Metadata Driven Multimedia Transcoding

by

Mazen Almaoui

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical & Computer Engineering
University of Toronto

Mazen Almaoui

Master of Applied Science, 2005

Graduate Department of Electrical & Computer Engineering

University of Toronto

Abstract

Advances in multimedia device processing capabilities and the Internet have contributed to the proliferation of collaborative applications such as distance learning (DL) webcasting. The challenge that remains to be addressed in existing DL systems is providing access to multimedia content to any user regardless of device capabilities, IP network heterogeneity, and personal preferences such as modality. A novel metadata-driven adaptation architecture is presented to overcome these challenges by performing 1) application layer transcoding that adapts the presentation format of DL content to match device media decoding capabilities and user desired modality, 2) bitstream transcoding to adapt multimedia to match device processing capabilities and encoding bit-rate supported by the network. Results demonstrate how the proposed system delivers personalized DL content with negligible transcoding overhead while meeting end-user environmental restrictions.

Acknowledgements

There are many people that I would like to show my gratitude. But first and foremost, I would like to thank God for the strength and will to endure this challenge.

Thanks especially to Professor Plataniotis for believing in me and giving me the opportunity to prove myself. To Azadeh, who has been a great help and friend from the beginning to the end of my Masters. And, to the rest of the Multimedia Group, thank you for an interesting experience.

Many thanks to my family, each of whom contributed in their own ways. To my mother, thanks for the constant support and believing in me (since I was five!). To my brothers and sister, I am a very lucky guy to be surrounded by such great role models.

To my friends, who had to endure my constant complaining over the past two years, many thanks. To the Lebanese gang, your comradeship made my university experience one to remember. Last but not least, to my oldest friend John, thanks for your friendship over the years (sorry for cheating off of you in our fourth grade spelling test).

Contents

List of Figures

List of Tables

List of Abbreviations

CA	Content Analysis
DCT	Discrete Cosine Transform
DI	Digital Item
DIA	Digital Item Adaptation
DIDL	Digital Item Declaration Language
DL	Distance Learning
DOM	Document Object Model
EES	Electronic Educational System
EBCC	Equation-Based Congestion Control
FGS	Fine Granularity Scalability
FTP	File Transfer Protocol
GOP	Group of Pictures
HDTV	High Definition Television
HTTP	Hypertext Transfer Protocol
IDCT	Inverse Discrete Cosine Transform
IP	Internet Protocol
LAN	Local Area Network
LOM	Learning Object Metadata

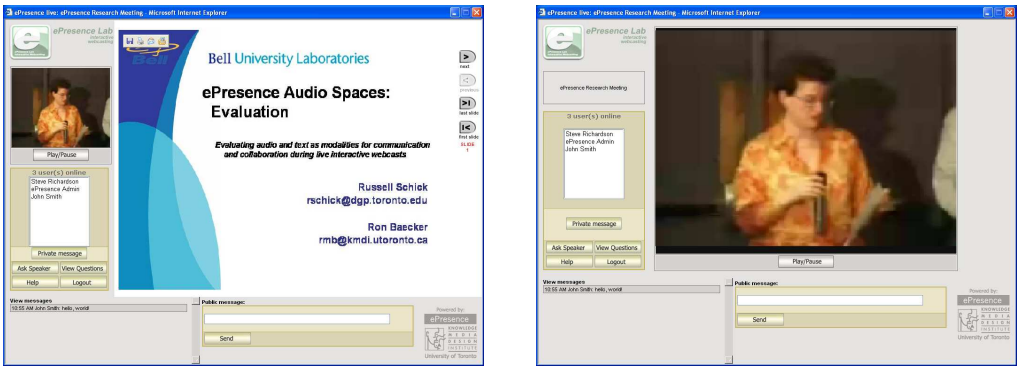
MAD	Motion Activity Descriptor
MPEG	Moving Pictures Experts Group
MV	Motion Vector
PC	Personal Computer
PF	Pre-filtering
PDA	Personal Digital Assistant
PSNR	Peak Signal-to-Noise Ratio
QoS	Quality of Service
RTSP	Real-time Streaming Protocol
RTCP	Real-time Control Protocol
RTP	Real-time Transfer Protocol
SNR	Signal-to-Noise Ratio
SQL	Structured Query Language
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
UED	Usage Environment Description
WAE	Weighted Absolute Error
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Transformation Language

Chapter 1

Introduction

The maturity of wired and wireless networks has added a new dimension to how people and groups collaborate. Furthermore, advances in multimedia content delivery as well as the increasing processing capability of mobile devices allow a variety of stationary and mobile users to enrich their collaborative experience. Among the plethora of collaboration applications conceived in the recent years is distance learning (DL) [1]. This refers to a learning paradigm that allows students and instructors to be in separate physical locations. Learning material such as video, audio, text chat, and slides are streamed between the "classroom" and end-users allowing them not only to attend remote lessons but to actively participate by communicating and interacting over the Internet. As an example, Figure 1.1 shows a screen shot of a DL webcasting system, ePresence [2].

This thesis proposes a *DL content adaptation* framework for the delivery of DL multimedia to any user despite their device processing constraint or network conditions.



(a) Video, audio, slides, & text chat (b) Video, audio, and text chat only

Figure 1.1: ePresence with different modalities depending on user preference.

1.1 Need for DL Content Adaptation

The ubiquity of wired and wireless multimedia processing devices means that DL content can be accessed by various users with unique environment conditions such as device capabilities, network conditions, and user preferences. In light of this, there is a need for a DL content adaptation framework [3] that can provide seamless access to multimedia to all such users by transfoming content to match their unique environmental conditions [4].

Conventional DL applications currently in use, such as ePresence, deliver the same content to all users without regard for user environment conditions. In other words, the burden is put on the user of the system to resolve device or network shortcomings. This thesis proposes an adaptation framework for real-time transformation of DL content to match each user's device capabilities, network conditions, and presentation preferences. In this work, adaptation refers to transforming the media streams in terms of parameters such as frame-rate, resolution, and interface presentation to accommodate for capabilities of the end-user device.

Description Tool	Parameters
Device Capabilities	codec supported, resolution, frame-rate, color depth, frequency range, number of audio channels
Network Characteristics	available bandwidth (bit-rate)
User Preference	modality (combination of video, audio, images, text and audio chat)

Table 1.1: User environment parameters used for content adaptation.

1.2 Challenges to Achieving DL Content Adaptation

In order to achieve DL content adaptation, three challenges imposed by environmental conditions are considered. These are unique device capabilities, heterogenous networks, and user defined Quality of Service (QoS). The rest of this section examines each of these challenges in detail.

The first challenge in multimedia delivery to end-users is the diversity in the processing capabilities of devices in terms of the parameters shown in Table 1.1. The diversity in device capabilities requires media content to be adapted (scaled and transformed) and media streams that cannot be processed by the device to be dropped. Whether for live or archived sessions, this must be performed in real-time to avoid significant adaptation delays in delivering DL content to the end-user. This real-time nature of a DL application is crucial in providing the user with an experience similar to that of attending the actual live event [5].

The second challenge is the delivery of multimedia over a multitude of heterogeneous and unreliable wired and wireless networks. Clearly, diverse networks exhibit unique behaviour in terms of bandwidth capacity and fluctuations due to factors such as packet loss, delay, and jitter [6]. These factors require varying considerations on media encoding and delivery from the content server to the end user. Specifically, the diversity in bandwidth characteristics

of the underlying network necessitates multimedia adaptation to provide variable bit-rate media encoding. In addition to challenges posed by bandwidth fluctuations, networks with a unicast [7] flow architecture may result in bandwidth capacity constraints on the DL content server. Unlike multicast [7] networks that are capable of setting up one-to-many connections, unicast networks impose a point-to-point connection with every client. This can deplete server-side network resources due to high bandwidth demands of rich media like video. The disadvantage of a multicast network is the lack of an ability to deliver personalized content to each user. A network topology must exist to ensure that a significant number of end-users can simultaneously conduct personalized streaming sessions with the server without depleting the available bandwidth.

The third consideration in multimedia access and delivery is the quality of the end-user experience [8]. Users must be allowed to negotiate a QoS based on their preferred multimedia modality. For example, one user may only demand audio and slides of a lecture with no video, while another may want all media to be delivered. This must be negotiated, not granted, since not all devices are capable of processing all types of media. For example, video should not be delivered to a user's device that is not capable of processing video.

The three aforementioned environmental challenges can be addressed in the framework of *metadata-driven transcoding* for the adaptation DL content to accommodate unique environmental conditions. An overview of the proposed solution is shown in Figure 1.2.

1.3 Metadata-driven Transcoding for DL

In order to match unique constraints imposed by each user, multimedia content must be adapted through a process known as *transcoding*. Multimedia transcoding is the conversion of one media file format to another or the re-encoding of the original media file with new encoding parameters (such as those listed in Table 1.1).

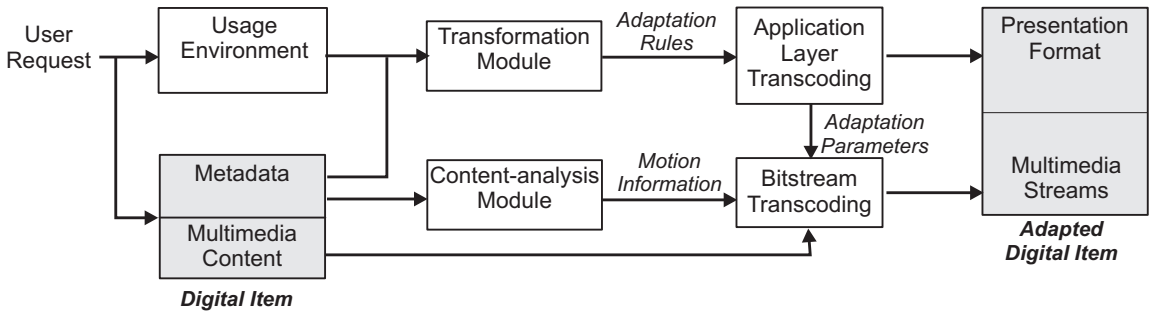


Figure 1.2: Proposed metadata-driven transcoding framework for a DL application.

We consider two types of transcoding for adaptation of DL content namely, application layer transcoding and multimedia bitstream transcoding. These are described in what follows.

Application layer transcoding is a higher level transcoding approach that adapts the media to match device processing capabilities and user preferences [4]. Note that this type of transcoding does not operate on the actual media bitstream. Instead, it determines the presentation format by producing a browser template, such as that in Figure 1.1, based on device capabilities and user desired modality. Furthermore, the application layer transcoding passes adaptation parameters/rules specifying how the underlying multimedia streams should be transformed to the bitstream transcoder.

Bitstream transcoding [9] is performed on the actual media to parse, transform, and truncate the underlying multimedia streams to conform to the device's processing capabilities and to adapt the encoding bit-rate to meet the network's fluctuating bandwidth capacity.

Both transcoding approaches require descriptions of varying device capabilities, network conditions, and user preferences. In this regard, metadata is a powerful tool that can be utilized for describing, indexing, and searching for the properties of diverse devices, networks, user preferences, and the properties of the actual media for the purpose of transcoding [4]. The metadata-driven transcoding framework [4] uses metadata descriptions for application

layer transcoding and providing adaptation parameters to perform bitstream transcoding.

Convention multimedia transcoding is performed at the physical layer. In order to perform metadata-driven transcoding, a framework is needed that associates multimedia streams, metadata describing the streams, and metadata describing the end-user environment. The framework of the MPEG-21 Standard [3] serves as an enabling technology for metadata-driven transcoding. The MPEG-21 Standard packages the actual multimedia and metadata describing the properties of the encapsulated media into a architectural unit known as a Digital Item (DI) [10]. Transcoding is then performed at the DI level and is known as Digital Item Adaptation (DIA) [11]. As shown in Figure 1.2, DI metadata describing the packaged multimedia and user environment metadata are used to produce adaptation rules to transcode DL content to meet end-user constraints. The application layer transcoder utilizes the adaptation rules to produce a presentation template by adapting the original template and adaptation parameters for the bitstream transcoder that conform to the end-user's environmental needs.

1.4 Overview of Proposed Solution

The overview of the proposed solution is depicted in Figure 1.2. The input to the system is a digital item that contains the requested media as well as metadata describing the media content, and characteristics such as encoding rate. In addition, a metadata description of the user environment metadata as shown in Table 1.1 is used in the transcoding process. Throughout the transcoding framework, this information is then used to produced an *adapted digital item* appropriate for delivery and presentation to the given user.

As shown in Figure 1.2, the environment description is used by a *transformation module* to generate a set of adaptation rules. The rules are then used by the application layer transcoder to produce the appropriate presentation template (e.g. HTML, JSP, etc.) for the

media. This transcoder also produces adaptation parameters, such as the frame-rate and resolution, for the bitstream transcoder. The bitstream transcoder uses this information together with analysis of the media content (e.g. motion activity) to transform the actual bitstream. Finally, the system outputs the adapted digital item containing the presentation format and media streams.

1.5 Thesis Contributions

This thesis addresses the problem of adapting DL content to match end-user environmental constraints . Specifically, a metadata-driven transcoding framework is proposed to deliver DL media content to users with diverse device capabilities, heterogeneous network conditions, and presentation preferences. The novel contributions of this work are explained in the following sections.

1.5.1 System Architecture

This work proposes a novel system architecture for achieving metadata-driven transcoding for adapting DL content. The system incorporates a real-time metadata-driven transcoding solution that uses various industry standard tools, such as MPEG-21 [12], MPEG-7 [13], and MPEG-4 [14] standards to provide seamless delivery of DL content to users with mobile devices. In particular, the flexibility of the proposed architecture allows DL designers to easily incorporate our proposed application layer transcoding solution with virtually any existing bitstream transcoders to provide a metadata-driven adaptation framework [15].

1.5.2 Application Layer Transcoding

A novel transformation module is proposed that produces adaptation rules for the application layer transcoder to meet the end-user environmental constraints. This component

incorporates a cache-based solution to significantly speedup the process of adaptation rule generation [16]. Existing metadata transformation solutions produce adaptation rules on-the-fly, which is computationally expensive. The proposed solution allows for real-time metadata transformation which is essential in a DL application.

1.5.3 Multimedia Bitstream Transcoding

As previously mentioned, conventional multimedia bitstream transcoding is performed at the physical layer. A novel bitstream transcoding framework is proposed that utilizes "adaptation hints" in the form of metadata from the application layer to provide enhanced transcoding. In particular, the underlying streams are transcoded taking into account end-user environment constraints and properties of the multimedia streams. Metadata is utilized to improve transcoding performance to ensure real-time encoding and to provide enhanced multimedia scalability by taking into account properties of the underlying streams such as motion information.

1.6 Thesis Outline

The rest of this thesis is organized as follows. A brief background on enabling technologies needed to achieve metadata-driven multimedia transcoding is discussed in Chapter 2. In particular, this chapter reviews the MPEG-21 Standard metadata-driven transcoding tools such as the DI for associating multimedia streams with their respective metadata descriptions and user environment description tools. We also discuss metadata description and transformations tools, the MPEG-4 multimedia bitstream encoding standard, and finally the MPEG-7 multimedia description interface that is utilized to extract useful metadata such as motion analysis information. Chapter 3 provides an overview of the shortcomings in the existing DL applications in delivering personalized DL content to meet the end-user con-

straints. This chapter also reviews prior work in metadata-driven transcoding and explain their limitations in producing adaptation rules in real-time (a characteristic which is essential for a DL application). The proposed method is presented in Chapters 4 and 5. Chapter 4 discusses the proposed application layer transcoding solution producing adaptation rules, a presentation template, and adaptation parameters that meet the end-user environmental constraints. Chapter 5 explains the bitstream transcoding framework and its utilization of the adaptation parameters from the application layer to provide enhanced multimedia adaptation. Experiments and results of the proposed transcoding approach are presented and discussed in Chapter 6. In particular, the results demonstrate how the proposed system produces personalized DL content that takes into account the end-user environmental constraints. Chapter 7 concludes this work and provides directions for future work. Lastly, Appendices A, B, and C, provide supporting information on the MPEG-21, MPEG-4, and MPEG-7 standard tools, respectively.

Chapter 2

Background

This chapter will present the enabling technologies and tools used in the proposed system architecture to achieve metadata-driven transcoding for varies applications, including DL. First, two approaches to multimedia adaptation, scalable coding and transcoding, are presented. After establishing motivation for the latter, the framework of metadata-driven transcoding and its constituent components are discussed. Specifically, background material will be provided on the Extensible Markup Language (XML) [17], which will be used to represent metadata. Furthermore, the Extensible Stylesheet Transformation Language (XSLT) [18] will be presented as a tool to transform the metadata to formats needed by the application layer and bitstream transcoders. Next, an overview of the Moving Pictures Experts Group [19] (MPEG) Standard will be given. In particular, the MPEG-21 Content Delivery Framework [12], MPEG-4 Video Coding [20], and the MPEG-7 Content Description Interface [21] Standards.

2.1 Scalable Coding vs. Transcoding

Multimedia adaptation is required to convert one media file format to another or to re-encode the original media file with new encoding parameters (Table 1.1. This is required to meet

the end-users preferences as well as device and network constraints. There are two main approaches to multimedia adaptation namely, scalable coding and transcoding.

The widely used approach of multimedia adaptation, known as offline scalable coding, is to encode and store various scaled versions of the same media content [3] to be delivered as required. Storing variations of the same media on the content server incurs a high storage cost. Also, a scaled media version can be streamed that meets the current network conditions, however, there remains the problem of adapting to varying network conditions like bandwidth fluctuations.

