Video adaptation for stored video delivery over resource-constrained networks

Tao, DaYong

2007

Tao, D.Y. (2012). Comparison of two methods of extracting carbohydrates from microalgae.
Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2465>

<https://doi.org/10.32657/10356/2465>

Nanyang Technological University

Downloaded on 26 Apr 2023 20:26:19 SGT

Video Adaptation for Stored Video Delivery over Resource-Constrained Networks

Tao DaYong

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2007

Acknowledgments

I am deeply grateful to my supervisor Dr. Cai Jianfei for giving me the opportunity to work under his research group. I would like to thank him for his guidance and advice for my research work over the past year. Without his encouragement and valuable feedback, I would not have completed this thesis and achieved so much.

I would also like to thank Dr. Andreas J. Kassler for his initiation of the project. His clear idea and explanation of the key issues help me to learn new things quickly and effectively.

Special thanks to Professor Chia Liang Tien and the technicians in the Center for Multimedia and Network Technology (CeMNet) laboratory for their kind assistance.

Last but not least, I would like to thank all my friends and colleagues in CeMNet for their support and help.

Tao DaYong

Contents

Acknowledgments	i
List of Figures	v
List of Tables	vii
Abstract	viii
1 Introduction	1
1.1 Research Background	1
1.1.1 Universal Multimedia Access	1
1.1.2 MPEG-21 Multimedia Framework	3
1.1.3 Video Adaptation	4
1.1.4 Scalable Video Coding	6
1.1.5 Bandwidth Smoothing	7
1.2 Problem Statement and Research Objective	9
1.3 Summary of Contributions	9
1.4 Outline of the Thesis	10
2 FGS Video Adaptation within MPEG-21	11
2.1 Overview of MPEG-21	12
2.2 MPEG-21 Part 7: Digital Item Adaptation	17
2.2.1 Usage Environment Description (UED)	17
2.2.2 BSDLink	19
2.2.3 Bitstream Syntax Description (BSD)	19
2.2.4 Terminal and Network Quality of Service (AQoS)	22
2.2.5 Universal Constraints Description (UCD)	23
2.2.6 Metadata Adaptability	23

2.2.7	Session Mobility	23
2.2.8	DIA Configuration	24
2.2.9	BSD-Based Adaptation Architecture	24
2.2.10	MPEG-21 DIA Current Status	25
2.3	FGS Video Adaptation Framework	27
2.3.1	FGS Video Coding Structure	27
2.3.2	Generating BSD and gBSD	28
2.3.3	Making Decisions	33
2.3.4	Transforming gBSD	35
2.3.5	Adapting Bitstream	37
2.4	Summary	39
3	Optimal Frame Selection under Constrained Resources	40
3.1	Introduction	40
3.1.1	VBR Video	40
3.1.2	Traffic Model	41
3.1.3	Workahead Smoothing	43
3.1.4	Online Smoothing	47
3.1.5	Lossy Smoothing	47
3.2	Proposed Optimal Frame Selection Algorithm	48
3.2.1	Motion Representation	49
3.2.2	Problem Setting	52
3.2.3	Channel Model	54
3.2.4	Non-optimal States	56
3.2.5	Virtual States	57
3.2.6	Optimal Preload	59
3.3	Simulation Results	61
3.3.1	Other Algorithms	61
3.3.2	Performance Evaluation	63
3.3.3	Effectiveness of Non-optimal State Elimination	65
3.4	Summary	66

4	Reverse Frame Selection under Dynamic Constrained Resources	67
4.1	Introduction	67
4.2	RFS for CBR Channels	69
4.2.1	RFS for VBR Channels	71
4.3	Simulation Results	73
4.3.1	Simulation for CBR Channels	73
4.3.2	Simulation for VBR Channels	80
4.4	Summary	83
5	Conclusion and Future Work	84
5.1	Conclusion	84
5.2	Future work	85
	References	86
	Publications	97

List of Figures

1.1	The UMA concept	2
1.2	General conceptual framework for video adaptation and associated concepts of resources and utility	5
2.1	Digital Item Adaptation architecture	15
2.2	Bitstream Syntax Description based adaptation architecture.	20
2.3	DIA description tools for content adaptation.	25
2.4	FGS scalability structure.	27
2.5	FGS bitstream syntax structure (the first FGS VOP contains 4 bitplanes, and second FGS VOP contains 5 bitplanes).	28
2.6	The video quality of foreman (at frame number = 60) after truncating bitplanes with “nBP” equals (a) 0.00, (b) 0.88, (c) 1.78, and (d) 2.49 respectively.	39
3.1	A feasible smoothing schedule with minimum peak rate and variability. .	45
3.2	Motion curve for (a) akiyo, (b) foreman, (c) mobile, and (d) stefan. . . .	51
3.3	The system setting for optimal frame selection.	52
3.4	The CBR channel model.	55
3.5	An illustration of a non-optimal state	57
3.6	The CBR channel model with virtual states.	58
3.7	Obtaining the optimal preload through GR and LR.	60
3.8	The Z-B algorithm with frame size stuffing.	62
3.9	The selection results for (a) akiyo, (b) foreman, (c) mobile, and (d) stefan. .	64
3.10	The effectiveness of non-optimal state elimination for stefan.	65
4.1	Reverse frame selection (a) in normal mode, (b) in mirrored mode.	70

4.2	Optimal path switching for video streaming over VBR channels.	72
4.3	Frame selection results at different preloads at buffer = 512 kbytes for (a) accumulated motion metrics (b) total number of selected frames, and at buffer = 1024 kbytes for (c) accumulated motion metrics (d) total number of selected frames	74
4.4	Frame Stuffing vs Number of Frame States.	75
4.5	Frame selection results under different bandwidth conditions for (a) buffer = 512 kbytes (b) buffer = 1024 kbytes.	78
4.6	Frame selection results for different buffers.	79
4.7	Frame selection results over channel 1 for (a) buffer = 512 kbytes (b) buffer = 1024 kbytes, and channel 2 for (c) buffer = 512 kbytes (d) buffer = 1024 kbytes.	81

List of Tables

2.1	MPEG-21 parts	13
2.2	Bitstream and description size properties for foreman	32
2.3	Properties of foreman video after adaptation at different bandwidth . . .	38
3.1	The properties of the two MPEG-4 video traces.	63
4.1	Gain Sensitivity to Frame Stuffing	76
4.2	Effectiveness of Non-optimal State Elimination at Buffer = 512K	77
4.3	Effectiveness of Non-optimal State Elimination at Buffer = 1024K	77
4.4	Number of State Ranges per Frame (stuffing = 200 bytes)	83

Abstract

Multimedia content delivery is subject to different user preferences, dynamic changing network conditions, and terminal devices with varying capabilities. To access rich multimedia content in such a heterogeneous environment, adaptation has to be performed along the delivery and consumption chain in an interoperable manner. This project studies content adaptation and bandwidth smoothing techniques for video streaming over resource-constrained networks. The objective is to provide end-to-end QoS management by adapting content to different terminal and network capabilities.

In this thesis, we present two methods for video delivery over resource-constrained networks: content adaptation within the MPEG-21 multimedia framework and optimal frame selection using bandwidth smoothing technique. For the content adaptation approach, we first give an overview of the various parts defined in the MPEG-21 standard, especially part 7 – Digital Item Adaptation (DIA). We then discuss several use cases for temporal-SNR scalable FGS video adaptation. Using the description tools defined in the DIA standard, adaptation decisions to maximize bandwidth utilization or content information flow can be easily obtained.

The bandwidth smoothing approach presents an optimal frame selection scheme based on dynamic programming. Given channel bandwidth and client buffer size, the selected optimal frame path can maximize user's perceptual quality while being guaranteed transmittable and playable. Unlike existing bandwidth smoothing techniques, the proposed scheme examines inter-frame dependencies and uses motion information to characterize frame importance. In addition, we propose a novel reverse frame selection scheme that can find all optimal results for different start-up delays given bandwidth and buffer size constraints. We further extend the reverse frame selection scheme to solve the problem of video delivery over variable bitrate (VBR) channels.

Chapter 1

Introduction

1.1 Research Background

The creation and consumption of multimedia resources have been increasing at a rapid pace over the past decade. More and more sophisticated access devices, such as mobile phones and personal digital assistants (PDA), are equipped with multimedia-enabled functionalities. However, difficulties arise for providing these devices seamless access to rich multimedia content because not only each device has different processing and presentation capabilities but also they are connected to networks characterized by different conditions, ranging from cellular, satellite to cable and optical communication channels. In addition, different users often have different preferences on how the content information should be presented. Given such a heterogeneous usage environment, it is necessary to dynamically adapt content to user's capabilities, conditions, and preferences along the resource delivery and consumption chain.

1.1.1 Universal Multimedia Access

To address this issue, the Universal Multimedia Access (UMA) [1, 2, 3] concept emerges for the next generation of multimedia applications especially for the mobile and wireless communication systems. "Universal" means seamless access to any content from anywhere at anytime. The content may be an adapted version of the original one due to

Figure 1.1: The UMA concept

environment constraints. Figure 1.1 illustrates the UMA concept. Each terminal can access to the media server's resources through wired or wireless networks and receive customized content according to its network conditions, device capabilities and personal preferences.

One way to provide such personalized service is by creating multiple formats (e.g., video, audio, image, and text) and qualities of the same content [4]. Upon receiving a request from a user, the server can make appropriate selection from multiple bitstreams to meet the user's environment constraints while maximizing quality of service (QoS). However, creating content in multiple formats and qualities is expensive and highly storage-intensive. It may be suitable for only limited known types of terminals and networks. For general and realtime applications, dynamic content adaptation is necessary. In [5], Pereira *et al.* vision that the future trend is to move from UMA to Universal Multimedia Experience (UME) because the ultimate consumer is the user and not the terminal. Personalized content can certainly improve user experience [6]. An example of UME is to include sign language window into video for deaf people.

1.1.2 MPEG-21 Multimedia Framework

Established in 1988, the Motion Picture Experts Group (MPEG) is well known for its pioneering work in creating the MPEG-x standards, namely MPEG-1/-2/-4/ and -7 (see <http://www.chiariglione.org/mpeg>). These standards define an extensive set of tools covering from audio-visual compression and transport to content description. Individual technologies are being created to meet the needs of different user communities. However, there lacks a big picture to describe how these technologies and standards fit together. No end-to-end solutions exist to allow different user communities to interact in an interoperable way [7]. The loose collection of technologies and standards is stalling the deployment of large-scale multimedia applications [8].

In June 2000, MPEG starts to work on a normative technology called MPEG-21, formally referred as ISO/IEC 21000 Multimedia Framework, to enable multimedia applications for the 21st century [7]. MPEG-21 is an open standards-based framework for multimedia delivery and consumption specifically taking into account digital rights management (DRM) and heterogeneity of terminals and networks. The vision is to enable transparent and augmented use of multimedia resources across a wide range of networks and devices for different user communities [9]. MPEG-21 intends to cover the entire value chain encompassing content creation, production, management, transport, distribution and consumption. It is a significant step toward achieving UMA [10].

MPEG-21 is based on two essential concepts [11]: the definition of a fundamental unit of distribution and transaction called the Digital Item (DI) and the concept of Users interacting with DIs. A digital item is a structured digital object with a standard representation, identification, and associated metadata within the framework [12]. In practice, a DI is a combination of resources, metadata, and structure. The resources are the individual assets or content such as videos and images. The metadata describe the

DI or individual resources. And the structure establishes the relationships among the parts of the DI, both resources and metadata.

A user is any entity that interacts with a DI in the context of MPEG-21 environment. Such users include individuals, consumers, communities, organizations, corporations, consortia, and governments. Technically, all parties interacting within the MPEG-21 framework are treated equally as users. Users are identified by their relationships to another user during an interaction. At the most basic level, MPEG-21 can be seen as providing a framework in which one user interacts with another user and the object of that interaction is a digital item. Some of these interactions are creating content, modifying content, delivering content, and consuming content.

1.1.3 Video Adaptation

Adaptation is emerging as an important tool for pervasive media applications. Hellwagner *et al.* [13] define adaptation as the capability of a system to dynamically change its behavior to keep the QoS above a certain level. Chang *et al.* [14] regard video adaptation as a tool or system that adapts one or more video programs to generate a new presentation with a video or multimedia format to meet user needs in customized situations. The authors in [14] propose a framework to unify and explore various issues and approaches in the video adaptation field (see Figure 1.2). An entity, which can be in the form of pixel, object, frame, shot, scene, syntactic components, or semantic components, is the basic unit that undergoes the adaptation process. An entity is associated with certain resource requirements (transmission bandwidth, picture resolution, and color depth) and utility values (user satisfaction of the content such as the peak signal-to-noise ratio (PSNR)). An adaptation operation transforms an entity and changes its associated resources and utility values. Thus video adaptation can be formulated as: given entity e and resource constraints C_R , find the optimal adaptation operation a_{opt} so that the utility value u_{max} of the adapted entity e' is maximized.

Figure 1.2: General conceptual framework for video adaptation and associated concepts of resources and utility

In the past few years, we have seen extensive research works on video adaptation [14, 15, 16]. The objective of video adaptation is to intelligently remove some information from the video signal itself so that end-to-end resource requirement can be reduced. Generally, video adaptation can be implemented in three ways. The first approach is to store many non-scalable bitstreams for each video sequence. Each bitstream is coded at different formats or different spatial/temporal/SNR (signal-to-noise ratio) resolutions. When a user requests access to the video sequence, the server can send the bitstream which is the closest to the user's requirements. Although this method usually costs more storage space in the video server and the chosen bitstream may not satisfy the user's requirement exactly, it is widely used in practical systems due to its simplicity. Recent research works such as [17] focus on how to efficiently switch among multiple non-scalable bitstreams in order to dynamically adapt to the time-varying network conditions. The second approach is to perform video transcoding [15] including decreasing the spatial resolution, reducing the SNR, or down sampling the temporal frame rate through re-encoding, re-quantization, and frame dropping etc. Although transcoding is very flexible and does not require extra storage space, it needs complex extra processing and is not suitable for large-scale diverse users. The third approach is to use scalable video coding

(SVC) [16], which has inherent ability to be adapted to different requirements. SVC schemes aim to encode a video once and decode it at multiple reduced rates and resolutions to provide simple and flexible adaptability.

Currently, adaptation approaches have been focusing on the following areas: semantic event-based adaptation, structural-level adaptation, video transcoding, and video skimming [14]. The semantic event-based adaptation approach involves content analysis, for example, the scoring moment in a soccer video, and thus the performance gain is very much dependant on the video content. In [18], Lim *et al.* present the usage of semantic information of user's region of interest (ROI). Resources such as videos or images are adapted based on the user's semantic ROI information so that the visual and graphical focus of attention can be best delivered to the user. The structure-level adaptation is often used for summarizing video through key frame extraction where the video structures are caused by event occurrence orders, camera control patterns, and the editing process [14]. In [19], Irani *et al.* present a structural level mosaic representation for video indexing. Continuous pictures captured by camera are transformed into a panoramic view. In addition, error-resistant video transcoding has also been studied to provide robust transmission for wireless applications [20]. Video skimming or rapid fast-forward is used when users want to finish viewing a video within a limited amount of time.

1.1.4 Scalable Video Coding

Scalable video coding provides efficient adaptation by simply discarding segments from the bitstream [16, 21]. The basic idea is to encode video at the highest resolution possible and decode it into reduced bit rate, frame rate, and spatial resolution depending on application constraints and requirements. Ideally, scalability should be achieved without any degradation in coding efficiency [15]. In the past few years, several scalable video coding schemes [22, 23, 24, 25, 26] have been proposed. For these scalable video coding

schemes, a video sequence is usually coded into a base layer and several enhancement layers. A better video quality can be obtained only when an enhancement layer is completely received and decoded.

Fine Granularity Scalability (FGS) [27] is a new scalable coding scheme adopted in the MPEG-4 standard. The basic idea of FGS is to encode a video sequence into a base layer and an enhancement layer. The base layer is coded with high efficiency but without scalability. The enhancement layer is bitplane coded after DCT transformation so that the enhancement layer bitstream can be truncated anywhere to provide fine-grained scalability. There is no inter-frame prediction in the FGS enhancement layer, which eliminates successive frame error propagation at the cost of low coding efficiency. In addition, FGS temporal scalability or FGST has been proposed to provide hybrid temporal-SNR scalability for best-effort internet video streaming [28, 29].

A fully scalable video can simultaneously provide temporal, spatial, and quality (SNR) scalability. Motion-compensated wavelet coding schemes make possible full scalability with fine granularity over a large range of bitrates [16]. Recent advances in motion-compensated temporal filtering (MCTF) have shown that combination with arbitrary motion compensation can achieve compression performance comparable with state-of-the-art single-layer coding schemes targeting for only one rate [30].

1.1.5 Bandwidth Smoothing

A video sequence can be encoded with either of the two compression modes [31]: constant bitrate (CBR) or variable bitrate (VBR). The CBR mode provides easy network management because of its predictable traffic patterns. The VBR mode offers better video quality for the same average bit rate by avoiding the need to adjust quantization parameters as in CBR [31]. Besides, VBR encoding offers greater opportunity for statistical multiplexing gains [32]. This has led to great interest in developing VBR coding

CHAPTER 1. INTRODUCTION

schemes. However, the VBR video traffic is difficult to manage because of its significant bitrate burstiness over multiple time scales [33]. While the short-term burstiness is due to coding structure algorithms (e.g., I, P, and B frames in MPEG videos), the long-term burstiness is due to scenes with significant motion or rapid chrominance and luminance changes such as those created by flashing lights [34]. The high peak and bursty bitrate will substantially increase bandwidth requirement for continuous video playback.

Bandwidth smoothing is a promising technique for reducing rate variability of VBR bitstreams. There are three basic smoothing techniques [35]:

- (i) Temporal multiplexing introduces a per-stream buffer in the end-to-end path to smooth individual stream's peak rate at the cost of increased delay.
- (ii) Statistical multiplexing gives collective multiplexing gain by sharing a network channel with multiple independent streams.
- (iii) Workahead smoothing sends video data ahead of schedule to reduce the variability of transmission rates, which is subject to two constraints: a) the data are available to be sent, and b) the client buffer space is sufficient to receive it.

Among these smoothing techniques, workahead smoothing [36, 37, 38] takes advantage of the *a priori* information in the encoded video such as frame sizes to create a transmission plan. The transmission plan which consists of a sequence of transmission rates controls the video delivery process. The basic idea is to build up buffer by sending more data than needed over small frame segments and drain buffer during bursts. By prefetching data ahead of each burst, large frames can be transmitted earlier at a slower rate to reduce peak bandwidth requirement.

1.2 Problem Statement and Research Objective

QoS is hard to achieve for content delivery in heterogeneous environments. Adaptation needs to be performed along the resource delivery chain in an interoperable manner. One objective of this research is to design a video streaming system with end-to-end QoS management by adapting content to different terminal capabilities, network conditions and user preferences. Additionally, VBR video exhibits significant bitrate burstiness over multiple time scales. Making use of the *a priori* information in a stored video, smoothing techniques can reduce bandwidth requirement. Another objective of the report is to design an optimal frame selection scheme given bandwidth and buffer constraints.

