Although efficient compression was not the only first-priority requirement in MPEG-4 (as it had been for MPEG-1 and MPEG-2), it is clear that it has been a central requirement in MPEG-4 in the sense that, whatever the type of data that had to be represented by binary encoding (e.g., shape for video objects, facial animation parameters for 3D facial models, or even scene composition data), the target was always to reach the smallest number of bits for a certain level of quality. Although some people claim that efficient compression of data is not a must today because of the growing availability of bandwidth, MPEG always acknowledged that bandwidth resources (either for transmission or for storage) are still limited, and thus efficient compression is required. This is even truer for some relevant recent transmission cases, such as the Internet and mobile networks, where bandwidth limitations and efficient compression are major issues. However, in the context of MPEG-4, efficient compression must be balanced against other major requirements, such as those related to interactivity capabilities (which have a price in terms of compression efficiency compared with noninteractive representation schemes), if new functionalities not supported by frame-based coding schemes are to be provided.

As noted, the MPEG-4 requirements' development process has been evolving since the beginning of the standardization process; this evolution has been targeting the inclusion of additional requirements related to functionalities, which fit and complement well the MPEG-4 vision. Because the MPEG-4 standardization process is not yet finished, it cannot be said that all the requirements have been addressed by means of an MPEG-4 tool. However, it is possible to say that all of the not-yet-addressed requirements are either being worked on or should be removed in a short time if the industries do not show sufficient support to move to the technical development phase.

1.1.3 Tools

The major trends mentioned—notably the mounting presence of audiovisual media on all networks, increasing mobility, and growing interactivity—have driven, and continue to drive, the development of the MPEG-4 standard.

To address the identified functionalities and requirements [N4319], a set of tools was developed to perform the following functions [Pere99]:

- Efficiently represent a number of data types through media codecs
  - Video from very low bit rates to very high-quality conditions
  - Music and speech data for a very wide bit-rate range, from transparent music to very low bit-rate speech
  - Generic dynamic 3D objects as well as specific objects such as human faces and bodies
  - Synthetic speech and music, including support for 3D audio spaces
  - Text and graphics

Google Exhibit No. 1021
Page 025

Google Exhibit No. 1063
Page 0789

- Provide fine granularity scalability in the quality, temporal, and spatial dimensions
- Provide, in the encoding layer, resilience to residual errors for the various data types, especially under difficult channel conditions such as mobile ones
- Independently represent the various objects in the scene, allowing independent access for their manipulation and reuse
- Compose audio and visual (natural and synthetic) objects into one audiovisual scene in a synchronized way
- Describe the objects and the events in the scene
- Provide interaction and hyperlinking capabilities
- Manage and protect intellectual property on audiovisual content and algorithms, so that only authorized users have access to the content
- Provide a delivery-media-independent representation format, to transparently cross the borders of different delivery environments

The major difference with previous audiovisual coding standards, at the basis of the new functionalities, is the object-based audiovisual representation model that underpins MPEG-4 (see Figure 1.1). An object-based scene is built using individual independent objects that have relationships in space and time. This representation approach offers a number of advantages: First, different object types may have different suitable coded representations—for example, a synthetic moving head is best represented using animation parameters, whereas video benefits from a smart representation of pixel values. Second, it allows the harmonious integration of different types of data into one scene, notably with natural and synthetic origins—for example, an animated cartoon character in a real world environment, or a real person in a virtual studio set. Third, interacting with the objects and hyperlinking from them is

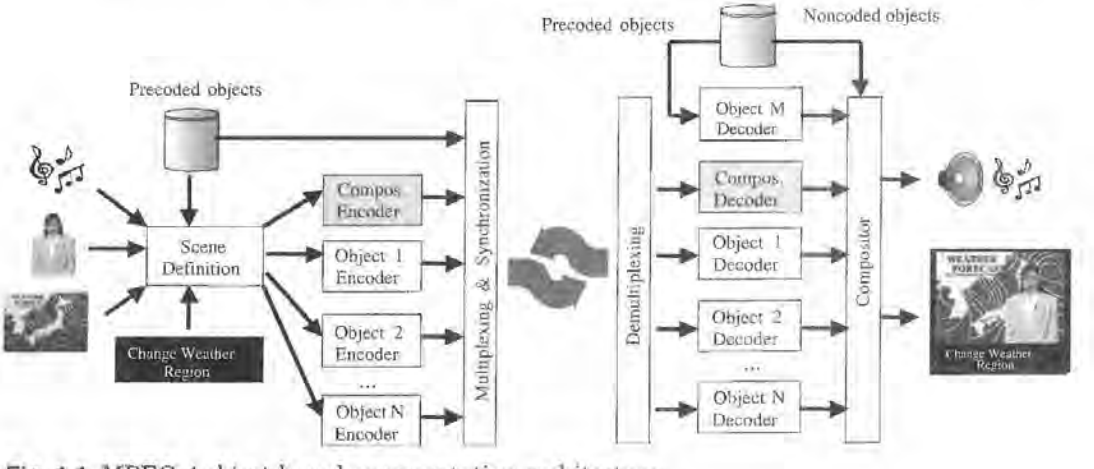


Fig. 1.1 MPEG-4 object-based representation architecture.