The second approach to adaptation is transcoding [22]. Transcoding only requires one version of the media to be stored on the content server; media is transformed in real-time to adapt to unique end-user environmental conditions such as device capabilities, fluctuating bandwidth, and user preference. Due to real-time constraints of a DL application as well as diversity in device capabilities and network conditions, this thesis uses a transcoding approach. Furthermore, a metadata transcoding solution is proposed where descriptions of the usage environment, such as device, are specified as metadata. This approach is discussed in the following section.

2.2 Metadata Description & Transformation Tools

In metadata-driven transcoding, the usage environment including user device capabilities, network conditions, and preferences are described as metadata. The prefix meta, meaning beyond, signifies information about the data that is more than just the media content. For example, metadata for a video clip may contain the creator and subject matter. The Extensible Markup Language (XML) is a popular language used for representing metadata. The particulars of this language and its advantages are discussed next.

2.2.1 Extensible Markup Language

The main functionality of XML is to provide structure and give context to data [17]. The simplest way to explain XML is by way of an example. Figure 2.1(a) shows a well structured, easy to read XML document that contains metadata pertinent to a student's personal information. As Figure 2.1(a) demonstrates, XML contains well defined open and close tags (similar to HTML) for every field, which makes it easy to search and parse by other applications. Hence metadata will be available for publishing, however, will need to be transformed into another presentation format like HTML in order to be displayed or used by another application. Figure 2.1(b) shows how the XML metadata containing student information can be utilized to create HTML code representing the personal web page of a student. Hence, an application such as a Internet browser can be utilized to display the HTML code as shown in Figure 2.1(c). The aforementioned transformation (accomplished using XSLT) will be explained in the next section. The following points highlight the advantages of using XML:

- Provides a well structured document that separates the data from its representation.
- Easily exchanged between vendors and utilized by different applications without the use of proprietor hardware or software.
- Text based which allows users to write and modify XML documents with simple text editors.

Details of XML syntax is outside the scope of this thesis, please refer to [17]. The next section will explain how XML metadata is transformed via XSLT to produce a representation format like HTML.

In metadata-driven transcoding, the XML metadata is processed and transformed to adapt the actual media content. The transformation rules are specified using the Extensible Stylesheet Transformation Language.

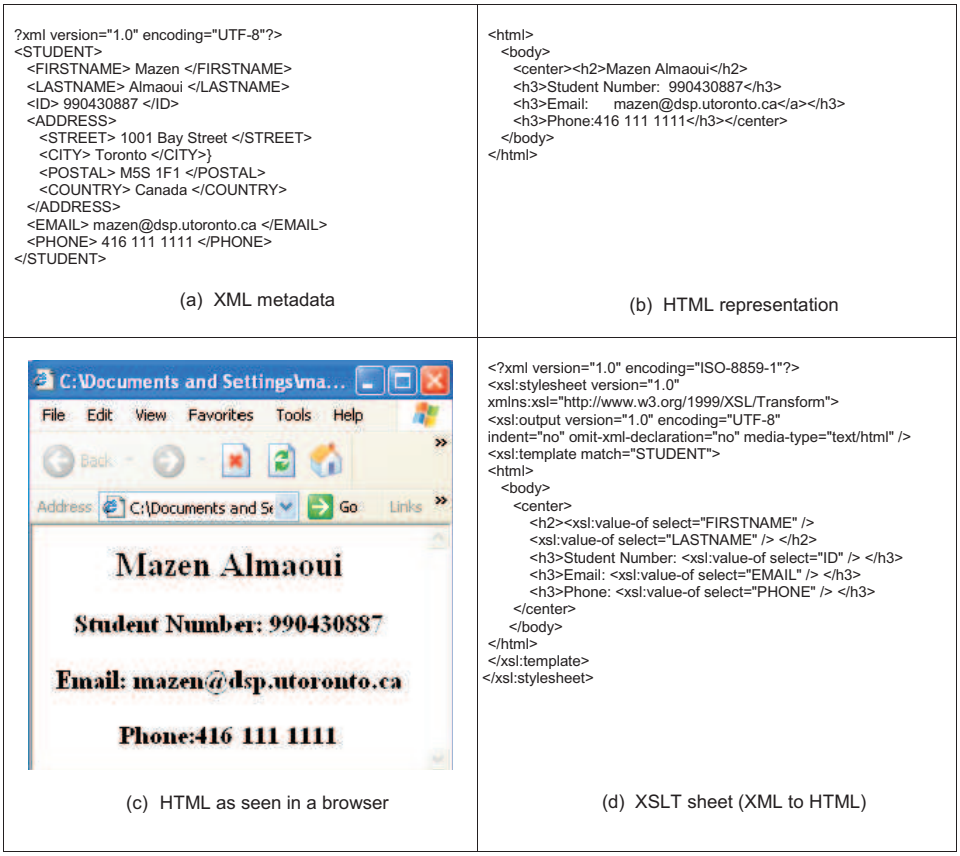


Figure 2.1: XML document transformed to HTML via XSLT sheet

2.2.2 Extensible Stylesheet Transformation Language

An XML document is essentially a text document; it cannot do anything by itself. Usually a transformation engine/parser that utilizes Extensible Stylesheet Transformation Language (XSLT) [18] is needed to interface between XML metadata and the application that requires the use of this metadata. In particular, XSLT is a tool for transforming and parsing the structure of an XML document into an alternative format such as HTML, text, or another XML document [18]. The following features make XSLT an attractive tool for transforming XML documents:

- XSLT uses XML syntax; everyone who is familiar with XML syntax does not have the

additional hassle of learning a new language/syntax.

- XSLT is a declarative language without side effects.

- XSLT is a powerful rule-based meta-language that allows writing a set of independent pattern matching rules. In other words, XSLT enables navigation through stylesheets or XML documents to extract and manipulate specific data. This behavior mimics Structured Database Query Languages (SQL).

- XSLT can serve as input or output; XSLT stylesheets can read and write other XSLT stylesheets.

It can be argued that programming languages such as Java or C++ can be used to perform the XML transformation process. However, XSLT is the preferred method because it was designed to describe the required transformations by using a set of independent pattern-matching rules rather than specifying a sequence of procedural instructions as would be the case with Java and C++ [18]. This set of independent pattern-matching rules make XSLT highly extensible for inserting new rules as required. Figure 2.1(d) shows the XSLT sheet responsible of transforming the XML metadata into HTML code that can be represented in a browser. As shown, the sheet specifies the output will be a HTML document (*media-type="text/html"*). The sheet searches for the root tag "STUDENT" (*match="STUDENT"*), parses the consequent tags and extracts the data associated with those tags (*value-of select="ID"*), and applies the required HTML formatting which produces the browser representation. This is a very simple example given in order to demonstrate the functionality of an XSLT sheet and its interaction with an XML document. The next section will introduce the MPEG-21 Standard Universal Multimedia Access Framework and explain how it utilizes XML and XSLT as enabling tools.

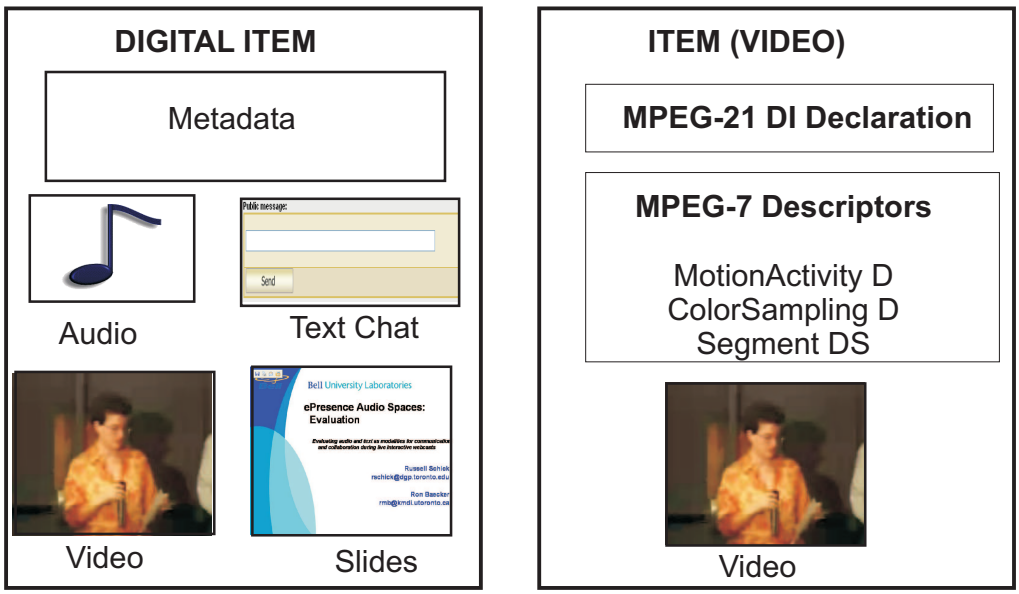
2.3 The MPEG-21 Standard

The MPEG-21 standard defines a framework for the transparent delivery and consumption of multimedia by all entities in the system chain [12]. An entity may refer to a client device receiving the multimedia or the content server that is delivering or processing the associated multimedia. The MPEG-21 Standard aims to replace the multitude of standard formats offering the same services (such as video codec standards) by a single access and storage framework that can be processed by all applications [12]. This unification of standards will not be finalized until 2010, however the MPEG-21 committee has already finalize parts of the standard [12] that can be utilized immediately for the purposes metadata-driven transcoding. The subsections below will provide detail regarding how MPEG-21 normalized tools will be utilized in the proposed transcoding framework.

2.3.1 Digital Item

The MPEG-21 Standard describes multimedia in terms of an elementary architectural unit called a Digital Item (DI) [3]. A DI packages a combination of media such as video, audio, images, text, and the XML metadata description of its contents. The DI can be thought of as the representation of the encapsulated multimedia that is manipulated, distributed, and transmitted throughout the delivery chain. Figure 2.2(a) shows a DI that represents the multimedia associated with the ePresence system shown in Figure 1.1. The Digital Item Declaration Language (DIDL) [23] provides a flexible and standardize language for declaring and creating a DI. The DIDL is flexible because it allows the DI to have any type (and number) of media. For example, a DI representing an ePresence lecture can be composed of only video and audio, or can be composed of slides and audio. The structured approach of the DIDL model allows for a standardized framework that can be parsed by any entity with DI processing capabilities, even without having knowledge of its contents. Appendix A

shows the metadata pertaining to the ePresence DI shown in Figure 2.2(a) and give details of the contents of the DI. The details of the syntax and the meaning of the metadata tags can be found in [10]. Transcoding within the MPEG-21 framework is not approached as adapting individual streams or media type. Instead, transcoding is performed on a DI level; this is referred to as Digital Item Adaptation (DIA) [3].



(a) Entire DI (b) Video Item

Figure 2.2: DI for an ePresence Lecture: (a) All media and (MPEG-21, MPEG-7) metadata pertaining to the DI (b) Close up of the video item and MPEG-7 metadata descriptors used

2.3.2 Digital Item Adaptation

The process of adapting a DI can be considered synonymous to the concept of metadata-driven transcoding; metadata describing the contents of the DI is transformed to provide transcoding parameters in order to transcode multimedia bitstreams at both the application and bitstream layers. Figure 2.3 shows a high level overview of the DIA engine used in this

thesis. Metadata pertaining to the DI is transformed by the application layer transcoder using XSLT sheets to produce a presentation template and adaptation parameters that is utilized by the bitstream transcoding engine. The presentation template produced by the application layer transcoder and the adapted multimedia streams produced by the bitstream transcoder represent the adapted DI. By performing transcoding on the DI level , the rich metadata embedded in the DI can be exploited to provide adaptation "hints" and to provide a relationship among the encapsulated media in the context of the intended application. The adaptation hints can include browser layout rules or content analysis parameters [24] to provide higher quality transcoding.

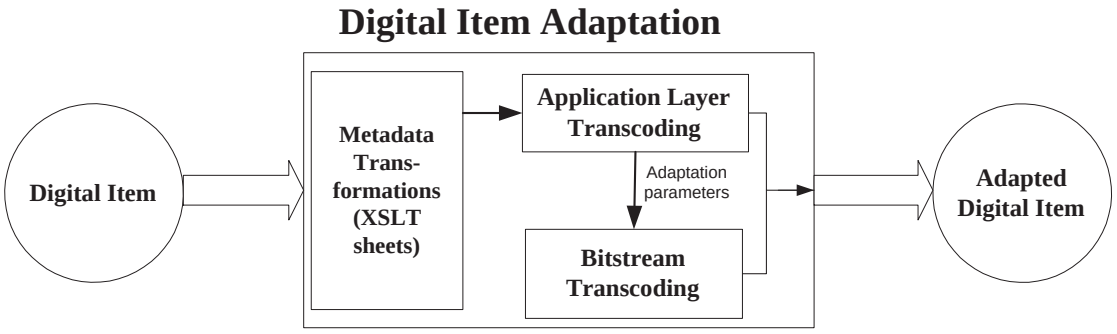


Figure 2.3: Proposed DIA engine architecture

To date, the DIA methodology proposed by the MPEG-21 Standard is still non normative. However, as mentioned above, there are normative in the DIA part of the standard that is utilized in this thesis, namely the Usage Environment Description (UED) Tools [11] and the DIDL.

2.3.3 Usage Environment Description Tools

The (UED) tools are used to index and store metadata that describes device capabilities, network conditions, and user characteristics as well as the natural environment characteristics. The UED tools provide a standardized description tool that contains all the information

needed about the user environment for the purpose of adaptation. For example, the DisplayCapabilities schema shown in Section A.1 of Appendix A describes a subset of a device properties which can be utilized to deduce the end-users device capabilities for processing multimedia. These standardized description tools allow for the use of metadata by the DIA engine or external entities within the system delivery pipeline that interface with the adaptation engine that might required information about the user's environment. Section A.1 of Appendix A lists all the UED descriptors used by the proposed DIA engine to represent the end-users environment. It should be noted that MPEG-21 purposes a larger set of parameters [25] for applications that require alternative information about the user's environment.

It is important for a DL framework to have a method of producing metadata to describe the user's environment in order to implement a metadata-driven transcoding. The MPEG-21 Standard does not specify how to obtain information about the user's environment. There are automated methods of obtaining user environment information like Session Description Protocol [26] to obtain device capabilities and Real Time Control Protocol [27] to obtain information about the user's network conditions. However, these protocols are outside the scope of this thesis and will not be discussed further.

2.4 Application Layer Transcoding

As previously mentioned a metadata-driven transcoding solution uses the DI metadata for DIA. The adaptation is done at two levels: application layer and bitstream layer.

An application layer transcoder takes in a set of adaptation rules, specified in XSLT sheet, and transforms the DI metadata (see Figure 2.1) to generate a presentation format for the media and provide parameters to the bitstream encoder. In the specific example of Figure 2.1, an XSLT processor uses the sheet in part (d), to generate the HTML document displaying the user information. Note that in DIA, the presentation format must also allow for inclusion

of multimedia (such as video) that will be displayed through a web-browser. Furthermore, the application layer transcoding passes adaptation parameters/rules specifying how the underlying multimedia streams should be transformed to the bitstream transcoder. The next section presents the particulars of bitstream level transcoding.

2.5 Bitstream Transcoding

The bitstream transcoder uses the parameters from the application layer transcoder to actually transform the media bitstream. At the heart of this component is a multimedia encoding standard such as MPEG-4 which is presented in the next section.

2.5.1 The MPEG-4 Standard

The MPEG-4 Standard [20] has emerged as a multimedia encoding industry standard of high quality content with low bit-rates. Of particular interest to this thesis are the scalable video encoding tools that the standard offers. These scalable tools include spatial scaling to encode at different resolutions and temporal scaling to encode at different frame-rates. The third type of scalability is Signal-to-Noise (SNR) scalability, namely, Fine Granularity Scalability (FGS). FGS varies the bit-rate by encoding video with a low bit-rate base layer with the ability to add enhancement layers to produce higher quality to produce higher a bit-rate. However, not too many mobile devices are capable of decoding streams encoded with FGS, hence FGS will not be incorporated into the proposed solution.

In [14], a media adaptation scheme is proposed that is based on MPEG-4 object based scaling. Audio-visual objects in media streams are individuality analyzed and streamed to meet device processing capabilities and available bandwidth. This method suffers from extremely high computation overhead and does not address the need for user media personalization. This overhead becomes a significant bottleneck once the number of objects is large

for the currently streamed media.

Appendix B contains a detailed explanation of the MPEG-4 Video Coding Standard. The next section will explain how the MPEG-7 Standard can be utilized as an enabling tool for content analysis.

2.6 The MPEG-7 Standard

While other MPEG Standards such as MPEG-1, MPEG-2, and MPEG-4 [20] specify how to encode audio/visual streams, the objective of the MPEG-7 Standard [13] is to serve as a tool for describing properties of audio/visual content. In MPEG-7 Multimedia Content Description Interface [21] plays an important role in the next generation of multimedia transcoding applications. It provides standard media descriptors that can be used to describe media properties associated with video, audio, and images. The aforementioned descriptors can represent the following properties into XML metadata:

- Video: motion intensity, key frames, color features of a GOP, GOP segment information
- Audio: spectral, parametric, temporal features of an audio signal
- Image: color, texture, and shape information

The above mentioned properties/features that are represented by the descriptors are only a subset of what the MPEG-7 Standard offers. The descriptors serve as content analysis tools that can be utilized to provide "adaptation hints" during transcoding for the purpose of content analysis. Descriptor metadata that represent properties of a media can be also be embedded into a DI as shown in Figure 2.2(b). This figure is a detailed look at the video item and its associated metadata. In this example the MotionActivity, ColorSampling, and Segment descriptors accompany the encapsulated video.