1.3 Summary of Contributions

This thesis mainly focuses on stored video adaptation under resource-constrained networks. The major contributions of the report are as follows:

- (i) Development of a video adaptation framework based on MPEG-21 Digital Item Adaptation (DIA) architecture. In particular, using the description tools defined in the DIA standard, optimal adaptation decisions that maximize bandwidth utilization or content information flow can be easily obtained.
- (ii) Development of an optimal frame selection algorithm based on dynamic programming. Given bandwidth and buffer size, our proposed optimal frame selection algorithm is able to find the optimal set of frames that can maximize the video's accumulated motion information flow.
- (iii) Development of a novel reverse frame selection algorithm that can find all optimal frame selection results for different preloads in one round given the bandwidth and buffer size constraints. In addition, we extend the proposed algorithm to solve the problem of video streaming under VBR channel conditions.

1.4 Outline of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 introduces the MPEG-21 multimedia framework. We start with an overview of the various parts defined in the standard. We then present several use cases for FGS video adaptation using the MPEG-21 Digital Item Adaptation architecture.

Chapter 3 presents an optimal frame selection algorithm for delivering VBR video over resource-constrained networks. The problem is first formulated using bandwidth smoothing technique. An optimal frame selection scheme based on dynamic programming is then proposed to maximize the accumulated motion information flow given bandwidth and buffer constraints.

Chapter 4 presents a novel reverse frame selection algorithm that is able to find all optimal results for different preloads in one round for different start-up delay requirements. In addition, the proposed algorithm is extended to solve the problem for VBR channels.

Chapter 5 concludes the thesis with a summary of the studies. Additionally, we give suggestions for future work directions.

Chapter 2

FGS Video Adaptation within MPEG-21

This chapter is organized into three parts. The first part gives an overview of the MPEG-21 framework. MPEG-21 consists of a series of independent standards with each standard created as a separate part in the framework to allow various technologies to be useful as stand-alones and thus used as much as possible [11, 39]. MPEG-21 aims to define an integrated multimedia framework so that when the different parts are used together, optimal results can be achieved. This framework will benefit the entire value chain from content creators and rights holders to service providers and consumers [40]. The overview briefly introduces the various standards defined in the MPEG-21 framework.

The second part focuses on MPEG-21 Digital Item Adaptation (DIA) for transparent use of multimedia across heterogeneous networks and devices, which is central toward achieving UMA [10]. In this part, we first introduce the description tools specified in the DIA standard that assist the adaptation of digital items. These tools include Usage Environment Description (UED), Bitstream Syntax Description Link (BSDLink), Bitstream Syntax Description (BSD), Terminal and Network Quality of Service (AdaptationQoS), Universal Constraints Description (UCD), Metadata Adaptability, Session Mobility, and DIA Configuration [41]. We also present the adaptation architecture based on these description tools. In particular, we show how the BSD tool can facilitate coding-format

independent adaptation that bridges the gap between the diversity of media formats and the variety of networks and devices.

The third part shows an example of FGS video adaption using the MPEG-21 DIA standard. The AdaptationQoS tool provides sampling points that associates adaptation operations with constraints and utility values. The Adaptation Decision Taking Engine (ADTE) decides the best choice of adaptation operation based on the AdaptationQoS to maximize the utility value (PSNR) while satisfying usage environment constraints. Among the output decision parameters of the ADTE, two parameters are used by the BSD transformation engine for truncation of FGS enhancement layer bitstream. These two parameters together enable the best utilization of the constrained network resources.

2.1 Overview of MPEG-21

Up to now, many of the normative parts of MPEG-21 have been published. Table 2.1 lists the various parts and their (expected) publication date. The most current development status can be found at: <http://www.itscj.ipsj.or.jp/sc29/29w42911.htm#MPEG-21>. In the following, we will briefly introduce each part that has been defined in the MPEG-21 framework.

Part 1—Vision, Technologies, and Strategy [11] is a technical report that describes the MPEG-21 multimedia framework and its architectural elements with functional requirements. The title has been chosen to reflect the purpose of the technical report:

- Define a vision for a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices to meet the needs of all users.
- Achieve the integration of components and standards to facilitate harmonization of technologies for the creation, modification, management, transport, manipulation, distribution, and consumption of Digital Items.

Table 2.1: MPEG-21 parts

Parts	Title	Publication
1.	Vision, Technologies and Strategy	Nov. 2004
2.	Digital Item Declaration	Oct. 2005
3.	Digital Item Identification and Description	Apr. 2003
4.	Intellectual Property Management and Protection Components	Apr. 2006
5.	Rights Expression Language	Apr. 2004
6.	Rights Data Dictionary	May 2004
7.	Digital Item Adaptation	Oct. 2004
8.	Reference Software	Apr. 2006
9.	File Format	Jul. 2005
10.	Digital Item Processing	Jan. 2006
11.	Evaluation Methods for Persistent Association Tools	Nov. 2004
12.	Test Bed for MPEG-21 Resource Delivery	Apr. 2005
13.	—	—
14.	Conformance Testing	Oct. 2006
15.	Event Reporting	Jul. 2005
16.	Binary Format	Oct. 2005
17.	Fragment Identification of MPEG Resources	—
18.	Digital Item Streaming	Oct. 2006

- Define a strategy for achieving a multimedia framework by the development of specifications and standards based on well-defined functional requirements through collaboration with other bodies.

Part 2—Digital Item Declaration (DID) [12, 42] provides a set of abstract terms and concepts to form a useful model so that any Digital Item (DI) can be uniquely represented and identified. There are many possible ways to represent the abstract model, and DID has chosen to use XML (eXtensible Markup Language) representation, which is termed the Digital Item Declaration Language (DIDL). There are three folds to use the DIDL [43]: 1) extensibility, 2) compatibility with other languages (e.g., WAP, JavaScript, SMIL, HTML), and 3) integration with other parts of the MPEG-21 framework such as the Right Expression Language (REL). Each element of the DIDL corresponds to a building block of the DID abstract model. The DID standard is thus divided into three clauses:

- **Model:** A set of abstract terms and concepts to form a useful model for declaring Digital Items. Within this model, a DI is a digital representation of an asset, and as such, it is the thing that is acted upon (managed, described, exchanged, collected, etc.) within the model.
- **Representation:** The description of the syntax and semantics of each of the DID elements, as represented in XML.
- **Schema:** An XML schema comprising the entire grammar of the DID representation in XML.

Part 3—Digital Item Identification (DII) [44, 42] provides a schema that can be used to include identifiers into a DID. Digital Items and their parts are identified by encapsulating uniform resource identifiers (URI) into the identifier element. The DII standard itself does not specify new identification systems. Rather, it allows for existing and new identification systems to be integrated into the MPEG-21 framework through the mechanism of a Registration Authority.

Part 4—Intellectual Property Management and Protection (IPMP) [45] defines an interoperable framework to allow controls on the flow and usage of Digital Items throughout their lifecycle. It includes standardized ways of retrieving IPMP tools from remote locations, exchanging messages between IPMP tools and between these tools and the terminals [11]. It also addresses authentication of IPMP tools, and has provisions for integrating the Rights Data Dictionary and the Rights Expression Language.

Part 5—Rights Expression Language (REL) [46, 47, 48] is an XML-based machine-readable declarative language used to specify rights and conditions for the authorized distribution and use of multimedia resources and services. REL provides a flexible and interoperable mechanism that addresses the issue of privacy by expressing rights and

conditions for the use of personal data. It declares rights and permissions using the terms defined in the Rights Data Dictionary standard.

Part 6—Rights Data Dictionary (RDD) [49, 48] comprises a set of clear, consistent, structured, integrated, and uniquely identified terms to support REL. RDD defines a single meaning for a term represented by a particular RDD name or headword. Terms are organized into families according to the roles they play in contexts. Contexts have two varieties: 1) events (through which change happens) and 2) situations (which are unchanging), which conform to the rules of the RDD context model. The context model is a logical structure ensuring the structure of the dictionary can never be broken.

Part 7—Digital Item Adaptation (DIA) [41, 50] specifies various description tools used to assist the adaptation of digital items. As shown in Figure 2.1, a digital item is subject to a description adaptation engine and a resource adaptation engine, which produces the adapted digital item. Though the description tools are within the scope of this standard, the two adaptation engines are not normative components for standardization.

Figure 2.1: Digital Item Adaptation architecture

Part 8—Reference Software [51] includes softwares that implement tools for all other parts in the framework based on the requirements that have been defined.

Part 9—File Format [52] is designed to contain a base MPEG-21 XML document with some or all of its referenced resources in a single package, which enables the interchange, editing, and playback of MPEG-21 documents.

Part 10—Digital Item Processing (DIP) [53, 54] provides tools to allow users to include sequences of instructions for adding specified functionality to a DID. Such a sequence of instructions is called a Digital Item Method (DIM). The basic idea is that on receipt of a DID, a list of methods that can be applied is presented to the user. And the user can choose one method which is then executed by the DIP Engine.

Part 11—Evaluation Methods for Persistent Association Tools [55] is a technical report focusing on the evaluation of two classes of technologies: watermarking and fingerprinting for audio content. Tools based on these two technologies offer a means to associate content, metadata and IPMP elements while the association can be directly embedded in or inferred from the content itself. In addition, the association still persist when the content is adapted. Such tools are termed Persistent Association Technologies (PAT). This technical report sets out a process and plan for evaluating PAT using a common methodology. Further enhancement to cover other media types such as videos, images, and texts is expected to be added to the report.

Part 12—Test Bed for MPEG-21 Resource Delivery [56] specifies a software test bed to evaluate resource delivery for streaming applications while taking into account varying network conditions.

Part 13—(currently not defined) was originally designed to specify Scalable Video Coding (SVC), which had been moved to MPEG-4 Advanced Video Coding standard later on.

Part 14—Conformance Testing [57] tests whether a particular software or hardware or document meets its requirements as specified in the framework.

Part 15—Event Reporting [58] provides a standardized way for sharing information about events. It defines conditions that an event is deemed to have occurred and the syntax and semantics for creating and processing the event reports.

Part 16—Binary Format [59] specifies the binary format for MPEG-21 descriptions using MPEG-7 BiM (ISO/IEC 15936-1). This enables the efficient interchange or storage of MPEG-21 descriptions.

Part 17—Fragment Identification for MPEG Resources [60] specifies syntax which is based on and extends the W3C XPointer specification (<http://www.w3.org/TR/2003/REC-xptr-framework-20030325>) for URI Fragment Identifiers to address parts of a resource.

Part 18—Digital Item Streaming [39] targets the incremental delivery of a digital item in a piecewise fashion with temporal constraints such that a receiving peer may incrementally consume the digital item.

As MPEG-21 is a framework driven by the requirements for interoperability, new technologies could be considered for inclusion as new MPEG-21 parts.

2.2 MPEG-21 Part 7: Digital Item Adaptation

As explained in the previous section, the scope of MPEG-21 part 7 is to standardize description tools to assist adaptation of digital items. In this section, we first briefly introduce the key aspects for each of the description tools and their roles in the Digital Item Adaptation (DIA) standard. Then, we illustrate the adaptation architecture focusing on the adaptation decision-making process.

2.2.1 Usage Environment Description (UED)

The usage environment includes the description of User characteristics, terminal capabilities, network characteristics, and natural environment characteristics [41, 50]. Such

descriptions provide the adaptation engine with usage environment information based on which optimal adaptation decisions can be made.

User Characteristics specify the characteristics of a single user, including general user information, usage preferences and history, presentation preferences, accessibility characteristics, mobility characteristics and destination [41]. *UserInfo* specifies general information about users, such as name and contact information, where a user can be a person, a group of persons, or an organization. *UsagePreferences* provide various tools that directly describe the preferences of a user. *UsageHistory* describes the history of actions on digital items by a user, e.g., recording a video program. It describes the preferences of a user indirectly. *PresentationPreferences* define a set of preferences related to the means of how digital items are to be presented. *AccessibilityCharacteristics* provide descriptions for adapting content according to the user's auditory, visual, or color impairments. *MobilityCharacteristics* provide a concise description of the movement of a user over time. *Destination* specifies a tool for describing the destination of a user, with which adaptive location-aware services could be provided.

Terminal Capabilities specify the consumption and processing constraints of a particular terminal. The terminal in general represents both the transmitting end and the receiving end, including servers, intermediate network nodes, and end devices [50]. Three major attributes of terminal capabilities are: 1) codec capabilities, 2) I/O characteristics, and 3) device properties. The codec capabilities specify the formats that a particular terminal is capable of encoding or decoding. The I/O (input/output) characteristics include display capabilities (e.g., resolution and color capabilities), audio output capabilities (e.g., number of output channels) and user interaction inputs such as the ability to input information through a keypad. The device properties cover a variety of attributes, including power, storage, and CPUBenchmark to define the device class.

Network Characteristics specify both static network capabilities and time-varying network conditions [41]. The static capabilities include description of the maximum network capacity, the minimum guaranteed bandwidth, as well as attributes indicating if the network can provide in-sequence packet delivery and how to deal with erroneous packets (forward, correct or discard them). The dynamic conditions specify the available bandwidth (e.g., minimum, maximum, and average available bandwidth), error (packet loss rate and bit error rate) and delay (one-way, two-way packet delay and delay variation). Due to its dynamic nature, time stamp information (start time and duration) is also needed for network conditions.

Natural Environment Characteristics describe the natural environment properties such as location and time of digital item usage as well as characteristics pertaining to audio-visual aspects [41]. For the visual aspects, illumination characteristics that may affect the perceived display of visual information are specified. For the audio aspects, the noise levels and noise frequency spectrum are specified. The location and time attributes are referenced by both mobility and destination tools specified in *User Characteristics*.

2.2.2 BSDLink

The BSDLink tool provides references to link the BSD tools and the steering description tools [41]. The steering description tool, such as the AdaptationQoS, steers and governs the whole adaptation process. In addition, the BSDLink tool may include a reference to the bitstream described by the BSD, a reference to the BSD transformation style sheet, and a reference to the parameters used by the BSD transformation style sheet.

2.2.3 Bitstream Syntax Description (BSD)

A bitstream is a sequence of structured binary symbols representing a media format specific resource. Due to the diversity of existing media formats, it is desired that an

Figure 2.2: Bitstream Syntax Description based adaptation architecture.

adaptation engine, unaware of the media formats, can perform adaptation operations on the bitstream. To facilitate such coding-format independent adaptation, DIA defines the XML based Bitstream Syntax Description (BSD) tool [41, 61]. The BSD in general does not describe the bitstream on a bit-per-bit basis, but rather address the high-level structure of the bitstream. Furthermore, the BSD itself is scalable, which enables the description of bitstream at different syntactic layers, e.g., GOP (Group of Picture) or scene levels, depending on the application. The BSD is also adaptable in order to properly reflect bitstream changes.

Figure 2.2 depicts the high-level BSD based adaptation architecture [10]. The BSD generator first parses the bitstream to generate the BSD. The BSD is then subject to the BSD transformation engine, which produces the transformed BSD. The adapted bitstream is generated by adapting the input bitstream according to the transformed BSD. Both the BSD and the bitstream generation processes are normatively specified by the standard. However, the BSD transformation process is not normative. Depending on the adaptation environments, BSD transformation can be performed by either the eXtensi-

ble Stylesheet Language Transformations (XSLT) or the Streaming Transformations for XML (STX) [62].

To facilitate both the BSD generation and the bitstream generation in a coding-format independent manner, a new language called the *Bitstream Syntax Description Language (BSDL)* has been created based on XML Schema [41, 61]. With BSDL, it is possible to design specific Bitstream Syntax Schemas (BS Schemas) to describe the syntax of a particular coding format [61]. A generic BintoBSD processor can automatically parse a bitstream and generate its description (BSD) based on the BS Schema. The same schema is also used by a generic BSDtoBin processor to generate the adapted bitstream from the transformed BSD.

BSDL requires the adaptation engine to know the coding-format specific BS Schema in order to process the BSD and generate the adapted bitstream. However, in some cases, it is desired to use a coding format independent schema, e.g., at remote devices with constrained resources. The *generic Bitstream Syntax (gBS) Schema* is defined to enable codec-agnostic description of bitstream that can be used when the availability of a specific BS Schema is not ensured [61, 63]. For gBS Description (gBSD), all coding format specific information will be specified using attributes. Additionally, gBS Schema allows to include semantic information into gBSD through the “marker” attribute.

DIA does not specify normatively a BintogBSD processor to generate the gBSD from a bitstream directly. There are two ways to generate the gBSD [63]: 1) generate the gBSD during bitstream encoding process because the encoder knows the structure and adaptation possibilities; 2) generate the BSD first using BintoBSD processor and then transform the BSD to gBSD with a XSLT style sheet. Unlike the BSDtoBin processor, a generic gBSDtoBin processor does not need codec specific schemas to parse and generate bitstreams since the gBS schema is standardized and coding format independent.

2.2.4 Terminal and Network Quality of Service (AQoS)

Terminal and network QoS (AdaptationQoS) addresses the problem of selecting optimal parameter settings for media resource adaptation to satisfy constraints imposed by terminals and networks while maximizing the QoS [41]. To facilitate this, the AdaptationQoS or AQoS tool specifies the relationship between constraints, feasible adaptation operations satisfying these constraints, and associated utilities (qualities) using modular representation formats. Three types of modules are specified by the standard [41, 50]:

- *UtilityFunction* provides a restricted set of adaptation operation points in a list format (N-dimensional vector), which is suitable for sparse and discrete data.
- *LookUpTable* provides a M x N matrix of adaptation operation points, which is suitable for dense and discrete data.
- *StackFunction* provides a functional representation format that expresses the relationship with a stack of operators and arguments. This module is suitable for functional and continuous data.

A module is a basic structural unit for grouping the data (constraints, adaptation operations, and utilities) and their relationship. The interface between modules are provided by IOPins. Each IOPin is a uniquely identifiable variable that is globally declared and referenced within a module. Two or more modules referencing the same IOPin are linked, which means that the IOPin should have the same value for all these modules. The value of an IOPin can be either resolved directly within the AQoS or retrieved externally from the UED. Further, the UCD may apply limitation and optimization constraints to the IOPin values. However, the UCD may not know the semantic meaning of an IOPin defined in the AQoS. In order to link the UCD constraints to the right IOPin in AQoS, DIA creates a number of *classification schemes* to standardize the semantics

for representing media characteristics, usage environment characteristics, and segment decompositions [64]. The AQoS associates the IOPins with the semantic terms, and the UCD uses only the semantic terms to specify constraints rather than using the IOPins.

2.2.5 Universal Constraints Description (UCD)

The UCD tool establishes mathematical abstractions for constraint specification [41]. The constraints are obtained either externally from UED or by a direct specification of numeric constants. The UCD is comprised by a set of limit and optimization constraints. The limit constraints are mathematically Boolean expressions which must be evaluated to true. The optimization constraints are numeric expressions which are to be maximized or minimized. Together, the limit and optimization constraints specify a generic single or multi-objective optimization problem.

