 to 8.18. Although the standard deviation in the MPEG-2 video cell arrivals (Table 8.10) is greater than that of the MPEG-1 video cell arrivals (Table 8.9), the burstiness of the traffic is actually quite similar, as can be seen by comparing either the "Peak/Mean" or the "Std dev/Mean" values in the same tables. This is due to the fact that the MPEG-2 videos also have a much higher mean cell rate.

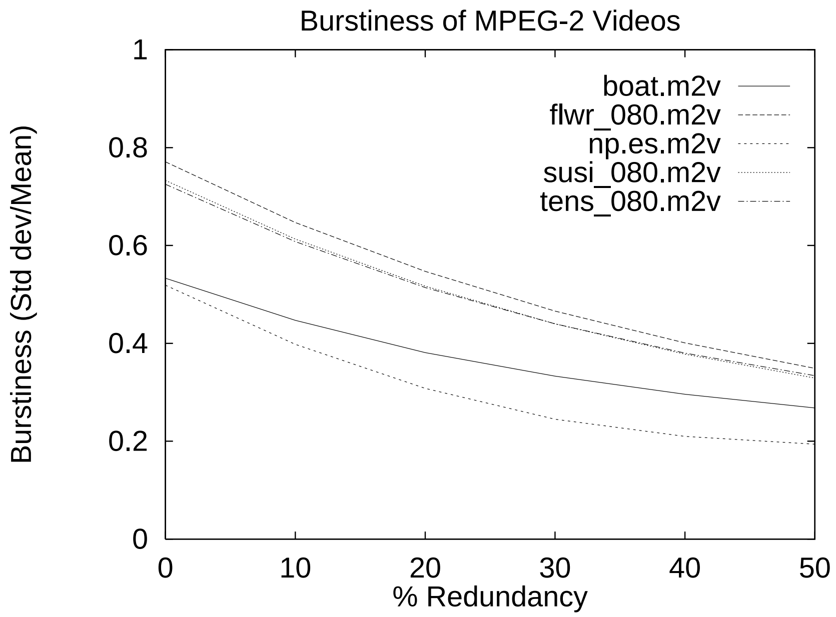

Figure 8.14: *This graph shows the burstiness of redundant data for MPEG-2 videos.*

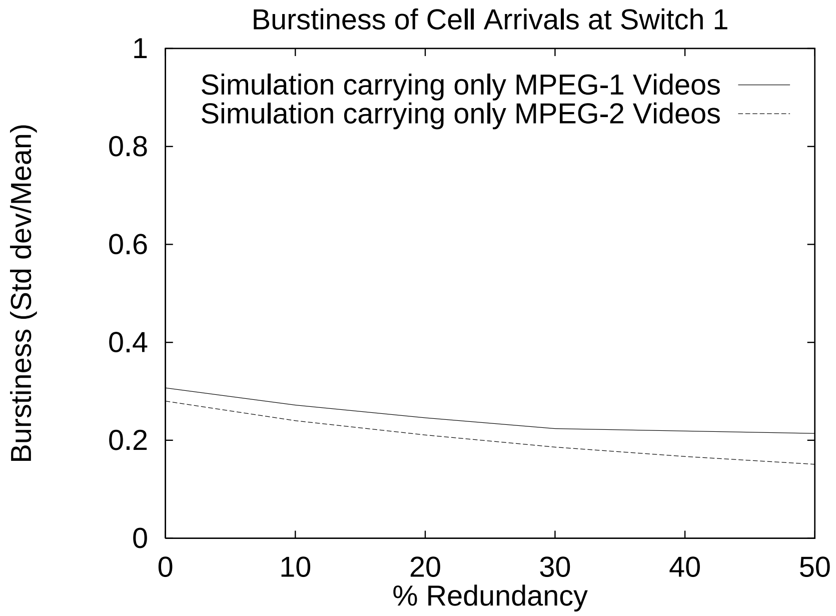

Figure 8.15: *This graph shows the burstiness of MPEG-1 and MPEG-2 video traffic containing redundancies. The burstiness is shown for cell arrivals at switch-1 during MPEG-1 and MPEG-2 simulations.*

Video Name		Burstiness of Video Data (Std dev/Mean) (where Target = 0 to 50% redundant data)					
	Target	0 %	10 %	20 %	30 %	40 %	50 %
aspen3.mpg		0.555	0.429	0.346	0.277	0.229	0.187
blossom.mpg		0.663	0.522	0.446	0.382	0.331	0.298
casino02.mpg		0.493	0.397	0.334	0.285	0.251	0.227
casinopr.mpg		0.517	0.412	0.334	0.271	0.226	0.193
Ext.lg.mpg		0.401	0.323	0.271	0.231	0.203	0.185
extraord.mpg		0.632	0.489	0.396	0.325	0.275	0.246
icecream.mpg		0.656	0.510	0.413	0.333	0.271	0.234
kiss.mpg		0.742	0.713	0.618	0.528	0.482	0.421
loubrock.mpg		0.579	0.559	0.490	0.416	0.370	0.302
neptune.mpg		0.554	0.451	0.375	0.312	0.261	0.222
boat.m2v		0.533	0.447	0.381	0.333	0.296	0.268
flwr_080.m2v		0.771	0.647	0.547	0.466	0.401	0.349
np.es.m2v		0.519	0.398	0.308	0.245	0.210	0.194
susi_080.m2v		0.733	0.613	0.517	0.440	0.378	0.329
tens_080.m2v		0.725	0.608	0.514	0.440	0.380	0.334

Table 8.8: *This table shows the burstiness of the video data when the data stream has 10 to 50 percent redundant data.*

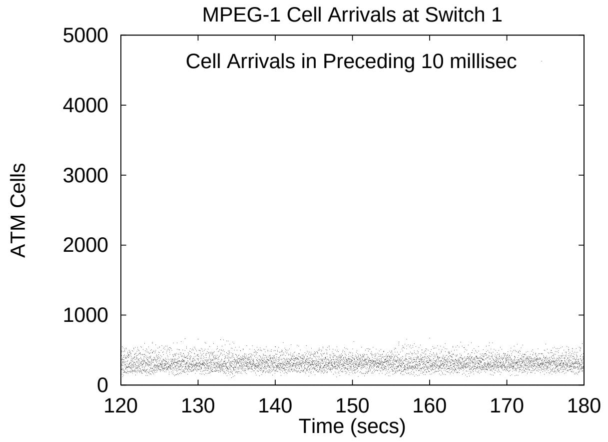

Figure 8.16: *This graph shows the number of ATM cell arrivals at a switch handling traffic for 10 MPEG-2 systems transport streams carrying only MPEG-1 videos. The cell arrivals are shown for each 10 msec time interval in the final 60 seconds of a 3 minute simulation. (Peak/Mean) Burstiness = 2.375*

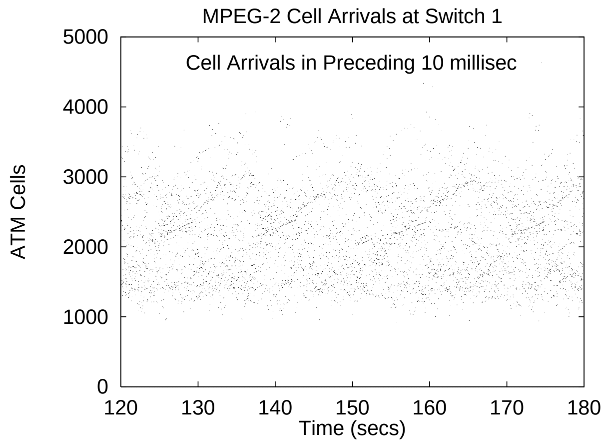

Figure 8.17: *This graph shows the number of ATM cell arrivals at a switch handling traffic for 10 MPEG-2 systems transport streams carrying only MPEG-2 videos. The cell arrivals are shown for each 10 msec time interval in the final 60 seconds of a 3 minute simulation. (Peak/Mean) Burstiness = 2.042*

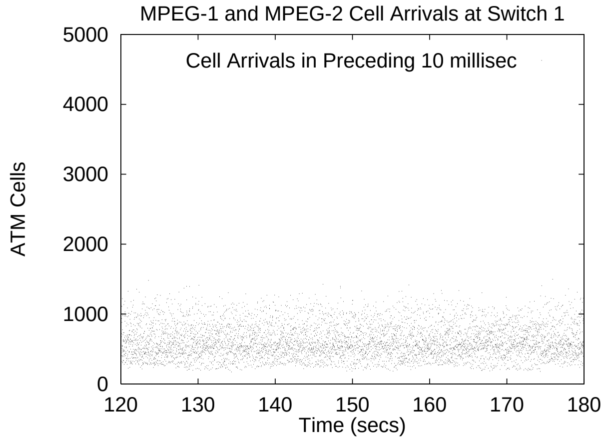

Figure 8.18: *This graph shows the number of ATM cell arrivals at a switch handling traffic for 10 MPEG-2 systems transport streams carrying MPEG-1 and MPEG-2 videos. The cell arrivals are shown for each 10 msec time interval in the final 60 seconds of a 3 minute simulation. (Peak/Mean) Burstiness = 2.512*

% Redundant Data	Mean Cell Rate	Peak Cell Rate	Std Dev Cell Rate	Burstiness (Peak/Mean)	Burstiness (Std dev/Mean)
0 %	319.13	758	97.85	2.375	0.307
10 %	347.46	751	94.48	2.161	0.272
20 %	377.50	763	92.88	2.021	0.246
30 %	409.32	781	91.62	1.908	0.224
40 %	437.32	773	95.74	1.768	0.219
50 %	467.41	839	99.86	1.795	0.214

Table 8.9: *This table shows the burstiness of cell arrivals at switch-1 during simulations carrying MPEG-1 videos. The cell arrival counts are for 10 millisecond intervals. The burstiness is measured two ways: the peak cell arrival rate divided by the mean arrival rate, and the standard deviation divided by the mean arrival rate. Both measures show a decrease in burstiness as the redundancy is increased.*

% Redundant Data	Mean Cell Rate	Peak Cell Rate	Std Dev Cell Rate	Burstiness (High/Mean)	Burstiness (Std dev/Mean)
0 %	2135.02	4359	597.14	2.042	0.280
10 %	2347.24	4452	563.46	1.897	0.240
20 %	2559.58	4753	540.13	1.857	0.211
30 %	2767.62	4813	515.24	1.739	0.186
40 %	2977.68	4965	495.96	1.667	0.167
50 %	3187.34	5068	479.98	1.590	0.151

Table 8.10: *This table shows the burstiness of cell arrivals at switch-1 during simulations carrying MPEG-2 videos. The cell arrival counts are for 10 millisecond intervals. The burstiness is measured two ways: the peak cell arrival rate divided by the mean arrival rate, and the standard deviation divided by the mean arrival rate. Both measures show a decrease in burstiness as the redundancy is increased.*

8.6 Structure of Simulated Videos

While preparing the video data for simulation, some interesting insights were gained into the structure of encoded MPEG videos. Although two thirds of the videos used in the simulations were MPEG-1, most of them still followed the convention of encoding pictures with one slice per row of macroblocks[1]. This was true for 13 of the 15 video clips used in the simulations. Encoding more than one slice per row of macroblocks would further aid in error recovery, but none of the simulation videos were encoded that way.

Although the MPEG standards allow motion vectors to reference pixels anywhere in the reference pictures, most encoders apparently only search a restricted area surrounding the screen position of the macroblock being predicted. This is evidenced by the number of reference picture slices that are referenced by predicted P-picture and B-picture slices, as shown in Tables 8.11 and 8.12. Looking at Table 8.11, the majority of P-picture slices reference about 3 slices of the reference picture. The references most likely correspond to the slice of macroblocks directly above, beneath, and in the same row as the slice being predicted. The MPEG-2 encoders seem to be a little more sophisticated, but it must be rememebered that all of the MPEG-2 videos used in the simulations are test videos designed to test the functionality of MPEG-2 decoders, and thus might include references that would not normally be found in commercial videos.

Similarly, the majority of MPEG-1 video slices in the B-pictures shown in Table 8.12 reference 6 or fewer reference picture slices—which could reasonably cor-

[1]Although MPEG-2 video standard requires a new slice at the beginning of each row of macroblocks, the MPEG-1 standard has no such requirement.

respond to 3 slices in each of the forward and backward reference pictures. This information could be helpful in determining the storage requirements for programs that might be developed to evaluate and/or store dependency information.

Looking from the referenced rather than the referencing point of view, an understanding of the number and distribution of macroblocks that reference given slices could be helpful in developing or evaluating new priority schemes for redundant data. The reference counts could be used to estimate the percentage of redundant slices that would produce the greatest effectiveness in redundant video streams. The study of such relationships is reserved for future research, but Appendix B contains slice reference graphs for each of the MPEG videos used in this thesis. The x-axes show the number of macroblocks that reference any given I or P-picture slice, and the y-axes show the number of slices being referenced.