2.6.1 Motion Activity Descriptor

A human perceives segments of a video as slow, medium, and fast action segments. For example, watching a formal lecture is perceived as a slow clip, but the video clip of a lab demonstration can be perceived as a fast clip. In other words, a video entails of sequences of high and low activity. The Motion Activity Descriptor (MAD) [21] extracts and expresses the activity of these video segments. It can be used in video transcoding to make intelligent frame dropping decisions when the allowable bit-rate encoding rate drops due to a drop in network bandwidth. For example, if there are two video segments A (with high motion intensity) and B (with low motion intensity), it would be ideal if the transcoder is provided with this knowledge in order to drop more frames from segment B. The MAD extracts and encodes the intensity of activity as an integer lying in the range of 1-5. A value of 5 indicates a high level of activity and the value of 1 indicates the lowest level of activity. The intensity of motion is a subjective measure, hence ground truth [28] was used as a fidelity measure for the MAD intensity measure. Refer to Appendix C for a detailed explanation of this descriptor.

2.7 Chapter Summary

This chapter provided a review of the tools used in the proposed solution. In general terms, this solution achieves metadata-driven transcoding by adapting the original digital item, containing the user requested media, based on the its metadata, usage environment, and media content. With reference to our systems diagram depicted in Figure 1.2, the concepts of DI and DIA in the MPEG-21 framework were discussed. We also reviewed how XML and XSLT sheets can be utilized during application-layer transcoding to transform the DI metadata, generate a presentation format, and obtain parameters for the bitstream transcoder. Next, the bitstream transcoder and the related aspects of the MPEG-4 standard were dis-

cussed. Finally, a brief outline of the tools offered by the MPEG-7 standard for the purpose of content analysis was given.

Chapter 3

Prior Work

This section provides a brief review of DL webcasting solutions and their shortcomings in addressing content adaptation to meet end-user environmental constraints. Furthermore, existing metadata-driven transcoding solutions are outlined.

3.1 Distance Learning

Web-based distance learning systems such as those discussed in [29–33] include a combination of video, audio, slides, chat session and whiteboard functionality. These applications utilize an Apache web server to interact with clients and provide web-based access to media on the server database (SQL backbone). Moreover, a RealNetworks server is employed to stream the media to clients. A similar approach is given in [5] with the exception that a H.263 video compression engine is used to stream an addition video feed (at low bit-rates) containing what is being demonstrated on a whiteboard. These distance learning solutions do not take into account that each user will have unique environmental conditions. In particular, they do not address the need for adapting media to meet the user device capabilities, user media preference (i.e. modality), and adapting the media encoding rate to take into account fluctuating network conditions.

A four-tier Electronic Educational System (EES) model was proposed in [34] to provide limited data personalization to the end-user. The top layer, known as the instructional layer, allows educators to specify which media components to include in the system (video, slides, chatroom, whiteboard, etc.). However, the end-users do not have the flexibility to specify which DL content is desired. The lower layers take the instructions from the top layer to create the final presentation format as seen through a browser. A similar a metadata-based approach for delivering personalized course material to target specific user learning needs was proposed in [35]. In particular, IEEE LTSC Learning Object Metadata (LOM) is used to create a flexible architecture to allow professors to deliver personalized content. For example, the user can choose the language or difficulty level of the DL content. Although both of the above mentioned models allow for personalized creation of the presentation content, they also do not address user personalization in terms of device and network capabilities, and media preference.

3.2 Metadata-Driven Transcoding

Section 1 presented a motivation for metadata-driven transcoding. As mentioned earlier, metadata syntax is represented using XML and metadata processing and transformations can be accomplished using XSLT [9]. Application layer transcoding requires transformation rules to process user metadata in order to determine the modality to stream and the stylesheet required to properly display this modality in a browser. Additionally, bitstream transcoding requires transformation rules to process user metadata in order to determine how to physically adapt the media to meet the device's multimedia processing and network bit-rate requirement.

A generic metadata transformation system is proposed in [22] and similarly in [36] as shown in Figure 3.1. These solutions consist of a Document Object Model (DOM) proces-

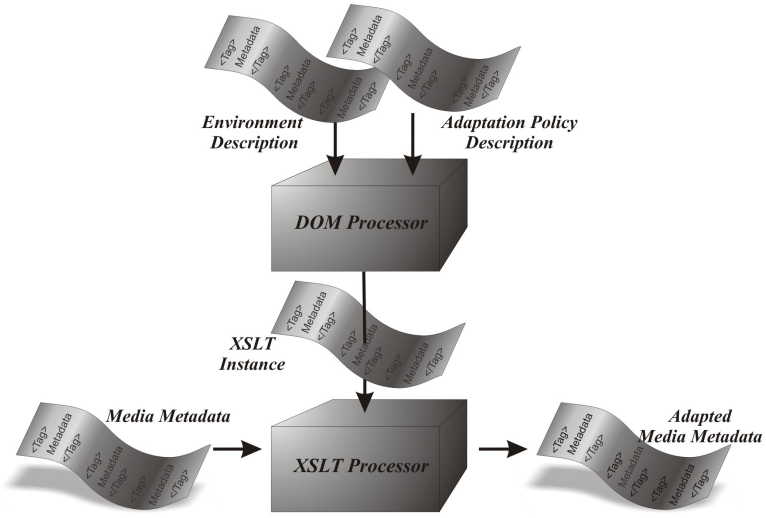


Figure 3.1: Metadata Transformation Engine.

sor that creates XSLT sheets and an XSLT processor that transforms the XML data with the created XSLT sheet. The DOM processor takes in the environment description and adaptation policy description as input and outputs an XSLT sheet that meets the current adaptation specifications. Although these implementations are intended to provide generic content adaptation solutions, implementing the DOM processor is application specific and hence requires knowledge of semantics of all applications that will have their contents transformed by this metadata transcoding engine. The problem with DOM processing is that it is requires intensive conditional programming, which is computationally expensive for a DL application that requires real-time transcoding.

Transcoding must be incorporated into existing DL applications to meet varying device capabilities, heterogeneous network conditions, and user modality preference. Since a DL application is considered, our method allows us to perform DOM processing a priori and cache XSLT sheets to transcode for unique environmental processing conditions. In essence, the proposed method replaces the DOM processor with a selection phase that is more efficient in the context of real-time transcoding. Hence, by having a fully cached XSLT solution,

metadata-driven transcoding can be accomplished in real-time.

3.3 Chapter Summary

This section provided a review of the state-of-the-art DL systems and metadata transcoding solutions. It was noted that existing DL systems offer very limited user personalization and thus do not comply to a DL content adaptation framework. We then proposed to use metadata-driven transcoding for achieving DL content adaptation and proceeded to review metadata transcoding solutions. Since such works are not specifically geared towards DL applications, they suffer several shortcomings including high computational complexity which prevents their use in realtime applications such as DL.

This thesis proposes a metadata-driven transcoding solution for adapting DL content to match end-user environmental conditions. By considering the particular characteristics of such applications, the proposed method can offer a real-time transcoding solution.

Chapter 4

Application Layer Transcoding

The objective of the proposed metadata-driven transcoding architecture [37] is to adapt the archived or live DL multimedia content requested by the user to match the requesting user's environment characteristics (device capability, network conditions, and personal preferences). The adaptation is achieved through application layer and bitstream level transcoding. This chapter presents a real-time solution for application level transcoding and generation of adaptation parameters for the bitstream transcoder.

4.1 Overview

The details of the proposed application layer transcoding engine is shown in Figure 4.1. During the first stage, metadata pertinent to a given DI is adapted using XSLT rules selected by the transformation module. The transformation module selects transcoding rules in a "best effort" manner to match the user's environment description (MPEG-21 usage environment). Next, the transformation rules and the MPEG-21 DI associated with the DL content are forwarded to the application layer transcoder. This transcoder uses the XSLT rules to perform application layer transcoding to produce a presentation (browser) template with the desired modality and produces adaptation parameters used to transform the underlying bitstreams

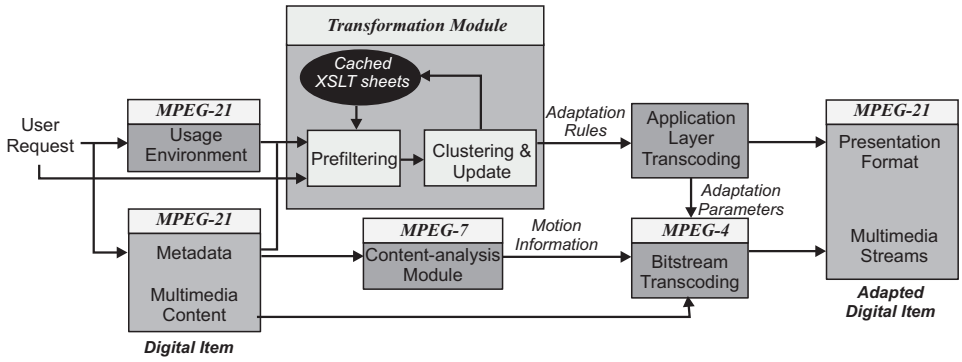


Figure 4.1: System Overview: adaptation engine that performs application layer and bitstream transcoding

to conform to device multimedia processing capabilities and current network conditions, thereby producing the adapted DI. A subset of these sheets can be found in Appendix A. The adapted DI represents the presentation template and DL multimedia that best matches the end-users environmental description needs. The final step is to stream the content to the end-user.

It must be noted that the adaptation can be performed at the content server, a proxy server, or the user terminal [25]. In this work, the DIA engine is implemented on the content server in order to create transparent access to end-user devices that might have minimal processing capabilities to keep the client (device) as thin as possible. The only requirement that is needed is JavaScript capabilities on the client. This is a fair assumption because most PC, PDA, and Smart Phones are capable of processing JavaScript. The details of the various components of the DIA engine are described in sections that follow.

4.2 Transformation Module

As seen in Figure 4.1, the objective of the transformation module is to choose the best matching XSLT rule from a set of cached sheets that will adapt the DL content to best match

end-user environment characteristics. The selection criterion depends on the parameters listed in Table 1.1. Furthermore, current available network bandwidth is taken into account when choosing the transformation rules. In contrast to previous approaches [22], [36], the transformation module does not create XSLT sheets to serve an incoming user request. Instead, a sheet best matching the user request environment needs is chosen from a set of previously cached sheets to provide best-effort service in real-time. This method avoids the high complexity and delay caused by creation of new sheets on the fly (i.e. DOM processing) as mentioned in Chapter 3. The transformation module also updates the cached sheets to better adapt to changing user environmental conditions by employing a clustering algorithm.

The XSLT sheet selection process is comprised of three steps. First, the cached sheets are pre-filtered to determine a small set of possible sheets matching the user request. Then, one of the sheets from this set is chosen to be used during transcoding and the cache is updated. Finally, the transformation rules are passed to a transcoder to produce the final adapted media. The details of the transformation module are discussed in the following subsections.

4.2.1 Pre-filtering

The first step in the proposed approach is to search the cached XSLT sheets and select sheets that satisfy the minimum parameter requirements (Table 1.1) that will enable the user's device to properly process the request multimedia. For example, if the user's device is capable of processing video at 15 frames per second (f/s), all sheets that process video at a rate greater than 15 frames per second are eliminated from the selection process. These parameters remain *unchanged* assuming the same device is used for the entire webcasting session. The pre-filtering of XSLT sheets is accomplished by partitioning the set of existing sheets using a tree structure [38] based on parameters such as device capabilities, modality, and available network bandwidth. The advantage of using a tree structure is that XSLT

sheets can be organized in a manner that allows efficient insertion, searching, and elimination of XSLT sheets in and out of the cache.

The pre-filtering component provides multiple XSLT sheets with the capability to transcode the requested media based on user device, and network conditions and preferences. Hence, a number of different sheets are available to be used by the transcoding engine for DIA which meet fixed session requirements such as the presentation format of the DL content and the multimedia modality that the device is capable of processing. Having multiple sheets allows the transcoder to switch between adaptation rules to account for dynamically fluctuating parameters such as available bandwidth. The next section describes how the clustering component i) chooses a unique XSLT sheet from the set of pre-filtered sheets for transcoding and ii) updates the cached sheets.

4.2.2 Clustering & Cache Updates

The objective of the clustering and update module is two fold. First, an appropriate XSLT sheet must be chosen based on the MPEG-21 UEDT for transcoding. Second, this section will explain how incoming user request UED (Appendix A) parameters are used to update the cached sheets. Updating the cache allows the system to cope with changing environmental conditions without the need for a system administrator to manually update the existing sheets. This section will demonstrate why a clustering algorithm is a suitable approach for sheet selection and updating. It is important here to note that the cache update can proceed in parallel to selecting appropriate sheets needed to support the user requests. Thus, no additional delay is perceived by the user.

In the following discussion, each sheet j is treated as a vector of transcoding parameters (all have numerical values) found in Table 1.1 denoted by $\mathbf{S}_j = [s_1, \ldots, s_m]^T$. In other words, the transformation rules of sheet j adapt the DL content to the environmental parameters

represented by vector $\mathbf{S}_j$. Similarly, the incoming user request i, $\mathbf{Q}_i = [q_1, \ldots, q_m]^T$, is a vector with the same dimensionality as the sheet vectors. Each incoming request vector is then compared against the set of pre-filtered sheets and the sheet most "similar" to the request is chosen for transcoding. The similarity comparison algorithm is discussed next.

Sheet Selection for Transcoding: The objective of the sheet selection algorithm is to map an incoming user request to a an XSLT sheet in the cache containing the adaptation rules for transcoding. In essence the incoming requests are clustered based on the sheet they map to. The main challenges during this clustering are determining the number of clusters needed and choosing initial data samples. For the purpose of our application, the number of clusters is 20, which is the number of cached sheets present in the proposed system. Furthermore, the cached sheets will serve as the initial data samples used. The set of requests served by a given sheet $\mathbf{S}_j$ generates a request cluster c_j which will be used for cache selection and update. It is appropriate that each sheet has its own cluster so that each sheet is individually updated according to the requests mapped to it. We denote the set of all clusters (corresponding to each cached sheet) after n requests have been received by the DL system as $\mathcal{C}_N^{(n)}$:

$$\mathcal{C}_N^{(n)} = \{\underbrace{\{\mathbf{Q}_{1,1}, \ldots \mathbf{Q}_{1,N_1}\}}_{c_1^{(n)}}, \ldots, \underbrace{\{\mathbf{Q}_{N,1}, \ldots \mathbf{Q}_{N,N_N}\}}_{c_N^{(n)}}\} \tag{4.1}$$

where N is the total number of cached sheets, $\mathbf{Q}_{i,j}$ and N_j are the i^{th} and total number of requests served by sheet j respectively, and $c_j^{(n)}$ denotes the cluster request for sheet j after n requests are received.

After the pre-filtering stage where M sheets are disqualified, $N - M$ sheets are available that match the end-user environmental restrictions. The clustering algorithm is used to pick one sheet out of the $N - M$ sheets in order to transcode according to the current environmental conditions. Clearly, finding a cached sheet that *exactly* matches all the environmental

criterions of a user request is unlikely. Thus the system must map each user request to a cluster (XSLT adaptation rule) that produces the lowest distance error. Hence, a similarity measure is needed in order to map an incoming data sample (i.e. user request) to a cluster (i.e. XSLT sheet). Choosing similarity measures that take into account the unique nature of the parameters in Table 1.1 is an difficult task to accomplish. This is because parameters such as frame-rate change continuously due to rapidly advances in technology. In light of the fact that popular similarity measures like variations of the Euclidean Distance [39] depend on properties such as the probability distribution of each parameter, it will be impossible to pin point a similarity measure for each unique parameter. The clustering algorithm proposed is based on a normalized Weighted Absolute Error (WAE) [39] similarity measure criterion. A normalized WAE criterion will provide the clustering algorithm with the following features:

- The absolute error can be calculated quickly. This quick calculation becomes important when the number of clusters is large. This is important for DL systems that require real-time operations.

- All parameters in Table 1.1 have numerical values that should be normalized because each parameter has a unique dynamic range. Normalization becomes important during the aggregation of similarity measures to ensure that parameters with large dynamic ranges do not mask the affect of parameters with small dynamic ranges.

- This similarity measure utilizes the user preferences [3] to weight the parameters according to user-perceived importance. Each weight represents a parameter priority between 1-3 (where 3 represents highest priority) which are assumed to be determined *a priori* by the user. For example, if the user preference states that the frame-rate is more important than display resolution, then more emphasis will be placed on finding a closely matching frame-rate request, hence providing he user with a degree of QoS personalization.

Sheet selection is accomplished by minimizing the losses over all clusters by mapping a request to a cluster that produces the lowest WAE. This loss is denoted as e and is calculated as follows:

$$e = \sum_{j=1}^{N-M} \sum_{i=1}^{N_j} d(\mathbf{Q}_{i,j}, \mathbf{S}_j) \tag{4.2}$$

where $d(\mathbf{Q}_{i,j}, \mathbf{S}_j)$ is the distance between a request and a sheet obtained as the WAE between the two vectors:

$$d(\mathbf{Q}_i, \mathbf{S}_j) = \sum_{k=1}^{m} w_k \frac{|q_k - s_k|}{\alpha_k} \tag{4.3}$$

where α_k is used to normalize the parameter distances to the range $[0, 1]$, and w_k indicates the importance of sheet parameter k. This process is repeated throughout the transcoding session. In other words, a user request if mapped to a particular sheets according to the following condition:

$$\mathbf{Q}_i \in c_j \quad \text{iff} \quad d(\mathbf{Q}_i, \mathbf{S}_j) \leq \min_{1 \leq j \leq N-M} d(\mathbf{Q}_i - \mathbf{S}_j) \tag{4.4}$$

A summary of the sheet selection algorithm is shown in Figure 4.2.