Google Exhibit No. 1021
Page 026

Google Exhibit No. 1063
Page 0790

now feasible. There are many more advantages—such as the selective spending of bits, the easy reuse of content without transcoding, and the provision of sophisticated coding solutions for scalable content on the Internet—all of them resulting from the adoption of the object-based representation model.

The applications that benefit from the new concepts and functionalities are found in many (and very different) environments [N2724]. Therefore, MPEG-4 is constructed as a toolbox rather than a monolithic standard, using profiles that provide solutions for these different application settings (see Chapter 13). This means that, although MPEG-4 is a standard comprising a vast array of technologies, it is structured so that solutions are available at the measure of the needs. It is the task of each implementer to extract from the MPEG-4 standard the technical solutions adequate to his or her needs (likely a small subset of the standardized tools) by choosing the adequate profiling combination.

Because a standard is always a constraint on freedom, it is important to make it as minimally constraining as possible [Chia97]. To MPEG, this means a standard must offer maximum advantages by specifying the minimum necessary, allowing for competition and evolution of the technology in the *non-normative areas;* these non-normative areas correspond to the tools for which normative specification is not essential for interoperability. On the contrary, normative tools are those defined by the standard whose specification is essential for interoperability. For example, whereas video segmentation and rate control are non-normative tools, the decoding process needs to be normative. The strategy of specifying the minimum for maximum usability ensures that good use can be made of the continuous technical improvements in the relevant technical areas. The consequence is that better non-normative tools can always be used, even after the standard is finalized, and it is possible to rely on competition for obtaining ever-better results. In fact, it is through the non-normative tools that products will distinguish themselves, which only reinforces the importance of this type of tools.

1.1.4 Applications

MPEG-4 wants to address a wide range of applications, many of them completely new, as there are very new functionalities to take benefit from, and many others improved regarding those already available [N2724]. Unlike MPEG-2 where the *killer application* was digital television, in a first approach just understood as the digital translation of the rather old analog version, MPEG-4 does not target a major and exclusive killer application but opens many new frontiers. Playing with audiovisual scenes and creating, reusing, accessing, and consuming audiovisual content will become easier and more powerful. New and richer applications can be developed, for example, in enhanced broadcasting, remote surveillance, personal communications, games, mobile multimedia, and virtual environments. MPEG-4 allows services combining the traditionally different service models: broadcast, (online) interaction, and communication. As such, MPEG-4 addresses *convergence,* under-

Google Exhibit No. 1021
Page 027

Google Exhibit No. 1063
Page 0791

stood as the proliferation of audiovisual information in all kinds of services and on all types of (access) networks.

The MPEG-4 Applications document [N2724] describes application examples benefiting from the MPEG-4 technology that will serve as inspiration to the industry to create many more exciting applications. In this sense, MPEG-4 is a technical playground where many application constructions may be built by the manufacturers and service providers. The MPEG-4 Applications document suggests the following applications, using both audio and visual information or just one of them: broadcast, collaborative scene visualization, content-based storage and retrieval, digital amplitude modulation (AM) broadcasting, digital television set-top box, DVD, infotainment, mobile multimedia, real-time communications, streaming video on the Internet/Intranet, studio and television postproduction, surveillance, and virtual meetings.

1.2 Formal Standardization Process

MPEG (Moving Picture Experts Group) formally called *Coding of Moving Picture and Audio* is Working Group 11 (WG11) of Subcommittee 29 (Coding of Audio, Picture, Multimedia and Hypermedia Information) of the ISO/IEC (International Organization for Standardization/International Electrotechnical Commission) Joint Technical Committee 1 (JTC1) [MPEG]. JTC1's scope is standardization in the field of information technology: According to JTC1 [JTC1], information technology includes the specification, design, and development of systems and tools dealing with the capture, representation, processing, security, transfer, interchange, presentation, management, organization, storage, and retrieval of information.

Since the major objective of MPEG is to produce technical specifications, it is important to have an idea, even if brief, about the types of documents that can be produced by a group like MPEG. Through JTC1, MPEG can produce the types of documents presented in the following:[5]

- **International standards:** These are documents with the technical specification of the standard. After the first edition of a standard, new editions may be published, notably to incorporate in one document the previous edition and all the amendments and corrigenda meanwhile issued. Whenever a new edition for a certain Part of the standard includes new technology (and thus is not just a compilation of previously approved technology), it must be approved following a process similar to amendments.

5. For each type of document, the various successive stages, as well as the minimum time the voting period by national standardization bodies may take for each stage (this means the minimum time to take the promotion of a certain document from one stage to the next), are indicated.