2.2.6 Metadata Adaptability

Three types of metadata adaptation operations are specified by the DIA standard to reduce the processing complexity: 1) filtering, 2) scaling, and 3) integrating [41]. Given terminal and network constraints, it is wasteful to transmit and process unnecessary or irrelevant metadata of a large and detailed description. Filtering and scaling can be used to reduce the depth and the number of occurrences of certain elements to reduce the parsing complexity [50]. On the other hand, if a resource is described by multiple descriptions with each description focusing on different syntactic or semantic levels, complementary information can be integrated to get a more concise description.

2.2.7 Session Mobility

Session mobility refers to the transfer of configuration-state information of a DID on one device to a second device. This enables the digital item to be received by the second

device in an adapted way [50]. Furthermore, application-state information, e.g., the track of a music currently being played, may be transferred as well.

2.2.8 DIA Configuration

The DIA configuration tools [41] provide the following functionality: 1) suggest the DIA descriptions for resource adaptation and DID configuration; 2) identify how the “Choice/Selection” elements in a DID should be processed, e.g., displayed to users or configured automatically; 3) indicate the location for adaptation, e.g., sender side, receiver side, or either side.

2.2.9 BSD-Based Adaptation Architecture

Figure 2.3 illustrates the BSD-based resource adaptation architecture using the description tools described previously. There are three major steps involved to adapt a bitstream based on its high-level structure: 1) the Adaptation Decision-Taking Engine (ADTE) makes decisions for optimal adaptation parameter settings based on the inputs of AQoS, UED and UCD; 2) a generic XSLT (or STX) processor transforms the BSD based on the decision parameters; 3) a BSDtoBin processor adapts the actual bitstream according to the transformed BSD. Note that it is the BSDLink that provides the referencing information for locating the steering description tool AQoS, the decision parameters, the BSD, the XSLT style sheet for BSD transformation, and the input bitstream to be adapted.

The ADTE optimizes adaptation parameters that can satisfy the usage environment constraints while maximizing the QoS [64]. The ADTE can have several inputs including an AQoS, several UEDs and UCDs from different sources. The outputs of the ADTE are one or more parameters in the form of IOPins defined in both the BSDLink and the AQoS. The output parameters are used by the XSLT style sheet for transforming the BSD. The implementation of the ADTE is not normative in the DIA standard. Furthermore, DIA does not restrict the delivery architecture within which an ADTE is used in practice.

Figure 2.3: DIA description tools for content adaptation.

2.2.10 MPEG-21 DIA Current Status

MPEG-21 DIA standard has become an enabling technology for providing seamless multimedia access to different users across diverse networks and terminals. However, challenging problems still exist and may become major obstacles for the future success of MPEG-21. In this section, we discuss some of the issues and the current research activities related to MPEG-21 DIA (MPEG homepage: <http://www.chiariglione.org/mpeg>).

Currently, video adaptation in the MPEG-21 DIA framework is performed in a rather static manner. Depending on the level of description details (scene, GOP, frame, etc), the gBSD could be very large in size. Standard XSLT processors today requires complete loading of the input XML into memory before transforming it [62]. This means that XSLT-based adaptation can not happen unless the complete gBSD is available. Transmitting the entire gBSD document during session setup may cause significant startup

delay. Even if the gBSD has been pre-stored, an adaptation proxy may have limited processing power and memory capacity to process the gBSD in real-time. Two solutions have been proposed in [62]: 1) use a new streaming transformation language STX to process SAX-based XML events as an alternative to XSLT, and 2) break down the gBSD into small fragments known as processing units (PUs) that are individually identifiable and processable by standard XSLT. The second approach has been approved as an amendment part to MPEG-21 DIA.

Another challenge is representing and processing DIA descriptions in binary domain [62]. It has become a normative standard for MPEG-21 part 16. The binary format of descriptions can obviously reduce the document size and consequently reduce the transportation resource requirement. Another advantage of using the binary format is that it is machine-friendly and may greatly improve the processing speed. Currently design of binary symbols as well as its enabling processing engines are being studied by the MPEG group.

In [62], a distributed adaptation concept is described as adapting media contents at the server and proxies in multiple steps along the delivery path. Distributed architectures are known for their abilities in providing load balancing and data sharing to achieve the best utilization of system resources. In [62], the server adapts the DIs according to its environment constraints and subsequent proxies continue to adapt the DIs according to their own environment constraints. The server or proxy environment constraints refer to the common constraints shared by all the users served by the server or proxy. Under this distributed adaptation, a proxy can oversee and reshape the common environment constraints in the downstream and then sends the common constraint sets to its upstream proxy or server for aggregate adaptation. Thus, network utilization is optimized through distributed adaptation.

Figure 2.4: FGS scalability structure.

2.3 FGS Video Adaptation Framework

In this section, we use FGS video as an example to illustrate the adaptation process and explain the usage of the description tools specified in the DIA standard. In particular, we show the process of BSD and gBSD generation, optimal decision making, gBSD transformation, and bitstream adaptation.

2.3.1 FGS Video Coding Structure

As shown in Figure 2.4, FGS video is encoded into a base layer bitstream and an enhancement layer bitstream. While the base layer is encoded using MPEG-4 simple profile, the enhancement layer is bitplane coded after DCT (discrete cosine transform) transformation. Unlike the base layer VOPs (Video Object Plane), there is no prediction among adjacent FGS VOPs. And each base layer VOP has exactly one FGS VOP for picture quality enhancement. Figure 2.5 shows the syntactical structure of the FGS enhancement layer bitstream. Each FGS VOP starts with the header information, which is followed by several bitplanes (BP). The most significant bitplane is placed in the front of each VOP. In other words, truncation always begins with the least significant bitplane, which helps to preserve as much video quality as possible.

Figure 2.5: FGS bitstream syntax structure (the first FGS VOP contains 4 bitplanes, and second FGS VOP contains 5 bitplanes).

In the conventional DCT coding, the quantized DCT coefficients are coded in the (run, level) format. The number of consecutive zeros before a nonzero DCT coefficient is called a run, and the absolute value of the nonzero coefficient is called a level. For 2-D frame, the (run, level) symbol is then variable length coded (VLC). The bitplane coding method considers each quantized DCT coefficient as a binary number of several bits instead of a decimal integer. For example, for a zigzag scanned DCT coefficients:

9, 0, 6, 0, 0, 3, 0, 2, 2, 0, 0, 2, 0, 0, 1, 0, ..., 0, 0,

the maximum value in this DCT block is 9, for which the binary format is 1001, and therefore, there are totally 4 bitplanes to represent this block:

1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, (MSB plane)

0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0, 0, (MSB-1 plane)

0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, ..., 0, 0, (MSB-2 plane)

1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ..., 0, 0, (MSB-3 plane)

the bitplane that represents the most significant bits for the block is called MSB plane, and MSB-1 plane for the second most significant, etc (see [27]).

2.3.2 Generating BSD and gBSD

A generic BintoBSD processor needs the codec specific schema, which is created based on the BSDL, to parse a bitstream and generate its BSD. We've designed a "MPEG4-FGSBL" schema and a "MPEG4-FGSEL" schema for the BSD generation of the base

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

layer bitstream and the enhancement layer bitstream respectively. A fragment of the generated BSD for the base layer bitstream is shown in Listing 2.1. The BSD shows the high-level structure of the bitstream at VOP level. The header information, such as Video Object (VO), Video Object Layer (VOL), and Video Object Plane (VOP), is completely captured from the bitstream. For each VOP, the coding type (0, 1, and 2 representing I_VOP, P_VOP, and B_VOP respectively) is also extracted out of the bitstream. Note that the VOP coding type is encoded using 2 bits. To make the VOP “Payload” start at next integer byte, the “Stuffing” element takes up the rest 6 bits. So the value of the “Stuffing” element is actually a decimal number represented by 6 binary digits. The first and second values of the “Payload” element represent the VOP starting position and length in the bitstream after subtracting 5 bytes, with 4 bytes for the start code and 1 byte for the coding type and “Stuffing”.

Listing 2.1: Generated BSD for foreman video base layer bitstream

```
<?xml version="1.0"?> <Bitstream xml:base="foreman_qcif.cmp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="MPEG4-FGSBL"
  xmlns:fgsbl="MPEG4-FGSBL"
  xsi:schemaLocation="MPEG4-FGSBL MPEG4-FGSBL.xsd">
  <VO>
    <vo_start_code>00000101</vo_start_code>
  </VO>
  <VOL>
    <vol_start_code>00000120</vol_start_code>
    <vol_data>8 11</vol_data>
  </VOL>
  <VOP>
    <vop_start_code>000001B6</vop_start_code>
    <vop_coding_type>0</vop_coding_type>
    <Stuffing>16</Stuffing>
    <Payload>24 1539</Payload>
  </VOP>
  <VOP>
    <vop_start_code>000001B6</vop_start_code>
    <vop_coding_type>1</vop_coding_type>
    <Stuffing>17</Stuffing>
    <Payload>1568 478</Payload>
  </VOP>
  <VOP>
    <vop_start_code>000001B6</vop_start_code>
    <vop_coding_type>2</vop_coding_type>
    <Stuffing>16</Stuffing>
    <Payload>2051 171</Payload>
  </VOP>
  <!-- and so on -->
</Bitstream>
```

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

Listing 2.2 shows a fragment of the generated BSD for the enhancement layer bitstream at bitplane (BP) level. The bitplane number in each FGS VOP is represented by the last byte in the bitplane start code.

Listing 2.2: Generated BSD for foreman enhancement layer bitstream

```
<?xml version="1.0"?> <Bitstream xml:base="foreman_qcif_fgs.cmp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="MPEG4-FGSEL"
  xmlns:fgsbl="MPEG4-FGSEL"
  xsi:schemaLocation="MPEG4-FGSEL MPEG4-FGSEL-BP.xsd">
  <VO>
    <vo_start_code>00000101</vo_start_code>
  </VO>
  <VOL>
    <vol_start_code>00000121</vol_start_code>
    <vol_data>8 10</vol_data>
  </VOL>
  <VOP>
    <fgs_vop_start_code>000001B9</fgs_vop_start_code>
    <fgs_vop_coding_type>0</fgs_vop_coding_type>
    <Stuffing>16</Stuffing>
    <Payload>23 4</Payload>
    <BP>
      <fgs_bp_start_code>00000140</fgs_bp_start_code>
      <Payload>31 1214</Payload>
    </BP>
    <BP>
      <fgs_bp_start_code>00000141</fgs_bp_start_code>
      <Payload>1249 2225</Payload>
    </BP>
    <BP>
      <fgs_bp_start_code>00000142</fgs_bp_start_code>
      <Payload>3478 3335</Payload>
    </BP>
  </VOP>
  <VOP>
    <fgs_vop_start_code>000001B9</fgs_vop_start_code>
    <fgs_vop_coding_type>0</fgs_vop_coding_type>
    <Stuffing>17</Stuffing>
    <Payload>6818 4</Payload>
    <BP>
      <fgs_bp_start_code>00000140</fgs_bp_start_code>
      <Payload>6826 192</Payload>
    </BP>
    <BP>
      <fgs_bp_start_code>00000141</fgs_bp_start_code>
      <Payload>7022 1306</Payload>
    </BP>
    <BP>
      <fgs_bp_start_code>00000142</fgs_bp_start_code>
      <Payload>8332 2522</Payload>
    </BP>
    <BP>
      <fgs_bp_start_code>00000143</fgs_bp_start_code>
      <Payload>10858 3681</Payload>
    </BP>
  </VOP>
  <!-- and so on -->
</Bitstream>
```

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

Listing 2.3 shows a fragment of the gBSD generated by transforming the base layer BSD with an XSLT style sheet. The basic building block of gBSD is the “gBSDUnit” element. The “syntacticalLabel” attribute describes the syntatic label, which allows codec aware application to use for adaptation. The “start” and “length” attributes specify the starting position and the length of the “gBSDUnit” in the bitstream. With these attributes, the bitstream segment described by a “gBSDUnit” can be easily identified and located, and subsequently adapted. Another “Parameter” element, not shown in this example, describes a syntactical element whose value might be altered during adaptation. Consequently, the “Parameter” element is used to make sure that the adapted bitstream is syntactically correct.

Listing 2.3: Generated gBSD for foreman base layer bitstream

```
<?xml version="1.0" encoding="UTF-8"?> <dia:DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<dia:Description xsi:type="gBSDType">
  <Header>
    <ClassificationAlias alias="FGSBL"
href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
    <DefaultValues addressUnit="byte" addressMode="Absolute"
globalAddress="foreman_qcif.cmp"/>
  </Header>
  <gBSDUnit syntacticalLabel=":FGSBL:VO" start="0" length="4"/>
  <gBSDUnit syntacticalLabel=":FGSBL:VOL" start="4" length="15"/>
  <gBSDUnit syntacticalLabel=":FGSBL:I_VOP" start="19" length="1544"/>
  <gBSDUnit syntacticalLabel=":FGSBL:P_VOP" start="1563" length="483"/>
  <gBSDUnit syntacticalLabel=":FGSBL:B_VOP" start="2046" length="176"/>
  <!-- and so on -->
</dia:Description>
</dia:DIA>
```

Listing 2.4 shows a fragment of the gBSD for the enhancement layer bitstream. From this example, we see that a “gBSDUnit” element can have child “gBSDUnit” (and “Parameter”) elements. Besides, a “gBSDUnit” can include a “marker” attribute specifying semantically or syntactically important information. It is up to the application to make use of the marking information. In this example, we use the “marker” attribute to indicate the bitplane number in a FGS VOP. Value “0” means the most significant bitplane, which should be truncated at last.

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

Listing 2.4: Generated gBSD for foreman enhancement layer bitstream

```

<?xml version="1.0" encoding="UTF-8"?> <dia:DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dia:Description xsi:type="gBSDType">
    <Header>
      <ClassificationAlias alias="FGSEL"
href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
      <DefaultValues addressUnit="byte" addressMode="Absolute"
globalAddress="foreman_qcif_fgs.cmp"/>
    </Header>
    <gBSDUnit syntacticalLabel="FGSEL:VO" start="0" length="4"/>
    <gBSDUnit syntacticalLabel="FGSEL:VOL" start="4" length="14"/>
    <gBSDUnit syntacticalLabel="FGSEL:I_VOP" start="18" length="6795"/>
      <gBSDUnit syntacticalLabel="FGSEL:VOPHeader" start="18" length="9"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="27" length="1218" marker="0"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="1245" length="2229" marker="1"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="3474" length="3339" marker="2"/>
    </gBSDUnit>
    <gBSDUnit syntacticalLabel="FGSEL:I_VOP" start="6813" length="7726"/>
      <gBSDUnit syntacticalLabel="FGSEL:VOPHeader" start="6813" length="9"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="6822" length="196" marker="0"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="7018" length="1310" marker="1"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="8328" length="2526" marker="2"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="10854" length="3685" marker="3"/>
    </gBSDUnit>
    <!-- and so on -->
  </dia:Description>
</dia:DIA>

```

Table 2.2 summarizes the size properties for the encoded 300-frame “foreman” video sequence. The base layer has a code size of 89.8 kbytes, whereas the enhancement layer’s code size is as high as 2407 kbytes. The generated BSD document size is 45.8 kbytes for the base layer and 169.4 kbytes for the enhancement layer. The gBSD document is 22.0 kbytes for the base layer and 147.3 kbytes for the enhancement layer. It can be seen that without compression, the size of the description document is comparable to the size of the actual bitstream. MPEG-21 part 16 has addressed this problem by compressing and processing the description documents in binary format.

Table 2.2: Bitstream and description size properties for foreman

Layer	Code (kbytes)	BSD (kbytes)	gBSD (kbytes)
Base	89.8	45.8	22.0
Enhancement	2407.0	169.4	147.3

2.3.3 Making Decisions

Listing 2.5 shows the AQoS document used by the ADTE for optimal decision parameter setting. The “LookUpTable” specifies the adaptation relationship between the defined IOPins. The “BITPLANE” IOPin is associated with a semantic meaning defined by the AQoS classification schemes (“semantics=AQoS:3.2.1”), which means the number of the least significant bitplanes to truncate from every FGS VOP. The “PSNR” IOPin indicates the utility value associated with each bitplane layer cutting. For example, “BITPLANE”=2, “PSNR”=33.59 dB. The “BITRATE” IOPin tells the source rate (base layer and enhancement layer) after removing certain layers of bitplanes. The “RATE” IOPin is created purposely for optimal decision making. Note that the values of the “RATE” IOPin are obtained by simply shifting the “BITRATE” IOPin values one position to the right, and the leftmost position takes a very large number.

For FGS video, truncation at bitplane boundary is no fine granularity. For example, if the bandwidth is 600 kbps, coarse adaptation will simply remove the two least significant bitplanes, which wastes $600 - 458 = 142$ kbps network resource. We can truncate the bitstream somewhere in the second least significant bitplane to make use of the 142 kbps bandwidth. The “BP” IOPin in the StackFunction module achieves exactly this goal. The “TRANSMITRATE” IOPin value is extracted externally from the UED. The difference between “TRANSMITRATE” and “BITRATE” tells the amount of data in the most significant truncated bitplane that can be transmitted without violating the bandwidth constraint. The difference between “RATE” and “BITRATE” tells the amount of data increase by adding one more bitplane layer. So the ratio between these two differences yields the percentage of data in the most significant truncated bitplane that can be transmitted. Note that the ratio becomes zero when the “BITRATE” IOPin takes the leftmost value because the “RATE” IOPin value is now very large. The value of the “BP” IOPin is obtained by subtracting the ratio from the “BITPLANE” IOPin. The

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

integer part of “BP” tells the number of bitplanes to cut, and in addition to that, the decimal part tells the percentage of data in the next least significant bitplane to cut.