Maintaining the UED metadata of every request poses two burdens on the content server. The first concern is that each metadata description can be 50Kb in size [11]; assuming a large number of user requests, significant storage space may be required to store these descriptions. The second concern is incorporating a efficient and organized method of caching and accessing the information from these files when needed. In order to reduce the storage and indexing requirements on the server, the actual clusters, which include the metadata files mapped to them, are not stored. Instead, only cluster centers $\mathbf{S}_j,\ j = 1 \ldots N$ are stored. The next

Inputs:
$\mathbf{S}_j^{(n-1)}$: j^{th} cached sheet after $n-1$ requests are received.
$\mathbf{Q}_i^{(n)}$: n^{th} user request.
$\mathcal{C}_N^{(n-1)}$: Cluster centers after $n-1$ requests are received.
Output:
Selection: XSLT sheet index to be used during trascoding
Algorithm:

```
for each new request
  Pre-filter based on UEDT parameters to obtain C_{N-M}^{(n-1)}
  Select sheet j to minimize loss function e:
    d_min = ∞
    for j = 1 : N − M
      if d(Q_i^{(n)}, S_j^{(n-1)}) ≤ d_min
        d_min = d(Q_i^{(n)}, S_j^{(n-1)})
        Selection = j
    end
end
```

Figure 4.2: Sheet selection algorithm.

section will explain how the user UED is used to update the clusters without the need to store the metadata files on the content server.

Updating the sheets: The sheets in the cache represent transformation rules to adapt to popular device capabilities such as PCs, PDAs, and Smart Phones. There are also an ample amount of sheets to represent the varying capabilities of each device. For example, there are numerous sheets to adapt to a range of PCs. Two methods are used to update the cache. The first method is manually creating new sheets to take into account new devices and network conditions. For example, if new devices such as High Definition Television (HDTV) or Large Displays [40] require access to the DL content, new rules, hence new XSLT sheets will be required to adapt the DL content to meet these unique devices needs. The second method explained in this section updates the cached sheets based on environment parameters from

incoming requests of devices recognized by our system.

A cluster structure similar to the one presented for sheet selection is used to update the XSLT sheets transcoding parameters based on device, network, and personal characteristics of the users accessing the system. This section will give details of how the multimedia bit-rate encoding parameter will be updated based on the bandwidth characteristics of the heterogenous network connecting the end-user to the content server. Updating can also be applied to other parameters like frame-rate and resolution. Updating sheet parameters can become effective over time as devices such as PDAs and Smart Phones advance in their capabilities in processing multimedia. For example, it is reasonable to assume that a PDA will be able to process more than 15 frames per second as they currently do. Most probably, they will soon be able to process 30 frames per second, matching the frame-rates of conventional PCs. This clustering update algorithm will change the frame-rate parameter in the metadata to capture this shift over time. This automated procedure will save the administrator time for analyzing and updating the sheets to meet this change in demand. The bit-rate encoding of each sheet will be updated based on the mean of the requests mapped to that sheet cluster.

User environmental metadata can be exploited to update the sheets to minimize the sheet selection error over time as the system learns more about the users characteristics. By updating the sheets based on user characteristics, the DL system can make a closer match to the criteria (user environmental parameters) during the sheet selection stage. Note that the sheet selection clustering algorithm operates only on pre-filtered sheets. This is done in light of the fact that a given request can be served only with XSLT rules that produce a presentation template and adapted multimedia that the user's device is capable of processing. For example, a user request with a given frame-rate can be served by any sheets with that frame-rate or lower. In this case, transcoding at a frame-rate higher than that requested is meaningless due to the lack of capabilities on the device side. The pre-

filtering operation ensures that this scenario of over-estimating capabilities does not occur. It must be noted that the error e taken over *all* sheets has a high probability of being smaller than the error taken over the pre-filtered set. This is because a request with slightly higher quality parameters than sheet $\mathbf{S}_j$ will be assigned to another sheet with possibly larger loss. In order to remedy this situation, the cache updating strategy aims to minimize the loss over all cached sheets. Hence a sheet updating algorithm is introduced by considering the modified loss function:

$$e' = \sum_{j=1}^{N} \sum_{i=1}^{N_j} d(\mathbf{Q}_{i,j}, \mathbf{S}_j) \tag{4.5}$$

Using this loss function and the entire set of cached sheets, the algorithm in the previous section is used to assign an incoming request to a particular cluster. In the case of updating the bit-rate encoding, the cluster representative, or sheet bit-rate is the new sample mean evaluated using all elements belonging to the cluster and is updated incrementally using the new request:

$$R_{S_j}^{(n)} = \frac{n-1}{n} R_{\mathbf{S}_j}^{(n-1)} + \frac{1}{n} R_{\mathbf{Q}_i}^{(n)} \tag{4.6}$$

Due to the incremental update policy utilized here, previous requests do not need to be stored on the system. Note that not all the sheets are updated. The system reserves one sheet for every type of device supported that adapts DL content assuming minimal processing capabilities. This grantees that users with outdated technology can still access the system. The experimentation reported in this work will highlight the converging properties of the clustering algorithm and will demonstrate the bandwidth utilization gains by the the proposed design.

4.3 Chapter Summary

This chapter presented a real-time application layer transcoder module. The objective of this component is to transform the DI metadata and generate a presentation format for the media. Additionally, encoding parameters are generated and passed to the bitstream transcoder discuss in the next chapter.

The proposed solution is advantageous over existing transcoding solutions in terms of its real-time performance. This is mainly due to a novel fully cached-based solution for selecting and updating transformation rules. This should be contrasted to existing techniques where such rules are actually generated through processing that is computationally intensive and not suitable for realtime applications.

Chapter 5

Bitstream Transcoding

The first step of transcoding is to initiate application layer objectives such as producing the proper template to display the desired modality in a browser (Figure 1.1). Components of the presentation template such as video, slides, and text chat are described in the DI metadata. The XSLT sheet chosen in the previous stages of the transformation module is used to transform the metadata describing these components into a browser-friendly HTML format taking into account the end-users environmental restrictions.

The application layer transcoding stages also produces adaptation parameters that are used by the bitstream transcoder. The MPEG-4 Standard scalable tools outlined in section 2.5.1 will be used to provide temporal, spatial, and SNR scalability (i.e. vary the bit-rate encoding), hence serving as the bitstream transcoder. The adaptation parameters obtained from obtained from the application layer transcoder are utilized to perform transcoding on the media streams by providing the proper configuration files for the MPEG-4 encoder. For example, parameters for temporal and spatial scaling of video and images and audio scaling parameters.

The proposed bitstream transcoding methodology is flexible and easily extensible to any

block-based coding schemes such as MPEG-(1,2,4) [19], H.264 [41], H.263 [42], H.261 [42], Windows Media Video [43], etc. Note that the proposed solution concentrates on video transcoding due to the fact that it is the most demanding media with respect to processing power and bandwidth utilization. For example, MPEG-4 audio requires an average bit-rate of 16-24Kb/s [44] while MPEG-4 video requires 64-192Kb/s [19].

5.1 Transcoder Design for a DL Application

The MPEG-4 Standard is application independent. The advantage of this attribute is that it can be used for a diverse number of applications. The disadvantage is that application specific properties are not taken into consideration to provide an enhanced encoding and decoding processes. In this section, a novel MPEG-4 transcoder is proposed that will exploit the properties of a DL application to decrease the multimedia encoding rate (while maintaining a high level of QoS); this is crucial for mobile devices such as PDAs and Smart Phones that have low memory and CPU processing power. This is also useful for streaming multimedia over bandwidth deprived networks. Transcoding performance improvements will be addressed in order to have a real-time solution.

The only rule that we must religiously abide to in the transcoder design is ensuring that encoded bitstreams conform to MPEG-4 syntax. This rule insures that the encoded streams can be played on any industry-based MPEG-4 player (decoder). Hence the end-user will not require any special or additional software to use our system making the end-client as thin as possible.

The proposed transcoder contains adaptation rules and optimizations specific to formal lectures that have low motion video and high motion video that uses MPEG-7 motion analysis to adapt lab demonstrations.

5.1.1 Formal Lectures: Low Motion Video

The following adaptation rules have been incorporated into a MPEG-4 transcoder [45] to exploit low motion video properties of formal lectures to provide enhance real-time video transcoding that produces low bit-rates without sacrificing the user-perceived QoS. In particular, if adaptation parameters passed on from the application layer indicates a formal lecture (i.e. low motion), the following motion compensation optimization is applied in order to improve transcoding performance.

Motion Compensation Optimization: Appendix B provides a detailed explanation of how motion vectors (MV) are for motion compensated prediction to decrease the video encoding bit-rate. Calculating the motion vectors (MV) for motion compensated prediction of prediction (P) or bi-directional (B) frames is an expensive operation.

A noticeable characteristic of a formal lecture is that the speaker is concentrated in the same location in the frame and the motion activity is usually minimal. This characteristic can be used to our advantage by not calculating motion vectors at all (i.e. send vertical and horizontal motion vectors equal to zero for every macroblock). Motion vectors are only crucial for video with objects and scenes that contain unpredictable and high motion activity. By recognizing that most of the activity is going to be segregated in the same area, video can be encoded with no motion vectors. That is, the assumption can be made that the most closely matching prediction macroblock in the previous frame is the macroblock that is in the same location in the current frame, which implies that the motion vectors will be equal to zero. This concept is illustrated in Figure B.2 in Appendix B. Hence, the calculation of motion vectors can be skipped and the prediction error can be calculated by taking the difference between the current macroblock and prediction macroblock that is in the same location in the previous or future frame. Although motion compensated prediction would be better with MV available, admirable compression rates will still be achieved without MV do

to low motion. For example, the most of the background (macroblocks) may stay the same throughout the session.

5.1.2 Lab Demonstrations: High Motion Video

The following section will introduce optimizations for lab demonstration type of DL that might contain high motion video. Furthermore explanation will be provided on how the MPEG-7 Standard Motion Activity Descriptor (MAD) [28] can be utilized to provide motion analysis in video and how it can be used to accomplish video transcoding.

Partial Motion Vector Optimization: As mentioned earlier, MV calculations for motion compensated prediction is an expensive operation and causes computational strain on the transcoder. As the frame size increases (i.e. resolution), the number of MV that needed to be calculated increases significantly. The MPEG-7 MAD can be utilized to efficiently decrease the amount of MV calculations needed to achieve admirable compression rates via motion compensation. In particular, we are interested in two values from MAD, the *intensity of activity* and *spatial distribution of activity* as described in Appendix C. In [46], a method for extracting this motion information is proposed and some of the ideas proposed were adopted by the MPEG-7 Standard. The problem with the proposed extraction method in [46] is that MV must be calculated for every macroblock of every frame in order to characterize the motion activity of every group of pictures (GOP); this contradicts our goal of decreasing the amount of MV calculated per frame. In this section, a variation of the extraction method proposed in [46] will be presented. An algorithm for improving transcoding performance with respect to MV calculations utilizing metadata from MAD. Improved transcoding performance will be crucial for the computationally expensive operation of transcoding from raw video to MPEG-4 format.

In [46], the magnitude of each macroblock is calculated. An average, C_{mv}^{avg}, is obtained

by taking the mean of all the calculated magnitudes. C_{mv}^{avg} is used as a threshold to label each macroblock as an area with motion (labelled as H), or with low or no motion (labelled as L) using the following equation:

$$Macroblock_i = \begin{cases} magnitude(MV)_i \geq C_{mv}^{avg} & \text{high motion(H)} \\ magnitude(MV)_i < C_{mv}^{avg} & \text{low motion (L)} \end{cases} \tag{5.1}$$

The proposed method uses a more efficient method to speed up the motion intensity calculation. Instead of calculating MV for every macroblock, each frame is partitioned into subpictures of 5x5 macroblocks (where each macroblock is 8x8 in size). Only the center macroblock MV, denoted as a seed, is calculated and is used to represent an approximation for the entire 5x5 group's motion information indicating the amount of activity in that subpicture. This operation is shown in Figure 5.1. Hence, C_{mv}^{avg} in the proposed optimization is obtained by only by taking the mean of all the seeds. If a subpicture is determined to have high motion (labelled H in Figure 5.1), then motion vectors are calculated for every macroblock in that subpicture. This allows us to approximate which subpictures contain a significant amount of motion and decrease the amount of motion vectors that need to be calculated. Using our method, we can also carry on from [46] to calculate *intensity of activity* more efficiently to classify a GOP into high or low motion intensity.

Because less MV are now going be used to calculate the *motion intensity of activity* and *spatial distribution of activity* parameters in [46], the values must be normalized to reflect the reduction in the number of MV used.

The following MV calculation algorithm will demonstrate the benefits of using less MV when encoding a P frame that requires motion compensation:

- If $motionintensity = [0, 2]$ in this GOP, do not calculate MV for this GOP (because motion is very low), only calculate prediction error like in low motion video algorithm explained above.

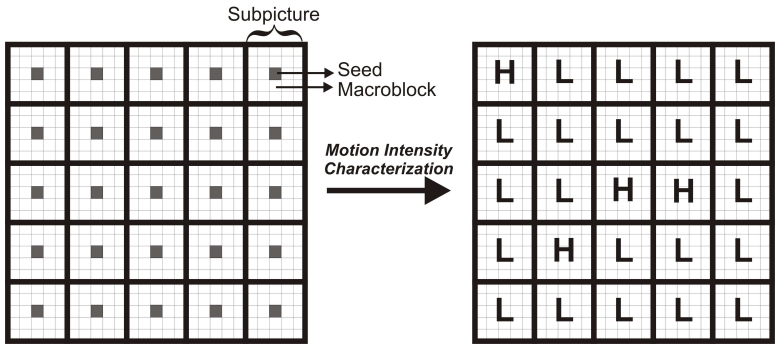


Figure 5.1: Subpicture are label into sections of high (H) or low (L) motion; only MV calculated are for subpictures labelled H

- If $motionintensity = [3, 5]$ in this GOP, only calculate MV for macroblocks in subpictures that are determined to have motion (Equation 5.1). A value of 3 or 4 indicates that some motion activity exists; calculating only some subpicture's MV (as opposed to all) in hope that some subpictures will be low in motion.

If there exist low motion activity, the prediction error will be small because the macroblock in the same location in the previous frame most likely be very similar to the macroblock being calculated, hence calculating MV for this macroblock will not produce significant coding efficiency (i.e smaller prediction error values). The same idea is applied to frames with moderate to high motion; chances are only parts of a frame will have significant movie, hence not all MV need to be calculated.

Temporal Video Transcoding Hints Utilizing MAD: The MAD can also be used to provide temporal scaling for the purpose of temporal video transcoding. The MAD would provide video content analysis to realize an intelligent transcoding solution. For example, for video captured that is captured at 30f/s by the input device, the frame-rate needs to be scaled down to 15f/s for some PDA. Or for situations where high bandwidth fluctuations exist, instead of dropping frames randomly, motion analysis via the MAD would enable making

critical frame dropping decisions based on the intensity of motion in the video segments (i.e. GOP).

The method implemented analyzes two video segments (150 frames each) at a time and drops frames from the video segments depending on their motion intensity (i.e. amount of activity in the segments). Note that you can use as many segments as required; two was a good tradeoff between Peak Signal-to-Noise (PSNR) [47] improvements and computation intensity. PSNR is ratio between the maximum value of a signal (image in our case) and the magnitude of background noise. It is a measure of quality of reconstruction in image compression. PSNR is not always an ideal choice for measuring perceptual quality. However, it will give some useful insight on the quality of the recover frames at the decoder side.

The main idea behind this approach is not to randomly drop frames. Let's say there are two segments A (low motion) and B (high motion), the proposed method implements a mapping in order to drop more frames from segment A rather than B because there is probably more key frames in segment B that need to be preserved. Without content analysis, important information can be lost because the same amount of frames would be dropped from each segment in conventional transcoders. In the case where the decoder needs to perform frame interpolation between $frame_{t-1}$ and $frame_{t+1}$ to recover the dropped frame $frame_t$, distortion will be less evident if more frames are dropped from segment A because there is a high probability that frames in segment A are very similar due to its low motion characteristic. Suppose we have the original video frame F_k that was dropped by the transcoder from segment A or B, and let G_k represent the interpolated frame on the decoder side, we want to maximize the following equation supposing that n frames are streamed to the end user:

$$AveragePSNR = \frac{\sum_{i=1}^{k-1} psnr(F_i, G_i) + psnr(F_k, G_k) + \sum_{j=k+1}^{n} psnr(F_j, G_j)}{n} \tag{5.2}$$

This equation would be maximized when the frame dropped is properly recovered at the decoder by means of interpolation. This equation is calculating the average PSNR when one frame is dropped, this can be extended to dropping multiple frames. Other methods [47] perform offline content analysis to find which frames dropped cause to largest drop in PSNR, which is only practice for video on demand applications . Although the proposed frame-dropping content analysis technique does not produce optimal PSNR results compared to [47], it does provide a real-time content analysis solution for providing PSNR improvement for temporal scaling.

SNR Transcoding: DCT Optimization The DCT and its role in block-based video coding is explained in detail in Appendix B. Performing DCT to convert YCbCr [20] planes from the spatial domain to the frequency domain requires optimization due to its high computational complexity in MPEG-4 video encoding. This high computation is due to the fact that the DCT operation on an NxN macroblock requires N^2 multiplications and N(N-1) additions.

The eye is much more sensitive to overall intensity (luminance) changes rather than to color changes. Usually most of the information pertaining to a frame is contained in its luminance rather than its color (chrominance). When transcoding the streams for mobile devices or low bit-rate networks we can take advantage of this human perceptual quality by performing a full 8x8 DCT on only the Y planes (hence preserving the entire plane) and only a partial (3x3) DCT on Cb and Cr planes (discarding a significant portion of these planes). If the bit-rate reduction need is low, then the transcoder can switch to a 5x5 DCT to send more coefficients and a higher QoS. The uncalculated coefficients default to zero. This method is a variation of [48] that uses frequency filtering to guarantee zero coefficients in well defined areas of the matrix. Besides the gain in saved calculations, the coefficients that are defaulted to zero are then run length entropy coded (Appendix B) and can all to be represented with one codeword. As a result, the only coefficients that are sent (in the

bitstream) for each Cb and Cr components are the 9 coefficients that were calculated and the 1 codeword representing the rest of the macroblock, for a total of 10 codewords as opposed to an upper bound of 64 (in the worst case). This optimization will also prove to be beneficial for the inverse DCT (IDCT) operation during decoding process on the client end because less coefficients will need to be processed, hence preserving CPU power and memory usage. The results in Chapter 6 will show the percentage of the calculations saved and show screen shots of how the perceptual quality is preserved in the decoded frames with this optimization.