CHAPTER 2

MPEG-4 Overview

by Olivier Avaro, Rob Koenen, and Fernando Pereira

Keywords: MPEG-4 design goals, multimedia architecture, audiovisual object representation, MPEG-4 tools, multimedia standards, MPEG-4 applications

The MPEG-4 project started in 1992 with the goal of setting a standard for *Very Low Bit Rate Audiovisual Coding* [N271]. The initial participants were companies from the consumer electronics, computer, and telecom industries, as well as academia. There was the promise of great new coding methodologies that would greatly improve (video) compression efficiency over the existing hybrid DCT-based coding schemes. The participants discussed whether MPEG-4 should be centered on *Very Low Bit Rates* or on *Very High Compression Efficiency*. Unfortunately, the "Great Plan" was built on quicksand, which became apparent in July of 1994 at the Grimstadt MPEG meeting. At that point, MPEG altered the goals of MPEG-4 considerably. After reassessing developments in the multimedia arena, MPEG redefined the goals to align them with the anticipated evolution in the production, delivery, and consumption of audiovisual content, as summarized in Table 2.1. The *linear* approach that had dominated the multimedia industry at all stages of the content lifecycle was replaced by a *nonlinear, object-based, interactive* way of dealing with audiovisual data. As a consequence, to reflect the focus of the work more accurately, the name of the MPEG-4 project was changed to *Coding of Audio-Visual Objects* [N998]. The scope of the MPEG-4 standard is defined in more detail in the MPEG Requirements document [N4319]. At the time of writing (February 2002), the MPEG-4 standard was

Google Exhibit No. 1021
Page 029

Google Exhibit No. 1063
Page 0793

Table 2.1 Evolution of multimedia content production, delivery, and consumption

	Traditional	MPEG-4
Production	• Mostly 2D content • Natural content, produced with cameras and microphones • Composition (mixing) of objects at production stage	• 2D and 3D content • Content with hybrid natural and computer-generated audio and video • Rectangular and presegmented video content • Explicit coding of composition of separate objects
Delivery	• Few networks carry audiovisual information (satellite, LAN, ISDN) • Many networks have their own content representation scheme • Communications and delivery over homogeneous networks	• Virtually all networks carry audiovisual information • Common content representation scheme • Communications and delivery over heterogeneous networks (multiple network types)
Consumption	• Mostly passive content • Content precomposed	• Composition of objects at consumption stage • Information is read, seen, and heard in an interactive and personalized way • More and more information is audiovisual and digital

largely finalized, although some additional tools for advanced functionality were still under development. A complete overview of the MPEG-4 technology is provided in the MPEG-4 Overview [N4505], which is updated at every MPEG meeting—about once every quarter.

2.1 Design Goals

The main MPEG-4 design goals, as formulated in 1994, can be summarized as follows:

1. To provide the technological foundations to represent multimedia content for a wide range of multimedia services and networks. To provide a common representation basis encompassing the interactive, broadcast, and conversational paradigms. To allow for seamless content usage across these different types of services.
2. To extend the interactivity found in text and images to content having a temporal dimension. To provide for this type of content client-side interactivity (i.e., without the need for a return channel), as well as server-side interactivity, exploiting an on-line connection.

Google Exhibit No. 1021
Page 030

Google Exhibit No. 1063
Page 0794

3. To extend multimedia content usage from high-bandwidth networks to all types of networks, including those with low bandwidth and high error rates. MPEG calls this *Universal Access*.
4. To integrate different media types into a single framework, giving each its own optimized coded representation.
5. To offer rich interactivity to content authors and content users.
6. To allow for consumption of the audiovisual content that respects the usage rights that are attached to it.

For all its complexity and power, the MPEG-4 standard is based on a very simple concept: an audiovisual scene is composed of audiovisual objects whose properties are defined using a scene description language. This concept allows

- Interaction with individual elements in the audiovisual content, named *audiovisual objects*
- Adaptation of the coding scheme on a per-object basis
- Easy reuse and customization of the audiovisual content

Audiovisual objects can differ in nature. They can be audio-only objects like a multichannel audio content or speech, or video-only objects, like traditional rectangular movies or arbitrarily shaped video objects. Objects can be natural, like audiovisual data captured from a microphone or from a camera, or synthetic, like text and graphics overlays, animated faces, or synthetic music. They can be 2D, like a Web page, or 3D, such as spatialized sound and 3D virtual worlds.

The scene description provides the spatial and temporal relationships between the audiovisual objects. These relationships can be purely 2D or 3D, or a mixture of 2D and 3D (for instance, a 2D "subscene" mapped in a 3D virtual world). The scene description includes commands specifying the behavior and interactive options embodied in the audiovisual objects and scenes. In addition, MPEG-4 specifies protocols to modify and animate the scene over time (for instance, adding or deleting objects, or changing objects' properties and behavior). The scene description is dynamic, which allows incremental build-up of the scene (as opposed to waiting for a download and then playing).

All information representing the audiovisual objects and the audiovisual scene description are provided as compressed binary streams, and all of these streams can be synchronized.

A possible audiovisual scene including natural and synthetic objects is shown in Figure 2.1. This figure shows how various visual and audio streams are composed on the top of a fixed still picture background (an object in its own right), according to a scene description stream. The diagram introduces the concept of *object descriptor* (OD). An object descriptor is a structure that lays the links between the scene description and the individual streams with media content. More details on these object descriptors may be found in Chapter 3.