Listing 2.5: AQoS for optimal decision parameter setting

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS AQoS.xsd">
  <DescriptionMetadata>
    <ClassificationSchemeAlias alias="AQoS"
      href="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoS-NS"/>
  </DescriptionMetadata>
  <Description xsi:type="AdaptationQoSType">
    <Module xsi:type="LookupTableType">
      <Axis iOPinRef="BITPLANE">
        <AxisValues xsi:type="IntegerVectorType">
          <Vector>0 1 2 3 4</Vector>
        </AxisValues>
      </Axis>
      <Content iOPinRef="PSNR">
        <ContentValues mpeg7:dim="5 1" xsi:type="FloatMatrixType">
          <Matrix>42.42 37.84 33.59 30.49 29.98</Matrix>
        </ContentValues>
      </Content>
      <Content iOPinRef="BITRATE">
        <ContentValues mpeg7:dim="5 1" xsi:type="FloatMatrixType">
          <Matrix>1998 1083 458 132 72</Matrix>
        </ContentValues>
      </Content>
      <Content iOPinRef="RATE">
        <ContentValues mpeg7:dim="5 1" xsi:type="FloatMatrixType">
          <Matrix>1000000 1998 1083 458 132</Matrix>
        </ContentValues>
      </Content>
    </Module>
    <Module xsi:type="StackFunctionType" iOPinRef="BP">
      <StackFunction>
        <Argument xsi:type="InternalIOPinRefType" iOPinRef="BITPLANE"
          <Argument xsi:type="InternalIOPinRefType" iOPinRef="TRANSMITRATE"/>
          <Argument xsi:type="InternalIOPinRefType" iOPinRef="BITRATE"/>
          <!-- subtract -->
          <Operation operator="SF0:16"/>
          <Argument xsi:type="InternalIOPinRefType" iOPinRef="RATE"/>
          <Argument xsi:type="InternalIOPinRefType" iOPinRef="BITRATE"/>
          <!-- subtract -->
          <Operation operator="SF0:16"/>
          <!-- divide -->
          <Operation operator="SF0:19"/>
          <!-- subtract -->
          <Operation operator="SF0:16"/>
        </StackFunction>
      </Module>
      <IOPin semantics="AQoS:3.2.1" id="BITPLANE"/>
      <IOPin semantics="AQoS:2.1" id="PSNR"/>
      <IOPin id="BITRATE"/>
      <IOPin id="RATE"/>
      <IOPin id="BP"/>
      <IOPin semantics="AQoS:1.1.1" id="TRANSMITRATE"/>
    </Description>
  </DIA>
```

Listing 2.6 shows the BSDLink document that provides the referencing information for AQoS, gBSD, bitstream, XSLT style sheet for gBSD transformation, and decision parameters. The “Parameter” element used by the XSLT style sheet are references to the IOPins of the AQoS. The “name” attribute appears also in the XSLT style sheet and the “Value” is a reference to an “id” of an AQoS IOPin.

Listing 2.6: BSDLink for referencing

```
<DIA>
  <Description xsi:type="BSDLinkType">
    <SteeringDescriptionRef uri="foreman_AQoS.xml"/>
    <BSDRef uri="foreman_gBSD.xml"/>
    <BitstreamRef uri="foreman_qcif_fgs.cmp"/>
    <BSDTransformationRef uri="foreman_gBSD_transform.xslt"
      type="http://www.w3.org/1999/XSL/Transform"/>
    <Parameter xsi:type="IOPinRefType" name="nBP">
      <Value>BP</Value>
    </Parameter>
  </Description>
</DIA>
```

2.3.4 Transforming gBSD

Listing 2.7 shows the XSLT style sheet for enhancement layer gBSD transformation, which can be processed by any XSLT processor. The style sheet takes in one parameter “nBP”, which means the amount of bitplane data to be removed from the bitstream. The value of “nBP” is resolved by the ADTE, and obtained from the BSDLink. The “nBP” represents a decimal number, where the integer part tells the number of bitplane layers to cut for each FGS VOP, and in addition to that, the decimal part tells the percentage of data in the next bitplane layer to cut.

The first template “tplAll” in the style sheet simply copies every element together with its attributes in the gBSD to the adapted gBSD. The second template “tplBP” provides instructions to the XSLT processor on how to copy each “gBSDUnit” describing a bitplane segment. The template first counts the total number of bitplanes in the current FGS VOP, and represents it with a variable “tBP”. Next, a new variable “pBP” is created by subtracting the “nBP” value and the “marker” attribute value from the “tBP”. The

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

“marker” value gives the current bitplane’s position in the FGS VOP. The most significant bitplane is marked by “0”, and the number increases by 1 for each subsequent bitplane. The value of “pBP” determines how to copy the current bitplane (BP):

- (i) $pBP \geq 1$: the entire bitplane is copied.
- (ii) $0 < pBP < 1$: a percentage of the bitplane data are copied, and the amount is given by multiplying the bitplane “length” value with “pBP”.
- (iii) $pBP \leq 0$: the bitplane is not copied.

Listing 2.7: XSLT style sheet for gBSD transformation

```
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet
version="1.0"
  xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  xmlns:gbsd="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:param name="nBP">0.0</xsl:param>
  <xsl:template name="tplAll" match="@*|node()"/>
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <xsl:template name="tplBP" match="gbsd:gBSDUnit[@syntacticalLabel=':FGSEL:BP']">
    <xsl:variable name="tBP">
      <xsl:value-of select="count(..//gbsd:gBSDUnit[@syntacticalLabel=':FGSEL:BP'])"/>
    </xsl:variable>
    <xsl:variable name="pBP">
      <xsl:value-of select="$tBP - $nBP - @marker"/>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="$pBP >= 1">
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:when>
      <xsl:when test="$pBP > 0 and $pBP < 1">
        <xsl:variable name="len">
          <xsl:value-of select="floor(@length * $pBP)"/>
        </xsl:variable>
        <xsl:copy>
          <xsl:apply-templates select="node()"/>
          <xsl:apply-templates select="@syntacticalLabel|@start|@marker"/>
          <xsl:attribute name="length">
            <xsl:value-of select="$len"/>
          </xsl:attribute>
        </xsl:copy>
      </xsl:when>
      <xsl:otherwise/>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

Listing 2.8 shows the transformed enhancement layer gBSD with “nBP”=1.78. For each FGS VOP, the least significant bitplane is removed (integer part of “nBP” is 1), and in addition to that, 78% of the second least significant bitplane data are truncated. For example, in the first FGS VOP (Listing 2.4), the least significant bitplane with “marker”=2 is completely removed, and the second least significant bitplane keeps an amount of $2229 \times 0.22 = 490$ bytes. Note that during the transformation process, some attribute values may not be updated to reflect the changes (e.g., “start” attribute).

Listing 2.8: Transformed gBSD with bitplane truncation parameter “nBP”=1.78

```
<?xml version="1.0" encoding="UTF-8"?> <dia:DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<dia:Description xsi:type="gBSDType">
  <Header>
    <ClassificationAlias alias="FGSEL"
href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
    <DefaultValues addressUnit="byte" addressMode="Absolute"
globalAddress="foreman_qcif_fgs.cmp"/>
  </Header>
  <gBSDUnit syntacticalLabel="FGSEL:V0" start="0" length="4"/>
  <gBSDUnit syntacticalLabel="FGSEL:VOL" start="4" length="14"/>
  <gBSDUnit syntacticalLabel="FGSEL:I_VOP" start="18" length="6795"/>
    <gBSDUnit syntacticalLabel="FGSEL:VOPHeader" start="18" length="9"/>
    <gBSDUnit syntacticalLabel="FGSEL:BP" start="27" length="1218" marker="0"/>
    <gBSDUnit syntacticalLabel="FGSEL:BP" start="1245" length="490" marker="1"/>
  </gBSDUnit>
  <gBSDUnit syntacticalLabel="FGSEL:I_VOP" start="6813" length="7726"/>
    <gBSDUnit syntacticalLabel="FGSEL:VOPHeader" start="6813" length="9"/>
    <gBSDUnit syntacticalLabel="FGSEL:BP" start="6822" length="196" marker="0"/>
    <gBSDUnit syntacticalLabel="FGSEL:BP" start="7018" length="1310" marker="1"/>
    <gBSDUnit syntacticalLabel="FGSEL:BP" start="8328" length="555" marker="2"/>
  </gBSDUnit>
  <!-- and so on -->
</dia:Description>
</dia:DIA>
```

2.3.5 Adapting Bitstream

The transformed gBSD is used by a generic gBSDtoBin processor to parse and generate the adapted bitstream. Table 2.3 shows several adaptation scenarios under different bandwidth conditions. For example, at 600 kbps bandwidth, the “nBP” parameter is optimized to be 1.78. The enhancement layer description and bitstream properties are: the bitrate = 524 kbps, the transformed gBSD (gBSD’) size = 134.0 kbytes, and the

CHAPTER 2. FGS VIDEO ADAPTATION WITHIN MPEG-21

Table 2.3: Properties of foreman video after adaptation at different bandwidth

Bandwidth (kbps)	nBP (decimal)	Bitrate(EL) (kbps)	gBSD'(EL) (kbytes)	gBSD''(EL) (kbytes)	PSNR (dB)
1200	0.88	1120.8	160.4	142.8	38.17
600	1.78	524	134.0	118.8	34.23
300	2.49	226.4	107.2	95.0	31.71

updated gBSD (gBSD'') size = 118.8 kbytes. The PSNR after decoding the base layer and the adapted enhancement layer bitstreams = 34.23 dB. Figure 2.6 shows the video quality of foreman at frame number 60 for each adaptation scenario.

To ensure further adaptation along the transmission path, the transformed gBSD (Listing 2.8) must be updated to reflect the bitstream changes. As shown in Listing 2.9, certain attribute values, such as the "start" and "length" attributes, have been updated by the gBSDtoBin processor. The updated gBSD and the adapted bitstream together form an adapted digital item, which may be adapted subsequently.

Listing 2.9: Updated gBSD with bitplane truncation parameter "nBP"=1.78

```

<?xml version="1.0" encoding="UTF-8"?> <dia:DIA
xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dia:Description xsi:type="gBSDType">
    <Header>
      <ClassificationAlias alias="FGSEL"
href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
      <DefaultValues addressUnit="byte" addressMode="Absolute"
globalAddress="foreman_qcif_fgs.cmp"/>
    </Header>
    <gBSDUnit syntacticalLabel="FGSEL:V0" start="0" length="4"/>
    <gBSDUnit syntacticalLabel="FGSEL:VOL" start="4" length="14"/>
    <gBSDUnit syntacticalLabel="FGSEL:I_VOP" start="18" length="1717"/>
      <gBSDUnit syntacticalLabel="FGSEL:VOPHeader" start="18" length="9"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="27" length="1218" marker="0"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="1245" length="490" marker="1"/>
    </gBSDUnit>
    <gBSDUnit syntacticalLabel="FGSEL:I_VOP" start="1735" length="2070"/>
      <gBSDUnit syntacticalLabel="FGSEL:VOPHeader" start="1735" length="9"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="1744" length="196" marker="0"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="1940" length="1310" marker="1"/>
      <gBSDUnit syntacticalLabel="FGSEL:BP" start="3250" length="555" marker="2"/>
    </gBSDUnit>
    <!-- and so on -->
  </dia:Description>
</dia:DIA>

```

Figure 2.6: The video quality of foreman (at frame number = 60) after truncating bit-planes with “nBP” equals (a) 0.00, (b) 0.88, (c) 1.78, and (d) 2.49 respectively.

2.4 Summary

In this chapter, we have given an overview of the parts defined in the MPEG-21 multimedia framework. Each part is considered as a distinctive technology and can be used as stand alones. We also present the Digital Item Adaptation (DIA) architecture defined in MPEG-21 part 7. We start with the description tools specified by the DIA standard to assist adaptation of digital items. A general framework for coding format independent resource adaptation using the description tools is given. Finally, we illustrate the adaptation process with an example of FGS video. We discuss in detail the BSD and gBSD generation, optimal decision making, gBSD transformation, and bitstream adaptation with XML documents. In a nutshell, MPEG-21 provides flexible and interoperable use of multimedia resources for the 21st century.

Chapter 3

Optimal Frame Selection under Constrained Resources

3.1 Introduction

Video streaming has become an increasingly important technology used by many applications, such as video on demand, digital library, distance learning, and video conferencing, etc. Unlike data downloading, however, video streaming is delay-sensitive. Each video frame has a strict display time associated with it, and for the continuous video playback at the client side, each frame has to be received by the client buffer before its display time. This implies that the network bandwidth has to be sufficiently large to cope with the large volume of delay-sensitive video data, and at the same time, perhaps more importantly, the video source needs to be compressed efficiently before transmission to reduce the bandwidth requirement because network bandwidth is still a competing resource in today's network infrastructure.

3.1.1 VBR Video

In general, video can be compressed with either constant bitrate (CBR) encoding or variable bitrate (VBR) encoding. Compared to CBR compressed video, VBR video offers at least three major advantages [65, 31]: 1) better picture quality for the same average

bitrate, 2) shorter delay with reduced encoder buffer size, and 3) increased statistical multiplexing capability with lower network bandwidth requirement per admission call. Clearly, the predictable traffic pattern of CBR video makes the task of network management much easier. On the other hand, VBR video traffic often exhibits significant bitrate burstiness over multiple time scales, which complicates the network management to support good-quality VBR video transport. There are two types of bitrate burstiness occurring on two different time scales. The short-term burstiness is due to the coding format structure of the encoded bitstream, where intra-coded I frames generally need more bits to encode than forward predicated P frames, which in turn require more bits than bi-directionally predicated B frames. The long-term burstiness is caused by scene content variations or rapid chrominance and luminance changes such as in a lightening scene. With such a bursty traffic, a common belief is that VBR video needs to be transmitted according to the peak rate. However, this may result in significant under-utilization of network resources if the peak-to-mean rate ratio is high.

3.1.2 Traffic Model

One of the solutions to the problem of VBR video traffic is to use traffic models to describe the VBR characteristics of a video source, and try to boost up network utilization through statistical multiplexing of different sources aggregated in a high-speed network (e.g., ATM network) [66]. Traffic modelling may be broadly classified into two categories: statistic and deterministic. In [67], Frost and Melamed described several statistical traffic models, such as Markov-modulated, autoregressive, TES (Transform-expand-sample), and self-similar. In [33], Garrett and Willinger studied the statistical properties of VBR videos that exhibit self-similarity or long-range dependent time correlation. The VBR video appeared to be self-similar over a large range of time scales. They also showed that multiplexed sources behaved statistically better than single source. In general, statisti-

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

cal traffic models are based on stochastic processes and hence provide no performance guarantees for good-quality video transport.

Deterministic traffic modelling is a worst-case characterization of a video source that provides an absolute upper bound for deterministic performance guarantees [68, 69, 70]. Let $A[\tau, \tau + t]$ denote the traffic arrivals in the interval $[\tau, \tau + t]$. The upper bound is given by a function $A^*(t)$ such that for $\tau \geq 0$ and $t \geq 0$, $A[\tau, \tau + t] \leq A^*(t)$. $A^*(t)$ is a time-invariant traffic constraint function, i.e., a source is bounded for every interval of length t . Before a new source can be admitted to the network, call admission control is performed based on the delay bound of the new connection to verify if any existing connection will be affected. If the network carrying capacity is large enough for the new source, the call is accepted. Otherwise, the call is rejected. To achieve promising network utilization, deterministic traffic models try to make the traffic constraint function $A^*(t)$ as tight as possible so that admission control tests do not over-estimate the network resources required by a new source. The tightest deterministic bound is called the empirical envelope [69]. In [71], Ferrari and Verma proposed the $(Xmin, Xave, I, Smax)$ model, where $Xmin$ is the minimum packet spacing with constraint maximum packet size $Smax$, and within every interval of length I , the maximum number of packets sent is $Xave/I$. In [72], Cruz described the (σ, ρ) or leaky bucket [73] model, where ρ is a rate factor and σ is a burstiness factor, and within every interval of t , the traffic is less than $\sigma + \rho t$. In [69], Wrege *et al.* extended the work in [72] and proposed the $(\vec{\sigma}, \vec{\rho})$ model, which is a number n of (σ, ρ) pairs, and the amount of traffic in a time interval t is limited by $\min_i \{\sigma_i + \rho_i t\}$. The $(\vec{\sigma}, \vec{\rho})$ model defines a n -segment piecewise-linear concave traffic constraint function, which can be implemented by a cascade of n leaky buckets [74]. A more general deterministic traffic model is the D-BIND or deterministic bounding interval dependent model [68, 70], which characterizes sources via a number of rate-interval pairs (R_k, I_k) , where R_k is a bounding rate over every interval of I_k . The

D-BIND model captures the intuitive property that a source may be bounded by a rate close to the mean traffic rate over longer interval length. In [75], Tryfonas and Varma proposed deterministic algorithms for computing the burstiness curve of a video source. The algorithms exploit the piecewise linearity of the burstiness curve and identify the points at which the burstiness curve changes. In general, deterministic traffic models are able to provide hard performance guarantees, however, their ability to achieve high network utilization via statistical multiplexing is limited. For example, the average network utilization using the D-BIND traffic model was shown to be 25-60% [70].

3.1.3 Workahead Smoothing

Another solution for delivering VBR video is through the use of a bandwidth smoothing technique, commonly referred as workahead smoothing [35], that can take advantage of client buffer capability and *a priori* frame size information in stored video to schedule transmission of video data ahead of traffic bursts. The basic idea is to build up buffer by sending more data over small frame segments and drain buffer during bursts. By prefetching data into client buffer before each burst, large frames can be sent earlier at a slower rate to reduce network bandwidth requirements. Any valid transmission schedule should comply to the playback buffer constraint, i.e., not overflow or underflow the client buffer. If data are sent too fast, buffer overflow may occur and the received excessive data that are beyond the buffer size capacity will be discarded, causing loss of video data. Furthermore, most of the VBR video encoding technologies use predictive coding to explore the temporal correlation in a video. If the dropped data belong to a reference key frame, all its following dependent frames will be affected, resulting in degraded video quality. On the other hand, if data are sent too slow, buffer underflow may occur, resulting in starvation of data in buffer, which will interrupt the continuous video playback. Therefore, a feasible transmission schedule, which consists of a sequence

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

of piecewise-linear transmission rates and their corresponding operation periods (each pair of rate and period is called a run), must ensure that no buffer overflow or underflow occurs. The objective of workahead smoothing is to “smooth” the computed transmission rates, for example, reduce the peak rate or rate variability.

To formulate buffer constraints (see [76] for detailed formulation of these constraints), consider a N -frame video stream, and let f_i denote the size of frame i , for $i = 1, 2, \dots, N$. Without loss of generality, a discrete-time model is assumed where one time unit corresponds to a frame slot period. For a 30 frame per second video stream, one time unit corresponds to 1/30-th of a second. Let $A(t) = \sum_{i=1}^t a(i)$ represent the amount of cumulated data received by time t , where $a(i)$ is the amount of data received in one time unit of frame i slot. To ensure continuous video playback without interruption, the amount of received data must be large enough to avoid buffer underflow, which gives the lower bound of the buffer constraint:

$$A(t) \geq L(t) = \sum_{i=1}^t f_i, \quad (3.1)$$

and similarly, to prevent buffer overflow for a client buffer size $= B$, which gives the upper bound of the buffer constraint, the amount of received data must not exceed:

$$A(t) \leq U(t) = \min\{L(t-1) + B, L(N)\}, \quad (3.2)$$

for all $t = 1, 2, \dots, N$. A vector $S = [a(1), a(2), \dots, a(N)]$ is considered to be a feasible transmission schedule if $L(t) \leq A(t) \leq U(t)$ for all t .

In general, multiple feasible transmission schedules exist for a given buffer size for a VBR video. The task is to find an optimal schedule S^* that is as smooth as possible. In [35], Salehi *et al.* proposed an algorithm that can find a transmission schedule with minimum variance and peak rate, and the algorithm is called minimum variance bandwidth smoothing or MVBA in [37]. The variance is given by $\sum_t (a(t) - \bar{a})^2 / N$, with

Figure 3.1: A feasible smoothing schedule with minimum peak rate and variability.