The quickest and most efficient method to achieve further SNR scalability is to increase the quantization step by multiplying the the quantization tables by a scaler factor. Instead of multiplying each YCbCr plane's quantization tables by the same factor, the multiplication factor of the Cb and Cr planes will be greater that the Y planes to take advantage of the fact that the human eye is more sensitive to distortion in the Y plane.

The results in Section 6 will show how the proposed optimizations decrease the start-up transcoding delay while keep low bit-rates in order to produce a live lecture experience. Furthermore, results will indicate how the proposed temporal scalability methodology increases the QoS of the received video with negligible content analysis delay.

Furthermore, the bitstream transcoder requires the knowledge of available bandwidth in order to encode multimedia at a bit-rate that meets the end-users network conditions. The next section will explain how the proposed method recommends to estimate the available bandwidth in order to encode and streamed DL content at a high level of QoS while dealing with the end-users unreliable IP network conditions such as packet loss due to congestion. Design guidelines and recommendations for incorporating a bandwidth estimation protocol suitable for a DL design to deal with bandwidth fluctuations in IP networks will be given.

5.2 Bandwidth Estimation

The proposed system is designed for the following encoding and streaming protocol to deliver DL media according to the available bandwidth of the network connecting the end-user to the content server. Due to the fluctuating nature of IP networks, available bandwidth must be constantly re-estimated to avoid device buffer overflow and multimedia degradation due to significant packet loss when network congestion occurs. Hence, a bandwidth estimation protocol is a critical component for ensuring a high level QoS multimedia delivery over best-effort networks like the Internet.

Bandwidth estimation is the determination of available bandwidth for data transmission based on dynamic factors like packet loss and delay in order to cope with congestion points in the underlying network. Congestion is usually caused by greedy protocols such as UDP traffic [27], which do not contain a (bandwidth) rate control mechanism for sharing available bandwidth like TCP streams [27] at congested Internet points. Existing TCP rate control algorithms are suitable for applications such as HTTP and FTP applications that can withstand bursty data transmission (i.e. can withstand abrupt bandwidth fluctuations). However, bursty transmission is not appropriate for a DL application due to the high sensitivity of the human observer to visual data bit-rate fluctuations [49], especially in video. In light of this fact, smooth bandwidth rate estimation dynamics, whether increasing or decreasing, are required for streaming multimedia such as video. In the particular case of video transmission, a TCP rate control algorithm would reduce the transmission rate by halve as a reaction to each reported packet lost. Since the occasional loss of packets can be tolerated during video and audio streaming [50], an aggressive rate control algorithm like TCP that can assure 100% packet success transfer is not required for our application.

The proposed method is designed to utilize a TCP Friendly Rate Control (TFRC) variant of the TCP congestion control algorithm called Equation-Based Congestion Control

(EBCC) [51]. The fact that EBCC is TFRC ensures that it will behave "ethically" by sharing the available bandwidth fairly with competing streams when congestion occurs. Unlike conventional TCP rate control, EBCC does not drastically change the bit-rate encoding when packets are lost; it reduces the bandwidth by a ratio of $\frac{7}{8}$. However, EBCC behaves in an aggressive manner when multiple packets are lost by decreasing the transmission rate by a factor of 2. Furthermore, the bit-rate is gracefully increased once congestion decreases to maintain smooth dynamics. Equation (5.3) [51] provides one example of calculation of a TFRC response function used to estimate and control the bandwidth rate. This formulation can be used to provide an upper bound on the transmission rate for a given packet size s, round-trip-time R, calculated packet loss p, and the retransmission timeout t_{RTO}. Figure 5.2 shows the smooth dynamics of EBCC compared to a typical TCP congestion control algorithm using the ns-2 [1] IP network emulator.

$$BW_{available} = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{2p}{3}})p(1+32p^2)} \tag{5.3}$$

A low packet loss ratio is tolerable in video; concerns grow when this ratio increases significantly. A significant packet loss ratio can make a frame no longer rendible. In our application where we use prediction (P) and bi-directional (B) frames, this could make multiple frames undecodable due to error in just one frame. This is because the construction of P and B frames rely on information from previous or future Intra (I) and P frames. As Figure 5.3 shows, systems that encode and stream multimedia at a fixed rate that do not monitor congestion and adjust their encoding rate could be disastrous in term of packet loss. Due to the overshooting of the fixed bit-rate encoding, the best-effort property of the IP network randomly drops packets when congestion is evident. However, due to the adaptive nature of TCP and EBCC, congestion control allows both methods to maintain a low level of

[1] http://www.isi.edu/nsnam/ns/

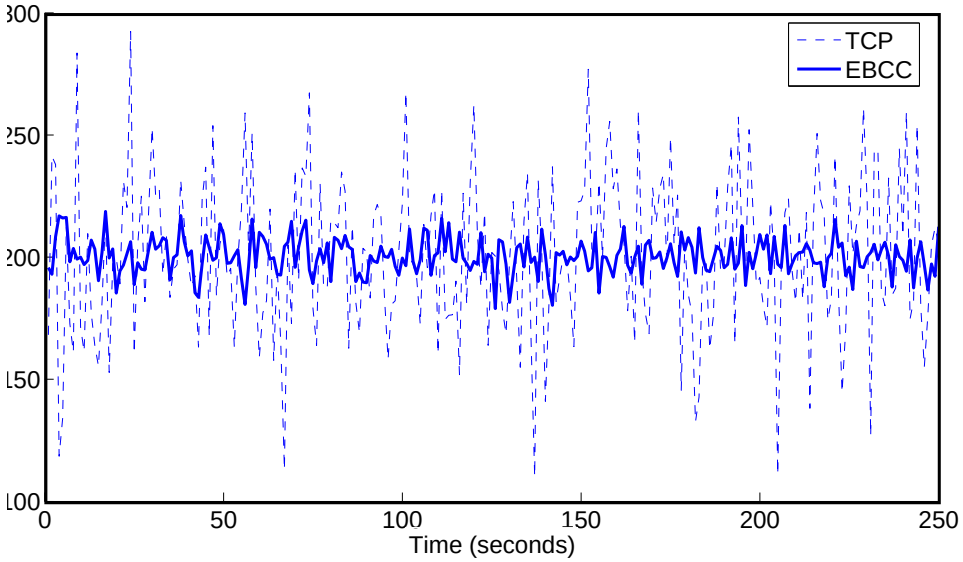


Figure 5.2: TCP and EBCC bandwidth estimation and rate control algorithms in response to packet loss.

packet loss (which can be a QoS parameter specified by the client or server). This plot also shows that EBCC can compete respectably with TCP algorithms in streaming multimedia with a low packet loss ratio. The details of EBCC is outside the scope of this paper, and the reader is referred to [51] for complete details.

5.3 Network Topology Design

Another key challenge in designing a DL content delivery system is dealing with a high traffic of users. The amount of users logging is to the system has great repercussions on system (DL content server) and network resources. If a high number of users are being catered to simultaneously by the DL server (e.g. 100), this will have a significant constraint on the processing power of the transcoder to adapt and customize DL material to meet each user's personal preference. In addition, available server side bandwidth has an upper limit on the amount of users in can deliver to simultaneously; this can be a big bottleneck in an application

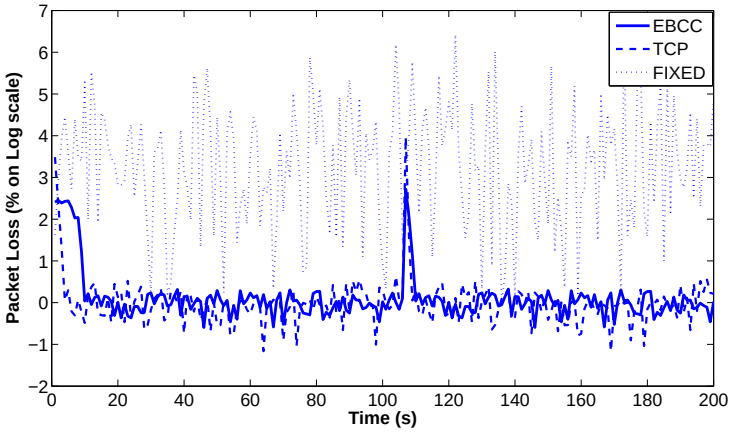


Figure 5.3: The ratio of packets lost over a period of 200 seconds at a congested network node.

like DL that delivers high bit-rate media such as video. The above mentioned problem is a constraint that is caused in a unicast network. A unicast network is an established connection between one sender (DL server) and one receiver (end-user). The above mentioned system constraint can be (partially or fully) eliminated by broadcasting DL material over a multicast network whereby one sender can stream content to multiple user simultaneously. There are many proposed solution in today's literature and commercial designs dealing with multicasting, however none of them provide a solution for delivering personalized content to end-users. This section will present a hybrid unicast/multicast design for the delivery of personalized content to end-users. Two methods will be presented; one based on relay transcoders and another one that utilizes the XSLT sheets mentioned in Chapter 4.

5.3.1 Method 1: Relay Transcoders

Method 1 shown in Figure 5.4(a) utilizes relay transcoders to produce a partial multicast solution even though the underlying network might be a unicast infrastructure. This method works as follows: There is a local DL content server that captures the live sessions, encodes and archives the various media streams. This server transcodes and delivers personalized

DL material to local users, for example within the city limits as shown in Figure 5.4(a). There also exists relay transcoders that might exist within the city, outside the city, or in other countries that are also capable of delivering personalized content to local users. The basic idea behind this approach is to send one unicast stream to relay transcoders instead of wasting bandwidth and sending unicast streams to each remote user. This method is effective in scenarios where there are many users are remote (i.e outside of the city) or for networks with no multicasting capabilities that offer low server-side bandwidth. Although extra hardware is needed, it is not a significant issue due to the fact that hardware is becoming inexpensive.

5.3.2 Method 2: Utilizing XSLT Sheets

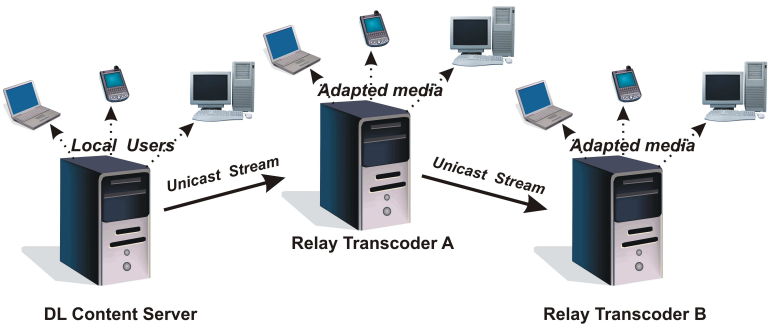
Method 2 shown in Figure 5.4(b) utilizes the clustering algorithm that was introduced in Section 4.2.2. The basic idea behind this approach is to send one multicast stream to all that map to the same XSLT sheet (i.e. require same transcoding rules) instead of sending unicast streams to each one of these users. This method achieves a partial multicast solution while providing personalized content to each user while saving significant amount of bandwidth like in Method 1. This method becomes effective in scenarios where there are many users logging on to the system that user the same type of device and require the same modality.

Section 6 will compare both methods with respect to system and network resource utilization. Choosing between both Method 1 and 2 would also depend on the system delivery architecture.

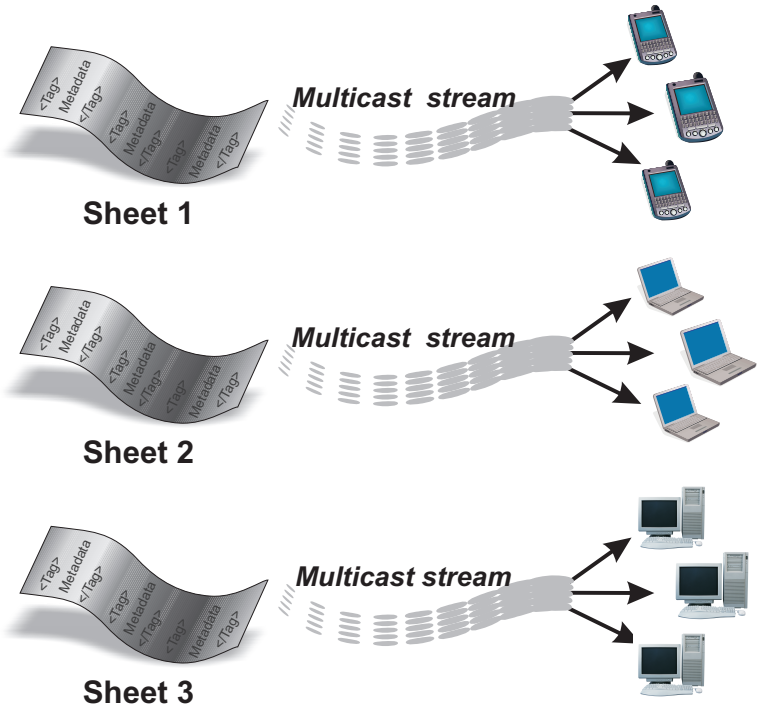
5.4 Chapter Summary

This chapter presented a real-time bitstream transcoding solution. The objective of this component is to transform the underlying bitstream using adaptation hints/parameters from the application layer.

Solutions for SNR and temporal transcoding were presented. Furthermore, motion analysis was incorporated for transcoder computational efficiency and for intelligent temporal scalability. Recommendations were given on bandwidth estimation and rate control that would be suitable for a DL application. The chapter concluded detailing network topologies that are capable of providing a partial multicast solution that deliveries personalized DL content to the end user.



(a) Method 1: Relay transcoder



(b) Method 2: Utilizing XSLT sheets

Figure 5.4: Hybrid unicast/mulicast approaches for DL content delivery for remote users.

Chapter 6

Results

This section will show the experimental setup and results of the proposed architecture to achieve the following criterions:

1. Perform application layer transcoding to produce and deliver personalized DL presentation template to meet the user's device constraints and preference needs.

2. Validate that the proposed clustering algorithm selects legitimate adaptation rules that matches the end-user's environmental needs. The selection process must be performed with negligible delay in order not to violate the real-time nature of a DL application. This subsection will also demonstrate the convergence property of the proposed cache updating algorithm.

3. Validate that the MV optimization schemes are capable of producing bit-rates comparable to conventional MPEG-4 transcoders while producing a significant improvement in transcoding delay.

4. Perform efficient frame dropping to achieve temporal transcoding. Verify that content (motion) analysis enhances temporal transcoding to produce higher QoS than conventional MPEG-4 methodology. Motion analysis must be efficient in order to meet the real-time demand of DL.

5. Perform efficient SNR transcoding that significantly decreases the encoding bit-rate and decoding complexity on mobile devices that consist of low processing power.

6. Achieve bitstream transcoding that adapts DL multimedia to match device processing capabilities and available bandwidth with acceptable start-up delay.

7. Investigate the bandwidth utilization capabilities of the proposed hybrid unicast/mulicast network topologies.

6.1 Experiment Setup

The proposed method is integrated into a setup similar to the ePresence system in order to take the next step to provide adaptive transcoding and streaming capabilities into the current architecture. The media that must be transcoded are video, audio, and images (i.e. slides) in the event stream. The experiments conducted and results obtained reflect the scenario of a end-user viewing a two hour lecture from a DL content server 600 Km away. This is intended to mimic a realistic DL session. The following sections provide details about what devices as well as what data sets used and how they were obtained.

6.1.1 Devices

The main objective of this thesis is too deliver DL content to end-users with devices such as a PC, PDA, or Smart Phone. The specifications of the PC and PDA that are used reflect the average processing capabilities (running Windows operating system) of users at the time this thesis was written. The PC consisted of an Intel Pentium 4, 2.8 GHz processor with 512 Mb of memory running Windows XP. A mainstream PDA is also used, namely the HP IPAQ hx2000 that consists of a Intel PXA270, 520 MHz processor, with 32 Mb of memory running Windows CE.

Experiments emulating the capabilities of a Smart Phone were conducted using Microsoft's Smart Phone Developer's Guide SDK [52]. Both the application layer and bitstream experimentation were verified using this SDK. The basic idea behind this SDK is to emulated how an application such as DL content being streamed to a user would be visual seen and decoded on a Smart Phone. This allows visual verification and also allows the measurement of QoS of the media as well as start-up or device processing delays.

6.1.2 DL Multimedia Data Set

As mentioned in previous system, the proposed architecture uses MPEG-4 as a means to encode video and audio. However, it should be reiterated that the proposed architecture, can apply to any block based coding standards. An FFMPEG [45] encoder is used as the physical transcoder witch is incorporated with temporal and SNR scalability as proposed by this thesis. The open source Darwin Streaming Server [53] is used as a means to stream MPEG-4 and 3GP (MPEG-4 format for mobile devices) content to end-users. For every streaming session used for experimentation, an archived or live lecture encoded in MPEG-4 format.

In light of the fact that DL video data sets are not standardized like the popular Foreman video clip, we use our own video clips for experimentation. Two videos are used to reflect low and high motion, namely, a two hour formal lecture that contains low motion and a two hour lab demonstrations with high motion. Both videos are encoded at 15 and 25 f/s at four different resolutions (sqcif, qcif, cif, and 4cif). The Results sections will clarify why the these particular encoding parameters are used.