Google Exhibit No. 1021
Page 031

Google Exhibit No. 1063
Page 0795

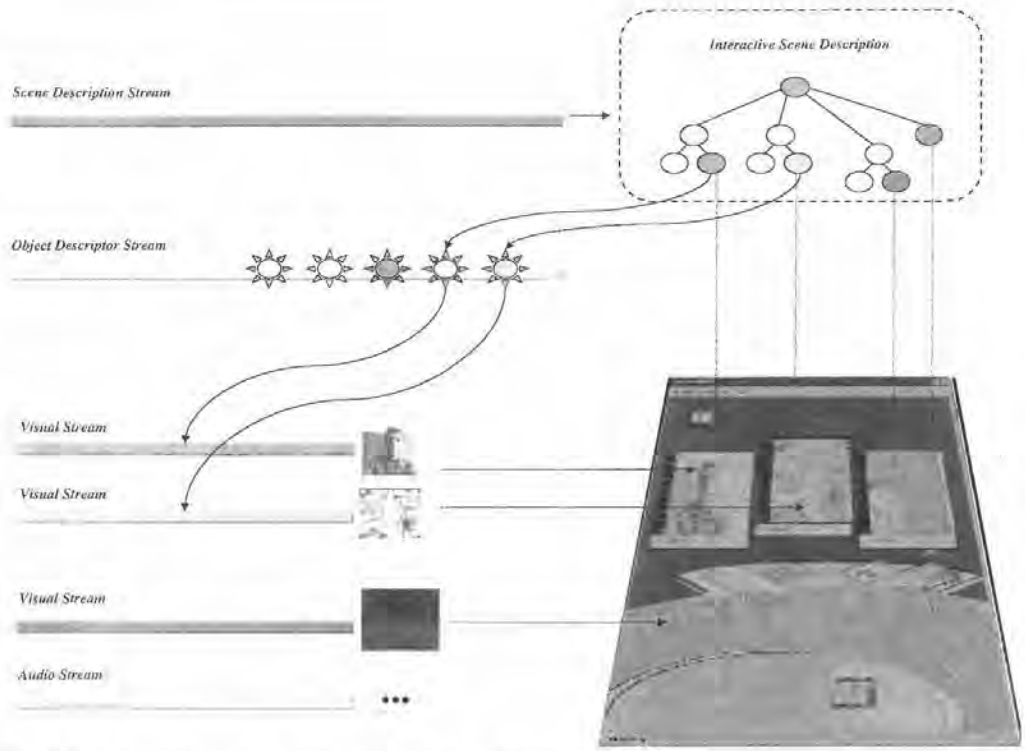


Fig. 2.1 Interactive audiovisual scene with objects.

2.2 An End-to-End Walkthrough

This section explains the high-level workings of the MPEG-4 standard using an end-to-end walkthrough of an MPEG-4 session. Although the MPEG-4 standard is designed to be used in many environments, the way MPEG-4 content is produced, delivered, and consumed often follows the same path. Such a typical path is shown in Figure 2.2, which highlights the main interoperability elements of the MPEG-4 standard as well as their respective position in the chain. While this walkthrough provides a typical scenario, MPEG-4 also can be used in other scenarios that are not covered by the walkthrough in Figure 2.2.

At the start of the walkthrough are content authors, who produce audiovisual content with the authoring tools of their preference. Part of the content creation process may be live; needless to say, though, off-line creation is also important in many MPEG-4-based applications. The content creation process can be separated into two major steps: authoring and publishing. Authoring involves the production of the audiovisual data, including the media, the scene description, and the interaction. Publishing involves adapting the content to the constraints imposed—by networks and terminals, for example.