$\bar{a} = \sum_t a(t)/N$, and the peak rate is given by $\max_t a(t)$. The strategy is to construct a piecewise-CBR transmission schedule, which generates m consecutive runs each with a constant rate r_j and a duration t_j . At $t = 0$, the longest feasible CBR segment is identified. When the transmission rate must be increased or decreased to ensure feasibility, the rate is changed as early as possible. The increased or decreased rate at the change point is chosen such that the CBR segment can extend the farthest possible. Figure 3.1 shows the first 3 runs of the algorithm (the solid line segments). Many optimal smoothing algorithms vary in the way of how to choose the starting point for the next run during rate changes, i.e., the dotted line segment in Figure 3.1. The critical bandwidth allocation or CBA algorithm [77, 78] starts a rate decrease at the leftmost point and searches every feasible starting point for a rate increase so that the CBR segment can extend as far as possible. The CBA algorithm is able to minimize the peak rate and the number of rate increases. The minimum changes bandwidth allocation or MCBA algorithm [79] extends CBA to perform searching for all rate changes. This results in a transmission schedule that has the minimum peak rate and the minimum number of rate changes.

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

There are also other algorithms proposed that focus on the lower bound of buffer constraint to construct a transmission schedule. In [36], Feng described a rate-constrained bandwidth smoothing or RCBS algorithm that is able to minimize the buffer size when a fixed maximum transmission rate is assumed. The piecewise constant rate transmission and transport algorithm or PCRTT algorithm [80] divides a video into fixed-size segments. Within each segment, the mean rate of the segment corresponds to one rate in the transmission schedule. To avoid buffer underflow, the PCRTT schedule is vertically shifted up until all of the runs lie above the buffer lower bound constraint, which introduces an initial playback delay and gives the minimum buffer size required. A further extension to [80], the PCRTT-DP algorithm [81], applies dynamic programming to find optimal PCRTT rates and intervals for a wide variety of optimization criteria, such as minimizing the peak rate and rate variability. In [82], Feng and Rexford compared the performance of the above mentioned algorithms, CBA, MCBA, MVBA, RCBS, PCRTT, and PCRTT-DP, for prerecorded video. In [83], Ye *et al.* proposed a wavelet-based traffic smoothing or WTS algorithm, which creates a full tree corresponding to the original VBR video traffic. A transmission schedule is a pruned version of the full tree for various optimization metrics such as peak rate, rate variability, and total number of rate changes.

From the above, a computed transmission schedule consists of a sequence of varying runs, which requires that the network needs to provide a flexible resource reservation mechanism to support the changing transmission rates. In [34], Grossglauser *et al.* proposed a renegotiated constant bit rate or RCBR network service model to support renegotiation of desired source transmission rate from the network. Although the RCBR service can effectively improve network utilization, problems could still happen when a rate renegotiation fails, i.e., the network may reject a higher bandwidth reservation request when the aggregate bandwidth in the network reaches the maximum capacity. In [84], Gan *et al.* proposed a dual-plan bandwidth smoothing algorithm for layer-encoded

video. Upon bandwidth renegotiation failure, enhancement layer video data are dropped to maintain the same frame rate at the current transmission rate.

3.1.4 Online Smoothing

For stored video, smoothing algorithms can use full knowledge of the frame sizes to compute an optimal transmission schedule. However, for live or interactive video such as video conferencing, a sender only has limited knowledge of frame sizes. Smoothing algorithms normally has to predict future frame sizes. In [85], Lam *et al.* proposed a lossless smoothing algorithm to smooth over a small set of frames based on the known information of existing frames and estimates of future frame sizes. Such smoothing can effectively reduce the short-term burstiness introduced by coding structure format such as I, P, and B frames in MPEG videos. In [86], Sen *et al.* proposed a window-based online smoothing algorithm for delay tolerant applications. Workahead smoothing is performed over a larger window of frames with prediction of future frame sizes for performance improvements. Different from the traffic-based future frame size prediction, in [87] and [88], Bocheck and Chang proposed to use the visual content of video for traffic prediction to facilitate dynamic resource allocation. In [89], Wu *et al.* make use of both content and short-term traffic information to predict future traffic patterns for dynamic resource allocation of real-time video streaming. In general, due to the lack of full knowledge of frame sizes, online smoothing can only yield suboptimal results.

3.1.5 Lossy Smoothing

While network bandwidth limits the amount of data that can be transmitted per unit time, buffer size regulates the amount of data that can be prefetched into the client buffer [90]. If the resources required by a transmission schedule can not be allocated, e.g., under tight network bandwidth and buffer size condition, lossy smoothing may be

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

unavoidable. Given a maximum bandwidth and a limited client buffer size, Ng and Song [91] suggested to introduce playback pauses or delete B and subsequently P frames of MPEG video when transmission exceeds the channel bandwidth limit. In [90], Zhang *et al.* proposed an selective frame discard algorithm to minimize the number of frames for Motion-JPEG video that must be discarded in order to meet both bandwidth and buffer size constraints. However, for MPEG video, the inter-frame dependencies among I, P, and B frames may substantially increase the computation complexity in [90]. In [92], Zhou and Liou proposed a nonlinear frame sampling strategy for video streaming under tight channel bandwidth and buffer size constraints. The objective is to obtain a set of frames that can maximize the accumulated salient scores assuming each frame is associated with an independent salient score. A so-called zigzag buffer (Z-B) model was proposed for analyzing different low bit-rate channel and limited buffer size scenarios. Dynamic programming is applied to the Z-B model for finding an optimal frame selection path. The Z-B model is designed under the assumption of constant frame sizes. For VBR video, they suggested to use the greatest common divisor (GCD) of all the frame sizes as a step size so that each frame size becomes a multiple of the step size, allowing the Z-B model to be still applicable. However, in practice, the GCD is most likely to be a small value, which results in a large number of states for each frame and thus significantly increases the computation complexity of dynamic programming. The authors suggested to use frame size stuffing to increase the step size at the cost of sacrificing bandwidth.

3.2 Proposed Optimal Frame Selection Algorithm

We limit our discussion for the problem of VBR video traffic to optimal lossy smoothing. In this chapter, we propose an dynamic programming based optimal frame selection or OFS algorithm that takes advantage of the *a priori* frame size and motion information in a video for optimal lossy smoothing. Since high motion objects are usually more perceptible

to human eyes, it is desired to select more frames in the high motion segments for better perception. The Pixel Change Map (PCM) method [93, 94] is adopted for computing the amount of motion in each frame. After noise reduction and normalization, the amount of motion in each frame is represented by a real number in the range $[0, 1]$, which is used as a motion metric for quantifying the importance of a frame. Our goal is to select a set of frames out of the video that can deliver the maximal accumulated motion metrics while being guaranteed not to violate any bandwidth or buffer size constraint. We emphasize on the best content delivery given the available resources instead of delivering the entire video through reserving resources.

The proposed OFS algorithm is different from the Z-B model in [92] in at least the following ways: 1) unlike the Z-B model that is designed for constant frame-size video, the proposed OFS algorithm focuses on VBR video; 2) for VBR video, the Z-B model needs to use frame size stuffing for computation complexity reduction of dynamic programming at the cost of sacrificing bandwidth resource, whereas the proposed OFS algorithm can reduce the computation complexity through non-optimal state elimination; 3) unlike the Z-B model which only considers intra-frame video coding such as motion JPEG, the proposed OFS algorithm considers more commonly used inter-frame video coding such as MPEG video.

3.2.1 Motion Representation

As described from the above, a fundamental problem we need to solve is how to represent the amount of motion for each frame. The common approach is to analyze the motion field and use the motion energy to quantify the amount of motion such as in [95]. Compared with the approaches directly based on motion fields, the pixel change map or PCM scheme [93, 94] is easy to implement with low complexity. In fact, any frame or content classification scheme can be used in the proposed optimal frame selection algorithm.

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

PCM is by no means the only or the best way to measure content importance. The PCM scheme is adopted due to its simplicity.

The perception of motion content for human visual system relies on the intensity of the motion. By intensity, it means how fast a certain object moves. The faster the object moves, the more perceptible it is to human. It can be observed that a higher intensity of motion would generally lead to a large number of pixel changes in the video frames. The pixel change map (PCM) of a frame gives a good characterization of the motion content in the video. In particular, PCM computes the absolute pixel value difference, $d_i(x, y) = |p_i(x, y) - p_{i-1}(x, y)|$, between adjacent frames. For each pixel in this frame, if the absolute difference d_i is greater than a fixed threshold (we arbitrarily choose a threshold = 10 for the computation), the corresponding PCM value is set to 1 and 0 if d_i is smaller than the threshold. The threshold is simply to undo the effect of any noise associated with the camera or the discretization process.

After getting the PCMs for all the frames, we form an 1-D pixel change sequence, where the i -th component denoted as $\bar{d}_i$ is the average pixel change in the i -th PCM for frame i . A filter is then applied to the pixel change sequence to remove the noise due to other factors other than motion and obtain a more accurate motion measurement. The PCMs may contain pixel changes due to other factors such as sudden change of lighting condition. Those pixel changes are regarded as noise that corrupts the true measurement of the amount of motion and thus must be removed by a filter [93, 94].

In this research, we simply define the motion metric $m_i = \tilde{d}_i$ for frame i , where $\tilde{d}_i$ is the filtered average pixel change values. Figure 3.2 shows the motion curves for four common interchange format (CIF) video sequences: (1) akiyo, (2) foreman, (3) mobile, and (4) stefan. For akiyo, the video contains little head movement motion and thus the motion curve (Figure 3.2(a)) is close to zero. The forman video contains various amount of motion at different time, especially when there is a large camera panning

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

Figure 3.2: Motion curve for (a) akiyo, (b) foreman, (c) mobile, and (d) stefan.

motion from frame 175 to frame 235 which is represented as a plateau in the motion curve (Figure 3.2(b)). For the mobile video, which contains smooth object and camera motion, the extracted motion curve (Figure 3.2(c)) is relatively flat. For the stefan video, the texture content is very rich and there is a periodical motion due the rhythm of playing tennis (Figure 3.2(d)). As shown in Figure 3.2(d), we can see that the extracted motion curve manifests the periodical rhythm very well. At the end of the stefan video sequence, the player rushes towards the net and we see an up drift of the motion curve there, which indicates the increasing amount of motion.

Figure 3.3: The system setting for optimal frame selection.

3.2.2 Problem Setting

Figure 3.3 shows the system setting for computing the optimal frame selection path or transmission plan. Two separate buffers are used at the client side for smoothing and decoding purposes respectively. The decoding buffer retrieves compressed frames from the receiving buffer and sends the decoded pictures to video sink for display. We assume that once a frame is retrieved from the receiving buffer, its space is immediately made available for future incoming data. In other words, we only need to examine the receiving buffer fullness to avoid buffer overflow and underflow when we compute the transmission plan. In the following, without specification, buffer size means the receiving buffer size.

In addition, for practical video coding, there usually exist inter-frame dependencies in the coded video. For example, MPEG video consists of I, P, and B frames. While I frames are intra-coded and can be decoded independently, forward predicted P and bidirectionally predicted B frames need their references for proper decoding. Thus, the encoding order of MPEG video is different from the display order. The proposed OFS algorithm selects frames according to the video's encoding order. In other words, for the receiving buffer, frames are consumed one by one in their encoding order at fixed intervals. It is the decoding buffer's responsibility to hold necessary references.

The transmission plan consists of the optimal frame selection path and the schedule for frame data transmission. The schedule tells the server the time and the period to stop transmitting data. In particular, for a long sequence of small-size frames, the client

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

consumes less data than the amount of data being received, which might cause buffer overflow eventually. Under such a circumstance, the server will have to either stay idle for some time or transmit at a reduced rate to prevent client buffer overflow.

Consider a N -frame video sequence, and let B denote the size of the client buffer and f_i and m_i denote the size and motion metric associated with the i -th frame respectively, where $i = 1, 2, \dots, N$. The problem of motion-based optimal frame selection can be expressed as follows:

to maximize the accumulated motion metrics M ,

$$\max(M = \sum_{i=1}^N m_i \cdot s_i), \quad (3.3)$$

subject to the bandwidth constraint

$$r_i \leq \text{bandwidth}/\text{framerate}, \quad (3.4)$$

and the buffer constraint, for $k = 1, 2, \dots, N$,

$$\sum_{i=1}^k f_i \cdot s_i - \text{preload} \leq \sum_{i=1}^k r_i \leq B - \text{preload} + \sum_{i=1}^{k-1} f_i \cdot s_i, \quad (3.5)$$

where s_i is an indicator function, which is equal to 1 if the i -th frame is selected and 0 otherwise, and r_i is the amount of data sent within the frame time slot between the $(i - 1)$ -th frame and the i -th frame. In Eqn. (3.4), if *bandwidth* is constant, then r_i is also constant unless buffer overflow occurs. The problem of variable channel bandwidth scenario is deferred to the next chapter.

The objective of optimal frame selection is to maximize the reward function given in Eqn. (3.3) by selecting a feasible set of frames, which satisfies both the network bandwidth and the buffer size constraints. Since each video frame is either selected or discarded, the problem can be considered as a 0-1 knapsack problem [96], which can be solved by dynamic programming. Dynamic programming is often used for solving optimization

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

problems whose solutions depend on their subproblems. It avoids overlap computation by solving each subproblem once and saves the answer to a table for future usage. At the final stage, it performs back tracing to find the optimal path that reaches the optimal solution. For a N -frame video, the number of possible path is 2^N in theory, which means the complexity grows exponentially with frame numbers. Our proposed dynamic programming based OFS algorithm has a worst-case complexity of $O(BN)$, where B is the client buffer size. In fact, the OFS algorithm is able to achieve a complexity of $O(TN)$, where the experimental results show that the value of T is about two magnitudes smaller than the value of B .

3.2.3 Channel Model

Figure 3.4 shows the constant bit-rate (CBR) channel model at frame level for client buffer management. Each discrete-time point along the horizontal direction is identified by the moment when the frame is being fetched out for decoding, and each buffer fullness level at any frame is called a state indicated by an arrow endpoint. The length of each downward arrow represents the frame size. The slope of the slanted lines determines the amount of data that can be transmitted in one frame time-slot period and is given by $bandwidth/frame\ rate$.

Let b_i^j denote the buffer fullness level at the j -th state at frame i . If state b_q^l is created by state b_p^k , or in other words, b_q^l is directly linked with b_p^k , the following relation holds:

$$b_q^l = \min\{b_p^k + (q - p) \cdot (bandwidth/frame\ rate), B\} - f_q, \quad (3.6)$$

where B is the buffer size and f_q is the size of frame q . Note that b_q^l becomes a full buffer state ($b_q^l = B - f_q$) if buffer overflow occurs during state transition from b_p^k to b_q^l . In this case, the server has to stay idle for some time or transmit at a reduced rate to avoid buffer overflow. The preload at the initial stage represents the amount of data that

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

Figure 3.4: The CBR channel model.

has been prefetched into the buffer before playback starts. Depending on the tolerable *startup delay* = *preload*/*bandwidth*, preload may vary in the range: $[0, B]$.

Let M_i^j denote the accumulated motion metrics for state b_i^j . For state transition from b_p^k to b_q^l :

$$M_q^l = M_p^k + m_q, \quad (3.7)$$

where m_q is the motion metric associated with frame q . If another state b_s^t also leads to state b_q^l , we resolve the collision with:

$$M_q^l = \max\{M_p^k, M_s^t\} + m_q, \quad (3.8)$$

from which, it is clear that the accumulated motion metrics for each state is always maximized. For example, in Figure 3.4, both the state b_2^1 with $M_2^1 = 0.08$ at P_2 and the state b_4^1 with $M_4^1 = 0.15$ at B_4 lead to the full buffer state b_5^1 at P_5 , but b_4^1 is chosen because it results in a larger $M_5^1 = 0.26$. For a tied situation where $M_p^k = M_s^t$, one can arbitrarily choose one state, or alternatively, choose the state that selects more frames as an additional gain. In summary, a state can be completely characterized by a five-tuple

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

vector $(f_i, m_i, b_i^j, M_i^j, p_i^j)$, where p_i^j is a pointer pointing back to the state that creates the current one and is useful for back tracing the path at the final stage. Eqn. (3.6), (3.7), and (3.8) together form the problem of bandwidth and buffer constraints as well as the optimization criterion, which can be solved by dynamic programming.

3.2.4 Non-optimal States

Theoretically, the number of possible states increase exponentially with frame number, which makes dynamic programming computationally prohibitive. However, we find that the number of states at each frame can be largely reduced by three factors:

- (i) buffer size: obviously, no state can fall outside the given buffer range. In fact, the number of states at frame i is confined within the range $[0, B - f_i]$.
- (ii) inter-frame dependency: for MPEG video, P and B frames need their reference frames for proper decoding. For example, in Figure 3.4, state transition from b_1^1 at I_1 to P_5 is not allowed because P_4 is a reference frame for P_5 , and hence any state transition to P_5 that bypasses P_4 is considered invalid.
- (iii) non-optimal states: described below.

Lemma 3.1 *For any two optimal states b_i^j and b_i^k at frame i (an optimal state means a state that could be included in the final optimal path), if $b_i^k < b_i^j$, then $M_i^k \geq M_i^j$, and vice versa. In other words, for optimal states at frame i , M_i^j increases monotonically as b_i^j decreases.*

Proof: Suppose $b_i^k < b_i^j$ and $M_i^k < M_i^j$, and b_i^k is on the final optimal frame selection path OP_k (see Figure 3.5). Obviously, b_i^j and b_i^k are mutually exclusive because frame i can only be selected once. Because b_i^j is at a higher buffer state, the least b_i^j can do is to select the same frames that b_i^k has selected from frame i until the end. The accumulated

Figure 3.5: An illustration of a non-optimal state

motion metrics of the new path OP_j is larger than that of OP_k by $M_i^j - M_i^k$. In fact, b_i^j has a better chance to select additional frames such as path OP_{j+} . Whatever the case is, b_i^j can make the accumulated motion metrics larger by at least $M_i^j - M_i^k$, which contradicts the claim that OP_k is the optimal path. Hence, b_i^k can never be an optimal state. For example, the state b_5^2 at P_5 with $M_5^2 = 0.23$ in Figure 3.4 is a non-optimal state. The elimination of non-optimal states can dramatically reduce the computation complexity because it not only reduces the number of states at each individual frame but also prevents those non-optimal states from propagating into subsequent frames.

3.2.5 Virtual States

Referring to Figure 3.4, we need to consider all the state transitions from P_2 , B_3 , and B_4 in order to obtain all possible states for P_5 . This may work fine for a small set of frames within a GOP (group of picture). However, it is not suitable for I frames in a long video sequence with a large number of GOPs since any state transition to an I frame from any previous frame is allowed. The number of possible state transitions to examine for the I frame in the next GOP can be potentially huge especially when the I frame is near to the end of the video sequence. To reduce the computation complexity and avoid the

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

Figure 3.6: The CBR channel model with virtual states.

inconvenient multi-step “look back”, we introduce the concept of virtual state, shown as the empty arrow endpoints in Figure 3.6. With virtual states, we only need to examine the states at the i -th frame to get all the optimal states for the $(i + 1)$ -th frame. In other words, virtual state facilitates frame-to-frame state transitions.