6.1.3 Bandwidth Estimation Data Set

In order to test the bandwidth estimation of our proposed method, a reproducible heterogenous network environment needs to exist that has similar behavior to the Internet. An IP

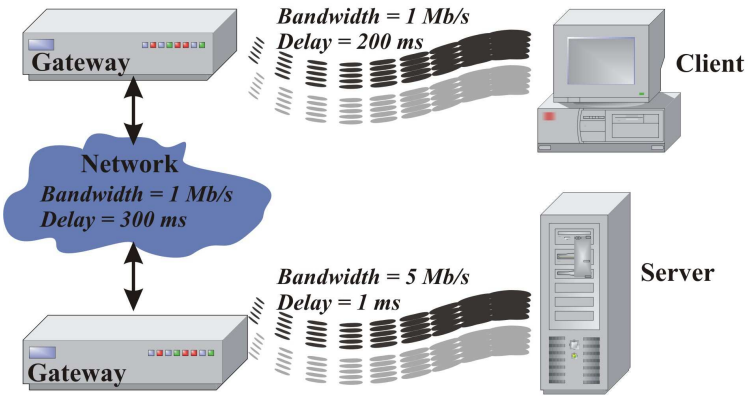


Figure 6.1: Client/Server network configuration for our experiments assuming a 600 Km streaming distance.

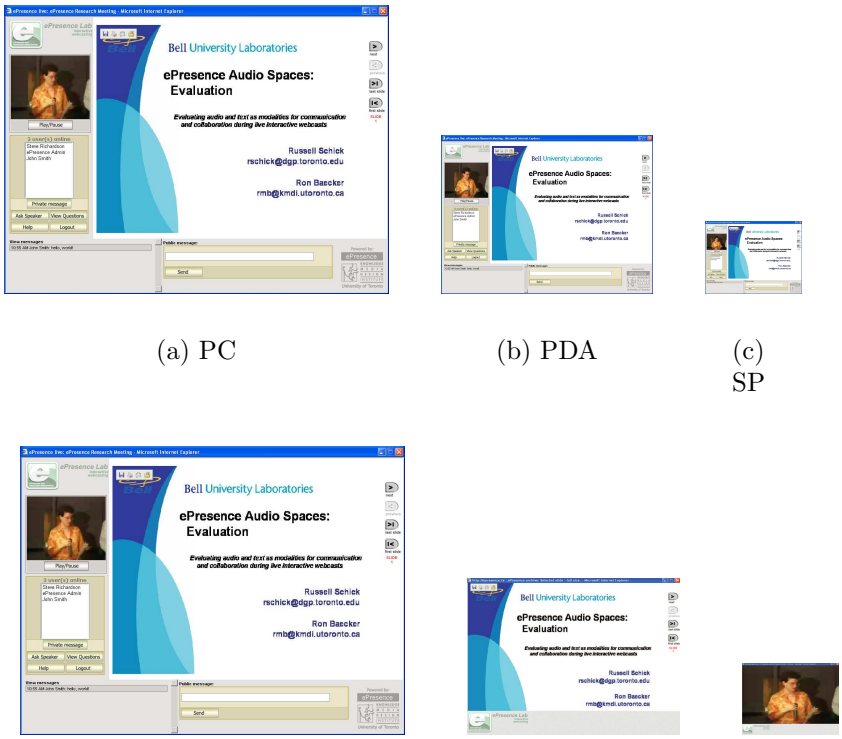
network emulator ns-2 is used to simulate a fluctuating heterogenous network for realistic interactive webcasting scenario. Figure 6.1 shows the IP network configuration used with the Real Time Streaming Protocol (RTSP) [6] and Real Time Control Protocol (RTCP) [6] plus TCP sever streams (dark and light colored waves) respectively and similar configuration on the client side. The RTSP connection between the server and the client is for the actual multimedia transportation and the RTCP plus the TCP connections are used for bandwidth estimation and for exchanging other client/server control information. The simulation was performed for a client connected with a 1Mb/s bandwidth capacity (e.g. located in Montreal, Canada) and the server is with uplink capabilities of 5Mb/s located in our University of Toronto Multimedia Lab (an approximate distance of 600 Km). This setup with varying parameters is used for experimentation.

6.2 Results

6.2.1 Application Layer Transcoding

Criteria 1: Application Layer Transcoding

Application layer transcoding is important to meet the user's preference and with respect desired modality and delivering the requested multimedia layout. More importantly, it is



(a) PC (b) PDA (c) SP

(d) PC (e) PDA (f) SP

Figure 6.2: ePresence: modality adaptation to meet the user's preference for PC, PDA, and Smart Phones (SP). Figure a-c show streaming without transcoding (assuming all devices can process all media). Figure d-f represent adaptation applied by proposed method.

also crucial in determining logically via XSLT transformation rules what are the device's capabilities in processing the requested modality. For example, it would not be logical to stream video, audio, slides, and text chat on a PDA with a QCIF display. This could be disastrous because the font size of the slides and the text chat would be too small to read (smaller than font size 8 for example). Figure 6.2 clearly displays this problem. Figure a-c show how the display size affects the clarity of the delivered media. Hence, it would not be logical to stream all the available media to a PDA or Smart Phone. Figure d-f show how application layer transcoding takes all these factors into account and adapts the available media and display format using adaptation rules. This ensures the proper delivery of DL

content to meet the device needs and while trying to meet the end-users modality request.

Sheet	Resolution	frame-rate
S1	CIF	15
S2	CIF	10
S3	QCIF	8

Table 6.1: XSLT sheets that adapt DL content to the specified parameters.

Request	Resolution	frame-rate	WAE S1	WAE S2	WAE S3
1	CIF	15	**0**	0.31	1.1
2	CIF	12	PF	**0.1**	0.9
3	QCIF	12	PF	PF	**0.3**

Table 6.2: Three user requests with unique resolution and frame-rate requirements. The WAE is calculated for the Sheets S1-S3 in Table 6.1.

Criteria 2: Validation of Clustering Algorithm

This subsection will address three aspects of the clustering algorithm: a) sheet selection process via an example, b) delay incurred due to sheet selection and metadata transformations, c) capability to update cached XSLT sheets.

Pre-filtering, Sheet Selection, & Metadata Transformations: A sheet selection example will be demonstrated to adapt DL content for a PDA. For simplicity, it is assumed that all parameters shown in Table 1.1 are matched perfectly between the sheets (Table 6.1) and the user requests (Table 6.2) except for the video resolution and frame-rate. The proposed sheet selection behaves as follows for the three user request shown in Table 6.2:

- *Request 1:* This is the case whereby the user environmental restrictions can be perfectly matched by a cached sheet. Sheet S1 produces a WAE of 0, and is selected for transcoding.

- *Request 2:* This request cannot be matched for every parameter requirement. Sheet S1 is pre-filtered (PF) because it does not match the requested frame-rate. Sheet S2

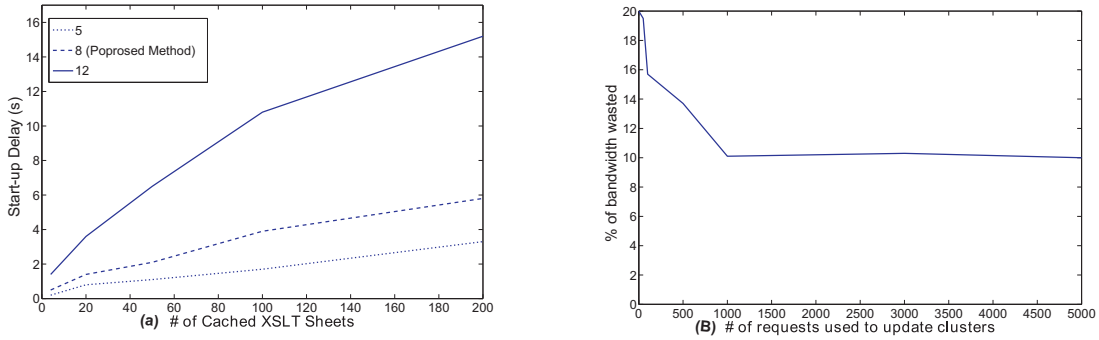


Figure 6.3: Start-up delays incurred by a) pre-filtering, sheet selection, and metadata transformation, b) Improvements in bandwidth utilization after updating sheets

is determined to be the best match; user is granted the requested resolution, however the frame-rate is lower than what the user's PDA is capable of processing.

- *Request 3:* As mentioned in chapter 4, base sheets must be cached and never updated to meet the needs of devices with minimal/poor processing capabilities; this is represented by sheet S3 in our example. Although sheet S2 meets the frame-rate requirement of this request, it is PF because it does not meet resolution requirements.

As shown in this example, best effort personalization is provided to the end-user with the guarantee of matching environmental restrictions to deliver DL content that can be properly processed.

Delay Incurred by Algorithm: One of the appealing aspects of the proposed metadata-driven transcoding solution lies in its efficiency to determine adaptation rules to meet the end-users environmental needs and performing metadata transformations. This is accomplished by having a fully cached XSLT solution that does not require DOM processing. Figure 6.3(a) shows the delay incurred by this algorithm. Our proposed method currently supports 20 cached sheets and utilizes 8 parameters (Table 1.1). As Figure 6.3(a) shows, the proposed

algorithm experiences a negligible delay of approximately 1.5 seconds. This graph also displays the trade-offs between delay and caching more XSLT rules and the repercussions of taking more environmental parameters into account. By caching more sheets, the DL system can provide more detailed adaptation at the cost of having a higher delay.

Convergence of Updating Algorithm: As mentioned in section 4.2.2, device and network parameters in the XSLT sheets (i.e. clusters) will be updated according to incoming requests (i.e. samples) to improve bandwidth utilization and to comply to changing demands in device parameters like frame-rate and resolution. Figure6.3 shows the improvement in bandwidth utilization with respect to the number of requests used to update the clusters. At approximately 1000 requests (used for updating the sheets), the updating operation of the clustering algorithm converges at a local minimum of 10% wasted bandwidth. Furthermore, no significant improvement is reached at the algorithm approaches 5000 requests, which gives confidence that the clustering algorithm converges. This make sense intuitively because after 1000 request, the system has probably been exposed to a wide array of users' devices and network properties accessing the system. Hence, parameters of the system were update to adapt to the network conditions of incoming requests.

6.2.2 Bitstream Transcoding

Criteria 3: Motion Vector Optimizations

Optimizations must be incorporated when transcoding from raw video to a compression scheme such as MPEG-4. One significant compression gain is accomplished by using MV to accomplish motion compensated prediction. In order for the MV optimizations mentioned in Chapter 5 to be incorporated into the proposed framework, it must be verified that the transcoding performance improvement does not cause a significant increase in the encoding bit-rate; this would defeat the objectives of a compression scheme.

The first MV optimization that must be verified is sending no MV for a formal lecture

that contains low motion. Figure 6.4(a) shows the compression gains at different resolutions all encoded at 25f/s comparing two methods: encoding with no MV and the MPEG-4 standard methodology. As this figure shows, both methods achieve similar compression rates. In this low motion scenario, the MPEG-4 method either calculates MV (which proved to be ineffective, or encodes the bitstream with (horizontal and vertical) MV equal to zero for low motion macroblocks. In either case, the proposed approach proved to be more effective because the MPEG-4 method wasted processing power on attempting to find similar macroblocks in the previous frame in order to obtain MV.

The partial MV calculation using the MPEG-7 MAD the *intensity of activity* and *spatial distribution of activity* metadata must also be verified with respect to coding gain for video with high motion activity such as lab demonstrations and conference type settings. Figure 6.4(b) shows the compression gains at different resolutions all encoded at 25f/s comparing two methods: encoding with partial MV calculations and the MPEG-4 standard methodology. As the figure shows, compression gains are not as admirable are the MV optimization method applied to low motion video (Figure 6.4(a)). There is roughly a 10% compression loss using the proposed method. Although there exists a minor compression loss, this optimization significantly decreased the transcoding delay compared to conventional MPEG-4 transcoding. Figure 6.5(b) verifies that approximately 15-20 seconds on average of transcoding delay is saved compared to the MPEG-4 method. This is a valuable tradeoff between compression gain and startup delay.

Criteria 4: Temporal Transcoding

This section will show the result of the temporal transcoding approach utilizing motion analysis. In particular, the MPEG-7 MAD motion intensity metadata is utilized with the algorithm presented in Chapter 5 . Results will be presented with regards to QoS gain in comparisons to the tradeoff with overhead delay incurred during content analysis.

The frame rate distribution algorithm discussed in Chapter 4 was tested against uniformly

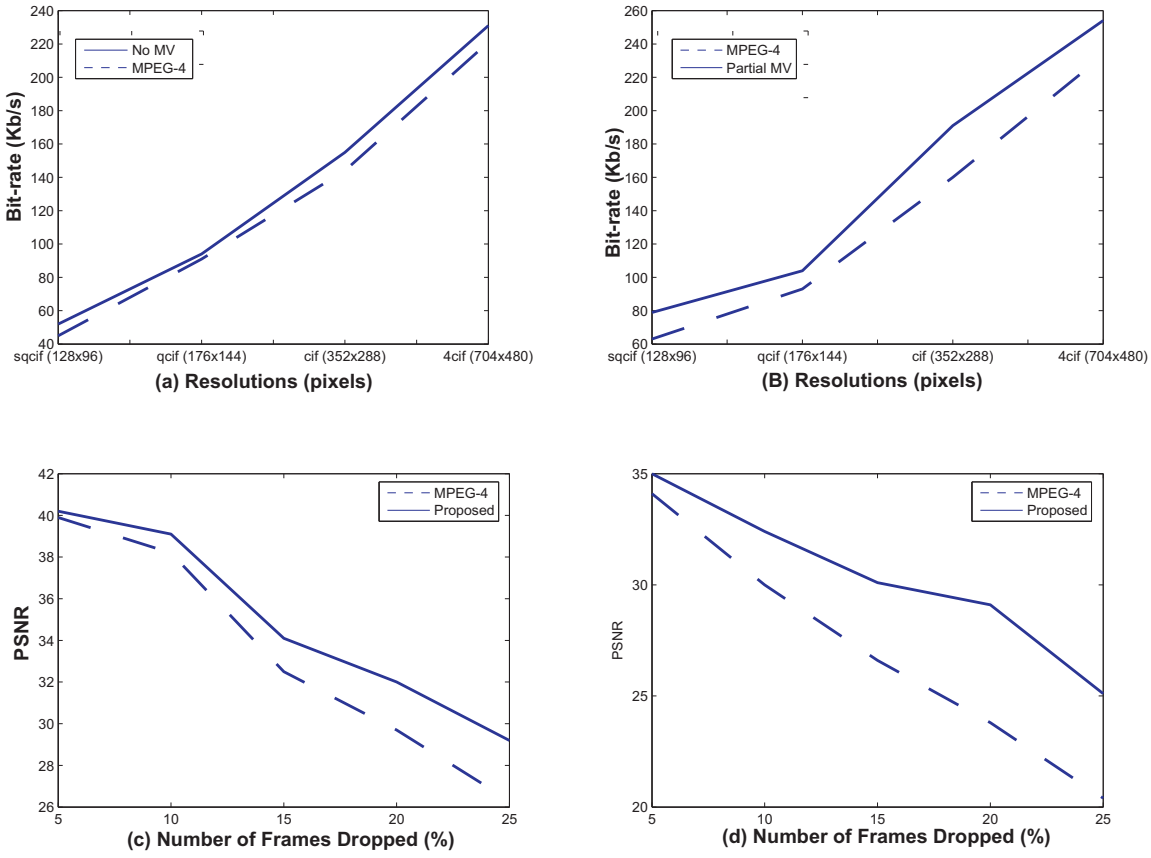


Figure 6.4: Optimizations compared against MPEG-4 (a)bit-rates achieved without MV, (b) bit-rate with partial MV, (c) PSNR with content analysis (original 25f/s), (d)PSNR with content analysis (original 15f/s)

dropping frames from video segments. A video with 800 segments (2 hour lab demonstration, 150 frames/segment) was used to conduct this experiment. In order to measure the effectiveness of the proposed method, we calculated the PSNR of the reconstructed frame at the decoder side for different percentages of frames dropped as shown in figure 6.4(c,d). Figure 6.4(c) shows the average PSNR values over the two hour video original encoded at 25f/s comparing uniform frame dropping in comparison to the proposed temporal transcoding method. For every frame dropped, the receiver reconstructs this frame by interpolating

between $frame_{t-1}$ and $frame_{t+1}$. If a segment has low motion, the hypothesis being tested is that there is a high probability that $frame_{t-1}$, $frame_t$ (that was dropped), and $frame_{t+1}$ have similar characteristics, hence the reconstruction will be accurate (in comparison to the same scenario in high motion segment. Higher PSNR values in Figure 6.4(c) demonstrate that QoS improvements can be reached when motion analysis is applied during temporal transcoding. Similarly, Figure 6.4(d) shows results for a video that was originally encoded at 15 f/s. Figure 6.4(d) shows improvements over figure 6.4(c), which shows that frames dropped from a low frame-rate video further decreases interpolation accuracy due to the dissimilarity of consecutive frames. Hence motion analysis is critical to preserve as many key frames as possible in high motion video.

Criteria 4: SNR Scalability

The following experimental results will show the admirable compression rates of the SNR scalability methodology. Section 5.1.1 demonstrated the computational complexity of the DCT transform. The DCT is symmetric to the inverse DCT (i.e. require the same amount of operations), hence by decreasing the number of operations needed at the encoder side, we also decrease the computational complexity at the decoder end. This will be beneficial for lowing the transcoding overhead at the encoder side and decreasing the decoding complexity for mobile devices with low processing power. Table 6.3 compares the number of operations needed to calculate the DCT for the proposed method, the conventional DCT, and an optimized DCT [54]. This table also shows a lower bound (only 3x3 calculations) and an upper bound (only 5x5 calculations) of the proposed method. The important result is that the proposed method reduces the computational complexity by 53% in the worst case scenario compared to the optimized DCT. In the best scenario where only a 3x3 DCT is calculated, a 61.2% upper bound improvement can be reached.