Google Exhibit No. 1021
Page 032

Google Exhibit No. 1063
Page 0796

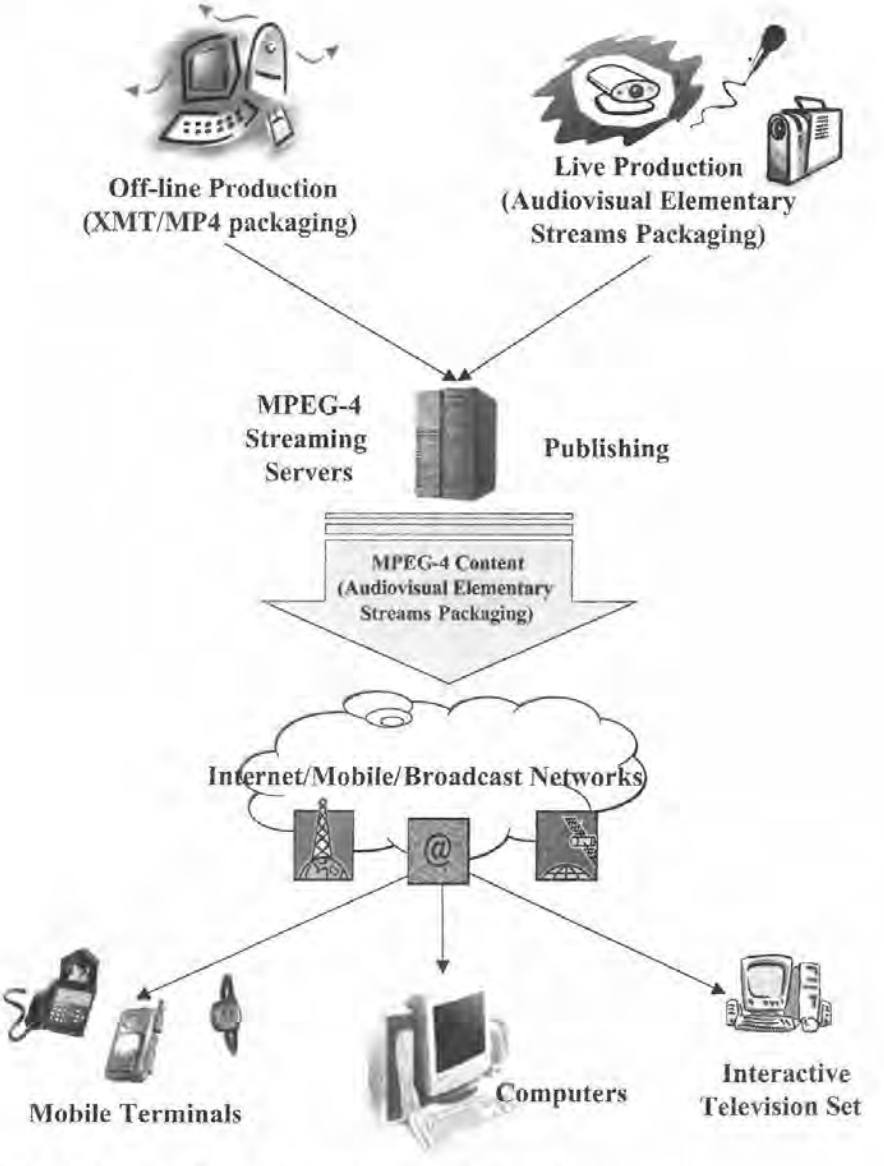


Fig. 2.2 Example MPEG-4 end-to-end walkthrough.

While streaming content is produced using MPEG-4's binary streaming formats, off-line multimedia content can be delivered to servers in an interoperable way using the MPEG-4 eXtensible MPEG-4 Textual format (XMT), as described in Chapter 6, or using the binary MP4 file format, as described in Chapter 7. The choice between these two depends on the freedom that authors want to offer users (or other authors) at the next stage of the delivery chain.

Google Exhibit No. 1021
Page 033

Google Exhibit No. 1063
Page 0797

XMT provides a structured and extensible representation of the scene description, allowing much flexibility for further manipulation or adaptation of this data (e.g., for automated processing like customization). It may contain additional information such as metadata, for instance, to express constraints on the adaptation to the delivery network. This makes it an appropriate format for exchanging content that needs to be further processed; XMT is therefore useful for the exchange of content between authors.

MP4 is more rigid than XMT in the sense that the scene description data is binary coded and hence hard to alter afterwards. The binary form is obviously more compact but also more deterministic with regard to what the users will see and hear, which is useful to ensure that an author's creations are presented as intended.

While MP4 and XMT files are the natural interoperability formats between authoring tools and MPEG-4 servers, this does not mean that the content will be stored as MP4 or XMT files on the server. Server implementations may have other ways to represent the content, optimized for specific software and hardware. What matters is that the served streams be compliant with the standard. But MPEG-4 servers can also directly use MP4 or XMT files to serve content on various networks. If XMT is used, the server needs to perform more than streaming; it will also need to encode the scene description data into binary form.

What goes out of the server are streams of data containing MPEG-4 content named *elementary streams*. The content of these elementary streams is discussed later in this chapter. What is important at this stage is that MPEG-4 audiovisual scenes can be split in several elementary streams, that these streams can be carried on different networks and that end terminals receiving these streams from different networks are capable of reconstructing the transmitted data in a synchronized manner. When content is produced live, it can be delivered to servers as interoperable elementary streams.

One of MPEG-4's design goals is to cover a wide range of access conditions, so that content can be created once and played on any network. This is achieved by abstracting the content delivery layer with an interface as specified in the Delivery Multimedia Integration Framework (DMIF) corresponding to MPEG-4 Part 6 [MPEG4-6]. This interface is named the DMIF Application Interface (DAI).

When streaming content, the MPEG-4 interoperability points are the specific formats of (and constraints on) the individual elementary streams, and the compliance to the walkthrough (i.e., the series of steps that has to be performed to access the elementary streams) defined by the DAI. In principle, what happens below the DAI is outside the scope of the MPEG-4 standard. In this way, the multimedia representation can be kept completely independent from the way it is delivered. There are exceptions to this rule: an efficient low-complexity multiplexing tool (FlexMux) and a dedicated file format (MP4) were developed because MPEG-4 needed specific tools for its transport (see Chapter 7).

Google Exhibit No. 1021
Page 034

Google Exhibit No. 1063
Page 0798

In order to transport the MPEG-4 content in existing environments, network-specific transport mechanisms have been defined. Transport in MPEG encompasses both wire formats and file formats. Currently, these transport mechanisms are available:

- Storage of MPEG-4 content in MP4 files [MPEG4-1]
- Carriage of MPEG-4 content on the Internet [MPEG4-8, GENT01]
- Carriage of MPEG-4 content in MPEG-2 transport streams [MPEG2-1]

These mechanisms are described in detail in Chapter 7. The spectrum of MPEG-4 consumption devices includes desktop computers, mobile devices, and interactive television sets. This last device nicely illustrates the various ways MPEG-4 content can be consumed.