In particular, a state b_i^j at frame i is carried forward and becomes the k -th virtual state at the $(i + 1)$ -th frame with the following relations:

$$b_{i+1}^{kv} = \min\{b_i^j + \text{bandwidth}/\text{framerate}, B\}, \quad (3.9)$$

$$M_{i+1}^{kv} = M_i^j, \quad (3.10)$$

$$p_{i+1}^{kv} = \begin{cases} p_i^j, & \text{if } b_i^j \text{ is a virtual state,} \\ \text{pointer to } b_i^j, & \text{if } b_i^j \text{ is an actual state,} \end{cases} \quad (3.11)$$

where k is an incremental counter for the total number of virtual states at the $(i + 1)$ -th frame. From the virtual states, the corresponding actual states are easily obtained by:

$$b_{i+1}^{ja} = b_{i+1}^{jv} - f_{i+1}, \quad M_{i+1}^{ja} = M_{i+1}^{jv} + m_{i+1}, \quad p_{i+1}^{ja} = p_{i+1}^{jv}. \quad (3.12)$$

If $b_{i+1}^{ja} < 0$, it means the actual state causes buffer underflow and must be removed from frame $i + 1$, although the virtual state b_{i+1}^{jv} is still valid (possible state transition to

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

future I frames). After obtaining all the virtual and actual states at frame $i + 1$, they are jointly verified for state optimality, i.e., states at frame $i + 1$, virtual or actual, must all satisfy Lemma 3.1. For example, for B_4 in Figure 3.6, the non-optimal virtual state b_4^{2v} is removed from B_4 since $(M_4^{2v} = 0.12) < (M_4^{1a} = 0.15)$. Note that eliminating b_4^{2v} also prevents it from propagating to P_5 .

Although the CBR channel model with virtual states as shown in Figure 3.6 solves the problem of frame-to-frame state transitions, it has another potential problem of losing possible optimal states at certain frames. Consider the following frames:

$$\dots P_{i-3} B_{i-2} B_{i-1} I_i B_{i+1} B_{i+2} P_{i+3} \dots,$$

for which we divide the virtual states at I_i into two sets: set 1 contains the virtual states pointing back to frames before P_{i-3} , and set 2 contains the virtual states pointing back to P_{i-3} , B_{i-2} , or B_{i-1} . Due to inter-frame dependencies, only the actual states created by set 2 virtual states at I_i are allowed to propagate to B_{i+1} and B_{i+2} . If set 1 and set 2 are jointly verified by Lemma 3.1, it is possible that a virtual state from set 2 is non-optimal and hence discarded. However, the discarded virtual state may create a potentially optimal state at B_{i+1} or B_{i+2} . A simple and safe solution is to treat the two sets of virtual and their corresponding actual states separately at I_i . Lemma 3.1 is applied individually within each set. At P_{i+3} , the two sets are combined and jointly verified by Lemma 3.1 again. The cost is that the number of I-frame states may have a worst-case value of $2B$, where B is the client buffer size.

3.2.6 Optimal Preload

Given fixed bandwidth and buffer size constraints, we may obtain different global optimal results for different preload values. Depending on the client's tolerable startup delay, the preload can vary in the range: $[0, B]$, where B is the buffer size. Intuitively, a larger

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

Figure 3.7: Obtaining the optimal preload through GR and LR.

preload should yield a larger or at least equal global optimal result. The question is: *what is the minimum preload required to obtain the maximal global optimal result given bandwidth and buffer size constraints?*

Obviously, $preload = B$ should yield the maximal global optimal result. But the problem is how to bring it down to the optimal value. We present a two-step approach to find the optimal preload. In Figure 3.7, OP_B denotes the optimal path with $preload = B$. First, we perform global reduction (GR), which is defined as the distance between the lowest state on OP_B and the empty buffer line. The entire optimal path can be brought down by GR without affecting the global optimal result. Second, we perform local reduction (LR) in addition to GR. The idea of LR comes from the observation that in the case of buffer overflow, some bandwidth is being wasted, which might be otherwise avoided if we can lower the preload by an appropriate amount. Consider two adjacent states b_i^j and b_k^l on OP_B , where b_k^l is a full buffer state. The amount of LR is:

$$LR_k = \min\{b_i^j + (k - i) \cdot (bandwidth/frame\ rate) - B, b_s^t\}, \quad (3.13)$$

where b_s^t ($s < k$) is the lowest state on OP_B before frame k after global reduction. It is clear that we can bring down OP_B from the first frame to frame k by LR_k without

affecting the global optimal result. Note that this local reduction has no effect on OP_B after frame k . The LR process terminates when b_s^t hits the empty buffer line or it reaches the frame where GR is computed. Therefore, by jointly applying GR and LR, we are able to bring down the preload from B to its optimal value and shift OP_B down to OP_{B-} .

3.3 Simulation Results

3.3.1 Other Algorithms

To evaluate the performance of our proposed OFS algorithm, we have also implemented three other frame selection algorithms: the just-in-time (JIT) algorithm [90], the greedy algorithm [90], and the Z-B algorithm with frame size stuffing [92]. Since the original algorithms only consider intra-frame video coding, we make some modifications in order for them to be applicable with inter-frame dependencies.

Just-In-Time Algorithm is probably the most intuitive approach for frame selection. It always drops the current frame if client buffer underflow occurs and reduces the transmission rate when buffer overflow occurs. Due to inter-frame dependencies, we orderly apply the JIT algorithm to different layers of frames: I, P and B, and make sure the references are selected before we select a new frame. The layered JIT algorithm has a worst-case complexity of $O(N^2)$.

Greedy Algorithm always tries to make the result look the best at the moment, i.e., the frame currently having the largest motion metric will get selected first. To overcome inter-frame dependencies, we use weighted metric for frame sorting:

$$m_i^w = \begin{cases} m_i, & \text{if frame } i \text{ is B frame,} \\ \sum_{k=i}^j m_k, & \text{if frame } i \text{ is I or P frame,} \end{cases}$$

where j is the index for the last frame in the current GOP. Reference frames are given greater weighted metric and thus will be considered first. However, we still need to check whether a frame's references have been selected before inserting the frame to the

Figure 3.8: The Z-B algorithm with frame size stuffing.

path. The overall complexity of the greedy algorithm is also $O(N^2)$. In addition, we also introduce another metric $m_i^r = m_i/f_i$, i.e., the ratio between motion metric and frame size, to replace the original motion metric m_i since the greedy algorithm is very sensitive to frame size.

Z-B Algorithm makes use of frame size stuffing for VBR video delivery. As shown in Figure 3.8, all the enqueue lines (slanted lines) are vertically separated by a fixed distance called the step size and every state at each frame is on an enqueue line. This configuration effectively bounds the number of states at each frame to $B/\text{stepsize}$. The larger the step size, the fewer the number of states and thus the less the computation complexity of $O(NB/\text{stepsize})$. To realize the configuration, the size of each frame must be a multiple of the step size. For example, for step size = 100 bytes, a frame size = 1009 bytes will be stuffed with 91-byte dummy data to make its size = 1100 bytes. The worst-case stuffing for a N -frame video is $N \times (\text{stepsize} - 1)$. A tradeoff has to be made between saving bandwidth resource and reducing computation complexity.

Table 3.1: The properties of the two MPEG-4 video traces.

Video Title	akiyo	foreman	mobile	stefan
Bitrate (kbps)	47.52	288.2	745.1	977.0
Motion Metric	3.05	63.20	109.3	127.0

3.3.2 Performance Evaluation

We compare the performance of six algorithms: OFS, OFS+OP, Z-B, Layered JIT, Greedy, and Greedy+WR, where OFS is our proposed optimal frame selection algorithm without optimal preload, OFS+OP is OFS with optimal preload, Z-B represents the Z-B algorithm with frame size stuffing, Layered JIT represents the layered just-in-time algorithm, Greedy represents the greedy algorithm with weighted metric, and Greedy+WR represents the greedy algorithm with weighted ratio metric. We choose four representative MPEG-4 CIF video traces: akiyo, foreman, mobile, and stefan, to evaluate the performance of the various algorithms. Each trace contains 300 frames with frame rate = 30 frames per second. All the traces are encoded in the pattern: $\dots I B B P B B P \dots$, with a GOP size of 90. Table 3.1 shows the coded average bit rate and motion metric for each video trace.

We fix the buffer size and vary the bandwidth for each video trace. Except for the OFS+OP algorithm, all the other algorithms use fixed preload value of half the buffer size. Figure 3.9 shows the frame selection results using the six different algorithms under different channel bandwidth conditions. It can be seen that OFS+OP outperforms all the other algorithms especially in the low bandwidth region because it can fully explore the buffer capacity. Under the fixed preload values, our proposed OFS always gives the optimal results. For the Z-B algorithm, we choose a step size = 100 bytes for all the video traces. The performance of Z-B for stefan is only slightly worse than OFS while the gap is much larger for akiyo. This is because stefan has a very high bit rate and the

CHAPTER 3. OPTIMAL FRAME SELECTION UNDER CONSTRAINED RESOURCES

Figure 3.9: The selection results for (a) akiyo, (b) foreman, (c) mobile, and (d) stefan.

stuffed data only occupy a small percentage of the total bandwidth, whereas for akiyo, the stuffed data severely degrade bandwidth utilization. As expected, the selection results of the Greedy+WR algorithm are very close to the optimal results except in the low bandwidth region. This is because at low bandwidth some P frames that have small motion metrics still tend to be selected due to the large weights assigned to them. As a result, Greedy+WR performs as poor as the Greedy algorithm and the JIT algorithm in the low bandwidth region. Note that for the Layered JIT algorithm, I and P frames are always considered first, which is equivalent to assign them weights. It is interesting that the Layered JIT algorithm which has no content awareness performs better than the

Figure 3.10: The effectiveness of non-optimal state elimination for stefan.

Greedy algorithm for stefan. The reason is that for stefan high motion frames have very large frame sizes and the Greedy algorithm completely ignores the cost of consuming large frames.

3.3.3 Effectiveness of Non-optimal State Elimination

Figure 3.10 shows the number of states at each frame for stefan using the OFS algorithm and the Z-B algorithm with bandwidth = 600 kbps, buffer = 150 kbytes, preload = 75 kbytes, and step size = 100 bytes for the Z-B algorithm. The initial linear increment indicates an exponential growth of states with frame numbers (note that a log scale is used at the vertical axis). The curves then become relatively constant because of the buffer size constraint. Those discontinuous points are mainly due to inter-frame dependencies. In particular, the spikes at I frames are the direct result of separate recording of the two

sets of I-frame states (see section 3.2.5). Figure 3.10 also shows that there exist a huge number of non-optimal states. By applying the proposed non-optimal state elimination, the number of states at each frame is reduced by approximately 100 times. Comparing the number of states at each frame (Figure 3.10) and the selection results (Figure 3.9(d)) for the OFS and Z-B algorithms, it appears that step size = 100 bytes is a good tradeoff for the Z-B algorithm for the stefan trace.

3.4 Summary

In this chapter, we have proposed an optimal frame selection (OFS) algorithm based on dynamic programming for optimal lossy smoothing under constrained resources. Given bandwidth and buffer size constraints, our proposed OFS algorithm can always find the optimal frame selection path that can maximize the delivered video's motion information content. We have also modified and implemented three other algorithms: the just-in-time algorithm, the greedy algorithm, and the Z-B algorithm with frame size stuffing. The experimental results have demonstrated that our proposed scheme can achieve the optimal performance especially in the case of low bandwidth. In addition, we have proposed non-optimal state elimination which can effectively reduce the number of states by 2 magnitudes.

Chapter 4

Reverse Frame Selection under Dynamic Constrained Resources

4.1 Introduction

In the previous chapter, we have discussed the problem of optimal lossy smoothing based on the *a priori* frame size and motion information in a video for constant bit-rate (CBR) channels. However, in practice, the time-varying channel rate, especially in the wireless communication networks, is hardly predictable. In [97], Feng and Liu proposed two solutions for streaming stored video over variable bit-rate (VBR) channels: 1) pre-compute a smoothing plan and adaptively drop frames under bad channel conditions, and 2) run the smoothing algorithm online under the new bandwidth condition for the rest of the frames. For the first approach, it is hard to choose the rate for computing a transmission plan because statistics about the VBR channel might not be available. Further, adaptive frame dropping degrades the global optimality. For the second approach, it requires online computation of transmission plans which may introduce delays and the situation becomes even worse when there are many concurrent connections. In [98], Stockhammer *et al.* proposed a scheme for deterministic VBR channel, which can guarantee certain quality of service (QoS) with certain probability by selecting the appropriate initial delay and buffer size. In [99], Hsu *et al.* studied the problem of real-time video transmission

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

over networks where VBR transmission is possible. Dynamic programming is applied to jointly select the source and channel rate, each has a set of possible values to choose from, to maximize the video quality.

Recently, we have seen extensive studies on rate-distortion optimized (RDO) video streaming over lossy channels [100, 101, 102]. The most representative work is the one in [100], where Chou *et al.* proposed a framework for streaming packetized media over a lossy packet network in a RDO way. The proposed framework is able to minimize the end-to-end distortion under a rate constraint by choosing the right packets to transmit at a given transmission opportunity. Although the framework is very comprehensive and theoretically sound, it requires an accurate network delay model, which is very difficult to obtain for a network such as Internet. In addition, the proposed optimal packet scheduling in [100] is very complex, which might limit its implementation in practice.

In this chapter, we assume the packet loss problem can be well handled by the error control techniques deployed in the transportation layer and the link layer. We focus on optimal lossy smoothing for dynamic VBR channels under a given buffer size. In particular, we first propose a reverse frame selection (RFS) algorithm for CBR channels, where the selection starts from the last frame and ends at the first frame. The major advantage of our proposed RFS algorithm is that by running RFS just once, we can easily retrieve any optimal frame selection path starting from any frame at any buffer state. Next, we extend the RFS algorithm to solve the problem of streaming stored video over the VBR channels that can be regarded as piecewise-CBR channels. We only need to run RFS k times, where k is the number of channel bandwidth samples, and it can apply to any type of VBR channels with bandwidth changes occurring at any time. For VBR channels, we assume the bandwidth does not change rapidly and remains constant in a magnitude of a few seconds. This is reasonable for slow-fading channels in mobile networks or using TRFC [103] in Internet, where the application rate is adjusted according to a certain feedback interval.

4.2 RFS for CBR Channels

In this section, we propose a reverse frame selection (RFS) scheme, which selects frames starting from the last frame of a video sequence until its beginning, to solve the problem of video streaming over CBR channels.

As shown in Figure 4.1(a), the symbols above the full buffer line tell the frame type (I, P, or B) as well as the frame number in the sequence. The real numbers below the empty buffer line are the motion metrics associated with each frame. It is obvious that after the client consumes the last frame, the buffer should become empty. Hence, the first state at the last frame is originated from the empty buffer line, which we refer it as an empty buffer state. A state is represented by the end point of an arrow, and all arrows are pointing upward because we can record the accumulated motion metrics only if the arrow end is within the full buffer line or its “buffer resource need” can be satisfied. An upward arrow means consumption of the buffer data. In contrast, reception of data during each frame slot is reflected by the downward slanted lines between frames. In case of “buffer underflow”, such as that from B_N to P_{N-1} , an empty buffer state will be created. It means that the amount of data transmitted during the period are more than enough to reach the current state, and the server needs to stay idle for some time or reduce the transmission rate. Because an path can terminate at any frame, if there is no empty buffer state at a frame, we will create one such as that at B_{N-2} .

Figure 4.1(a) is not intuitive to interpret. We use a simple “buffer mirror” to make the computation easier and more intuitive. Imagine there is a mirror aligned with the empty buffer line in Figure 4.1(a). If we look from the full buffer line side, we shall see a mirrored buffer model as shown in Figure 4.1(b). The buffer mirroring effect makes the computation just like what we do in the optimal frame selection algorithm except that the selection is from the end of the video sequence to the beginning. All the concepts, including non-optimal state elimination, virtual state to facilitate frame-to-frame state

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

Figure 4.1: Reverse frame selection (a) in normal mode, (b) in mirrored mode.

transition, discussed for the optimal frame selection algorithm in chapter 3 can also be applied to the mirrored buffer model.

Note that at the first frame, RFS generates multiple accumulated motion metric gains at different states and each gain corresponds to the optimal result that we can achieve at that preload level. In other words, RFS runs only once and gets all the optimal results for different start-up delays, which is very useful for the case of multiple clients with different start-up delay requirements. For those preloads that are not exactly matched in the RFS

results, they are the same as their nearby lower preloads. On the contrary, the optimal frame selection in chapter 3 has to run exhaustive searching in order to obtain all the optimal results for multiple start-up delay requirements, which is very time-consuming.

4.2.1 RFS for VBR Channels

For the VBR channels (piecewise constant channels), the optimal frame selection becomes extremely difficult since we do not know when and how the channel is going to change in the future, i.e., the channel variation is unpredictable. Suppose we know the range of bandwidth variation, one possible approach is to offline compute the optimal frame selection path according to the middle value of the bandwidth variation range. Note that this middle value is not the average bandwidth, which we do not know. We can also compute the optimal path according to the minimum (or maximal) channel bandwidth, but it will lead to high channel bandwidth wasting (or high occurrence of buffer underflows). During transmission, the just-in-time (JIT) algorithm is applied for on-the-fly frame dropping in response to bandwidth changes. If the current bandwidth is larger, JIT raises buffer occupancy, which reduces the probability of future buffer underflow. When buffer overflow occurs, JIT reduces the transmission rate. Clearly, all these approaches use the pre-computed frame selection path and the actual path cannot be better than the pre-computed one. Another possible approach is to use JIT directly without any pre-computed frame selection path, which tries to send all the frames without content awareness.

We extend the RFS algorithm for video streaming over VBR channels. In particular, we sample the bandwidth variation range into a finite sequence of channel rates. For a given client buffer size, we run RFS for each channel rate. During transmission, if the starting buffer occupancy status is b_1 and the current bandwidth is C , we will first classify it into one of the pre-sampled channel rates C_1 and then retrieve the optimal

Figure 4.2: Optimal path switching for video streaming over VBR channels.

frame path for the channel rate C_1 with a starting state of b_1 . If at frame N_2 the buffer state is b_2 and the bandwidth is changed to C_2 , we will retrieve the optimal frame path starting at frame N_2 for the channel rate C_2 with a starting state of b_2 . Figure 4.2 shows such an example. In this way, the global optimality is approximately preserved under dynamic changing bandwidth conditions. The key advantage here is that we only need to run RFS k times, where k is the number of channel bandwidth samples, and it can apply to any pattern of piecewise constant channels with bandwidth changes occurring at any time as long as the changes are within the variation range.

Another problem for video delivery over CBR and VBR channels is the end-to-end delay jitter introduced during video transmission. The delay jitter is the sum of fixed transmission and propagation delays, and a variable delay due to network queueing. In [35], the stochastic delay jitter problem is formulated as follows. Let $\tau = \sigma + \nu$ denote the end-to-end delay jitter, where σ is a constant and ν is a stochastic variable. Suppose the maximum network jitter $\nu_{max} = \tau_{max} - \tau_{min}$ can be determined or measured. Delay jitter can be accommodated by ensuring the transmission schedule is always at least ν_{max} time ahead of the client. For lossy smoothing, the optimal frame selection path must be shifted to the left by ν_{max} time units. A slightly extra buffer space is required to increase the safety margin. We leave the delay jitter problem as a future work.