Video Name	Percent of P-picture Slices referencing 0–9 other slices (where Ref. Count = number of other slices referenced)									
Ref. Count	0	1	2	3	4	5	6	7	8	9
aspen3.mpg	—	15.1	31.7	53.2	—	—	—	—	—	—
blossom.mpg	0.6	28.1	44.4	26.8	—	—	—	—	—	—
casino02.mpg	0.4	2.0	20.5	41.1	22.4	13.6	—	—	—	—
casinopr.mpg	0.2	3.4	28.8	47.8	15.2	4.6	—	—	—	—
Ext.lg.mpg	6.0	3.2	32.4	38.0	16.6	3.8	—	—	—	—
extraord.mpg	20.3	10.4	25.5	43.8	—	—	—	—	—	—
icecream.mpg	23.4	17.2	24.2	35.2	—	—	—	—	—	—
kiss.mpg	—	100.0	—	—	—	—	—	—	—	—
loubrock.mpg	—	100.0	—	—	—	—	—	—	—	—
neptune.mpg	0.3	5.2	36.1	58.3	—	—	—	—	—	—
boat.m2v	—	0.1	4.5	22.6	51.0	18.9	2.7	0.3	—	—
flwr_080.m2v	—	—	0.4	4.4	34.3	26.0	21.8	8.3	4.4	0.4
np.es.m2v	—	—	0.3	7.5	22.3	25.6	17.1	15.7	11.2	0.1
susi_080.m2v	—	—	1.2	1.3	5.4	14.9	28.7	25.8	18.8	4.0
tens_080.m2v	—	0.2	4.8	6.0	23.5	15.8	18.9	13.9	13.9	3.0

Table 8.11: *This table shows the percent of P-picture slices that reference the designated number (0-9) of reference picture slices.*

Video Name	Percent of B-picture Slices referencing 0–11 other slices (where Ref. Count = number of other slices referenced)											
Ref. Count	0	1	2	3	4	5	6	7	8	9	10	11
aspen3.mpg	—	0.3	11.5	12.2	24.9	23.9	25.2	1.7	0.3	—	—	—
blossom.mpg	—	0.4	20.5	8.5	21.9	29.0	19.7	—	—	—	—	—
casino02.mpg	0.1	0.2	3.0	6.0	22.0	14.1	44.2	8.6	1.8	—	—	—
casinopr.mpg	0.1	0.1	2.4	4.4	19.9	22.7	46.1	3.9	0.5	—	—	—
Ext.lg.mpg	0.2	17.7	8.9	14.5	16.6	14.5	18.6	6.3	2.1	0.5	0.1	—
extraord.mpg	0.4	1.4	21.7	19.6	20.4	15.8	20.6	—	—	—	—	—
icecream.mpg	—	1.0	36.7	16.1	19.4	10.5	16.2	—	—	—	—	—
kiss.mpg	—	1.3	98.7	—	—	—	—	—	—	—	—	—
loubrock.mpg	—	0.2	99.8	—	—	—	—	—	—	—	—	—
neptune.mpg	—	0.1	3.8	4.5	28.7	24.3	36.7	1.9	0.1	—	—	—
boat.m2v	—	—	—	—	1.0	3.6	25.4	49.0	19.0	1.4	0.6	—
flwr_080.m2v	—	—	—	0.2	2.3	5.5	16.8	32.2	28.8	11.4	2.5	0.3
np.es.m2v	—	—	0.1	0.4	1.1	2.0	21.6	20.4	18.7	11.4	8.2	5.2
susi_080.m2v	—	—	—	—	3.3	3.2	24.7	37.5	21.0	6.6	2.3	1.1
tens_080.m2v	—	0.1	3.4	1.0	4.5	3.7	24.1	22.5	21.9	12.0	5.1	1.5

Video Name	Percent of B-picture Slices referencing 12–16 slices				
Ref. Count	12	13	14	15	16
np.es.m2v	3.2	2.2	2.6	1.7	1.3
susi_080.m2v	0.3	—	—	—	—
tens_080.m2v	0.2	—	—	—	—

Table 8.12: *This table shows the percent of B-picture slices that reference the designated number (0-16) of reference picture slices. (Note: In order to reduce the width of the table, the percentages for slices referencing 12 or more slices are shown at the bottom of the table.)*

8.7 Conclusions

This thesis has explored one solution to the error recovery problem for applications containing a high degree of internal dependencies in their data streams. By taking advantage of these dependencies and transmitting critical information redundantly, the DART system has shown the usefulness of the remaining data to increase for MPEG videos being carried in MPEG-2 systems transport streams. DART's replication scheme also has the added side benefit of reducing the burstiness of the video traffic. The general principles developed for MPEG video streams can be carried over into other types of data streams having similarly strong internal data dependencies.

The DART system programs were developed to identify dependencies in actual MPEG-1 and MPEG-2 video streams, construct MPEG-2 systems transport streams with redundant data for each video, send the resulting streams through an ATM network simulator that simulates network errors by dropping occasional ATM cells, and evaluate the quality and effectiveness of the resulting transmissions. Some preliminary results and an early description of the DART system were presented by the author at the Fourth International Conference on Telecommunication Systems [23], held in Nashville, Tennessee, in March 1996.

This thesis lays the ground work for future research, which will further refine the data replication techniques, optimize DART system parameters, apply more sophisticated cell loss models to the data streams, and look at other quality measurements for the received videos. The work presented in Chapter 3, which explicitly identifies the dependencies that exist between encoded MPEG video elements, should also prove useful to other researchers working with MPEG video streams.

Chapter 9

Related and Future Research

9.1 Related Research

Related research in error modeling is being done by Richard A. Butler at The Robert Gordon University in Scotland [3]. Dr. Butler's research includes analysis of burst error activity in real networks, and attempts to predict network behavior by looking at error patterns. Dr. Butler's research could be used in future simulations to generate cell loss patterns that are more characteristic of real-world situations.

Other researchers are exploring error concealment, reduction, or recovery techniques. Susanna Aign and Khaled Fazel, for example, are developing error concealment techniques that can be applied to MPEG-2 videos that have missing or corrupted macroblocks [1]. Their techniques estimate missing macroblock data by using data or motion vectors from other nearby successfully-received macroblocks. Error concealment techniques have the advantage of low encoding and transmission overhead, although their decoding overhead is sometimes high.

Iain Richardson and Martyn Riley, also from The Robert Gordon University in

Scotland, are looking at forward error correction techniques for recovering lost MPEG data [21, 22]. Because I-pictures have a larger impact on video decoding than P-pictures, and P pictures have a larger impact than B pictures, they are also looking at reducing errors by sending I, P, and B pictures over separate virtual connections—each with its quality of service guarantees tailored to the associated pictures' importance to the decoding process [18, 19]. Although they have achieved success using forward error correction schemes that consume between 10 and 20% of the normal bandwidth, using forward error correction to protect against long bursts of errors can be extremely costly and might be better handled by a system like DART.

The ATM Forum is working on a standard (95-0012R5) for sending MPEG-2 data over ATM using the AAL-5 protocol with quality of service guarantees. According to e-mail correspondence with an ATM Forum technical committee member in July 1995, the document describing the ATM Forum's proposal for MPEG-2 over ATM is available only to ATM Forum members.

9.2 Areas for Future Research

Several topics for future research were mentioned in earlier chapters. Some of the ideas are listed here.

1. Add support for the MPEG-2 scalable video streams mentioned in Section 3.1.1.

2. Combine or compare the effects of MPEG-2 scalability extensions with the effects of the DART system.

3. Implement the graph coloring scheme outlined in Section 3.5.2, and use it to assign data to the redundant video streams.

4. Explore the use of multiple qualities of service and multiple virtual connections.

5. Modify the DART system to prioritize MPEG data using the CLP bit found in the ATM cell headers. Compare the results of transmitting normal video data using the CLP bit with those produced by both the modified and the unmodified DART system.

6. Use remote feedback to dynamically readjust data priorities based on receipt/loss of earlier data.

7. Examine new methods of partitioning the data stream into epochs.

8. Evaluate the implications of assigning target bandwidths based on factors other than the mean bit rate of the original data. The target bandwidth could be based, for example, on some combination of the mean bit rate and the burstiness of the data.

9. Fully implement the pre- and postprocessors to transmit the actual MPEG data in the ATM cells, and have the postprocessor present the resulting data to an actual MPEG decoder.

10. Use signal-to-noise or subjective measurements to evaluate the quality of transmitted videos.

11. Implement the DART system on an actual MPEG network, rather than using an ATM simulator.

12. Use either formal cell loss models or empirical data to produce better simulations of the DART system.

13. Examine the effects of smaller video slices on the quality of received pictures when using the DART scheme.

Bibliography

[1] Susanna Aign and Khaled Fazel. "Temporal & Spatial Error Concealment Techniques for Hierarchical MPEG-2 Video Codec". In *ICC '95 Seattle: 1995 IEEE International Conference on Communications*, volume 3, pages 1777–1783, June 18-22, 1995.

[2] Vasudev Bhaskaran and Konstantinos Konstantinides. *Image and Video Compression Standards, Algorithms and Architectures.* Kluwer Academic Publishers, 1995.

[3] Richard A. Butler and G. C. McPherson. "Investigation of Failure Forecasting Algorithms". In *4th International Conference on Telecommunication Systems Modeling and Analysis*, pages 79–86, March 21-24, 1996.

[4] Thomas M. Chen and Stephen S. Liu. *ATM Switching Systems.* Artech House, Inc., 1995.

[5] The ATM Forum. *ATM User-Network Interface (UNI) Specification, Version 3.1.* The ATM Forum, September 1994.

[6] Frank Gadegast. MPEG FAQ version 4.0.
Internet URL= http://www.cs.tu-berlin.de/mpegfaq/, 1995.

[7] Didier Le Gall. “MPEG: A video Compression Standard for Multimedia Applications”. *Communications of the ACM*, 34(4):305–313, April 1991.

[8] Mohammad Ghanbari and Charles J. Hughes. Packing Coded Video Signals into ATM Cells. *IEEE/ACM Transactions on Networking*, 1(5):505–508, October 1993.

[9] MPEG Software Simulation Group. Source code for MPEG-2 codec. Internet URL= ftp://ftp.netcom.com/pub/cf/cfogg/mpeg2/mpeg2codec_v1.1a.tar.gz, July 1994.

[10] Taesop Han and Luis Orozco-Barbosa. “Performance Requirements for the Transport of MPEG Video Streams over ATM Networks”. In *ICC '95 Seattle: 1995 IEEE International Conference on Communications*, volume 1, pages 221–225, June 18-22, 1995.

[11] ISO/IEC. ISO/IEC DIS 13818-1, Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems, 1994.

[12] ISO/IEC. ISO/IEC DIS 13818-2, Information technology – Generic coding of moving pictures and associated audio information – Part 2: Video, 1994.

[13] Filippini Luigi. MPEG moving picture expert group FAQ. Internet URL= http://www.crs4.it/ luigi/MPEG/mpegfaq.html, July 1995.

[14] David E. McDysan and Darren L. Spohn. *ATM Theory and Application*. McGraw-Hilll Series on Computer Science. McGraw-Hill, Inc., 1995.

[15] Naohisa Ohta. *Packet Video: Modeling and Signal Processing.* Artech House, Boston, 1994.

[16] Davis Pan. "A Tutorial on MPEG/Audio Compression". *IEEE Multimedia*, 2(2):60–74, Summer 1995.

[17] Iain E. G. Richardson and Martyn J. Riley. Improving the error tolerance of MPEG video by varying slice size. *Signal Processing Journal*, 46(3), October 1995.

[18] Iain E. G. Richardson and Martyn J. Riley. Intelligent packetising of MPEG video data. In *S.P.I.E. Visual Communications and Image Processing*, volume 2501, May 1995.

[19] Iain E. G. Richardson and Martyn J. Riley. MPEG Coding for Error-Resilient Transmission. In *Fifth International Conference on Image Processing and its Applications*, July 1995.

[20] Iain E. G. Richardson and Martyn J. Riley. Usage Parameter Control Cell Loss Effects on MPEG Video. In *ICC '95 Seattle: 1995 IEEE International Conference on Communications*, volume 2, pages 970–851, June 18-22, 1995.

[21] Martyn J. Riley, B. U. Köhler, and Iain E. G. Richardson. Statistical Multiplexing MPEG Video Traffic with FEC. In *IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, July 1995.

[22] Martyn J. Riley and Iain E. G. Richardson. Adapting MPEG Video Traffic to Avoid Network Congestion. In *Fifth IEE Conference on Telecommunications*, March 1995.

[23] Dean L. Scoville and Mark J. Clement. "A Dependency-Based Strategy for Handling ATM Cell Loss in MPEG-2 Transport Streams". In *4th International Conference on Telecommunication Systems Modeling and Analysis*, pages 152–161, March 21-24, 1996.

[24] Brian Christopher Smith. *Implementation Techniques for Continuous Media Systems and Applications*. PhD thesis, University of California at Berkeley, 1994.