Assume that the interactive TV set in Figure 2.2 receives through a satellite connection an MPEG-2 transport stream containing several MPEG-2 digital TV programs. Using the MPEG-2 extension that specifies the carriage of MPEG-4 on MPEG-2 transport streams [MPEG2-1], MPEG-4 content related to such a TV program can be sent along with the linear programming. This can provide the user with an enhanced interactive experience, such as a local interaction with a 3D model of a car in an ad or a multimedia electronic program guide. Note that all the interactivity is sent in the broadcast stream and a return path is not required.

Assume now that the TV set is also connected to the Internet. It is then possible to augment the broadcast experience with client-server functionality as well as to send rich media that can be blended with the basic TV program. One can imagine a range of services encompassing program enhancements with video clips streamed from the Internet on-demand up to multiuser games related to the TV programs with votes, 3D chats, and interaction with the scenario of the broadcast content.

2.3 Terminal Architecture

The overall architecture of an MPEG-4 terminal is depicted in Figure 2.3. At the bottom of the figure is the storage or transmission medium; this refers to the lower layers of the delivery infrastructure. MPEG-4 data can be transported on a variety of delivery systems as described in Section 2.2. This includes MPEG-2 transport streams, RTP/UDP over IP, and MPEG-4 (MP4) files. MPEG-4 data can also be sent over other transport mechanisms, such as Asynchronous Transfer Mode (ATM) or Digital Audio Broadcast (DAB), using one or a combination of the previous stacks of protocols.

Most of the currently available transport layer systems provide native means for multiplexing information. There are, however, a few exceptions, such as RTP Internet connections. In addition, the existing multiplexing mechanisms do not always fit MPEG-4 needs in terms of low delay, or they

Google Exhibit No. 1021
Page 035

Google Exhibit No. 1063
Page 0799

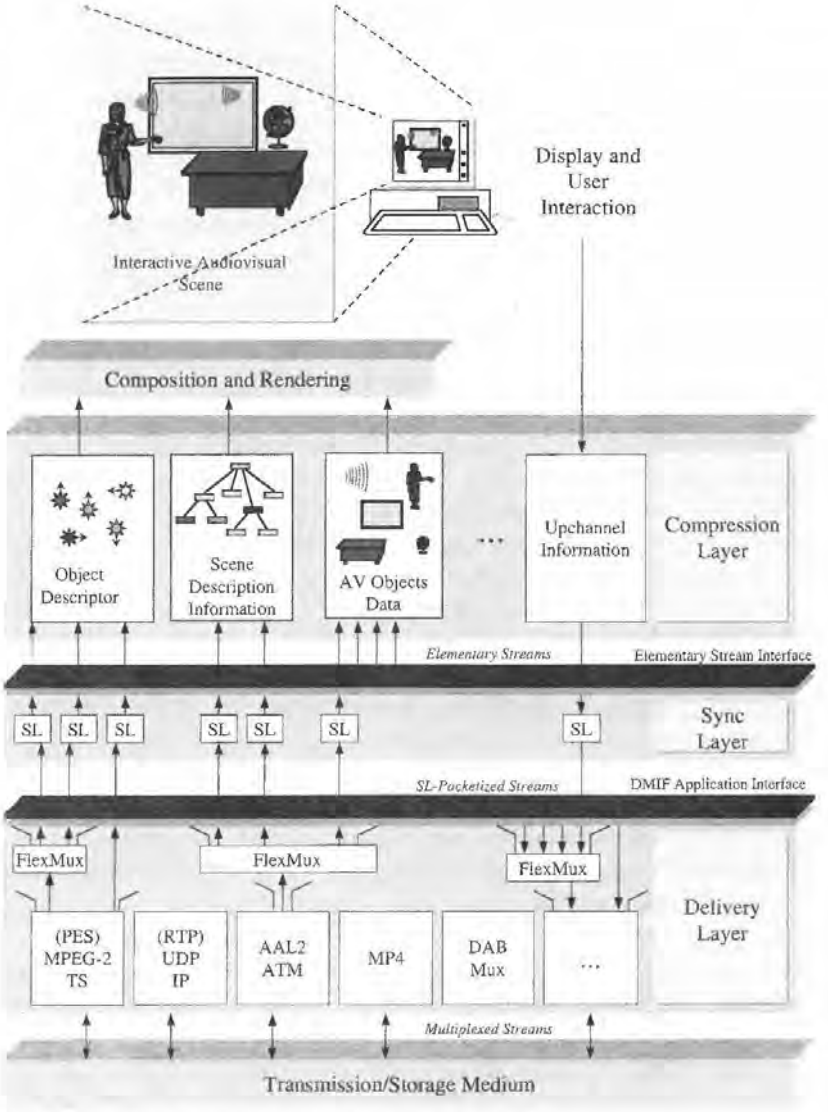


Fig. 2.3 MPEG-4 terminal architecture [MPEG4-1].

may incur prohibitive overhead in handling the expected large number of streams associated with an MPEG-4 session. As a result, MPEG-4 has defined a multiplexing tool, FlexMux, that can be used on top of the existing transport delivery layer (see Chapter 7).