4.3 Simulation Results

In this section, we run several experiments to evaluate the performance of our proposed scheme under both CBR and VBR channel conditions. The four video traces: akiyo, foreman, mobile, and stefan, with each having only 300 frames, are too short to evaluate the performance gains for different preloads and buffer sizes. To create a longer video sequence, we equally choose 15 times of each of the four video traces and randomly shuffle the sixty 300-frame traces to form a mixed video sequence. The mixed sequence is encoded into MPEG-4 code stream in the pattern: IPBBPBBP..., with a GOP (group of picture) size of 60. The encoded bitstream has an accumulated motion metrics of 4598.21 with an average bitrate of 542.33 kbps.

4.3.1 Simulation for CBR Channels

4.3.1.1 Impact of Preload

Figures 4.3(a) and 4.3(b) show the optimal accumulated motion metrics and the total number of selected frames that we can achieve at different preloads given bandwidth = 300 kbps and buffer = 512 kbytes. With such a small buffer, large preloads can not provide proportionate gains for the accumulated motion metrics (reflected by the decreasing rate of increment in Figure 4.3(a)). Obviously, the largest preload is limited to the buffer size. However, as shown in Figure 4.3(a), with frame stuffing = 1 byte (i.e. no stuffing), the accumulated motion metrics stop increasing at the preload time a little over 4 seconds (after that preload time, the plot becomes flat), which is less than half of the largest allowable preload time = 13.65 seconds ($8 * 512 \text{ kbits} / 300 \text{ kbps}$). The reason is because a small buffer is more likely to incur buffer overflow at an early stage, and once buffer overflow occurs, increasing the preload becomes useless (see chapter 3.2.6). Unlike the accumulated motion metrics, the total number of selected frames tend to accelerate with increasing preload time. At low preloads, frames that can contribute the most to

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

Figure 4.3: Frame selection results at different preloads at buffer = 512 kbytes for (a) accumulated motion metrics (b) total number of selected frames, and at buffer = 1024 kbytes for (c) accumulated motion metrics (d) total number of selected frames

the overall gain are selected, and hence more frames need to be selected at large preloads in order to contribute roughly the same amount of gains.

Figures 4.3(c) and 4.3(d) show the frame selection results with an increased buffer size = 1024 kbytes. The largest preload time is now 27.30 seconds ($2 * 13.65$ seconds). As shown in Figure 4.3(c), the accumulated motion metrics stop increasing at the preload time close to 18 seconds (see the plot with frame stuffing = 1 byte). The larger buffer size provides more space for building up data before buffer overflow occurs. The accumulated motion metrics increase almost linearly with preload time to a certain extent. Preloads tend to have greater impacts on large buffers.

Figure 4.4: Frame Stuffing vs Number of Frame States.

4.3.1.2 Effectiveness of Frame Stuffing vs Non-optimal State Elimination

Dynamic programming is known for its high computation complexity. In chapter 3.2.4, we have proposed to eliminate the intermediate non-optimal frame states to reduce the complexity. In addition, for large buffers, tradeoffs can be made to use frame stuffing to further reduce the complexity at the cost of sacrificing some bandwidth resources. In this section, we are going to evaluate the effectiveness of frame stuffing as well as non-optimal state elimination for reducing the complexity of dynamic programming.

Figure 4.4 shows the average number of states per frame for different frame stuffing step sizes under different buffer conditions. As we can see, the number of states per frame reduces with increasing stuffing sizes. For instance, for buffer = 1024 kbytes, the number of states reduces from over 128k (2^{17}) at stuffing = 1 byte (i.e. no stuffing) down to a little over 4k (2^{12}) at stuffing = 200 bytes, a dramatic reduction of 32 times. However, from Figure 4.4, we see that the rate of reduction keeps decreasing with increasing stuffing sizes. This phenomenon is expected because each time we double the stuffing size, the

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

Table 4.1: Gain Sensitivity to Frame Stuffing

Sensitivity (%) at buffer (byte)	Frame Stuffing Step Size (byte)					
	2	50	100	200	400	800
512K	99.99	99.33	98.68	97.33	94.71	89.70
1024K	99.98	99.20	98.44	96.86	94.07	88.66

number of states will reduce to half (in theory), and consequently the rate of reduction will decrease to a quarter. As the number of states becomes less, further reduction by increasing the stuffing size is insignificant.

While frame stuffing can be used to reduce the computation complexity, we can not underestimate the cost of sacrificing bandwidth. As shown in Figures 4.3(a) and 4.3(c), the amount of gains for the accumulated motion metrics keep decreasing with increased stuffing sizes. And the gap between adjacent plots is enlarged each time we double the stuffing size. To estimate the cost of frame stuffing, we introduce a parameter called sensitivity. Sensitivity for each stuffing is defined as the ratio between the accumulated motion metrics obtained at that stuffing and the accumulated motion metrics obtained with no stuffing (i.e. stuffing = 1 byte). From Figures 4.3(a) and 4.3(c), the gaps among different plots are relatively constant for different preloads, so we simply choose the preload time = 0 second. Table 4.1 shows the gain sensitivities for different frame stuffing sizes for both buffers of 512 kbytes and 1024 kbytes. Even with a stuffing size of 200 bytes, the selection result accuracy is close to 97%.

From Figure 4.4, for buffer = 512 kbytes, the recorded average number of states per frame is less than 2^{15} , which is much less than the theoretical value $512K = 2^{19}$. The reason is because a large number of non-optimal intermediate frame states have been eliminated based on Lemma 3.1. Let 2^T represent the theoretical number of states per frame. For buffer = 512 kbytes and stuffing = 200 bytes, the theoretical value $T = \log_2(512k / 200) = 11.36$. Let 2^R represent the recorded average number of states (see Figure 4.4). To evaluate the effectiveness of non-optimal state elimination, we calculate

Table 4.2: Effectiveness of Non-optimal State Elimination at Buffer = 512K

Number of States per Frame ($\log_2$)	Frame Stuffing Step Size (byte)						
	1	2	50	100	200	400	800
Theory	19	18	13.36	12.36	11.36	10.36	9.36
Record	14.43	14.34	12.51	11.81	10.98	10.09	9.25
Reduction (%)	95.79	92.09	44.52	31.70	23.16	17.07	7.34

Table 4.3: Effectiveness of Non-optimal State Elimination at Buffer = 1024K

Number of States per Frame ($\log_2$)	Frame Stuffing Step Size (byte)						
	1	2	50	100	200	400	800
Theory	20	19	14.36	13.36	12.36	11.36	10.36
Record	17.25	16.88	14.03	13.11	12.17	11.22	10.30
Reduction (%)	85.13	77.00	20.45	15.91	12.34	9.25	4.07

the ratio: $(2^T - 2^R)/2^T = 1 - 2^{R-T}$, i.e., the percentage of state reduction due to non-optimal state elimination. From Table 4.2, at no stuffing, the reduction percentage is as high as 95.79% for buffer = 512 kbytes. However, as we increase the stuffing size, the reduction becomes less effective. For instance, at stuffing = 200 bytes, the reduction percentage is 23.16%. This is no surprising result because frame states are spaced out by at least a distance equal to the stuffing size. As the stuffing size increases, it becomes less likely to create non-optimal frame states. Table 4.3 shows the reduction results for buffer = 1024 kbytes. It appears that a larger buffer tends to create a smaller percentage of non-optimal states. Non-optimal state elimination is more effective for small stuffing sizes and small buffers.

4.3.1.3 Performance of Different Algorithms

In this section, we evaluate the performance gains of our proposed reverse frame selection (RFS) algorithm under different bandwidth and buffer size conditions. The performance gains are compared with three other implemented algorithms: Layered JIT, Greedy, and Greedy+WR. We use OFS+n to represent a particular optimal result of RFS for one

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

Figure 4.5: Frame selection results under different bandwidth conditions for (a) buffer = 512 kbytes (b) buffer = 1024 kbytes.

preload since the selection results of OFS and RFS are the same for the same preload. The only difference is that RFS can find all the optimal results for different preloads in just one round. The Z-B algorithm is not compared here because it is the same as OFS+n, where n is the frame stuffing size.

To evaluate the performance gains under different bandwidth conditions, we use fixed preload = 375 kbytes (300 kbps * 10 seconds) and vary the bandwidth from 100 kbps to 500 kbps. Figures 4.5(a) and 4.5(b) show the frame selection results of different algorithms for buffer = 512 kbytes and 1024 kbytes respectively. It can be seen that OFS+100 (OFS with stuffing = 100 bytes) outperforms all the other algorithms under all bandwidth and buffer conditions. And all the OFS+n algorithms, where n is the frame stuffing step size in bytes, outperform the other three algorithms in the low bandwidth regions for both buffers. As expected, the selection results of Greedy+WR are close to the optimal results only in the high bandwidth regions. At low bandwidth, Greedy+WR is always trying to select more I and P frames, even if they have little contribution to the overall gain, due to the large weights assigned to them. Whereas OFS may skip some I and P frames and select those B frames that can contribute more. It seems surprising that the Layered JIT algorithm, which has no motion awareness, outperforms the Greedy

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

Figure 4.6: Frame selection results for different buffers.

algorithm. The reason is perhaps that high motion frames usually have large frame sizes, and the Greedy algorithm always selects the frame currently having the largest weighted motion metric, and completely ignores the cost of consuming large frames, which over consumes the bandwidth resource in the beginning and compromises the overall gain.

Figure 4.6 shows the frame selection results of different algorithms by varying the buffer from 256 kbytes to 2048 kbytes. We set the bandwidth = 300 kbps and use fixed preload time = 10 seconds. It can be seen that all the OFS+n algorithms outperform the other three algorithms under all buffer conditions. It is interesting that the Greedy algorithm has a worse performance when buffer increases from 1024 kbytes to 2048 kbytes. The reason is perhaps that a larger buffer allows the Greedy algorithm to select more large-size frames in the beginning, which consumes most of the bandwidth, and as a result, the overall gain is compromised. Due to the bandwidth constraint, a larger buffer = 2048 kbytes does not yield significant gain.

4.3.2 Simulation for VBR Channels

To simplify the experiment, we use piecewise-CBR channels to approximate the bandwidth variations for VBR channels. Further, we assume the bandwidth can only be one of the values in the set: $\{100, 200, 300, 400, 500\}$ kbps. We also introduce a parameter T , which defines the minimum time interval that a CBR segment can last, or in other words, the VBR channel has the freedom to change its bandwidth every T seconds. We create two random channels, channel 1 and channel 2, with $T = 2$ and $T = 10$ seconds respectively.

4.3.2.1 Performance of Different Algorithms

The frame selection results of four algorithms: JIT, AVE+200, RFS+200, and UPB+200, are compared for both fast-changing channel 1 and slow-changing channel 2. JIT represents the just-in-time algorithm, which always drops the current frame if buffer underflow occurs and reduces the transmission rate if buffer overflow occurs. AVE represents the approach that uses the middle bandwidth value to pre-compute a frame selection path and applies the JIT algorithm on-the-fly to discard frames under bad channel conditions. RFS represents our proposed reverse frame selection algorithm. UPB represents the upper bound computed with our proposed optimal frame selection algorithm assuming known *a priori* channel bandwidth variation behavior, which is unrealistic in practice without an accurate traffic model. For AVE, RFS, and UPB, we use frame stuffing = 200 bytes considering the tradeoff between the reserved number of states and the wasted bandwidth (see section 4.3.1.2).

Figures 4.7(a) and 4.7(b) show the frame selection results of different algorithms over channel 1. While UPB+200 gives the upper bound, both AVE+200 and OPT+200 outperform the JIT algorithm, and AVE+200 has a better performance than RFS+200. The reason is because RFS is optimal on the condition that the new bandwidth will remain constant until the end of the sequence. With bandwidth changes occur so frequent, RFS

CHAPTER 4. REVERSE FRAME SELECTION UNDER DYNAMIC CONSTRAINED RESOURCES

Figure 4.7: Frame selection results over channel 1 for (a) buffer = 512 kbytes (b) buffer = 1024 kbytes, and channel 2 for (c) buffer = 512 kbytes (d) buffer = 1024 kbytes.

obtains many local optimal results over small CBR segments, and the global optimality is severely deviated. On the other hand, the fast-changing channel rate may reflect the middle bandwidth approach and results in a better performance for AVE.

Figures 4.7(c) and 4.7(d) show the frame selection results over slow-changing channel 2. This time, RFS+200 outperforms AVE+200 and JIT especially for the large buffer = 1024 kbytes. This is because with less frequent bandwidth changes, RFS+200 can better preserve the global optimality over longer CBR segments. The performance of AVE+200 is severely degraded by the long-term low bandwidth CBR segments. And

even when there is a long-term high bandwidth CBR segment, AVE+200 can not select more frames to compensate the loss, and thus deviates largely from the global optimal result of UPB+200.

4.3.2.2 Channel Sampling Error and Overhead

For VBR channel simulation, we assume the bandwidth can only be one of the values in the set $\{100, 200, 300, 400, 500\}$ kbps. We run RFS 5 times for each of the bandwidth samples and record all the states for each sample separately. A CBR segment can then be retrieved in realtime from the recorded states for the CBR bandwidth sample. However, there are two problems for this approach:

- (i) It is highly unlikely that the realtime bandwidth is exactly one of the values in the set, and consequently there will be bandwidth sampling errors. For example, bandwidth = 119 kbps may be classified to 100 kbps.
- (ii) Even with non-optimal state elimination and frame stuffing, the average number of states per frame to record is still too high. From Table 4.2, for buffer = 512 kbytes and stuffing = 200 bytes, the average number of states per frame is close to 2K ($2^{10.98}$).

For the first problem, we can use a fine-grained bandwidth sampling strategy. For example, for a wireless network with channel rate variation ranges from 100 kbps to 300 kbps, we can sample the range at a step of 20 kbps. In addition, it is not necessarily bad to classify the realtime bandwidth to a pre-sampled value because we can still transmit data at the real channel rate and raise the buffer status to benefit future CBR segments.

For the second problem, which becomes even more important with fine-grained bandwidth sampling strategy, we propose to use run-length coding to group the states. We find that at each frame, many consecutive states point to the same frame to be selected

Table 4.4: Number of State Ranges per Frame (stuffing = 200 bytes)

State Ranges per Frame at buffer (byte)	Bandwidth (kbps)				
	100	200	300	400	500
512k	50.02	48.08	82.64	70.06	54.74
1024k	78.05	61.26	155.79	148.77	95.96

next. For example, at frame = 9000 (RFS run for buffer = 512 kbytes and bandwidth = 300 kbps with stuffing = 200 bytes), there are a total of 2548 states. However, all the states above buffer status = 8280 bytes point to the next frame = 9001, and all the states below 8280 bytes, including 8280 bytes, point to the next frame = 9058. Using run-length coding, we only need to record two state ranges: (8280, 9001) and (0, 9058), where the first number in each pair represents the range boundary and the second number indicates the next frame to select. Table 4.4 shows the average number of state ranges per frame under different bandwidth conditions. We see that for bandwidth = 300 kbps with stuffing = 200 bytes, the average number of state ranges per frame is around 83 for buffer = 512 kbytes and 156 for buffer = 1024 kbytes. Because the number of state ranges per frame is much less than the number of states per frame, realtime path retrieval becomes even faster.

4.4 Summary

In this chapter, we have studied the problem of optimal frame selection for streaming stored video over both CBR and VBR channels using dynamic programming. We have proposed the RFS algorithm, which can find the optimal results for any preload in one round for CBR channels. In addition, the proposed RFS algorithm has been smartly extended for the VBR channels, which can be modelled as piecewise CBR channels. Experimental results have demonstrated that with modest complexity our proposed RFS algorithm achieves much better performance than the common just-in-time algorithm, especially in the case of slow-changing VBR channels.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we have presented our current work on multimedia content adaptation based on MPEG-21 framework. Our objective is to provide end-to-end QoS by dynamically adapting content to different terminal and network conditions.

MPEG-21 digital item adaptation standard defines description tools used to assist in the adaptation process. For FGS video, we first obtain terminal and network constraints using the Usage Environment Description tool. Then, the AdaptationQoS tool is used to select the optimal adaptation operation that can maximize user utility value while meeting all the environment constraints.

Given bandwidth and buffer constraints, we proposed an optimal frame selection (OFS) algorithm based on dynamic programming. The OFS algorithm is able to find an optimal set of frames for maximum delivery of the motion information content in a video for better perceptual quality.

In addition, we have also proposed a reverse frame selection (RFS) algorithm for video transmission over both CBR and VBR channels. For CBR channels, the proposed RFS algorithm can find all the optimal results for different preloads by running just once, whereas other algorithms have to use exhaustive searching to get all the optimal results. We further extended the RFS algorithm to solve the problem of VBR channels that can

be regarded as piecewise CBR channels. Simulation results show that RFS has a very good performance for slow-changing VBR channels.

5.2 Future work

Our proposed algorithms generally cannot be directly applied for live or interactive video applications, which only has a limited knowledge of frame sizes. As discussed in chapter 3, one of the future works is to study window-based online smoothing techniques by introducing delay or developing frame sizes prediction models to achieve suboptimal video quality.

Another limitation of the proposed algorithms is that they consider only temporal sampling. Future trend of video coding is to enable full scalability where a single bit-stream supports scaling in temporal, spatial, and quality three-dimensional domains. Smoothing techniques should also consider changing the source rate. Joint design of source rate control and bandwidth smoothing should be studied for more flexible adaptive video streaming.

The third problem of the proposed frame selection algorithms is that they all have to perform smoothing for each given buffer size. This may limit its application since terminal devices can vary largely in their physical configurations. One of the solutions to the buffer problem is to compute a smoothing plan without frame dropping and find the minimum buffer requirement. During transmission, if the client buffer is smaller than the minimum buffer requirement, on-the-fly adaptation is applied. Another solution is to explore the relationships among bandwidth change points and store the transmission plans for all buffers in a compact manner as in [104].

References

- [1] R. Mohan, J. R. Smith, and C.-S. Li, “Adapting multimedia internet content for universal access,” *IEEE Transactions on Multimedia*, vol. 1, pp. 104–114, Mar. 1999.
- [2] A. Perkis, Y. Abdeljaoued, C. Christopoulos, T. Ebrahimi, and J. Chicharo, “Universal multimedia access from wired and wireless systems,” *Birkhauser Boston Transactions on Circuits, Systems and Signal Processing (Special Issue on Multimedia Communication)*, vol. 20, pp. 387–402, Sep. 2001.
- [3] P. van Beek, J. R. Smith, T. Ebrahimi, T. Suzuki, and J. Askelof, “Metadata-driven multimedia access,” *IEEE Signal Processing Magazine*, vol. 20, pp. 40–52, Mar. 2003.
- [4] J. R. Smith, R. Mohan, and C.-S. Li, “Scalable multimedia delivery for pervasive computing,” in *Proc. ACM Multimedia*, pp. 131–140, 1999.
- [5] F. Pereira and I. Burnett, “Universal multimedia experiences for tomorrow,” *IEEE Signal Processing Magazine*, vol. 20, pp. 63–73, Mar. 2003.
- [6] B. L. Tseng, C.-Y. Lin, and J. R. Smith, “Using MPEG-7 and MPEG-21 for personalizing video,” *IEEE Multimedia*, vol. 10, pp. 42–52, Oct.–Dec. 2003.
- [7] J. Bormans, J. Gelissen, and A. Perkis, “MPEG-21: The 21st century multimedia framework,” *IEEE Signal Processing Magazine*, vol. 20, pp. 53–62, Mar. 2003.
- [8] I. Burnett, R. V. de Walle, K. Hill, J. Bormans, and F. Pereira, “MPEG-21: Goals and achievements,” *IEEE Multimedia*, vol. 10, pp. 60–70, Oct.–Dec. 2003.