Appendix A

Simulation Data

A.1 Data from MPEG-1 Video Simulations

A.1.1 SEQ, GOP, Picture and Slice Usefulness Data

Video Name	Normal SEQs		Redundant SEQs					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	8.3	100.0	100.0	8.3	8.3	0.0	8.3	100.0
blossom.mpg	0.0	—	100.0	0.0	0.0	—	0.0	—
casino02.mpg	2.8	100.0	99.7	2.8	3.8	0.0	2.8	100.0
casinopr.mpg	5.6	100.0	100.0	5.6	11.1	0.0	5.6	100.0
Ext.lg.mpg	0.0	—	100.0	0.0	0.0	—	0.0	—
extraord.mpg	4.4	100.0	100.0	4.4	4.4	0.0	4.4	100.0
icecream.mpg	3.9	100.0	100.0	3.9	5.5	0.0	3.9	100.0
kiss.mpg	0.0	—	100.0	0.0	0.0	—	0.0	—
loubrock.mpg	0.0	—	100.0	0.0	0.0	—	0.0	—
neptune.mpg	0.0	—	100.0	0.0	20.0	0.0	0.0	—
OVERALL	3.4	100.0	99.9	3.4	4.5	0.0	3.4	100.0

Table A.1: *This table shows the usefulness of redundant Sequence of Pictures headers (SEQs) from a simulation of MPEG-1 videos with 10 percent redundant data.*

Video Name	Normal GOPs		Redundant GOPs					
			Sent		Lost		Useful	
	% Lost	% Replaced	% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	3.7	93.9	100.0	3.7	3.2	6.9	3.4	100.0
blossom.mpg	5.4	88.0	100.0	5.4	5.4	12.0	4.8	100.0
casino02.mpg	2.8	100.0	99.7	2.8	3.8	0.0	2.8	100.0
casinopr.mpg	3.2	100.0	99.7	3.2	6.9	0.0	3.2	100.0
Ext.lg.mpg	4.3	94.1	100.0	4.3	4.3	5.9	4.1	100.0
extraord.mpg	4.4	100.0	100.0	4.4	4.4	0.0	4.4	100.0
icecream.mpg	3.9	100.0	100.0	3.9	5.5	0.0	3.9	100.0
kiss.mpg	3.1	100.0	100.0	3.1	3.1	0.0	3.1	100.0
loubrock.mpg	5.2	94.7	99.7	5.2	2.2	12.5	4.9	100.0
neptune.mpg	3.5	100.0	100.0	3.5	2.6	0.0	3.5	100.0
OVERALL	3.9	95.6	99.9	3.9	3.9	4.4	3.7	100.0

Table A.2: *This table shows the usefulness of redundant Group of Pictures headers (GOPs) from a simulation of MPEG-1 videos with 10 percent redundant data.*

Video Name	Normal Pictures		Redundant Pictures					
			Sent		Lost		Useful	
	% Lost	% Replaced	% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	4.1	90.5	100.0	4.1	4.0	9.7	3.7	100.0
blossom.mpg	4.3	92.7	100.0	4.3	4.3	7.4	4.0	100.0
casino02.mpg	3.9	94.3	100.0	3.9	3.9	5.7	3.7	100.0
casinopr.mpg	3.8	96.1	100.0	3.8	3.8	3.9	3.6	100.0
Ext.lg.mpg	4.3	96.1	100.0	4.3	4.5	3.7	4.1	100.0
extraord.mpg	3.9	97.1	100.0	3.9	3.7	3.0	3.8	100.0
icecream.mpg	4.3	94.8	100.0	4.3	4.1	5.4	4.1	100.0
kiss.mpg	4.0	95.8	100.0	4.0	2.3	7.1	3.8	100.0
loubrock.mpg	4.3	94.4	100.0	4.3	2.6	9.4	4.1	99.1
neptune.mpg	4.1	97.3	100.0	4.1	3.8	2.9	4.0	100.0
OVERALL	4.1	94.9	100.0	4.1	3.7	5.7	3.9	99.9

Table A.3: *This table shows the usefulness of redundant picture headers from a simulation of MPEG-1 videos with 10 percent redundant data.*

Video Name	Normal Slices		Redundant Slices					
	% Lost	% Replaced	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	6.4	17.2	27.1	4.4	4.5	7.0	4.0	71.9
blossom.mpg	5.0	24.2	29.9	4.3	4.1	5.4	4.1	63.9
casino02.mpg	7.0	13.8	21.2	4.9	5.3	6.2	4.5	36.2
casinopr.mpg	9.9	12.2	20.0	6.6	6.4	8.4	6.0	18.9
Ext.lg.mpg	8.5	11.3	19.0	5.3	5.9	4.4	5.1	50.1
extraord.mpg	6.9	22.0	38.6	4.1	4.0	4.4	3.9	46.7
icecream.mpg	7.2	23.9	41.0	4.5	4.9	6.3	4.2	47.1
kiss.mpg	38.2	0.7	2.3	16.4	18.0	22.7	12.3	0.0
loubrock.mpg	39.2	0.1	0.5	17.9	21.4	33.3	10.7	0.0
neptune.mpg	6.7	14.6	21.3	4.8	4.9	4.0	4.6	49.6
OVERALL	7.9	15.2	27.0	4.7	4.9	5.9	4.5	45.3

Table A.4: *This table shows the usefulness of redundant slices from a simulation of MPEG-1 videos with 10 percent redundant data.*

A.1.2 Quality and Effectiveness Data

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431840	254520	299330	0.648	0.724	0.589	0.693	1.50
blossom.mpg	534138	312477	349877	0.437	0.494	0.585	0.655	0.91
casino02.mpg	1780328	591316	659912	1.420	1.573	0.332	0.371	1.08
casinopr.mpg	1781626	358446	399982	2.611	2.878	0.201	0.225	1.13
Ext.lg.mpg	810000	307050	338940	1.412	1.555	0.379	0.418	1.02
extraord.mpg	1782000	485848	563376	1.409	1.563	0.273	0.316	1.46
icecream.mpg	1781098	554774	655424	1.410	1.564	0.311	0.368	1.66
kiss.mpg	1780680	60390	60390	1.411	1.442	0.034	0.034	0.00
loubrock.mpg	1781670	113190	113190	1.411	1.435	0.064	0.064	0.00
neptune.mpg	1538145	716965	791654	1.325	1.468	0.466	0.515	0.97
TOTALS	14001525	3754976	4232075	13.494	14.697	0.268	0.302	1.43

Table A.5: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 videos with 10 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431840	249610	0.648	0.578	0.693	1.70
blossom.mpg	534138	298507	0.437	0.559	0.655	1.31
casino02.mpg	1780328	593626	1.420	0.333	0.371	1.04
casinopr.mpg	1781626	352264	2.611	0.198	0.225	1.33
Ext.lg.mpg	810000	320160	1.412	0.395	0.418	0.58
extraord.mpg	1782000	474320	1.409	0.266	0.316	1.71
icecream.mpg	1781098	570284	1.410	0.320	0.368	1.37
kiss.mpg	1780680	32670	1.411	0.018	0.034	37.43
loubrock.mpg	1781670	109230	1.411	0.061	0.064	2.14
neptune.mpg	1538145	715046	1.325	0.465	0.515	1.00
TOTALS	14001525	3715717	13.494	0.265	0.302	1.56

Table A.6: *This table shows the quality and effectiveness results from two simulations: one with 10 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431840	282370	302250	0.648	0.787	0.654	0.700	0.33
blossom.mpg	534138	326546	360415	0.437	0.535	0.611	0.675	0.46
casino02.mpg	1780328	628760	701074	1.420	1.710	0.353	0.394	0.56
casinopr.mpg	1781626	335148	432564	2.611	3.131	0.188	0.243	1.46
Ext.lg.mpg	810000	258405	292665	1.412	1.696	0.319	0.361	0.66
extraord.mpg	1782000	482196	567314	1.409	1.694	0.271	0.318	0.87
icecream.mpg	1781098	586696	713790	1.410	1.693	0.329	0.401	1.08
kiss.mpg	1780680	45870	47190	1.411	1.575	0.026	0.027	0.25
loubrock.mpg	1781670	82170	83160	1.411	1.555	0.046	0.047	0.12
neptune.mpg	1538145	696825	785080	1.325	1.598	0.453	0.510	0.62
TOTALS	14001525	3724986	4285502	13.494	15.974	0.266	0.306	0.82

Table A.7: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 videos with 20 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431840	249610	0.648	0.578	0.700	0.99
blossom.mpg	534138	298507	0.437	0.559	0.675	0.93
casino02.mpg	1780328	593626	1.420	0.333	0.394	0.89
casinopr.mpg	1781626	352264	2.611	0.198	0.243	1.14
Ext.lg.mpg	810000	320160	1.412	0.395	0.361	-0.43
extraord.mpg	1782000	474320	1.409	0.266	0.318	0.97
icecream.mpg	1781098	570284	1.410	0.320	0.401	1.25
kiss.mpg	1780680	32670	1.411	0.018	0.027	3.81
loubrock.mpg	1781670	109230	1.411	0.061	0.047	-2.35
neptune.mpg	1538145	715046	1.325	0.465	0.510	0.48
TOTALS	14001525	3715717	13.494	0.265	0.306	0.83

Table A.8: *This table shows the quality and effectiveness results from two simulations: one with 20 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431840	251590	303660	0.648	0.850	0.583	0.703	0.66
blossom.mpg	534138	298507	363924	0.437	0.576	0.559	0.681	0.69
casino02.mpg	1780328	621390	709060	1.420	1.851	0.349	0.398	0.46
casinopr.mpg	1781626	357214	427526	2.611	3.390	0.200	0.240	0.66
Ext.lg.mpg	810000	315690	356970	1.412	1.836	0.390	0.441	0.44
extraord.mpg	1782000	509740	623634	1.409	1.826	0.286	0.350	0.75
icecream.mpg	1781098	605066	745910	1.410	1.826	0.340	0.419	0.79
kiss.mpg	1780680	58410	59400	1.411	1.720	0.033	0.033	0.08
loubrock.mpg	1781670	119130	127380	1.411	1.723	0.067	0.071	0.31
neptune.mpg	1538145	716927	817076	1.325	1.729	0.466	0.531	0.46
TOTALS	14001525	3853664	4534540	13.494	17.327	0.275	0.324	0.62

Table A.9: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 videos with 30 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431840	249610	0.648	0.578	0.703	0.69
blossom.mpg	534138	298507	0.437	0.559	0.681	0.69
casino02.mpg	1780328	593626	1.420	0.333	0.398	0.64
casinopr.mpg	1781626	352264	2.611	0.198	0.240	0.72
Ext.lg.mpg	810000	320160	1.412	0.395	0.441	0.38
extraord.mpg	1782000	474320	1.409	0.266	0.350	1.06
icecream.mpg	1781098	570284	1.410	0.320	0.419	1.05
kiss.mpg	1780680	32670	1.411	0.018	0.033	3.73
loubrock.mpg	1781670	109230	1.411	0.061	0.071	0.75
neptune.mpg	1538145	715046	1.325	0.465	0.531	0.47
TOTALS	14001525	3715717	13.494	0.265	0.324	0.78

Table A.10: *This table shows the quality and effectiveness results from two simulations: one with 30 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431840	276900	311080	0.648	0.906	0.641	0.720	0.31
blossom.mpg	534138	320683	369215	0.437	0.618	0.600	0.691	0.37
casino02.mpg	1780328	610258	721644	1.420	1.988	0.343	0.405	0.46
casinopr.mpg	1781626	336710	445632	2.611	3.640	0.189	0.250	0.82
Ext.lg.mpg	810000	302040	353715	1.412	1.974	0.373	0.437	0.43
extraord.mpg	1782000	489236	644204	1.409	1.960	0.275	0.362	0.81
icecream.mpg	1781098	587906	784762	1.410	1.955	0.330	0.441	0.87
kiss.mpg	1780680	71610	75900	1.411	1.800	0.040	0.043	0.22
loubrock.mpg	1781670	102960	108240	1.411	1.815	0.058	0.061	0.18
neptune.mpg	1538145	676723	784130	1.325	1.858	0.440	0.510	0.40
TOTALS	14001525	3775026	4598522	13.494	18.515	0.270	0.328	0.59

Table A.11: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 videos with 40 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431840	249610	0.648	0.578	0.720	0.62
blossom.mpg	534138	298507	0.437	0.559	0.691	0.57
casino02.mpg	1780328	593626	1.420	0.333	0.405	0.54
casinopr.mpg	1781626	352264	2.611	0.198	0.250	0.67
Ext.lg.mpg	810000	320160	1.412	0.395	0.437	0.26
extraord.mpg	1782000	474320	1.409	0.266	0.362	0.92
icecream.mpg	1781098	570284	1.410	0.320	0.441	0.97
kiss.mpg	1780680	32670	1.411	0.018	0.043	4.79
loubrock.mpg	1781670	109230	1.411	0.061	0.061	-0.03
neptune.mpg	1538145	715046	1.325	0.465	0.510	0.24
TOTALS	14001525	3715717	13.494	0.265	0.328	0.64

Table A.12: *This table shows the quality and effectiveness results from two simulations: one with 40 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431840	253880	322120	0.648	0.965	0.588	0.746	0.55
blossom.mpg	534138	288970	390214	0.437	0.658	0.541	0.731	0.69
casino02.mpg	1780328	604208	775082	1.420	2.125	0.339	0.435	0.57
casinopr.mpg	1781626	330022	502150	2.611	3.904	0.185	0.282	1.05
Ext.lg.mpg	810000	324555	387870	1.412	2.109	0.401	0.479	0.39
extraord.mpg	1782000	477730	705122	1.409	2.097	0.268	0.396	0.98
icecream.mpg	1781098	583286	834174	1.410	2.085	0.327	0.468	0.90
kiss.mpg	1780680	78210	83490	1.411	1.907	0.044	0.047	0.19
loubrock.mpg	1781670	88440	95040	1.411	1.956	0.050	0.053	0.19
neptune.mpg	1538145	511936	810597	1.325	1.984	0.333	0.527	1.17
TOTALS	14001525	3541237	4905859	13.494	19.790	0.253	0.350	0.83

Table A.13: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 videos with 50 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431840	249610	0.648	0.578	0.746	0.59
blossom.mpg	534138	298507	0.437	0.559	0.731	0.61
casino02.mpg	1780328	593626	1.420	0.333	0.435	0.62
casinopr.mpg	1781626	352264	2.611	0.198	0.282	0.86
Ext.lg.mpg	810000	320160	1.412	0.395	0.479	0.43
extraord.mpg	1782000	474320	1.409	0.266	0.396	1.00
icecream.mpg	1781098	570284	1.410	0.320	0.468	0.97
kiss.mpg	1780680	32670	1.411	0.018	0.047	4.42
loubrock.mpg	1781670	109230	1.411	0.061	0.053	-0.34
neptune.mpg	1538145	715046	1.325	0.465	0.527	0.27
TOTALS	14001525	3715717	13.494	0.265	0.350	0.69

Table A.14: *This table shows the quality and effectiveness results from two simulations: one with 50 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 videos.*