Figure 6.5(a) shows the compression rates that can be reached by the proposed SNR scalability compared to the conventional a MPEG-4 encoder, and the optimized DCT im-

plemented in an MPEG-4 encoder. The graph was obtained by taking the average bit-rates of a two hour lecture at 25f/s at different resolutions. As the figure shows, significant compression rates can be reached by the proposed method; this is crucial for user residing over bandwidth deprived networks and users with mobile devices with low processing power.

Resolution (pixels)	Proposed (3x3)	Proposed (5x5)	DCT	Fast DCT
sqcif (128x96)	41856	50688	125952	107904
qcif (176x144)	86328	114444	2599777	222552
cif (352x144)	345312	457776	1039104	890208
4cif (704x576)	9292032	12318336	27961344	23954688

Table 6.3: Comparison of the number of operations (addition multiplication) need to computation frames with different resolutions.

Criteria 5: Transcoding Start-up Delay

Real-time applications like DL systems that deliver video and audio to the end-user must stream content with negligible delay in order to produce a live lecture experience. The RTSP protocal is capable of delivering a continues flow of video over a RTP with low streaming overhead. This is accomplished by pre-buffering a portion of video at the user device to ensure that the buffer can maintain a steady flow of media to playback while the rest of the video is streamed over the entire session. Although RTSP can ensure a steady flow of media in the user device buffer, the effect of bitstream transcoding delay must also be addressed and how it effects this steady flow. Transcoding inevitably will cause delay due to the fact that video and audio must be converted from one format to another (e.g raw video to MPEG-4 format) or scaled (e.g. change frame-rate).

One of the main objectives of the proposed MV and DCT optimizations presented above is to decrease the computational complexity incurred during encoding and decoding. Figure 6.5(b) shows the amount of start-up delay saved by each individual optimization, namely DCT, using no MV, and using the partial MV approach. As the Figure shows, significant overhead can be save by each individual optimization in comparison to MPEG-4 methodology

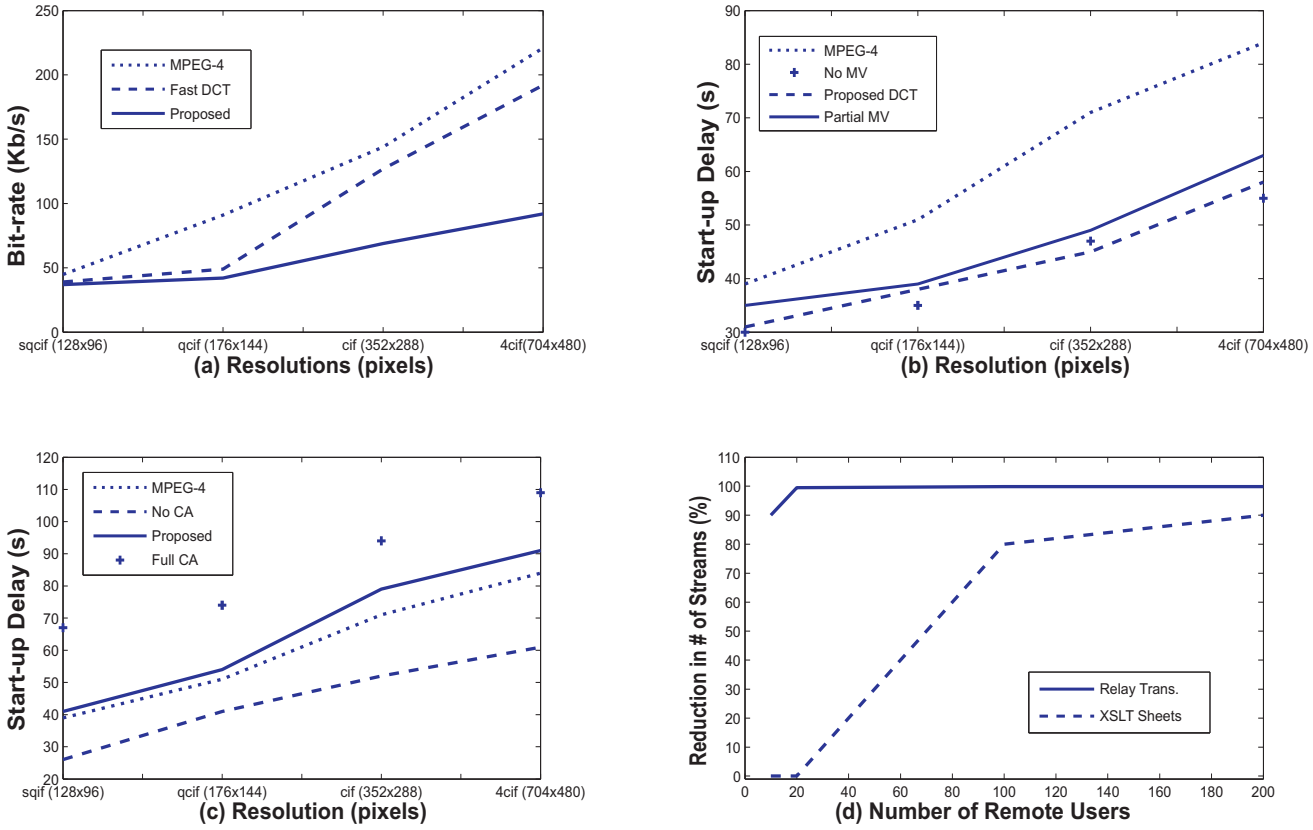


Figure 6.5: Optimizations compared against MPEG-4 (a) Compression gains in comparison also to Fast DCT, (b) Delay reductions using proposed optimizations, (c) delay incurred by entire proposed method and content analysis (CA)

for a video encoded at 25f/s. The start-up delay reduction gained by these optimizations are important to counter balance the overhead caused by motion analysis.

The proposed temporal transcoding approach would be useless unless it can be conducted with reasonable delay overhead. Figure 6.5(c) shows an approximation of start-up delays at the beginning of an DL webcasting session as a result of application layer and bitstream transcoding. These results are from real experiments using the above mentioned MPEG-4 delivery framework by streaming to a user that is 600Km away from the DL system server. The most important observation in this figure is that the proposed method that consists of the above mentioned optimizations plus the content analysis consists of practically the

same computational complexity as a MPEG-4 transcoder (same start-up delay). Hence we are able to introduce content analysis into the transcoding framework without comprising the real-time nature of DL. Other notable observations in Figure 6.5(c) is that applying a full content motion analysis (Full CA) as proposed by the MPEG-7 MAD, it would not be favorable to extract and use this metadata for real-time temporal transcoding. This is why applications use MPEG-7 descriptors/description scheme for on-demand applications whereby the MAD metadata can be extracted a priori, indexed, and used at a later time (as opposed to being conducted in real-time).

The encouraging conclusion that can be drawn from Figure 6.5(c) is that start-up delay overhead is going to be approximately under one minute. Assuming that an DL webcasting session will be on average one hour in duration (e.g. course lecture), this is an acceptable delay for the end-user to cope with. The benefits of transcoding will be worth the trade-off of a one minute start-up delay. Also note that Figure 6.5(c) is for video encoded at 25f/s, there this data serves as an upper bound on the delay incurred by the proposed method. For example, most mobile device to date decode video at 15f/s, hence, start-up delays will be less that the figures demonstrate.

Criteria 6: Hybrid Unicast/Multicast

Chapter 5 presented an overview of the tradeoffs between unicast and multicast flows with respect to bandwidth utilization and providing personalized DL content to the end user. This section will present a theoretical insight to the proposed hybrid unicast/multicast with respect to the number of streams that must be encoded by both proposed methods in comparison to a unicast approach.

Figure 6.5(d) gives an indication of the number of streams that would need to be encoded for remote users that are either in the U.S.A. or Italy. As the figure shows, using the relay transcoder approach would be effective no matter how many users are logged on remotely because this method one needs to send one stream to remote relay transcoders. However, it

must be mentioned again that the bottlenecks of this approach is the extra hardware/software needed to setup relay transcoders and the possible delay of relaying the stream from one transcoder to another as opposed to encoding and sending the stream directly to the user. The hybrid unicast/multicast approach using the (XSLT sheet) clustering algorithm mapping would not prove to be as effective as the relay transcoder methodology as shown in Figure 6.5(d). Assuming that 20 XSLT adaptation sheets are used, and assuming further that in the worst case scenario requests are mapped evenly between sheets, this method will not be affective until the number of users exceeds the number of adaptation sheets as the figure indicates. However, as the number of users grows to 100 and 200, the number of streams encoded and delivered decreases significantly to 80% and 90% respectively. The advantage of using this approach is that no extra hardware is needed and the architecture is already in place (i.e. mapping user requests to sheets).

6.3 Chapter Summary

This chapter presented results from the proposed metadata-driven transcoding framework.

The application layer transcoder proved to be effective in proving varies interface suitable for a wide range of devices. Furthermore, the application layer results demonstrated that the cache selection algorithm can be conducted in real-time and the cache update algorithm does in fact converge.

Results of the bitstream transcoder demonstrated that the addition of the proposed optimization enables the proposed framework to incorporate content analysis while providing a real-time content delivery service.

Chapter 7

Conclusions

Advances in multimedia content delivery as well as the increasing processing capability of mobile devices have created the opportunity for enriched collaborative experiences through distance learning (DL). A bottleneck hindering the DL experience is the diversity of end-users' environmental conditions such as unique device properties, heterogeneous and unreliable IP networks, and user preference. In contrast to conventional DL applications that deliver the same content to all users, this thesis proposes a metadata-driven transcoding framework whereby DL content is adapted in realtime to meet end-users' environmental needs.

The proposed metadata-driven multimedia transcoding framework utilizes a metadata-driven multimedia transcoding approach to deliver DL content to any user irrespective of their environment constraints. The descriptive power of (XML) metadata is utilized for describing, indexing, and searching for the properties of diverse devices, networks, user preferences, and the properties of the actual media to incorporate this rich data into the transcoding framework. The proposed architecture uses two types of transcoding for adaptation of DL content namely, application layer transcoding and multimedia bitstream transcoding.

Application layer transcoding is a higher level transcoding approach that adapts the media to match device processing capabilities and user preferences. It transforms metadata to

determine a presentation format by producing a browser template and transforms metadata to produce "adaptation hints" to provide an enhanced bitstream transcoding solution. The proposed metadata transcoding solution utilizes application-specific properties of DL to offer a realtime adaptation solution. In particular, our solution replaces the computationally expensive DOM processing for generation of adaptation rule with an efficient cache-based method. Experimental results indicate that the proposed solution is effective in achieving DL content personalization while achieving low delay overhead.

Bitstream transcoding is performed on the actual media to parse, transform, and truncate the underlying multimedia streams to conform to the device's processing capabilities and to adapt the encoding bit-rate to meet the network's fluctuating bandwidth capacity. The bitstream transcoder utilizes metadata provided by the application layer such as user device capabilities, the type of DL content (e.g. lecture, demonstration), and video motion analysis to provide enhanced (spatial, temporal, and SNR) adaptation. The adaptation is done by taking into account user and application specific properties. Experimental results demonstrate how content analysis can be accomplished in real-time to provide higher QoS adaptation than conventional methods.

7.1 Future Work

Several interesting problems remain to be investigated in the field of metadata-driven multimedia transcoding for DL webcasting.

- This thesis has only touched the surface with respect to content analysis. There are no limitations to how much content (or what type of) analysis can be incorporated into the proposed framework. Audio content analysis can be used to enhancing transcoding. For example, by classifying audio as speech, audio, and silent segment, this metadata can be used as adaptation hints by the bitstream transcoder as an indicator of where

to allocate more bits during encoding. This is similar to the motion analysis approach adopted by the proposed method to provide a higher QoS to the end user.

- Another important issue to address is image transcoding in order to adapt slides for different devices and bit-rates. Also, an effective method is needed to transcode slides from one format to another, which is important for mobile devices that cannot display Microsoft PowerPoint slides.

Appendix A

MPEG-21 Standard

The proposed architecture utilizes the MPEG-21 Standard's tools to achieve metadata-driven multimedia transcoding. In particular, Usage Environment Description (UED) tools and Digital Item Declaration Language (DIDL). The following sections present these tools and how they are utilized.

A.1 UED Tools

As mentioned in Chapter 2, UED tools are used to index and store metadata that describes device capabilities, network conditions, and user characteristics as well as the natural environment characteristics. Adaptation rules are chosen from the cached XSLT that best matches the constraints of the end-user based on this indexed metadata. Figure A.1 shows the *DisplayCapabilities* tools for describing parameters of the end-user device such as Resolution and color capabilities. Other tools utilized by this thesis include:

- *UserCharacteristics*: User characteristics in terms of general User information, content preference, presentation preferences, accessibility characteristics.
- *ModalityConversion*: Modality conversion preferences of a user.
- *PresentationPriority*: Presentation priority of resources according to user preference.

- *TerminalCapabilities*: Capabilities of the terminal in terms of decoding and encoding capabilities, input-output capabilities, and device properties.

- *NetworkCharacteristics*: Static and time-varying capabilities of a network.

```
<complexType name="DisplayCapabilitiesType">
 <complexContent>
  <extension base="dia:DIABaseType">
   <sequence>
    <element name="Resolution" type="ResolutionType"
     minOccurs="0" max Occurs="unbounded"/>
    <element name="SizeChar" minOccurs="0">
     <complexType>
     <attribute name="horizontal" type="integer" use="required"/>
     <attribute name="vertical" type="integer" use="required"/>
     <complexType/>
    <element/>
    more elements declared.......
   <attribute name="bitsPerPixel" type="integer" use="optional"/>
   <attribute name="colorCapable" type="boolean" use="optional"/>
   more elements declared.......
```

Figure A.1: DisplayCapabilities UEDT: Schema for describing the device display capabilities.

A.2 DIDL

The DIDL provides a flexible and standardize language for declaring and creating a DI. The section displays a sample DI describing contents of a typical lecture followed by the XSLT sheets that transforms this code to match PDA constraints for displaying slides.

Appendix B

MPEG-4 Standard

The Appendix will give an overview of block based video coding standards such as MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, and Windows Media Video. Although the intended application uses MPEG-4 as an encoding standard, the proposed transcoding solution can be applied to all the above standards. Figure B.1 shows all the components that all block-based video coding standards have in common. The following subsections will present more detail about these components.

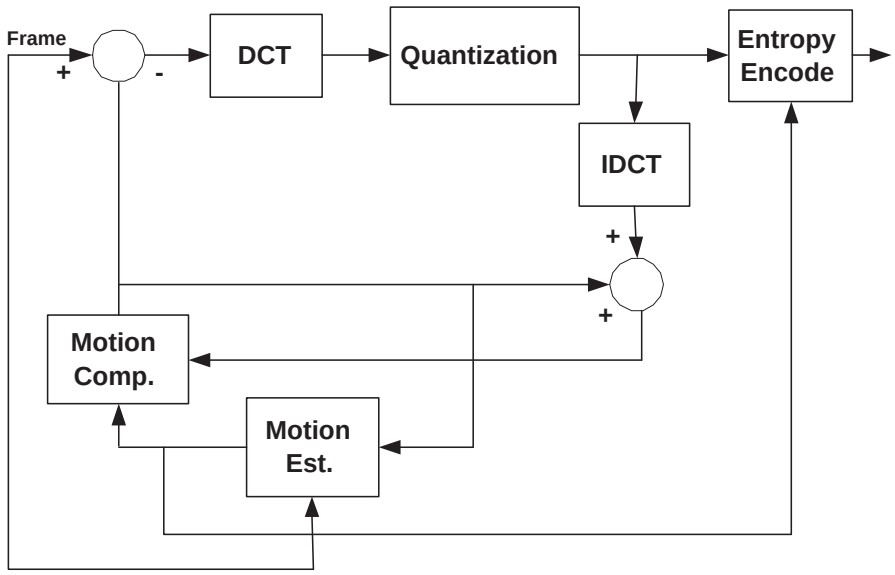


Figure B.1: Block-based video encoding system

B.1 Discrete Cosine Transform (DCT)

The DCT transforms a block of pixel values into a set of "spatial frequency" coefficients. This is analogous to transforming a time domain signal into a frequency domain signal using a Fast Fourier Transform. The DCT operates on a 2-dimensional block of pixels and produces good energy compaction in the block of values into a small number of coefficients to capture spatial redundancy between neighboring pixels. This means that only a few DCT coefficients are required to recover the original block of pixels.

An $N \times N$ image $\mathbf{G}$ can be transformed using the matrix $\mathbf{F}$ as follows

$$\mathbf{T} = \mathbf{F}\mathbf{G}\mathbf{F}^T \tag{B.1}$$

where $\mathbf{T}$ is an $N \times N$ matrix called the transform of $\mathbf{G}$. When the elements of $\mathbf{F}$ are computed using the DCT

$$\mathbf{F}_{[i,j]} = f(i,j) = \begin{cases} \sqrt{\frac{1}{N}} \cos(\frac{(2j+1)i\pi}{2N}) & i = 0, j = 0, 1, \ldots, N-1 \\ \sqrt{\frac{2}{N}} \cos(\frac{(2j+1)i\pi}{2N}) & i = 1, 2, \ldots, N-1, j = 0, 1, \ldots, N-1 \end{cases} \tag{B.2}$$

then equation (B.1) is called the forward DCT of $\mathbf{G}$. Noting that the variable i (the row number) in Equation (B.2) represents frequency, it can be seen that each row of the transform matrix $\mathbf{F}$ represents one of the one-dimensional DCT basis vectors.