The delivery layer provides a number of elementary streams to the MPEG-4 terminal. These streams can be audiovisual objects data, scene

Google Exhibit No. 1021
Page 036

Google Exhibit No. 1063
Page 0800

description data, control data in the form of object descriptors, or metadata describing the content or associating intellectual property rights to it.

Note that not all the streams have to be downstream (server to client); upstream channels can also exist, for the purpose of conveying data from the terminal to the transmitter or server. MPEG-4 standardizes the mechanisms that trigger the transmission of such upstream data at the terminal, as well as the formats in which they are transmitted back to the sender.

It is important that there be a common mechanism for conveying timing and framing information that applies to all types of elementary streams. The basic MPEG-4 timing model and the Sync Layer (SL) provide this functionality (see Chapter 3). The SL is a flexible and configurable packetization tool that allows the inclusion of timing, fragmentation, and continuity information in the associated data packets. Such information is attached to data units called *access units*, which comprise complete presentation units (e.g., an entire video frame or an audio frame).

Elementary streams are sent to their respective decoders, which process the data and produce composition units (e.g., a decoded video frame or, in MPEG-4 terminology, a Video Object Plane [VOP]). In order for the receiver to know what type of information is contained in each stream, control information in the form of ODs is used. The ODs describe the properties of elementary streams and are used to associate one or more elementary streams to each of the audio and visual objects in the scene. Through these ODs, the terminal can identify the content being delivered to it. Without an OD, the terminal wouldn't know how to use the elementary stream. ODs may also contain—or point to—Object Content Information (OCI). OCI elements can contain data such as content classification, rating, language, and keywords. (Because it was ready to use first, OCI forms sort of a precursor to MPEG-7.) MPEG-4 scenes can also contain MPEG-7 streams through a dedicated elementary stream type. Detailed descriptions of synchronization in MPEG-4 and of the OD framework are given in Chapter 3.

Advanced synchronization mechanisms (described in Chapter 4) augment the basic MPEG-4 timing model to permit synchronization of multiple streams and objects that can originate from a number of sources. The Flextime toolkit allows the definition of simple temporal relationships among MPEG-4 objects, such as `CoStart`, `CoEnd`, and `Meet`, as well as the specification of constraints for the timing relationship between MPEG-4 objects, as if the objects were on springs. The duration of an object is represented by the length of the spring, and these springs can be expanded or contracted as needed, to adjust the duration of each of the objects (adjusting the playback speed). Flextime is modeled after the W3C's Synchronized Multimedia Integration Language (SMIL) [SMIL01], and one of the reasons to include it in MPEG-4 was to provide compatibility with SMIL content.

In the context of an MPEG-4 audiovisual scene, at least one of the streams must contain the scene description information associated with the content. The scene description information defines the spatial and temporal

Google Exhibit No. 1021
Page 037

Google Exhibit No. 1063
Page 0801

position of the various objects, their dynamic behavior, and any interactivity features embedded in the content. As mentioned above, the (coded) data that represents an audiovisual object is carried in separate elementary streams. The scene description contains *pointers* to ODs when it refers to a particular audiovisual object. The MPEG-4 scene description language, called *binary format for scenes* (BIFS) [MPEG4-1], is further explained in Chapter 4. It complements the OD framework that defines the objects and their characteristics.

A key feature of the scene description is that, since it is carried in its own elementary stream(s), it can contain full timing information. This implies that the scene can be dynamically updated over time, a feature that provides considerable power for content creators. There is also a special lightweight mechanism to animate parts of the scene: *BIFS Anim*. This animation is accomplished by coding, in a separate stream, only the parameters that need to be updated.

The system's compositor uses the scene description information to aggregate the various natural and synthetic audiovisual objects and render the final scene for presentation to the user. MPEG-4 natural video objects (see Chapter 8) include rectangular and shaped, scalable and error-resilient video as well as still pictures [MPEG4-2]. Synthetic visual objects can be 2D and 3D meshes, as well as animated 3D faces and bodies as described in Chapter 9 and specified in both MPEG-4 Visual [MPEG4-2] and MPEG-4 Systems [MPEG4-1].

The palette of the MPEG-4 Audio tools includes speech coding from 2 to 24 kbit/s (see Chapter 10), general audio coding from 4 to 64 kbit/s/channel and above (see Chapter 11), as well as synthetic audio such as text-to-speech synthesis and synthetic music coding (see Chapter 12). All these tools are specified in MPEG-4 Audio [MPEG4-3], and the interfaces exposed at the scene description level are specified in MPEG-4 Systems [MPEG4-1].

The scene description tools provide mechanisms to capture user or system events. In particular, these mechanisms allow the association of events to user operations on desired objects and they can—in turn—modify the behavior of the object. Event processing is the core mechanism with which application functionality and differentiation/customization can be provided. In order to provide flexibility in this respect, MPEG-4 allows the use of ECMAScript (also known as JavaScript) scripting within the scene description. Use of scripting tools is essential in order to access state information and implement sophisticated interactive applications.