A.1.3 General Usefulness Data

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	6.3	17.9	28.1	4.3	4.5	7.1	4.0	73.1
blossom.mpg	5.0	24.8	30.6	4.3	4.1	5.5	4.1	65.1
casino02.mpg	7.0	13.9	21.5	4.9	5.3	6.2	4.5	37.0
casinopr.mpg	9.9	12.3	20.3	6.5	6.4	8.3	6.0	19.7
Ext.lg.mpg	8.5	11.6	19.5	5.3	5.9	4.4	5.0	51.5
extraord.mpg	6.9	22.2	38.8	4.1	4.0	4.3	3.9	47.2
icecream.mpg	7.2	24.0	41.2	4.5	4.9	6.3	4.2	47.5
kiss.mpg	38.1	0.8	2.6	14.8	16.1	22.4	11.2	4.2
loubrock.mpg	39.1	0.2	0.8	12.7	14.2	31.7	8.2	19.2
neptune.mpg	6.7	14.8	21.6	4.8	4.9	4.0	4.6	50.3
OVERALL	15.1	6.6	21.7	4.9	5.1	6.9	4.6	42.4

Table A.15: *This table shows the usefulness of redundant data in a simulation of MPEG-1 videos with 10 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-	Sent		Lost		Useful	
		placed	% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	6.6	30.8	44.8	4.8	4.8	6.1	4.5	63.7
blossom.mpg	5.2	39.4	48.0	4.5	4.2	5.9	4.3	67.9
casino02.mpg	7.2	26.5	37.2	5.4	5.5	5.6	5.1	37.1
casinopr.mpg	9.9	24.7	37.0	7.2	6.8	7.5	6.6	18.8
Ext.lg.mpg	8.6	22.5	32.8	6.3	5.9	7.4	5.9	37.2
extraord.mpg	7.3	37.2	55.6	5.1	5.2	4.8	4.9	36.7
icecream.mpg	7.0	37.8	58.2	4.8	5.2	4.4	4.6	47.2
kiss.mpg	37.9	10.0	22.8	22.3	22.1	26.1	16.6	2.3
loubrock.mpg	39.5	7.4	16.7	23.3	24.3	24.2	17.5	2.3
neptune.mpg	7.1	28.6	38.1	5.6	5.1	5.9	5.3	51.8
OVERALL	15.2	16.8	38.4	7.6	7.6	13.1	6.6	27.9

Table A.16: *This table shows the usefulness of redundant data in a simulation of MPEG-1 videos with 20 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-	Sent		Lost		Useful	
		placed	% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	6.4	43.1	59.1	5.0	5.2	5.1	4.7	65.7
blossom.mpg	5.3	52.3	60.8	4.8	4.5	5.8	4.6	61.3
casino02.mpg	7.1	39.2	50.2	5.9	5.7	5.8	5.6	37.6
casinopr.mpg	10.1	35.8	50.9	7.6	7.2	7.1	7.1	16.8
Ext.lg.mpg	8.9	34.6	45.0	7.4	6.9	8.3	6.9	43.4
extraord.mpg	6.9	44.4	65.1	5.0	5.2	4.9	4.7	37.2
icecream.mpg	7.1	46.3	67.5	5.2	5.2	5.8	4.9	45.9
kiss.mpg	38.5	18.9	41.8	24.5	24.7	28.5	17.5	0.9
loubrock.mpg	38.8	17.2	35.6	27.4	31.8	27.1	18.8	7.1
neptune.mpg	6.9	41.5	52.2	5.8	5.8	5.3	5.4	51.4
OVERALL	15.2	26.6	52.2	9.5	9.8	17.3	7.8	24.4

Table A.17: *This table shows the usefulness of redundant data in a simulation of MPEG-1 videos with 30 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Replaced	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	6.4	54.1	68.8	5.3	5.4	5.1	5.0	67.8
blossom.mpg	5.1	57.6	70.2	4.4	4.9	4.1	4.2	70.5
casino02.mpg	7.2	46.8	60.5	5.9	5.9	6.1	5.5	36.3
casinopr.mpg	10.1	43.5	61.7	7.7	7.8	8.1	7.1	19.4
Ext.lg.mpg	8.9	42.9	55.7	7.3	7.1	7.3	6.8	43.5
extraord.mpg	7.0	52.8	72.3	5.4	5.5	5.7	5.1	38.2
icecream.mpg	6.9	52.5	73.9	5.3	5.7	6.2	4.9	46.1
kiss.mpg	37.3	25.7	53.2	24.4	25.4	25.2	18.0	2.6
loubrock.mpg	38.0	23.9	47.4	26.3	28.7	24.9	19.2	3.4
neptune.mpg	6.9	49.2	64.0	5.7	5.6	6.2	5.3	47.4
OVERALL	14.9	33.9	62.1	9.8	10.2	16.2	8.1	23.1

Table A.18: *This table shows the usefulness of redundant data in a simulation of MPEG-1 videos with 40 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Replaced	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	6.2	58.9	76.2	5.1	5.7	5.8	4.8	70.0
blossom.mpg	5.1	65.2	77.3	4.6	5.2	5.0	4.3	74.4
casino02.mpg	7.0	55.6	69.3	6.0	6.2	6.2	5.6	43.4
casinopr.mpg	10.1	52.8	70.6	8.3	8.3	9.0	7.6	23.5
Ext.lg.mpg	8.7	52.7	65.3	7.5	7.2	7.1	7.0	43.6
extraord.mpg	7.0	60.0	78.8	5.7	6.0	5.9	5.3	40.9
icecream.mpg	7.1	60.0	79.4	5.7	6.0	6.0	5.3	49.9
kiss.mpg	37.5	32.2	66.8	25.0	26.5	26.2	18.1	2.5
loubrock.mpg	40.5	34.0	62.7	30.5	28.4	30.0	22.0	2.8
neptune.mpg	6.7	58.9	73.5	5.7	5.8	5.4	5.4	47.4
OVERALL	15.2	42.0	71.6	11.0	11.1	18.4	8.9	22.9

Table A.19: *This table shows the usefulness of redundant data in a simulation of MPEG-1 videos with 50 percent redundant data.*

A.2 Data from MPEG-2 Video Simulations

A.2.1 Quality and Effectiveness Data

Video Name	Macroblocks			Mean bit rate		Quality		Effective-ness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
boat.m2v	8748000	4024215	4366980	7.133	7.804	0.460	0.499	0.90
	8743140	4159935	4431555	7.133	7.804	0.476	0.507	0.69
flwr_080.m2v	7126680	2499596	2573428	9.329	10.255	0.351	0.361	0.30
	7122720	2445432	2483800	9.329	10.255	0.343	0.349	0.16
np.es.m2v	8745570	3557295	3638745	8.424	9.263	0.407	0.416	0.23
	8741520	3343725	3447180	8.423	9.262	0.383	0.394	0.31
susi_080.m2v	7125360	2347972	2389992	9.326	10.253	0.330	0.335	0.18
	7121400	2186228	2263932	9.327	10.253	0.307	0.318	0.36
tens_080.m2v	8553600	2707408	2982276	11.153	12.260	0.317	0.349	1.02
	8548848	3031248	3086116	11.154	12.262	0.355	0.361	0.18
TOTALS	80576838	30303054	31664004	90.731	99.671	0.376	0.393	0.46

Table A.20: *This table shows the quality and effectiveness results from a single simulation of MPEG-2 videos with 10 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
boat.m2v	8748000	4194945	7.133	0.480	0.499	0.44
	8743140	4190175	7.133	0.479	0.507	0.61
flwr_080.m2v	7126680	2529340	9.329	0.355	0.361	0.18
	7122720	2467476	9.329	0.346	0.349	0.07
np.es.m2v	8745570	3312945	8.424	0.379	0.416	0.99
	8741520	3456540	8.423	0.395	0.394	-0.03
susi_080.m2v	7125360	2331956	9.326	0.327	0.335	0.25
	7121400	2251920	9.327	0.316	0.318	0.05
tens_080.m2v	8553600	3008236	11.153	0.352	0.349	-0.09
	8548848	2976160	11.154	0.348	0.361	0.37
TOTALS	80576838	30719693	90.731	0.381	0.393	0.31

Table A.21: *This table shows the quality and effectiveness results from two simulations: one with 10 percent redundant data and one without any redundant data. Both simulations included the same MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effective-
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	ness
boat.m2v	8748000	4035015	4518720	7.133	8.497	0.461	0.517	0.63
	8743140	4139055	4712040	7.133	8.498	0.473	0.539	0.72
flwr_080.m2v	7126680	2363460	2441384	9.329	11.182	0.332	0.343	0.17
	7122720	2425192	2542364	9.329	11.182	0.340	0.357	0.24
np.es.m2v	8745570	3514590	3612285	8.424	10.089	0.402	0.413	0.14
	8741520	3395340	3508335	8.423	10.089	0.388	0.401	0.17
susi_080.m2v	7125360	2183324	2242284	9.326	11.179	0.306	0.315	0.14
	7121400	2315500	2352152	9.327	11.180	0.325	0.330	0.08
tens_080.m2v	8553600	2968592	3061740	11.153	13.370	0.347	0.358	0.16
	8548848	2890624	3190924	11.154	13.371	0.338	0.373	0.52
TOTALS	80576838	30230692	32182228	90.731	108.637	0.375	0.399	0.33

Table A.22: *This table shows the quality and effectiveness results from a single simulation of MPEG-2 videos with 20 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
boat.m2v	8748000	4194945	7.133	0.480	0.517	0.40
	8743140	4190175	7.133	0.479	0.539	0.65
flwr_080.m2v	7126680	2529340	9.329	0.355	0.343	-0.18
	7122720	2467476	9.329	0.346	0.357	0.15
np.es.m2v	8745570	3312945	8.424	0.379	0.413	0.46
	8741520	3456540	8.423	0.395	0.401	0.08
susi_080.m2v	7125360	2331956	9.326	0.327	0.315	-0.19
	7121400	2251920	9.327	0.316	0.330	0.22
tens_080.m2v	8553600	3008236	11.153	0.352	0.358	0.09
	8548848	2976160	11.154	0.348	0.373	0.36
TOTALS	80576838	30719693	90.731	0.381	0.399	0.24

Table A.23: *This table shows the quality and effectiveness results from two simulations: one with 20 percent redundant data and one without any redundant data. Both simulations included the same MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effective-
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	ness
boat.m2v	8748000	4151250	4817295	7.133	9.179	0.475	0.551	0.56
	8743140	4072005	4779450	7.133	9.179	0.466	0.547	0.61
flwr_080.m2v	7126680	2386252	2518604	9.329	12.088	0.335	0.353	0.19
	7122720	2501576	2651572	9.329	12.089	0.351	0.372	0.20
np.es.m2v	8745570	3467430	3754305	8.424	10.916	0.396	0.429	0.28
	8741520	3236085	3540060	8.423	10.916	0.370	0.405	0.32
susi_080.m2v	7125360	2224772	2299088	9.326	12.082	0.312	0.323	0.11
	7121400	2220284	2307668	9.327	12.083	0.312	0.324	0.13
tens_080.m2v	8553600	2999524	3144768	11.153	14.455	0.351	0.368	0.16
	8548848	2895420	3072652	11.154	14.456	0.339	0.359	0.21
TOTALS	80576838	30154598	32885462	90.731	117.444	0.374	0.408	0.31

Table A.24: *This table shows the quality and effectiveness results from a single simulation of MPEG-2 videos with 30 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
boat.m2v	8748000	4194945	7.133	0.480	0.551	0.52
	8743140	4190175	7.133	0.479	0.547	0.49
flwr_080.m2v	7126680	2529340	9.329	0.355	0.353	-0.01
	7122720	2467476	9.329	0.346	0.372	0.25
np.es.m2v	8745570	3312945	8.424	0.379	0.429	0.45
	8741520	3456540	8.423	0.395	0.405	0.08
susi_080.m2v	7125360	2331956	9.326	0.327	0.323	-0.05
	7121400	2251920	9.327	0.316	0.324	0.08
tens_080.m2v	8553600	3008236	11.153	0.352	0.368	0.15
	8548848	2976160	11.154	0.348	0.359	0.11
TOTALS	80576838	30719693	90.731	0.381	0.408	0.24

Table A.25: *This table shows the quality and effectiveness results from two simulations: one with 30 percent redundant data and one without any redundant data. Both simulations included the same MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effective-
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	ness
boat.m2v	8748000	4130235	4944150	7.133	9.848	0.472	0.565	0.52
	8743140	4158090	4883535	7.133	9.849	0.476	0.559	0.46
flwr_080.m2v	7126680	2265604	2872452	9.329	12.999	0.318	0.403	0.68
	7122720	2387220	2758932	9.329	13.000	0.335	0.387	0.40
np.es.m2v	8745570	3445200	4048335	8.424	11.743	0.394	0.463	0.44
	8741520	3515715	4111380	8.423	11.744	0.402	0.470	0.43
susi_080.m2v	7125360	2126740	2408076	9.326	13.002	0.298	0.338	0.34
	7121400	2114156	2376088	9.327	13.003	0.297	0.334	0.31
tens_080.m2v	8553600	3006212	3402916	11.153	15.571	0.351	0.398	0.33
	8548848	2930796	3381664	11.154	15.572	0.343	0.396	0.39
TOTALS	80576838	30079968	35187528	90.731	126.332	0.373	0.437	0.43

Table A.26: *This table shows the quality and effectiveness results from a single simulation of MPEG-2 videos with 40 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
boat.m2v	8748000	4194945	7.133	0.480	0.565	0.47
	8743140	4190175	7.133	0.479	0.559	0.43
flwr_080.m2v	7126680	2529340	9.329	0.355	0.403	0.34
	7122720	2467476	9.329	0.346	0.387	0.30
np.es.m2v	8745570	3312945	8.424	0.379	0.463	0.56
	8741520	3456540	8.423	0.395	0.470	0.48
susi_080.m2v	7125360	2331956	9.326	0.327	0.338	0.08
	7121400	2251920	9.327	0.316	0.334	0.14
tens_080.m2v	8553600	3008236	11.153	0.352	0.398	0.33
	8548848	2976160	11.154	0.348	0.396	0.34
TOTALS	80576838	30719693	90.731	0.381	0.437	0.37