REFERENCES

- [9] J. Bormans and K. Hill, “MPEG-21 overview v.5,” *ISO/IEC JTC1/SC29/WG11/N5231*, Shanghai, Oct. 2002.
- [10] A. Vetro, “MPEG-21 digital item adaptation: Enabling universal multimedia access,” *IEEE Multimedia*, vol. 11, pp. 84–87, Jan.–Mar. 2004.
- [11] “Information technology–multimedia framework (MPEG-21)–Part 1: Vision, Technology and Strategy,” *ISO/IEC JTC1/SC29/WG11/N6388*, Mar. 2004.
- [12] “Information technology–multimedia framework (MPEG-21)–Part 2: Digital Item Declaration,” *ISO/IEC FDIS 21000-2*, Apr. 2005.
- [13] L. Boszormenyi, H. Hellwagner, H. Kosch, M. Libsie, and S. Podlipnig, “Metadata driven adaptation in the ADMITS project,” *Signal Processing: Image Communication*, vol. 18, pp. 749–766, Sep. 2003.
- [14] S.-F. Chang and A. Vetro, “Video adaptation: concepts, technologies, and open issues,” in *Proceedings of the IEEE*, vol. 93, pp. 148–158, Jan. 2005.
- [15] A. Vetro, C. Christopoulos, and H. Sun, “Video transcoding architectures and techniques: an overview,” *IEEE Signal Processing Magazine*, vol. 20, pp. 18–29, Mar. 2003.
- [16] J.-R. Ohm, “Advances in scalable video coding,” *Proceedings of the IEEE*, vol. 93, pp. 42–56, Jan. 2005.
- [17] M. Karczewisz and R. Kurceren, “The sp- and si- frames design for h.264/avc”, journal = IEEE Transactions on Circuits and Systems for Video Technology, volume = 13, pages = 637–644, month = Jul., year = 2003,”
- [18] J. Lim, M. Kim, J.-N. Kim, and K. Kim, “Semantic transcoding of video based on regions of interest,” in *VCIP*, pp. 1232–1243, Jul. 2003.
- [19] M. Irani and P. Anandan, “Video indexing based on mosaic representation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 86, pp. 905–921, May 1998.

REFERENCES

- [20] G. D. L. Reyes, A. Reibman, S.-F. Chang, and J.-I. Chuang, "Error-resilient transaction for video over wireless channels," *IEEE Journal on Selected Areas in Communications*, pp. 1063–1074, Jun.
- [21] D. Mukherjee, A. Said, and S. Liu, "A framework for fully format-independent adaptation of scalable bit-streams," *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Video Adaptation*, Oct. 2005.
- [22] T. K. Tan, K. K. Pang, and K. N. Ngan, "A frequency scalable coding scheme employing pyramid and subband techniques," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 4, pp. 203–207, April 1994.
- [23] H. Sun, W. Kwok, and J. W. Zdepski, "Architectures for MPEG compressed bit-stream scaling," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, pp. 191–199, April 1996.
- [24] G. J. Conklin and S. S. Hemami, "A comparison of temporal scalability techniques," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, pp. 909–919, Sept. 1999.
- [25] D. Wilson and M. Ghanbari, "Exploiting interlayer correlation of SNR scalable video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, pp. 783–797, Aug. 1999.
- [26] M. Domanski, A. Luczak, and S. Mackowiak, "Spatial-temporal scalability for MPEG video coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, pp. 1088–1093, Oct. 2000.
- [27] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, Mar. 2001.
- [28] M. van der Schaar and H. Radha, "A hybrid temporal-SNR fine-granular scalability for internet video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 318–331, March 2001.

REFERENCES

- [29] H. M. Radha, M. van der Schaar, and Y. Chen, “The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP,” *IEEE Transactions on Multimedia*, vol. 3, pp. 53–68, Mar. 2001.
- [30] J.-R. Ohm, M. van der Shaar, and J. W. Woods, “Interframe wavelet coding—motion picture representation for universal scalability,” *Singal Processing: Image Communication*, vol. 19, pp. 877–908, 2004.
- [31] T. V. Lakshman, A. Ortega, and A. R. Reibman, “VBR video: Tradeoffs and potentials,” in *Proceedings of the IEEE*, vol. 86, pp. 952–973, May 1998.
- [32] I. Dalgic and F. A. Tobagi, “Performance evaluation of ATM networks carrying constant and variable bit-rate video traffic,” *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1115–1131, Aug. 1997.
- [33] M. W. Garrett and W. Willinger, “Analysis, modeling and generation of self-similar VBR video traffic,” in *ACM SIGCOMM*, pp. 269–280, Aug. 1994.
- [34] M. Grossglauser, S. Keshav, and D. N. C. Tse, “RCBR: A simple and efficient service for multiple time-scale traffic,” *IEEE Transactions on Networking*, vol. 5, Dec. 1997.
- [35] J. D. Salehi, Z. L. Zhang, J. F. Kurose, and D. Towsley, “Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing,” *IEEE Transactions on Networking*, vol. 6, pp. 397–410, Aug. 1998.
- [36] W.-C. Feng, “Rate-constrained bandwidth smoothing for the delivery of stored video,” in *SPIE Multimedia Compting and Networking*, pp. 316–327, Feb. 1997.
- [37] W.-C. Feng and J. Rexford, “A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video,” in *Proceedings of IEEE INFOCOM97*, pp. 58–66, Apr. 1997.
- [38] W.-C. Feng, B. Krishnaswami, and A. Prabhudev, “Proactive buffer management for the streamed delivery of stored video,” in *ACM Multimedia*, pp. 285–290, Sep. 1998.

REFERENCES

- [39] I. S. Burnett, F. Pereira, R. V. de Walle, and R. Koenen, *The MPEG-21 Book*. John Wiley & Sons, Ltd, 2006.
- [40] F. Pereira, J. R. Smith, and A. Vetro, “Introduction to the special section on mpeg-21,” *IEEE Transactions on Multimedia*, vol. 7, pp. 397–399, Jun. 2005.
- [41] “Information technology–multimedia framework (MPEG-21)–Part 7: Digital Item Adaptation,” *ISO/IEC FDIS 21000-7*, Mar. 2004.
- [42] I. S. Burnett, S. J. Davis, and G. M. Drury, “MPEG-21 digital item declaration and identification–principles and compression,” *IEEE Transactions on Multimedia*, vol. 7, pp. 400–407, Jun. 2005.
- [43] S. Hashemipour and M. Ali, “Mpeg-21 & didl: Dawn of a new multimedia era,” in *IEEE International Symposium on Consumer Electronics*, pp. 91–95, Sep. 2004.
- [44] “Information technology–multimedia framework (MPEG-21)–Part 3: Digital Item Identification,” *ISO/IEC JTC1/SC29/WG11/N4939*, Jul. 2002.
- [45] “Information technology–multimedia framework (MPEG-21)–Part 4: Intellectual Property Management and Protection,” *ISO/IEC JTC1/SC29/WG11/N7196*, Apr. 2005.
- [46] “Information technology–multimedia framework (MPEG-21)–Part 5: Rights Expression Language,” *ISO/IEC FDIS 21000-5*, Jul. 2003.
- [47] X. Wang, “MPEG-21 rights expression language: Enabling interoperable digital rights management,” *IEEE Multimedia*, vol. 11, pp. 84–87, Oct.–Dec. 2004.
- [48] X. Wang, T. DeMartini, B. Wragg, M. Paramasivam, and C. Barlas, “The MPEG-21 rights expression language and rights data dictionary,” *IEEE Transactions on Multimedia*, vol. 7, pp. 408–417, Jun. 2005.
- [49] “Information technology–multimedia framework (MPEG-21)–Part 6: Rights Data Dictionary,” *ISO/IEC FDIS 21000-6*, Jul. 2003.

REFERENCES

- [50] A. Vetro and C. Timmerer, “Digital item adaptation: Overview of standardization and research activities,” *IEEE Transactions on Multimedia*, vol. 7, pp. 418–426, Jun. 2005.
- [51] “Information technology–multimedia framework (MPEG-21)–Part 8: Reference Software,” *ISO/IEC FDIS 21000-8*, May 2005.
- [52] “Information technology–multimedia framework (MPEG-21)–Part 9: File Format,” *ISO/IEC FDIS 21000-9*, Jan. 2005.
- [53] “Information technology–multimedia framework (MPEG-21)–Part 10: Digital Item Processing,” *ISO/IEC JTC1/SC29/WG11/N6780*, Oct. 2004.
- [54] F. D. Keukelaere, S. D. Zutter, and R. V. de Walle, “MPEG-21 digital item processing,” *IEEE Transactions on Multimedia*, vol. 7, pp. 427–434, Jun. 2005.
- [55] “Information technology–multimedia framework (MPEG-21)–Part 11: Evaluation Tools for Persistent Association Technologies,” *ISO/IEC TR 21000-11*, Mar. 2004.
- [56] “Information technology–multimedia framework (MPEG-21)–Part 12: Test Bed for MPEG-21 Resource Delivery,” *ISO/IEC TR 21000-12*, Jan. 2005.
- [57] “Information technology–multimedia framework (MPEG-21)–Part 14: Conformance Testing,” *ISO/IEC JTC1/SC29/WG11/N7213*, Apr. 2005.
- [58] “Information technology–multimedia framework (MPEG-21)–Part 15: Event Reporting,” *ISO/IEC JTC1/SC29/WG11/N6783*, Oct. 2004.
- [59] “Information technology–multimedia framework (MPEG-21)–Part 16: Binary Format,” *ISO/IEC FDIS 21000-16*, May. 2005.
- [60] “Information technology–multimedia framework (MPEG-21)–Part 17: Fragment Identification for MPEG Resources,” *ISO/IEC JTC1/SC29/WG11/N6948*, Jan. 2005.

REFERENCES

- [61] G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers, and M. Amiellh, "Bitstream syntax description: A tool for multimedia resource adaptation within MPEG-21," *Signal Processing: Image Communication*, vol. 18, pp. 721–747, 2003.
- [62] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream syntax description-based adaptation in streaming and constrained environments," *IEEE Transactions on Multimedia*, vol. 7, pp. 463–470, Jun. 2005.
- [63] C. Timmerer, G. Panis, H. Kosch, J. Heuer, H. Hellwagner, and A. Hutter, "Coding format independent multimedia content adaptation using XML," in *Proceedings of SPIE International Symposium ITCOM 2003*, vol. 5242, Sep. 2003.
- [64] D. Mukherjee, E. Delfosse, J.-G. Kim, and Y. Wang, "Optimal adaptation decision-taking for terminal and network quality-of-service," *IEEE Transactions on Multimedia*, vol. 7, pp. 454–462, Jun. 2005.
- [65] A. R. Reibman and A. W. Berger, "Traffic descriptors for VBR video teleconferencing over atm networks," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 329–339, Jun. 1995.
- [66] Z.-L. Zhang, J. D. Salehi, and D. Towsley, "Smoothing, statistical multiplexing, and call admission control for stored video," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1148–1166, Aug. 1997.
- [67] V. S. Frost and B. Melamed, "Traffic modeling for telecommunications networks," *IEEE Communications Magazine*, pp. 70–81, Mar. 1994.
- [68] E. W. Knightly and H. Zhang, "Traffic characterization and switch utilization using a deterministic bounding interval dependent traffic model," in *INFOCOM*, pp. 1137–1145, Apr. 1995.
- [69] D. E. Wrege, E. W. knightly, H. Zhang, and J. Liebeherr, "Deterministic delay bounds for VBR video in packet-switching networks: Fundamental limits and practical trade-offs," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 352–362, Jun. 1996.

REFERENCES

- [70] E. W. Knightly and H. Zhang, "D-BIND: An accurate traffic model for providing QoS guarantees to VBR traffic," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 219–231, Apr. 1997.
- [71] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 368–379, Apr. 1990.
- [72] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, Jan. 1991.
- [73] M. Butto, E. Cavallero, and A. Tonietti, "Effectiveness of the "leaky bucket" policing mechanism in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 335–342, Apr. 1991.
- [74] R. L. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Transactions on Information Theory*, vol. 37, pp. 132–141, Jan. 1991.
- [75] C. Tryfonas and A. Varma, "Efficient algorithms for computation of the burstiness curve of video sources," *IEEE Transactions on Multimedia*, vol. 6, pp. 862–875, Dec. 2004.
- [76] A. R. Reibman and B. G. Haskell, "Constraints on variable bit-rate video for ATM networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 361–372, Dec. 1992.
- [77] W. chi Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," in *Proceedings of SPIE Multimedia Computing and Networking*, pp. 234–242, Mar. 1995.
- [78] W. chi Feng and S. Sechrest, "Critical bandwidth allocation for the delivery of compressed video," *Computer Communications*, vol. 18, pp. 709–717, Oct. 1995.
- [79] W.-C. Feng, F. Jahanian, and S. Sechrest, "An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video," *Multimedia Systems*, vol. 5, pp. 297–309, Sep. 1997.

REFERENCES

- [80] J. M. McManus and K. W. Ross, "Video-on-demand over ATM: Constant-rate transmission and transport," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1087–1098, Aug. 1996.
- [81] J. M. McManus and K. W. Ross, "A dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks," *Telecommunication Systems*, vol. 9, no. 2, pp. 223–247, 1998.
- [82] W. chi Feng and J. Rexford, "Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video," *IEEE Transactions on Multimedia*, vol. 1, pp. 302–312, Sep. 1999.
- [83] D. Ye, J. C. Barker, Z. Xiong, and W. Zhu, "Wavelet-based VBR video traffic smoothing," *IEEE Transactions on Multimedia*, vol. 6, pp. 611–623, Aug. 2004.
- [84] T. Gan, K.-K. Ma, and L. Zhang, "Dual-plan bandwidth smoothing for layer-encoded video," *IEEE Transactions on Multimedia*, vol. 7, pp. 379–392, Apr. 2005.
- [85] S. S. Lam, S. Chow, and D. K. Y. Yau, "A lossless smoothing algorithm for compressed video," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 697–708, Oct. 1996.
- [86] S. Sen, J. L. Rexford, J. K. Dey, J. F. Kurose, and D. F. Towsley, "Online smoothing of variable-bit-rate streaming video," *IEEE Transactions on Multimedia*, vol. 2, pp. 37–48, Mar. 2000.
- [87] P. Bocheck and S.-F. Chang, "Content-based dynamic resource allocation for VBR video in bandwidth limited networks," in *Sixth International Workshop on Quality of Service (IWQoS)*, pp. 204–206, May 1998.
- [88] S.-F. Chang and P. Bocheck, "Principles and applications of content-aware video communication," in *ISCAS*, pp. 33–36, May 2000.
- [89] M. Wu, R. A. Joyce, H.-S. Wong, L. Guan, and S.-Y. Kung, "Dynamic resource allocation via video content and short-term traffic statistics," *IEEE Transactions on Multimedia*, vol. 3, pp. 186–199, Jun. 2001.

REFERENCES

- [90] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal, and R. P. Tsang, "Efficient selective frame discard algorithms for stored video delivery across resource constrained networks," in *INFOCOM*, pp. 472–479, Mar. 1999.
- [91] J. K.-Y. Ng and S. Song, "A video smoothing algorithm for transmitting MPEG video over limited bandwidth," in *Proceedings of Fourth International Workshop on Real-Time Computing Systems and Applications*, pp. 229–236, Oct. 1997.
- [92] X. S. Zhou and S.-P. Liou, "Optimal nonlinear sampling for video streaming at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 535–544, Jun. 2002.
- [93] H. Yi, D. Rajan, and L.-T. Chia, "Global motion compensated key frame extraction from compressed videos," in *ICASSP*, Mar. 2005.
- [94] H. Yi, D. Rajan, and L.-T. Chia, "A new motion histogram to index motion content in video segment," in *Pattern Recognition Letters*, 2005.
- [95] Y.-F. Ma and H.-J. Zhang, "A new perceived motion based shot content representation," in *ICIP*, 2001.
- [96] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. MA:MIT Press, 1990.
- [97] W.-C. Feng and M. Liu, "Critical bandwidth allocation techniques for stored video delivery across best-effort networks," in *20th IEEE International Conference on Distributed Computing Systems (ICDCS'00)*, pp. 56–63, 2000.
- [98] T. Stockhammer, H. Jenkac, and G. Kuhn, "Streaming video over variable vit-rate wireless channels," *IEEE Transactions on Multimedia*, vol. 6, pp. 268–277, Apr. 2004.
- [99] C.-Y. Hsu, A. Ortega, and A. R. Reibman, "Joint selection of source and channel rate for VBR video transmission under atm policing constraints," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 1016–1028.

REFERENCES

- [100] P. A. Chou and Z. Miao, “Rate-distortion optimized streaming of packetized media,” *Microsoft Research Technical Report*, Feb. 2001.
- [101] J. Chakareski and P. A. Chou, “Application layer error-correction coding for rate-distortion optimized streaming to wireless clients,” *IEEE Transactions on Communications*, vol. 52, pp. 1675–1687, Oct. 2004.
- [102] J. Chakareski, S. Han, and B. Girod, “Layered coding vs. multiple descriptions for video streaming over multiple paths,” *Multimedia Systems*, pp. 275–285, Jan. 2005.
- [103] M. Handley, S. Floyd, J. Padhye, and J. Widmer, “TCP friendly rate control TFRC: Protocol specification,” *RFC 3448, Internet Engineering Task Force*, Jan. 2003.
- [104] S. G. Rao and S. V. Raghavan, “Fast techniques for the optimal smoothing of stored video,” *Multimedia Systems*, vol. 7, pp. 222–233, May 1999.

Publications

- [1] Dayong Tao, Jianfei Cai, Haoran Yi, Deepu Rajan, Liang-Tien Chia, and King Ng Ngan, “Dynamic Programming-Based Reverse Frame Selection for VBR Video Delivery Under Constrained Resources,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 11, Nov. 2006, pp. 1362-1375.
- [2] Dayong Tao and Jianfei Cai, “Optimal Reverse Frame Selection for Stored Video Delivery under Constrained Resources,” *IS&T/SPIE VCIP 2007, San Jose, California, USA*.
- [3] Dayong Tao and Jianfei Cai, “A Novel Reverse Frame Selection Scheme for Video Streaming over VBR Channels,” *IEEE ICME 2006, Toronto, Ontario, Canada*, pp. 1845-1848.
- [4] Dayong Tao and Jianfei Cai, “VBR Video Delivery under Constrained Resources using Motion-Aware Optimal Frame Selection,” *IEEE ISCAS 2006, Island of Kos, Greece*, pp. 1327-1330.