Subsequently, using the IDCT to recover $\mathbf{G}$ can be expressed in terms of $\mathbf{T}$ using the inverse transform:

$$\mathbf{G} = \mathbf{B}\mathbf{T}\mathbf{B}^T \tag{B.3}$$

$$= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \mathbf{T}_{[u,v]} \mathbf{I}_{\mathbf{u},\mathbf{v}} \tag{B.4}$$

where $\mathbf{B} = \mathbf{F}^{-1} = \mathbf{F}^T$.

B.2 Quantization

The DCT allows us to represent an image in a more meaningful manner for compression (i.e., based on frequency), but does not provide compression itself. Lossy compression is generally achieved through quantization. In a DCT-based scheme such as MPEG-4, it is the DCT coefficients that are quantized so that distortion may be intelligently controlled.

In the MPEG-4 in particular, an image is divided into pixel blocks of size 8×8. The DCT is performed independently on each block and then the coefficients are quantized using a uniform midtread quantizer. The quantization step size for each of the 64 coefficient locations within the 8×8 block can be controlled by way of an 8×8 quantization matrix Q. The quantization is performed for each coefficient in each block in the following manner,

$$\mathbf{T}^q_{[u,v]} = \text{round}(\mathbf{T}_{[u,v]}/Q_{[u,v]}) \tag{B.5}$$

where the coefficient $\mathbf{T}^q_{[u,v]}$ is a coded quantized version of $\mathbf{T}_{[u,v]}$. The rounding operation does the actual quantization and the division is a normalization for step size (without it, the quantization would be fixed with a step size of 1). To get the actual quantized value (the reconstructed approximation of the original value $\mathbf{T}_{[u,v]}$ with precision determined by the quantization step size), the following operation is performed

$$\widehat{\mathbf{T}}_{[u,v]} = \mathbf{T}^q_{[u,v]} \cdot Q_{[u,v]} \tag{B.6}$$

where $\widehat{\mathbf{T}}_{[u,v]}$ is the reconstructed approximation of $\mathbf{T}_{[u,v]}$. For example, if our step size $Q_{[u,v]}$ were 3, we could get the following possible reconstructions:

$\mathbf{T}_{[u,v]}$	$\mathbf{T}^{q}_{[u,v]}$	$\widehat{\mathbf{T}}_{[u,v]}$	Absolute Error
5	2	6	1
7	2	6	1
-22	-7	-21	1
-17	-6	-18	1
39	13	39	0

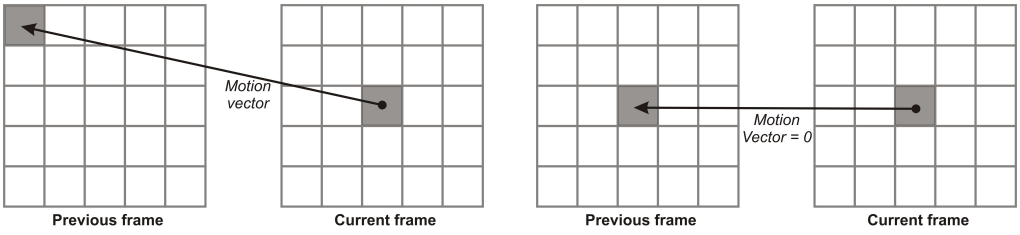
By choosing the particular quantization step sizes $Q_{[u,v]}$, the designer may control which frequency components are given greater emphasis (by using small step sizes) and which are quantized more heavily (by using large step sizes).

For a typical block of pixels, many of the coefficients produced by the DCT are close to zero. The quantizer module reduces the precision of each coefficient so that the near-zero coefficients are set to zero and only a few significant non-zero coefficients are left. This is done in practice by dividing each coefficient by an integer scale factor and truncating the result. It is important to realize that the quantizer "throws away" information.

B.3 Motion Estimation and Compensation

Another method used to decrease the encoding rate is encoding Prediction (P) frames as opposed to Intra (I) frames using motion compensation. This is accomplished by subtracting the previous transmitted frame from the current frame so that only the difference or residue needs to be encoded and transmitted. This means that areas of the frame that do not change (for example the background) are not encoded. Further reduction is achieved by attempting to estimate where areas of the previous frame have moved to in the current frame (motion estimation) and compensating for this movement (motion compensation). The motion estimation module compares each 4x4, 8x8, or 16x16 pixel block (macroblock) in the current frame with its surrounding area in the previous frame and attempts to find a match as

shown in Figure B.2(a). The matching area is moved into the current macroblock position by the motion compensator module. The motion compensated macroblock is subtracted from the current macroblock. If the motion estimation and compensation process is efficient, the remaining "residual" macroblock should contain only a small amount of information.



(a) MV calculation for macroblock matching

(b) Not calculating MV; choose macroblock in same location in prev frame

Figure B.2: Motion Compensated Prediction

B.4 Entropy Encoding

An entropy encoding (such as a Huffman or Runlength) replaces frequently-occurring values with short binary codes and replaces infrequently-occurring values with longer binary codes. The Entropy encoding is used to compress the quantized DCT coefficients. The result is a sequence of variable-length binary codes. These codes are combined with synchronization and control information (such as the motion "vectors" required to reconstruct the motion-compensated reference frame) to form the encoded MPEG-4 bitstream.

Appendix C

MPEG-7 Standard

C.1 MPEG-7 Motion Descriptors

The MPEG-7 Standard finalized four motion descriptors that capture complementary motion features. These descriptors are called Motion Activity, Camera Motion, Motion Trajectory, and Parametric Motion [28]. The Motion Activity Descriptor (MAD) captures the overall activity of motion in a video segment (i.e. the level of action in the segment). The Camera Motion Descriptor (CMD) describes the motion of the camera in a video segment. The Motion Trajectory Descriptor (MTD) captures the overall displacement of a region, in particular, the evolution of its center positioning in time and space. The Parametric Motion Descriptor (PMD) deals with the lower level features of the MTD to provide more detailed description about the region displacement.

This project will utilize the MAD in order to perform video transcoding. A human perceives segments of a video as slow, medium, and fast action segments. For example, in a news clip, watching the news caster speak is perceived as a slow clip, but the video clip of the news story (e.g. a goal being scored) can be perceived as a fast clip. In other words, a video entails of sequences of high and low activity. The MAD extracts and expresses the activity of these video segments. It can be used in video transcoding to make intelligent frame dropping decisions when the allowable bitrate encoding rate drops due to a drop in

network bandwidth.

C.1.1 Extracted Parameters of the MAD

The MAD extracts and encodes the following parameters:

1. **Intensity of Activity:** This is an integer lying in the range of 1-5. A value of 5 indicates a high level of activity and the value of 1 indicates the lowest level of activity.

2. **Direction of Activity:** This specifies the dominant direction of objects in the scene. This is a three-bit integer representing eight equally spaced directions.

3. **Spatial distribution of activity:** This parameter indicates whether the activity is spread across many regions or restricted to one region. It gives a value for the number and size of the active regions in a frame. For example, a news casters speaking would have 1 large active region while an aerial shot of city would have many small active regions. This parameter is expressed by three integers (with 16 bits).

4. **Temporal distribution of activity**: This parameter expresses the variation of activity over the duration of the video segment. It gives an indication of whether the activity is sustained throughout the duration of the segment of whether it is contained in part of the segment. For example, if the segment is 100 frames, the activity might only be contained in the the first 50 frames. This is expressed in 6 bits.

Note that this descriptor can be expressed by only the first (intensity of activity), or with all parameters. Hence this compact descriptor can be encoded from 3-52 bits. XML metadata code expresses the structure of the MAD encoding, which facilitates extensibility and metadata transformations. This thesis will only utilize the intensity of activity and spatial distribution activity parameter for simplicity of implementation.

C.1.2 Intensity Extraction of the MAD

Ground Truth: The intensity of motion is a subjective measure. In light of this fact, ground truth [28] is used as a fidelity measure for the MAD intensity measure. Hence, the performance of intensity of motion activity of the MAD is accessed by finding the fidelity to the ground truth. The MPEG-7 test set consists of 637 video segments used to produce this ground truth. The segments are a few second long and cover a wide range of genre (news, sports, action, drama, talk shows, etc.). Human subjects are used to rate the intensity of motion activity of each of the segments(range of 1 to 5). Then, the average rating across the subjects were used to produce the ground truth.

Measure of Motion Activity: The MPEG-7 Standard uses the fact that motion-vector magnitude is an indication of the magnitude of the motion itself and use statistical properties of the motion-vector magnitude of macroblocks to measure intensity of motion activity [28]. They found that the standard of deviation and the average of the motion-vector magnitude approximately matched the ground truth after quantization. They also discovered that the standard deviation of the motion-vector magnitude provides a better approximation of the ground truth. In light of this fact, the use the quantized standard deviation of the motion-vector magnitude to compute the intensity of motion activity. Table 1 shows the thresholds used for quantization of the standard deviation.

Activity Value	Range of σ
1	$0 \leq \sigma < 3.9$
2	$3.9 \leq \sigma < 10.7$
3	$10.7 \leq \sigma < 17.1$
4	$17.1 \leq \sigma < 32$
5	$32 \leq \sigma$

Table C.1: Standard deviation of motion vector magnitude

Bibliography

[1] A. Pahwa, D. Gruenbacher, S. Starrett, and M. Morcos, "Distance learning for power professionals: virtual classrooms allow students flexibility in location and time," *IEEE Power and Energy Magazine*, vol. 3, no. 1, pp. 53–58, January.

[2] R. Baecker, G. Moore, and A. Zijdemans, "Reinventing the lecture: Webcasting made interactive," in *Proceedings of HCI International*, vol. 1. Lawrence Erlbaum Associates, 2003, pp. 896–900.

[3] J. Bormans, J. Gelissen, and A. Perkis, "MPEG-21: The 21st century multimedia framework," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 53–62, 2003.

[4] P. van Beek, "Metadata-driven multimedia access," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 40–52, 2003.

[5] S. Deshpande and J. Hwang, "A real-time interactive virtual classroom multimedia distance learning system," *IEEE Transactions on Multimedia*, vol. 3, no. 4, pp. 432–444, 2001.

[6] D. Wu, Y. Hou, W. Zhu, Y.-Q. Zhang, and J. Peha, "Streaming video over the Internet: approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, 2001.

[7] D. Wu, Y. Hou, and Y. Zhang, "Transporting real-time video over the internet: Challenges and approaches," in *IEEE Transactions on Circuits and Systems for Video Technology*, March 2001.

[8] F. Pereira and I. Burnett, "Universal multimedia experiences for tomorrow," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 63–73, 2003.

[9] C. Timmerer, "Resource adaptation using XML with the MPEG-21 multimedia framework," Dissertation, Institut fur Informaionstechnologie, Universitat Klagenfurt, Germany, 2003.

[10] "Information technology-multimedia framework MPEG-21-part1: Vision, technology and strategy," ISO/IEC, Tech. Rep. TR 21000-1:2004, 2004.

[11] "Information technology-multimedia framework MPEG-21-part7: Digit item adaptation," ISO/IEC, Tech. Rep. TR 21000-7:2004, 2004.

[12] I. Burnett, R. V. de Walle, K. Hill, J. Bormans, and F. Pereira, "MPEG-21: goals and achievements," *IEEE Multimedia*, vol. 10, no. 4, pp. 60–70, 2003.

[13] S. Chang, T. Sikora, and A. Purl, "Overview of the mpeg-7 standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 12–36, June.

[14] K. Cha and S. Kim, "Adaptive scheme for streaming MPEG-4 contents to various devices," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1061–1066, 2003.

[15] M. Almaoui and K. Plataniotis, "Scalable e-learning multimedia adaptation architecture," in *Springer-Verlog Lecture Notes in Computer Science*, September 2005.

[16] M. Almaoui, A. Kushki, and K. Plataniotis, "Adaptive multimedia transcoding using cached xslt sheets," in *22nd Biennial Symposium on Communications*, May 2004.

[17] (2000, October) Extensible markup language (xml) 1.0 (second edition). [Online]. Available: http://www.w3.org/TR/REC-xml, W3C Recommendation

[18] (2001, October) Extensible stylesheet language (xsl) version 1.0. [Online]. Available: http://www.w3.org/TR/xsl

[19] Mpeg: Moving pictures expert group. [Online]. Available: http://www.chiariglione.org/mpeg/

[20] "Information technology-coding of audio-visual objects," ISO/IEC, Tech. Rep. 14496-2:2001, 2001.

[21] "Information technology-multimedia content description interface," ISO/IEC, Tech. Rep. 15938-3:2001, 2001.

[22] A. Kinno, Y. Yonemoto, M. Morioka, and M. Etoh, "Environment adaptive XML transformation and its application to content delivery," in *Proceedings of the Symposium of Applications and the Internet*, 2003, pp. 31–36.

[23] "Information technology-multimedia framework (mpeg-21)-part 2: Digital item declaration," ISO/IEC, Tech. Rep. TR 21000-2:2004, 2004.

[24] Y. Wang, Z. Liu, and J. Huang, "Multimedia content analysis using both audio and visual clues," *IEEE Signal Processing Magazine*, vol. 17, no. 6, pp. 12–36, November.

[25] J. Magalhes and F. Pereira, "Using MPEG standards for multimedia customization," *Signal Processing: Image Communication*, vol. 19, no. 5, pp. 437–456, 2004.

[26] J. Pan, C. Tan, and W. Lee, "Context-aware service protocol. an extensible and configurable framework for user context awareness services in pervasive computing systems,"

in *Proceeding of the IEEE Wireless Communications and Networking*, vol. 3, 2003, pp. 2058–2063.

[27] N. Wakamiya, M. Miyabayashi, M. Murata, and H. Miyahara, "Dynamic quality adaptation mechanisms for tcp-friendly MPEG-4 video transfer," in *Second International Workshop on Quality of Service in Multiservice IP Networks*, 2003, pp. 539–550.

[28] S. Jeannin and A. Divakaran, "MPEG-7 visual motion descriptors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 720–725, 2001.

[29] J. Brotherton, J. Bhalodia, and G. Abowd, "Automated capture, integration, and visualization of multiple media streams," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1998, pp. 54–63.

[30] H. Tsang, L. Hung, and S. Ng, "A mulitmedia distance learning system on the internet," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, 1999, pp. 243–246.

[31] C. Bouras, P. Destounis, J. Garofalakis, and et. al., "Efficient web-based open and distance learning services," *Journal of Telematics and Informatics*, vol. 17, no. 3, pp. 213–237, 2000.

[32] L. Rowe, D. Harley, P. Pletcher, and S. Lawrence, "Bibs: A lecture webcasting system, Tech. Rep. FCD 21000-1:2007, 2001.

[33] J. Padhye and J. Kurose, "Continuous-media courseware server: A study of client interactions," *IEEE Internet Computing*, pp. 65–73, 1999.

[34] E. Cloete, "Electronic education system model," *Computers and Education*, vol. 36, pp. 171–182, 2001.

[35] F. Cirillo, A. Cozzolino, M. D. Santo, M. Marsella, and S. Salerno, "A metadata based distance learning platform," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, 2000, pp. 44–48.

[36] T. Lemlouma and M. Layaida, "Adapted content delivery for different contexts," in *Proceedings of the Symposium of Applications and the Internet*, January 2003.

[37] M. Almaoui, A. Kushki, and K. Plataniotis, "Metadata-driven multimedia transcoding for distance learning," *Springer-Verlog Multimedia Systems Journal*, p. Submitted, January.

[38] B. Preiss, *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. University of Waterloo, Waterloo: Wiley, 1999.

[39] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," *Lecture Notes in Computer Science*, vol. 1973, pp. 420–434, 2001.

[40] J. Friesen and T. Tarman, "Remote high-performance visualization and collaboration," *IEEE Computer Graphics and Applications*, vol. 20, no. 4, pp. 45–49, 2000.

[41] "Information technology- h.264: Advanced video coding (avc)," ISO/IEC, Tech. Rep. 14496-10:2004, 2004.

[42] G. Ashraf and M. Chong, "Performance analysis of h.261 and h.263 video coding algorithms," *in Proceedings of 1997 IEEE International Symposium on Consumer Electronics*, pp. 153–156, December.

[43] S. Srinivasan, J. Pohsiang, and T. Holcomb, "Windows media video 9: overview and applications," *Signal Processing: Image Communication*, vol. 19, no. 9, pp. 851–875, October.

[44] J. Herre and B. Grill, "Overview of MPEG-4 audio and its applications in mobile communications," in *Proceedings of 5th International Conference on Signal Processing*, 2000, pp. 604–613.

[45] Ffmpeg mpeg-4 encoder. [Online]. Available: http://sourceforge.net/projects/ffmpeg/

[46] X. Sun, A. Divakaran, and B. Manjunath, "A motion activity descriptor and its extraction in compressed domain," in *Proceedings of the IEEE Pacific-Rim Conference on Multimedia (PCM '01)*, October 2001.

[47] K. Lepold, "Quality controlled temporal video adaptation," Dissertation, Institut fur Informaionstechnologie, Universitat Klagenfurt, Germany, 2003.

[48] V. Thomas, J. Blin, and M. Hias, "Experiments on visibility thresholds of dct coefficients," in *Proceedings Picture Coding Symposium, Italy*, September 1988.

[49] N. Cranley, L. Murphy, and P. Perry, "Video adaptation: User-perceived quality-aware adaptive delivery of MPEG-4 content," in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, 2002, pp. 42–49.

[50] N. Cranle, P. Perry, and L. Murphy, "Perceptual quality adaptation (PQA) algorithm for 3GP and multi-tracked MPEG-4 content over wireless ip networks," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2004.

[51] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM*, 2000.

[52] Windows media 9 series. [Online]. Available: http://www.microsoft.com/windows/windowsmedia

[53] "Apple's open-source darwin streaming server," http://developer.apple.com/darwin/projects/streaming/.

[54] N. Cho and S. Lee, "Fast algorithm and implementation of 2-d discrete cosine transform," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 3, pp. 296–305, March.