Table A.27: *This table shows the quality and effectiveness results from two simulations: one with 40 percent redundant data and one without any redundant data. Both simulations included the same MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effective-
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	ness
boat.m2v	8748000	4085640	5072850	7.133	10.542	0.467	0.580	0.51
	8743140	4153095	5167530	7.133	10.543	0.475	0.591	0.51
flwr_080.m2v	7126680	2441384	3200692	9.329	13.908	0.343	0.449	0.63
	7122720	2572240	3337664	9.329	13.909	0.361	0.469	0.61
np.es.m2v	8745570	3308850	4293675	8.424	12.586	0.378	0.491	0.60
	8741520	3502350	4385565	8.423	12.586	0.401	0.502	0.51
susi_080.m2v	7125360	2182092	2645280	9.326	13.915	0.306	0.371	0.43
	7121400	2025364	2790348	9.327	13.916	0.284	0.392	0.77
tens_080.m2v	8553600	2998952	3688916	11.153	16.659	0.351	0.431	0.47
	8548848	3145604	3893560	11.154	16.661	0.368	0.455	0.48
TOTALS	80576838	30415571	38476080	90.731	135.225	0.377	0.478	0.54

Table A.28: *This table shows the quality and effectiveness results from a single simulation of MPEG-2 videos with 50 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
boat.m2v	8748000	4194945	7.133	0.480	0.580	0.44
	8743140	4190175	7.133	0.479	0.591	0.49
flwr_080.m2v	7126680	2529340	9.329	0.355	0.449	0.54
	7122720	2467476	9.329	0.346	0.469	0.72
np.es.m2v	8745570	3312945	8.424	0.379	0.491	0.60
	8741520	3456540	8.423	0.395	0.502	0.54
susi_080.m2v	7125360	2331956	9.326	0.327	0.371	0.27
	7121400	2251920	9.327	0.316	0.392	0.49
tens_080.m2v	8553600	3008236	11.153	0.352	0.431	0.46
	8548848	2976160	11.154	0.348	0.455	0.62
TOTALS	80576838	30719693	90.731	0.381	0.478	0.51

Table A.29: *This table shows the quality and effectiveness results from two simulations: one with 50 percent redundant data and one without any redundant data. Both simulations included the same MPEG-2 videos.*

A.2.2 General Usefulness Data

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
boat.m2v	2.4	12.9	21.2	1.5	1.5	1.7	1.5	55.6
	2.3	12.1	21.2	1.3	1.5	1.3	1.3	53.4
flwr_080.m2v	3.2	13.3	27.3	1.6	1.6	1.8	1.6	29.6
	3.3	14.3	27.3	1.7	1.8	1.3	1.7	31.2
np.es.m2v	2.6	12.7	22.7	1.5	1.6	1.8	1.5	27.7
	2.7	12.4	22.8	1.5	1.8	1.9	1.4	24.3
susi_080.m2v	3.1	11.5	20.4	1.8	2.1	2.1	1.8	22.5
	3.2	12.4	20.4	2.0	1.9	1.4	2.0	25.5
tens_080.m2v	3.3	12.9	22.5	1.9	2.0	1.9	1.9	33.2
	3.3	12.6	22.5	1.9	1.8	1.6	1.8	31.3
OVERALL	2.9	12.7	22.7	1.7	1.7	1.7	1.6	33.0

Table A.30: *This table shows the usefulness of redundant data in a simulation of MPEG-2 videos with 10 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
boat.m2v	2.3	23.9	35.5	1.6	1.7	1.6	1.6	56.9
	2.3	25.1	35.5	1.7	1.5	1.4	1.6	56.4
flwr_080.m2v	3.3	25.9	45.1	1.9	2.0	2.7	1.9	25.2
	3.2	25.2	45.1	1.8	1.9	2.0	1.8	28.1
np.es.m2v	2.6	24.6	39.8	1.7	1.8	2.3	1.6	27.5
	2.6	23.6	39.8	1.6	1.8	3.0	1.5	29.4
susi_080.m2v	3.2	24.2	38.5	2.0	2.0	2.3	2.0	23.8
	3.2	23.0	38.5	1.9	2.1	1.4	1.9	23.2
tens_080.m2v	3.3	24.8	40.1	2.1	1.9	2.2	2.0	28.9
	3.3	25.0	40.1	2.1	2.2	2.1	2.1	32.5
OVERALL	2.9	24.6	39.7	1.8	1.9	2.1	1.8	32.6

Table A.31: *This table shows the usefulness of redundant data in a simulation of MPEG-2 videos with 20 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
boat.m2v	2.4	36.4	48.6	1.8	1.6	2.2	1.8	54.5
	2.2	35.3	48.6	1.7	1.7	1.5	1.6	54.7
flwr_080.m2v	3.2	36.0	58.8	2.0	2.0	1.8	2.0	27.0
	3.2	35.5	58.8	2.0	2.1	2.7	1.9	28.1
np.es.m2v	2.6	35.1	51.5	1.8	1.9	1.5	1.8	35.6
	2.7	34.6	51.5	1.9	1.9	2.3	1.8	32.7
susi_080.m2v	3.3	35.3	53.9	2.2	2.2	3.2	2.1	24.7
	3.3	34.8	53.9	2.2	2.2	2.2	2.1	24.1
tens_080.m2v	3.3	35.8	54.2	2.2	2.1	2.8	2.1	28.7
	3.3	35.5	54.2	2.2	2.1	1.9	2.2	31.2
OVERALL	2.9	35.4	53.2	2.0	2.0	2.2	1.9	33.4

Table A.32: *This table shows the usefulness of redundant data in a simulation of MPEG-2 videos with 30 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
boat.m2v	2.2	45.7	59.5	1.7	1.8	2.1	1.7	55.1
	2.3	46.6	59.5	1.8	1.8	2.1	1.8	53.6
flwr_080.m2v	3.2	44.9	68.8	2.1	2.1	2.0	2.1	38.0
	3.2	44.9	68.8	2.2	2.2	2.4	2.1	34.8
np.es.m2v	2.7	44.3	60.2	2.0	2.0	2.3	2.0	43.0
	2.6	44.7	60.1	2.0	2.0	2.5	1.9	44.8
susi_080.m2v	3.2	44.4	65.5	2.2	2.3	2.9	2.2	31.5
	3.2	43.6	65.5	2.2	2.3	2.6	2.1	28.8
tens_080.m2v	3.2	44.3	64.6	2.3	2.3	3.1	2.2	36.1
	3.2	43.2	64.6	2.2	2.3	2.2	2.1	35.2
OVERALL	2.9	44.6	63.4	2.1	2.1	2.4	2.0	39.6

Table A.33: *This table shows the usefulness of redundant data in a simulation of MPEG-2 videos with 40 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
boat.m2v	2.3	54.6	68.8	1.9	1.9	2.0	1.9	56.3
	2.3	55.2	68.8	1.9	2.0	1.7	1.9	58.2
flwr_080.m2v	3.3	56.2	75.9	2.5	2.5	2.3	2.4	44.6
	3.2	54.9	75.9	2.3	2.4	2.5	2.3	48.0
np.es.m2v	2.8	54.0	68.1	2.2	2.1	2.5	2.2	50.9
	2.6	53.6	68.1	2.1	2.1	2.3	2.0	50.0
susi_080.m2v	3.3	51.9	72.5	2.4	2.4	3.3	2.4	35.5
	3.2	53.1	72.5	2.4	2.5	3.3	2.4	38.5
tens_080.m2v	3.2	52.7	72.4	2.4	2.5	3.0	2.3	40.7
	3.2	54.8	72.4	2.5	2.4	3.0	2.4	43.7
OVERALL	2.9	54.1	71.3	2.3	2.3	2.6	2.2	46.3

Table A.34: *This table shows the usefulness of redundant data in a simulation of MPEG-2 videos with 50 percent redundant data.*

A.3 Data from Mixed (MPEG-1 and MPEG-2) Video Simulations

A.3.1 Quality and Effectiveness Data

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431810	150510	192520	0.648	0.724	0.349	0.446	2.39
blossom.mpg	534105	193303	233178	0.437	0.494	0.362	0.437	1.57
casinopr.mpg	1781340	145486	186956	2.611	2.878	0.082	0.105	2.79
Ext.lg.mpg	810000	141300	175770	1.412	1.555	0.174	0.217	2.40
extraord.mpg	1781956	298650	400818	1.409	1.563	0.168	0.225	3.12
kiss.mpg	1780680	5280	5280	1.411	1.442	0.003	0.003	0.00
loubrock.mpg	1781340	6270	8580	1.411	1.435	0.004	0.005	21.72
neptune.mpg	1537860	327750	411920	1.326	1.468	0.213	0.268	2.39
boat.m2v	8743095	465840	541530	7.133	7.804	0.053	0.062	1.73
np.es.m2v	8739135	439785	485865	8.424	9.263	0.050	0.056	1.05
TOTALS	27921321	2174174	2642417	26.221	28.627	0.078	0.095	2.35

Table A.35: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 and MPEG-2 videos with 10 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431810	160310	0.648	0.371	0.446	1.72
blossom.mpg	534105	164120	0.437	0.307	0.437	3.20
casinopr.mpg	1781340	128062	2.611	0.072	0.105	4.50
Ext.lg.mpg	810000	131430	1.412	0.162	0.217	3.32
extraord.mpg	1781978	313148	1.409	0.176	0.225	2.55
kiss.mpg	1780680	4950	1.411	0.003	0.003	2.94
loubrock.mpg	1781340	16500	1.411	0.009	0.005	-28.30
neptune.mpg	1537898	237614	1.326	0.155	0.268	6.84
boat.m2v	8743140	418860	7.133	0.048	0.062	3.11
np.es.m2v	8739225	417825	8.424	0.048	0.056	1.64
TOTALS	27921516	1992819	26.221	0.071	0.095	3.55

Table A.36: *This table shows the quality and effectiveness results from two simulations: one with 10 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 and MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431800	173460	198510	0.648	0.787	0.402	0.460	0.67
blossom.mpg	534105	187066	245487	0.437	0.535	0.350	0.460	1.39
casinopr.mpg	1781340	160710	195690	2.611	3.132	0.090	0.110	1.09
Ext.lg.mpg	810000	152625	193935	1.412	1.696	0.188	0.239	1.34
extraord.mpg	1781890	313346	407572	1.409	1.694	0.176	0.229	1.49
kiss.mpg	1780680	7260	7260	1.411	1.575	0.004	0.004	0.00
loubrock.mpg	1781340	7260	7260	1.411	1.554	0.004	0.004	0.00
neptune.mpg	1537860	322962	418437	1.326	1.598	0.210	0.272	1.44
boat.m2v	8742825	525960	643140	7.133	8.498	0.060	0.074	1.16
np.es.m2v	8738955	454815	505395	8.424	10.089	0.052	0.058	0.56
TOTALS	27920795	2305464	2822686	26.221	31.157	0.083	0.101	1.19

Table A.37: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 and MPEG-2 videos with 20 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431810	160310	0.648	0.371	0.460	1.11
blossom.mpg	534105	164120	0.437	0.307	0.460	2.21
casinopr.mpg	1781340	128062	2.611	0.072	0.110	2.65
Ext.lg.mpg	810000	131430	1.412	0.162	0.239	2.36
extraord.mpg	1781978	313148	1.409	0.176	0.229	1.49
kiss.mpg	1780680	4950	1.411	0.003	0.004	4.00
loubrock.mpg	1781340	16500	1.411	0.009	0.004	-5.53
neptune.mpg	1537898	237614	1.326	0.155	0.272	3.70
boat.m2v	8743140	418860	7.133	0.048	0.074	2.80
np.es.m2v	8739225	417825	8.424	0.048	0.058	1.06
TOTALS	27921516	1992819	26.221	0.071	0.101	2.21

Table A.38: *This table shows the quality and effectiveness results from two simulations: one with 20 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 and MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431800	179690	213230	0.648	0.850	0.416	0.494	0.60
blossom.mpg	534105	173162	256047	0.437	0.576	0.324	0.479	1.50
casinopr.mpg	1781318	116996	169268	2.611	3.390	0.066	0.095	1.50
Ext.lg.mpg	810000	110625	209280	1.412	1.836	0.137	0.258	2.97
extraord.mpg	1781890	326172	443190	1.409	1.826	0.183	0.249	1.21
kiss.mpg	1780680	4950	5610	1.411	1.720	0.003	0.003	0.61
loubrock.mpg	1781340	10560	16170	1.411	1.723	0.006	0.009	2.41
neptune.mpg	1537860	354122	450699	1.326	1.729	0.230	0.293	0.90
boat.m2v	8742735	506340	664155	7.133	9.179	0.058	0.076	1.09
np.es.m2v	8738910	456345	543600	8.424	10.916	0.052	0.062	0.65
TOTALS	27920638	2238962	2971249	26.221	33.745	0.080	0.106	1.14

Table A.39: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 and MPEG-2 videos with 30 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431810	160310	0.648	0.371	0.494	1.06
blossom.mpg	534105	164120	0.437	0.307	0.479	1.76
casinopr.mpg	1781340	128062	2.611	0.072	0.095	1.08
Ext.lg.mpg	810000	131430	1.412	0.162	0.258	1.97
extraord.mpg	1781978	313148	1.409	0.176	0.249	1.40
kiss.mpg	1780680	4950	1.411	0.003	0.003	0.61
loubrock.mpg	1781340	16500	1.411	0.009	0.009	-0.09
neptune.mpg	1537898	237614	1.326	0.155	0.293	2.95
boat.m2v	8743140	418860	7.133	0.048	0.076	2.04
np.es.m2v	8739225	417825	8.424	0.048	0.062	1.02
TOTALS	27921516	1992819	26.221	0.071	0.106	1.71

Table A.40: *This table shows the quality and effectiveness results from two simulations: one with 30 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 and MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431800	142350	213070	0.648	0.906	0.330	0.493	1.25
blossom.mpg	534105	179344	259721	0.437	0.618	0.336	0.486	1.08
casinopr.mpg	1781318	149270	202092	2.611	3.640	0.084	0.113	0.90
Ext.lg.mpg	810000	168015	208755	1.412	1.974	0.207	0.258	0.61
extraord.mpg	1781890	320804	472846	1.409	1.960	0.180	0.265	1.21
kiss.mpg	1780680	3630	3960	1.411	1.800	0.002	0.002	0.33
loubrock.mpg	1781340	13530	19470	1.411	1.814	0.008	0.011	1.54
neptune.mpg	1537860	349638	454024	1.326	1.858	0.227	0.295	0.74
boat.m2v	8742600	510750	689805	7.133	9.848	0.058	0.079	0.92
np.es.m2v	8738820	476910	612585	8.424	11.744	0.055	0.070	0.72
TOTALS	27920413	2314241	3136328	26.221	36.163	0.083	0.112	0.94

Table A.41: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 and MPEG-2 videos with 40 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431810	160310	0.648	0.371	0.493	0.83
blossom.mpg	534105	164120	0.437	0.307	0.486	1.41
casinopr.mpg	1781340	128062	2.611	0.072	0.113	1.47
Ext.lg.mpg	810000	131430	1.412	0.162	0.258	1.48
extraord.mpg	1781978	313148	1.409	0.176	0.265	1.30
kiss.mpg	1780680	4950	1.411	0.003	0.002	-0.72
loubrock.mpg	1781340	16500	1.411	0.009	0.011	0.63
neptune.mpg	1537898	237614	1.326	0.155	0.295	2.27
boat.m2v	8743140	418860	7.133	0.048	0.079	1.70
np.es.m2v	8739225	417825	8.424	0.048	0.070	1.18
TOTALS	27921516	1992819	26.221	0.071	0.112	1.51

Table A.42: *This table shows the quality and effectiveness results from two simulations: one with 40 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 and MPEG-2 videos.*

Video Name	Macroblocks			Mean bit rate		Quality		Effectiveness
	S	R_n	R_r	B_n	B_r	Q_n	Q_r	
aspen3.mpg	431800	183980	244030	0.648	0.965	0.426	0.565	0.67
blossom.mpg	534105	162624	277541	0.437	0.658	0.304	0.520	1.40
casinopr.mpg	1781318	127468	237732	2.611	3.904	0.072	0.133	1.75
Ext.lg.mpg	810000	157080	215175	1.412	2.109	0.194	0.266	0.75
extraord.mpg	1781890	333520	533280	1.409	2.097	0.187	0.299	1.23
kiss.mpg	1780680	9570	11220	1.411	1.907	0.005	0.006	0.49
loubrock.mpg	1781340	16170	16830	1.411	1.956	0.009	0.009	0.11
neptune.mpg	1537860	277780	505970	1.326	1.984	0.181	0.329	1.65
boat.m2v	8742600	516465	732240	7.133	10.543	0.059	0.084	0.87
np.es.m2v	8738820	473940	700380	8.424	12.586	0.054	0.080	0.97
TOTALS	27920413	2258597	3474398	26.221	38.708	0.081	0.124	1.13

Table A.43: *This table shows the quality and effectiveness results from a single simulation of MPEG-1 and MPEG-2 videos with 50 percent redundant data.*

Video Name	Macroblocks		Mean bit rate	Quality		Effectiveness
	S_n	R_n	B_n	Q_n	Q_r	
aspen3.mpg	431810	160310	0.648	0.371	0.565	1.07
blossom.mpg	534105	164120	0.437	0.307	0.520	1.37
casinopr.mpg	1781340	128062	2.611	0.072	0.133	1.73
Ext.lg.mpg	810000	131430	1.412	0.162	0.266	1.29
extraord.mpg	1781978	313148	1.409	0.176	0.299	1.44
kiss.mpg	1780680	4950	1.411	0.003	0.006	3.60
loubrock.mpg	1781340	16500	1.411	0.009	0.009	0.05
neptune.mpg	1537898	237614	1.326	0.155	0.329	2.27
boat.m2v	8743140	418860	7.133	0.048	0.084	1.57
np.es.m2v	8739225	417825	8.424	0.048	0.080	1.37
TOTALS	27921516	1992819	26.221	0.071	0.124	1.56

Table A.44: *This table shows the quality and effectiveness results from two simulations: one with 50 percent redundant data and one without any redundant data. Both simulations included the same MPEG-1 and MPEG-2 videos.*

A.3.2 General Usefulness Data

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Replaced	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	15.3	16.4	28.1	10.6	11.6	14.7	8.9	47.7
blossom.mpg	11.5	22.8	30.6	9.6	8.9	11.7	8.6	42.4
casinopr.mpg	19.9	12.3	20.3	14.3	14.2	15.9	12.1	6.9
Ext.lg.mpg	18.3	11.7	19.5	12.8	13.4	14.3	10.9	31.6
extraord.mpg	17.0	23.3	38.8	13.6	12.9	26.6	10.2	34.5
kiss.mpg	59.9	0.8	2.6	31.5	32.3	40.4	18.4	6.4
loubrock.mpg	63.8	0.3	0.8	26.0	13.3	2.3	25.7	14.3
neptune.mpg	16.1	16.4	21.5	13.7	9.6	14.9	12.3	31.9
boat.m2v	22.4	11.2	21.2	15.0	17.0	18.1	11.9	8.0
np.es.m2v	25.1	10.4	22.8	17.4	21.4	27.6	11.5	0.7
OVERALL	26.8	8.8	20.6	15.4	17.0	22.8	11.5	12.0

Table A.45: *This table shows the usefulness of redundant data in a simulation containing a mixture of MPEG-1 and MPEG-2 videos with 10 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	15.1	28.2	44.8	11.8	12.8	17.6	9.5	40.5
blossom.mpg	11.1	36.8	48.0	9.4	8.8	10.8	8.5	42.0
casinopr.mpg	19.3	22.6	37.0	14.0	14.4	15.3	11.8	6.4
Ext.lg.mpg	17.9	20.5	32.8	13.4	14.3	15.2	11.2	28.1
extraord.mpg	16.4	33.5	55.6	13.5	14.1	25.4	9.9	25.1
kiss.mpg	59.3	8.9	22.8	42.6	42.1	46.4	23.1	0.5
loubrock.mpg	62.7	6.4	16.7	44.6	45.5	45.0	24.1	0.7
neptune.mpg	16.1	29.0	38.1	13.5	10.2	12.5	12.3	26.5
boat.m2v	21.5	21.2	35.5	16.1	17.4	18.6	12.9	7.6
np.es.m2v	24.3	20.2	39.8	17.1	20.5	23.2	12.3	0.7
OVERALL	26.1	18.0	36.8	17.4	18.9	24.4	12.8	8.4

Table A.46: *This table shows the usefulness of redundant data in a simulation containing a mixture of MPEG-1 and MPEG-2 videos with 20 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	13.7	39.2	59.1	11.3	14.0	16.0	9.1	41.4
blossom.mpg	10.8	48.5	60.8	9.7	9.7	11.7	8.6	43.4
casinopr.mpg	19.6	32.8	50.9	15.1	15.4	15.8	12.7	5.2
Ext.lg.mpg	17.1	30.9	45.0	14.0	14.4	15.8	11.7	25.5
extraord.mpg	16.0	40.9	65.1	13.1	13.8	22.1	10.0	26.0
kiss.mpg	60.0	17.6	41.8	46.2	43.9	47.6	25.3	0.6
loubrock.mpg	61.3	14.5	35.6	48.7	47.1	50.4	24.9	0.5
neptune.mpg	16.0	40.1	52.2	13.9	11.3	14.1	12.3	25.0
boat.m2v	21.0	30.6	48.6	16.4	17.6	17.8	13.2	7.7
np.es.m2v	24.0	28.7	51.5	17.7	19.8	21.7	13.4	3.9
OVERALL	25.7	26.9	50.0	18.9	19.8	25.8	13.8	8.6

Table A.47: *This table shows the usefulness of redundant data in a simulation containing a mixture of MPEG-1 and MPEG-2 videos with 30 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	14.3	48.1	68.9	12.0	12.5	16.0	10.0	41.7
blossom.mpg	11.2	56.5	70.2	10.1	9.4	11.1	9.0	43.7
casinopr.mpg	19.4	41.2	61.7	15.5	15.2	16.6	13.0	6.8
Ext.lg.mpg	16.7	38.9	55.7	13.9	14.8	14.9	11.7	23.8
extraord.mpg	15.0	49.9	72.3	12.6	12.7	17.5	10.4	27.9
kiss.mpg	58.6	22.1	53.2	45.1	44.4	46.9	24.3	0.2
loubrock.mpg	61.6	18.5	47.4	48.8	48.5	51.2	24.0	1.2
neptune.mpg	15.0	49.2	64.0	13.3	11.8	14.6	11.5	22.1
boat.m2v	20.8	38.3	59.5	16.6	18.0	17.7	13.4	7.4
np.es.m2v	23.9	37.1	60.2	19.2	19.5	22.4	14.8	6.6
OVERALL	25.5	33.9	60.0	19.6	20.1	26.3	14.4	9.2

Table A.48: *This table shows the usefulness of redundant data in a simulation containing a mixture of MPEG-1 and MPEG-2 videos with 40 percent redundant data.*

Video Name	Normal Elements		Redundant Elements					
	% Lost	% Re-placed	Sent		Lost		Useful	
			% Sent	% Need	% Lost	% Need	% Usable	% Used
aspen3.mpg	13.3	56.1	76.2	11.4	11.8	14.3	9.7	54.7
blossom.mpg	11.2	63.7	77.3	10.3	9.6	11.2	9.2	51.0
casinopr.mpg	19.2	48.3	70.6	15.9	16.3	17.0	13.1	12.3
Ext.lg.mpg	17.0	47.2	65.3	14.8	15.5	16.3	12.3	26.3
extraord.mpg	14.3	56.2	78.8	12.1	12.2	15.2	10.2	29.4
kiss.mpg	58.1	26.8	66.8	44.7	46.8	45.7	23.3	0.8
loubrock.mpg	61.8	25.0	62.8	50.6	50.7	51.1	24.7	0.4
neptune.mpg	14.2	55.3	73.5	12.4	12.1	14.7	10.7	26.8
boat.m2v	21.2	46.4	68.8	17.8	18.6	19.2	14.3	7.9
np.es.m2v	23.4	44.5	68.1	19.8	19.9	22.4	15.3	10.1
OVERALL	25.3	40.8	69.3	20.6	21.1	27.4	14.9	11.0

Table A.49: *This table shows the usefulness of redundant data in a simulation containing a mixture of MPEG-1 and MPEG-2 videos with 50 percent redundant data.*

Appendix B

Slice Reference Graphs

B.1 References to I-Picture Slices

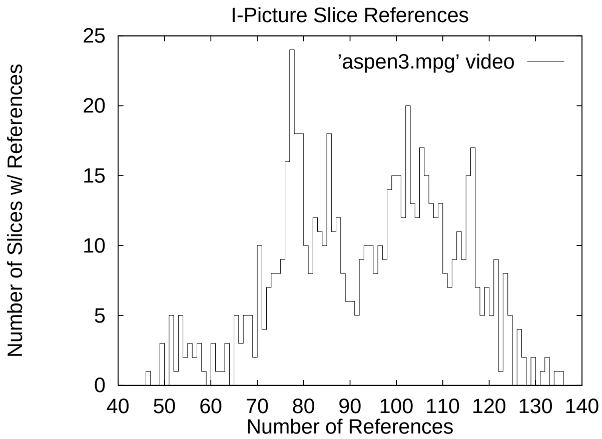

Figure B.1: *This figure shows the distribution of references to I-picture slices in the "aspen3.mpg" video.*

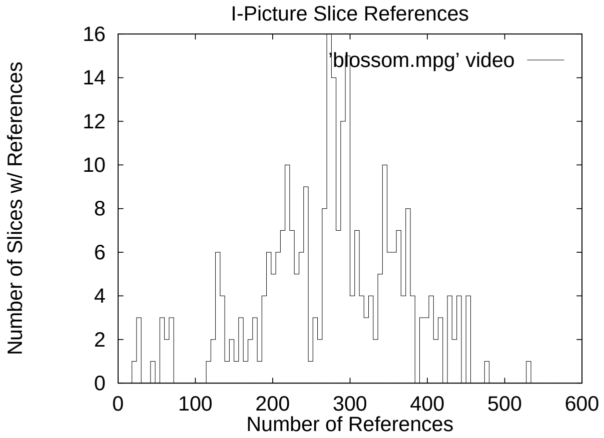

Figure B.2: *This figure shows the distribution of references to I-picture slices in the "blossom.mpg" video.*

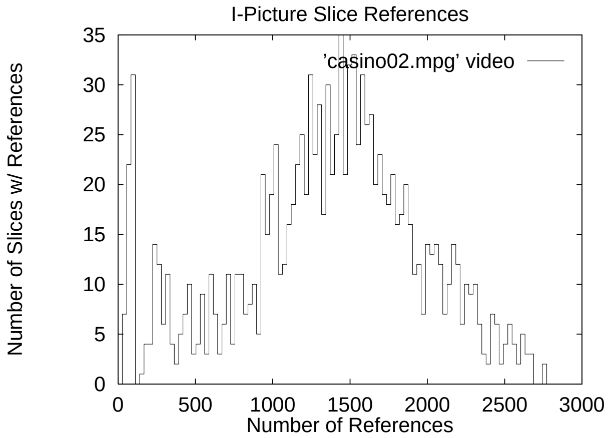

Figure B.3: *This figure shows the distribution of references to I-picture slices in the "casino02.mpg" video.*

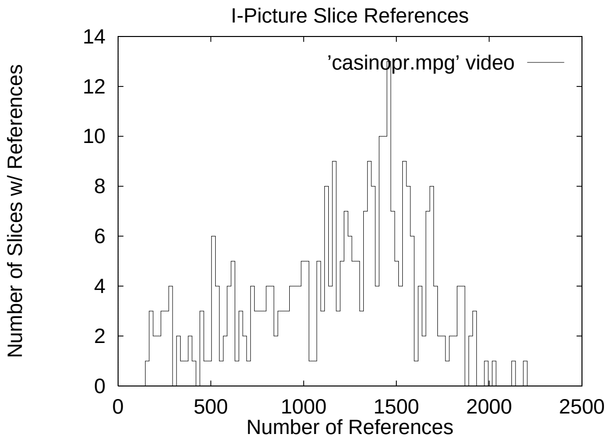

Figure B.4: *This figure shows the distribution of references to I-picture slices in the "casinopr.mpg" video.*

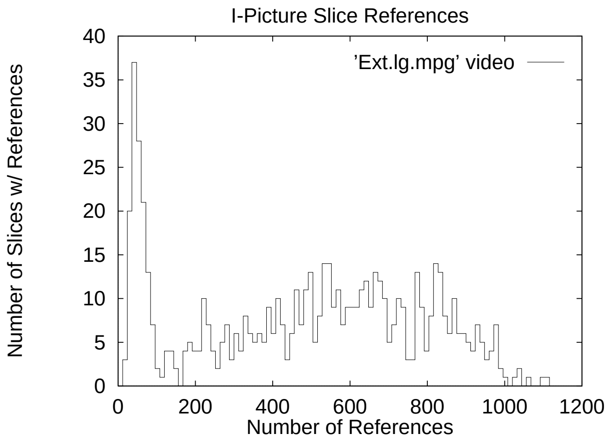

Figure B.5: *This figure shows the distribution of references to I-picture slices in the "Ext.lg.mpg" video.*

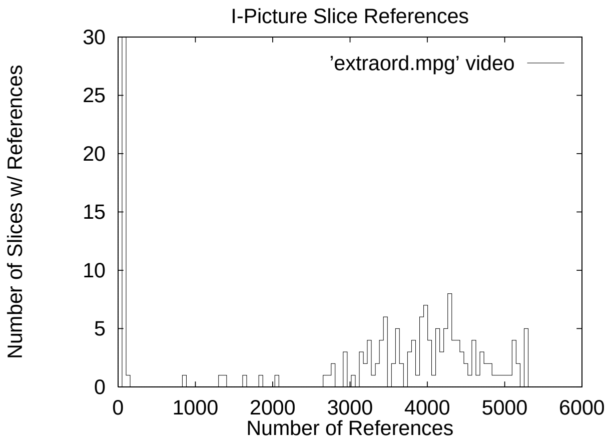

Figure B.6: *This figure shows the distribution of references to I-picture slices in the "extraord.mpg" video.*

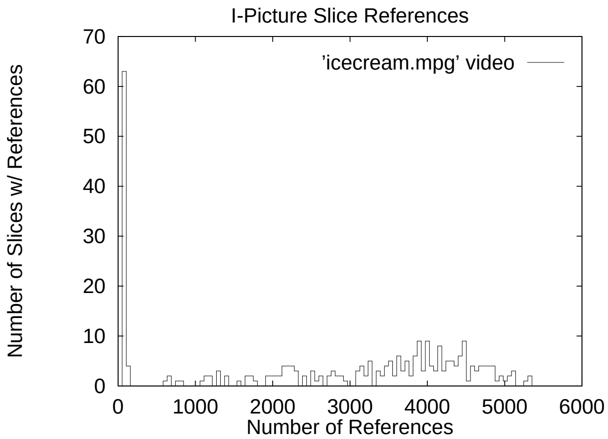

Figure B.7: *This figure shows the distribution of references to I-picture slices in the "icecream.mpg" video.*

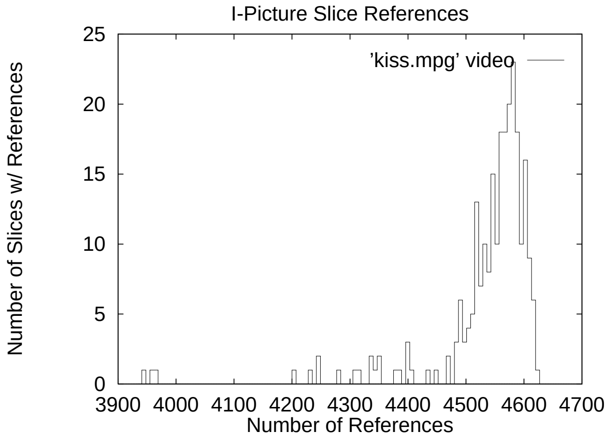

Figure B.8: *This figure shows the distribution of references to I-picture slices in the "kiss.mpg" video.*

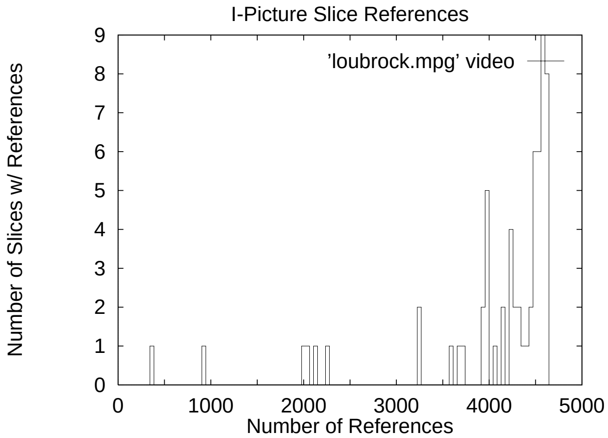

Figure B.9: *This figure shows the distribution of references to I-picture slices in the "loubrock.mpg" video.*

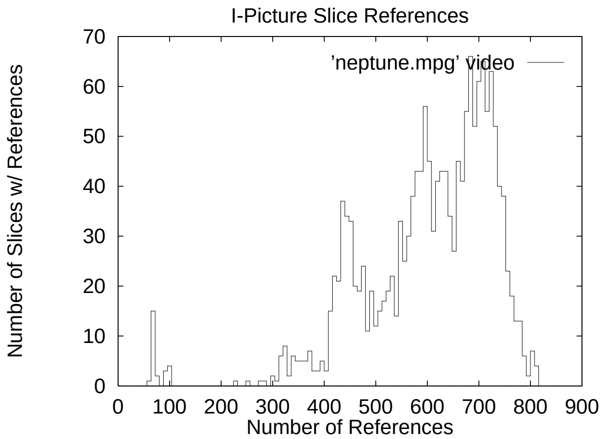

Figure B.10: *This figure shows the distribution of references to I-picture slices in the "neptune.mpg" video.*

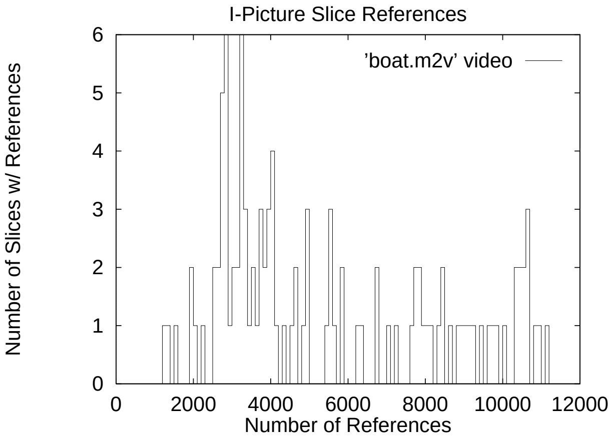

Figure B.11: *This figure shows the distribution of references to I-picture slices in the "boat.m2v" video.*

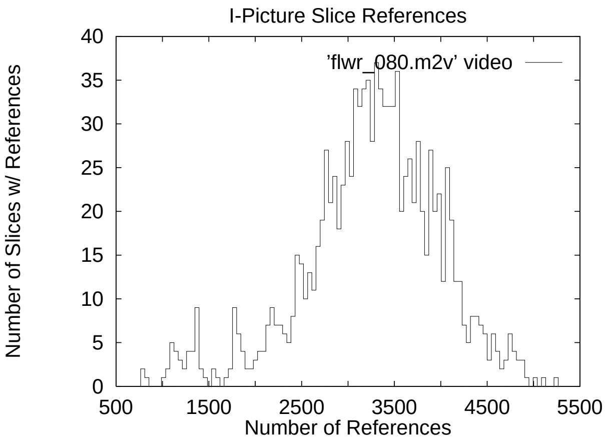

Figure B.12: *This figure shows the distribution of references to I-picture slices in the "flwr_080.m2v" video.*

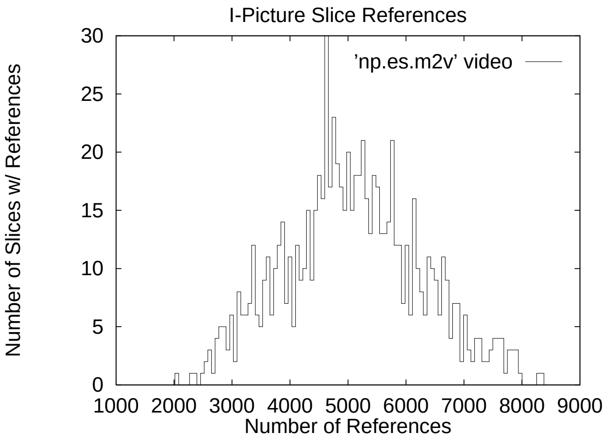

Figure B.13: *This figure shows the distribution of references to I-picture slices in the "np.es.m2v" video.*

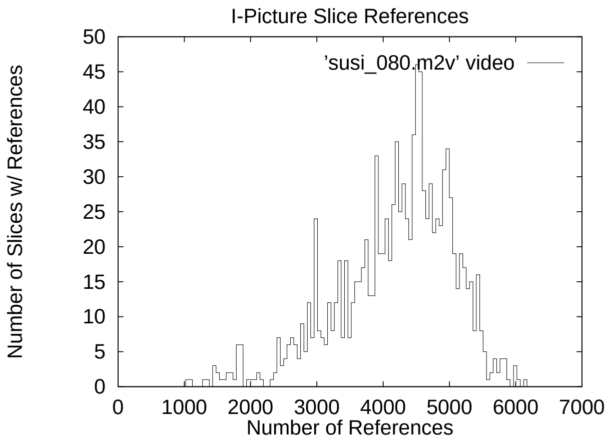

Figure B.14: *This figure shows the distribution of references to I-picture slices in the "susi_080.m2v" video.*

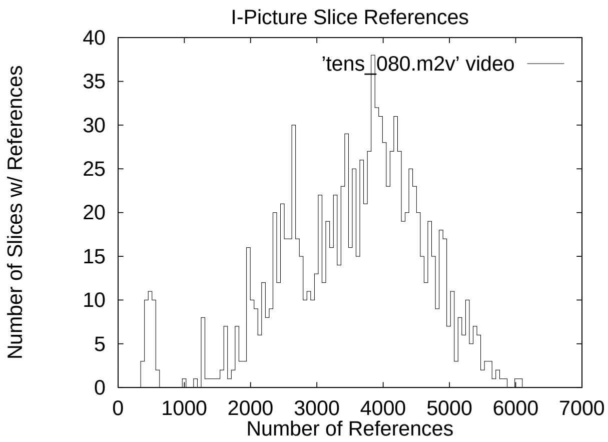

Figure B.15: *This figure shows the distribution of references to I-picture slices in the "tens_080.m2v" video.*

B.2 References to P-Picture Slices

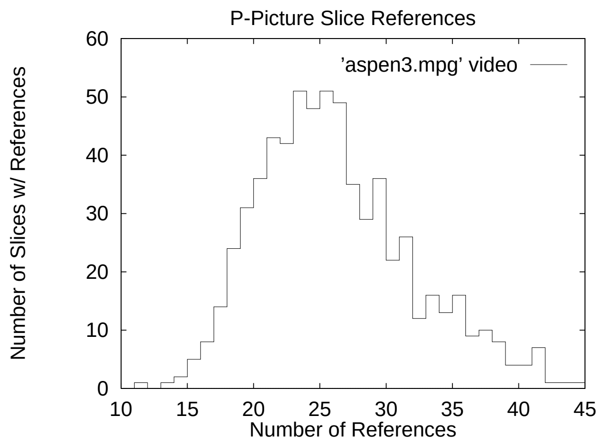

Figure B.16: *This figure shows the distribution of references to P-picture slices in the "aspen3.mpg" video.*

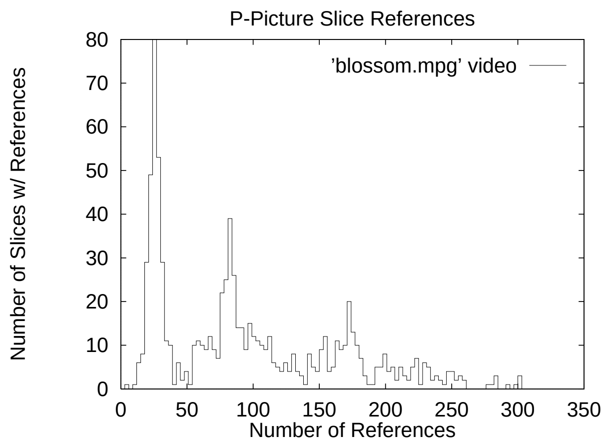

Figure B.17: *This figure shows the distribution of references to P-picture slices in the "blossom.mpg" video.*

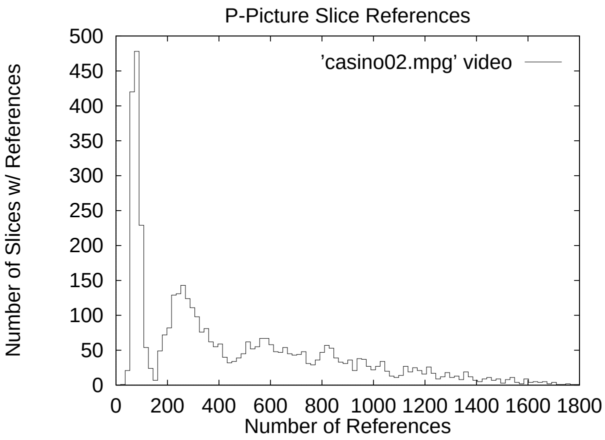

Figure B.18: *This figure shows the distribution of references to P-picture slices in the "casino02.mpg" video.*

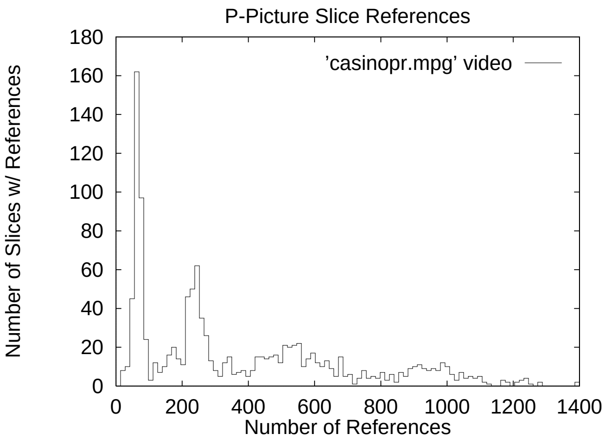

Figure B.19: *This figure shows the distribution of references to P-picture slices in the "casinopr.mpg" video.*

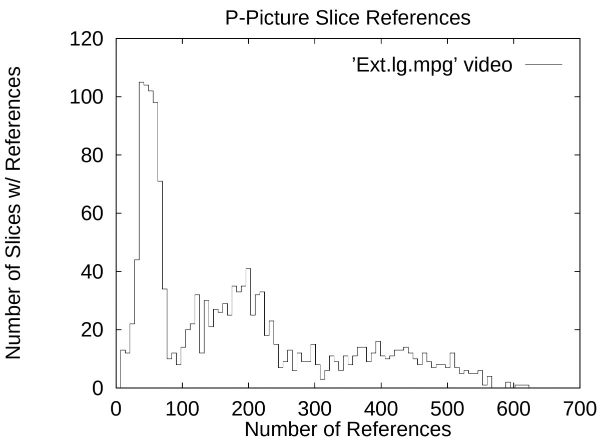

Figure B.20: *This figure shows the distribution of references to P-picture slices in the "Ext.lg.mpg" video.*

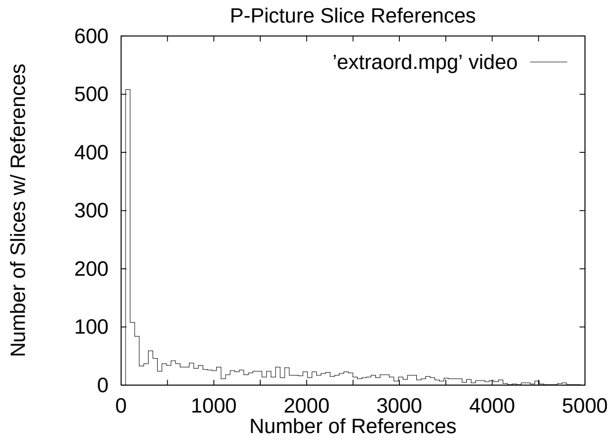

Figure B.21: *This figure shows the distribution of references to P-picture slices in the "extraord.mpg" video.*

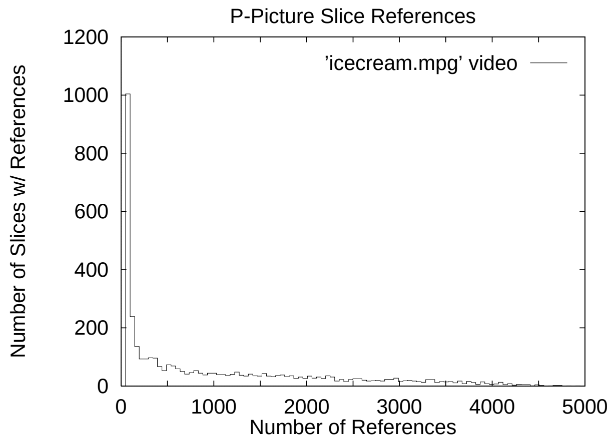

Figure B.22: *This figure shows the distribution of references to P-picture slices in the "icecream.mpg" video.*

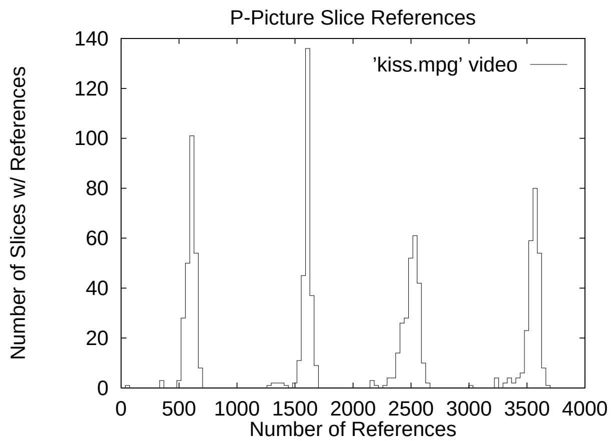

Figure B.23: *This figure shows the distribution of references to P-picture slices in the "kiss.mpg" video.*

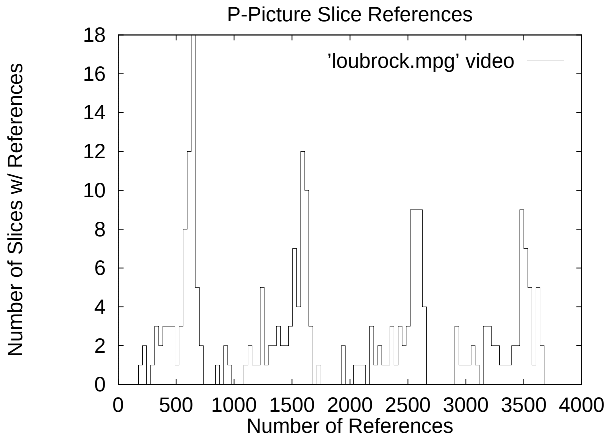

Figure B.24: *This figure shows the distribution of references to P-picture slices in the "loubrock.mpg" video.*

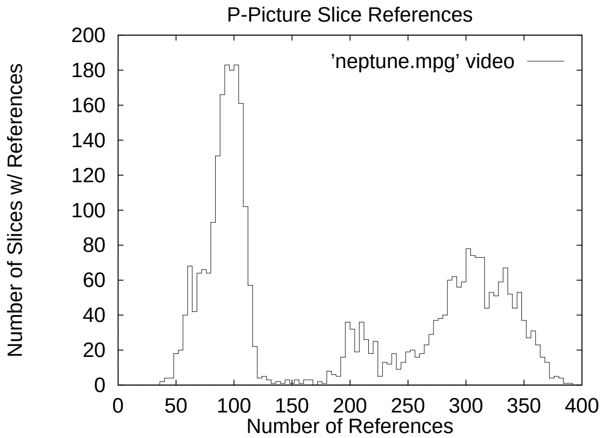

Figure B.25: *This figure shows the distribution of references to P-picture slices in the "neptune.mpg" video.*

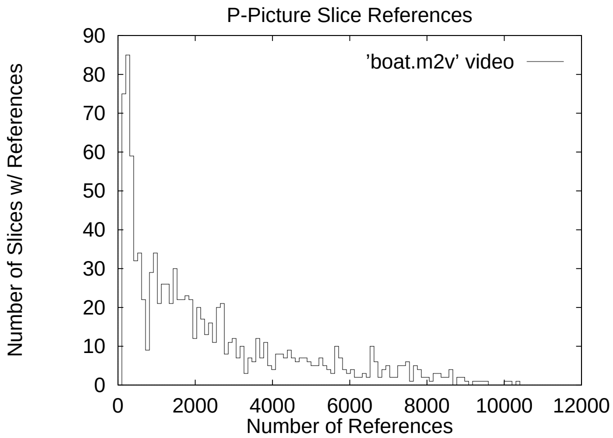

Figure B.26: *This figure shows the distribution of references to P-picture slices in the "boat.m2v" video.*

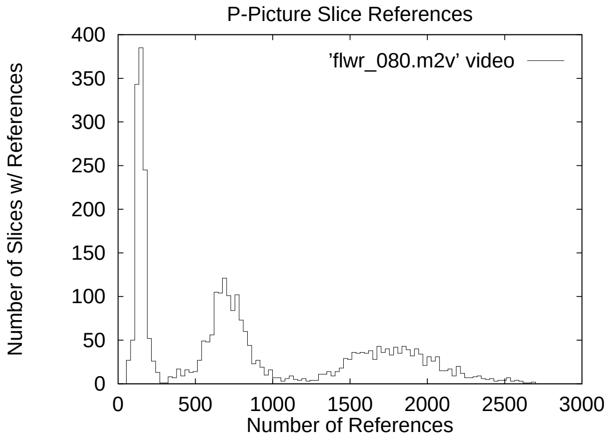

Figure B.27: *This figure shows the distribution of references to P-picture slices in the "flwr_080.m2v" video.*

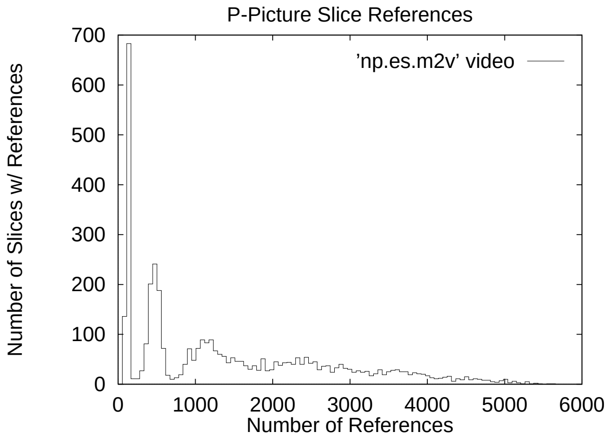

Figure B.28: *This figure shows the distribution of references to P-picture slices in the "np.es.m2v" video.*

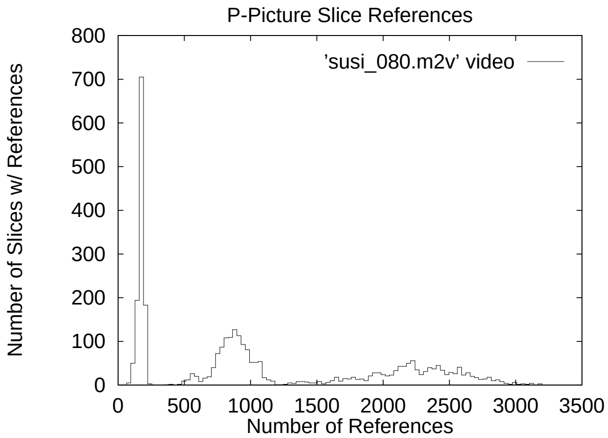

Figure B.29: *This figure shows the distribution of references to P-picture slices in the "susi_080.m2v" video.*

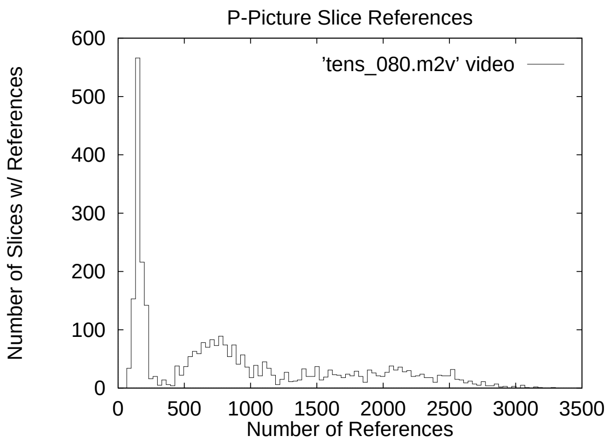

Figure B.30: *This figure shows the distribution of references to P-picture slices in the "tens_080.m2v" video.*

A Dependency-based Strategy for Handling ATM Cell Loss in MPEG-2 Transport Streams

Dean L. Scoville

Department of Computer Science

M.S. Degree, July 1996

ABSTRACT

Whenever data is transmitted over networks, congestion and other factors can cause the data to become lost or corrupted. Retransmission is impractical for many applications such as long-distance real-time video. This thesis proposes a solution to the data loss problem for data streams containing extensive internal dependencies. The proposed solution, referred to as the DART (Dependency-based Approach for Redundant Transmissions) system, assigns priorites to individual elements in the data streams and sends a redundant copy of the most critical elements. The effects of the DART system on MPEG video transmissions are presented here.

Actual MPEG video streams, along with redundant copies of their most critical data, were packaged into MPEG-2 systems transport streams and sent over a simulated ATM network. Results showed the quality of the received videos to increase with the added redundancy. The effectiveness (percent increase in quality per percent increase in bandwidth) of the transmissions was highest for networks with high error rates. Adding redundant data decreased the burstiness of MPEG traffic. Future work will further enhance the DART system and explore related error recovery methods.

COMMITTEE APPROVAL:

Mark J. Clement, Committee Chairman

Evan L. Ivie, Committee Member

R. Alan Whitehurst, Committee Member

Scott N. Woodfield, Graduate Coordinator