MPEG-4 also defines a set of Java language Application Programming Interfaces (APIs), generally called MPEG-J, through which access to an underlying MPEG-4 player can be provided to Java applets (called *MPEGlets*). These MPEGlets can form the basis for very sophisticated applications, opening up rich possibilities for audiovisual content creators to author attractive content. Chapter 5 provides a complete description of the MPEG-J application engine.

In addition to the new interactive features for end users, the object-based content structure provides great benefits to content creators as well. The use of an object-based structure, where composition happens at the receiver, con-

Google Exhibit No. 1021
Page 038

Google Exhibit No. 1063
Page 0802

siderably simplifies the content creation process. Starting from a set of audiovisual objects, it is easy to define a scene description that combines these objects in a meaningful presentation. An essentially similar approach is used in HTML and Web browsers, which makes it easy for nonexpert users to create their own content. The fact that the content's structure survives the process of coding and distribution also allows for the reuse of the objects in the scene. For example, content filtering and searching applications can be easily implemented using ancillary OCI carried in object descriptors. Finally, when allowed by the content author, consumers with the right tools can easily extract and reuse individual objects.

2.4 MPEG-4 Tools

This section provides a brief overview of the tools available in the MPEG-4 standard. A more detailed description can be found in the MPEG-4 Overview [N4505]. Each of these tools is explained in detail in one of the remaining chapters of this book.

The tools are organized in one of four categories—Systems, Visual, Audio, and DMIF—corresponding to the four technology specification Parts of the MPEG-4 standard. The tools listed cover the state of the standard as of February 2002. Since that date, more tools may have been standardized in MPEG-4; for up-to-date information, consult the MPEG Web site, *mpeg.telecomitalialab.com* [MPEG].

2.4.1 Systems Tools

The major tools standardized in MPEG-4 Systems [MPEG4-1] are

- **Object descriptor framework:** The OD framework defines the relationship between the individual elementary streams and the media objects in the scene (e.g., the audio or video streams associated with an object representing a participant in a videoconference). ODs provide information such as the elementary streams available to represent a given media object, the characteristics of the decoders needed to consume the elementary streams, and the location (possibly a URL) of the elementary streams data. ODs include some *Intellectual Property Management and Protection* (IPMP) information in the form of data sets covering the identification of intellectual property rights associated with the media objects. See Chapter 3 for further detail.
- **Systems decoder model:** The systems decoder model provides the basic synchronization and streaming features of the MPEG-4 standard. The model defines the initialization and continuous management of the receiving terminal's buffers. It also defines the timing-identification, synchronization, and recovery mechanisms. For further details see Chapter 3.

Google Exhibit No. 1021
Page 039

Google Exhibit No. 1063
Page 0803

- **Binary format for scenes:** BIFS describes the spatio-temporal arrangements of the MPEG-4 objects in the scene. The scene description provides a rich set of functionality for 2D and 3D composition as well as text and graphics primitives. It also provides forms of interactivity (e.g., general event handling and routing between objects in the scene) among events triggered by the user or the scene, as well as client- and server-based interaction. See Chapter 4 for details.
- **MPEG-J:** MPEG-J allows the use of Java programs to complement the logic and programmatic part of MPEG-4 content. These programs are called MPEGlets when they are transmitted along with the multimedia content. MPEG-J defines interfaces to various aspects of the terminal and networks in the form of Java APIs. MPEG-J also defines a delivery mechanism allowing MPEGlets and other Java classes and objects to be streamed separately. Chapter 5 provides greater detail.
- **Extensible MPEG-4 textual format:** XMT is a framework for representing MPEG-4 scene descriptions using a textual syntax. The XMT format allows the content authors to exchange their content with other authors or with tools or service providers, and it facilitates interoperability with both the X3D format being developed by the Web3D consortium [X3D01] and the Synchronized Multimedia Integration Language (SMIL) format from the W3C consortium [SMIL01]. See Chapter 6 for more detail.
- **Transport tools:** Although transport of multimedia content is not in the core activity of the standard, MPEG-4 defines two transport tools, MP4 and FlexMux, for which there was no clearly identified standardization body where they could have been standardized. MP4 is a tool for storing MPEG-4 data in a file (the MPEG-4 file format). FlexMux is a tool for the interleaving of multiple streams into a single stream, including timing information. See Chapter 7 for a discussion of these tools.
- **IPMP "hooks":** APIs that provide an interface to IPMP systems and a way to identify which IPMP system was used. The IPMP system itself is not specified by MPEG. At the time of writing (February 2002), MPEG-4 was developing an extension to these MPEG-4 hooks that will provide more digital rights management capabilities as well as more flexibility for the IPMP system [N4270]. The goal of this work is to allow end users to consume protected content in a transparent manner and to provide renewability of IPMP systems (see Chapter 3).

2.4.2 Visual Tools

The major tools standardized in MPEG-4 Visual [MPEG4-2] are introduced here.

- **Video compression tools:** Algorithms providing the efficient compression of video for bit rates between 5 kbit/s and 1 Gbit/s and for resolutions from sub-QCIF to studio editing resolutions (4k×4k pixels).

Google Exhibit No. 1021
Page 040

Google Exhibit No. 1063
Page 0804

Progressive and interlaced video are supported. Compact coding of textures with a quality adjustable from *acceptable* for very high compression ratios up to *near lossless* is supported, notably targeting texture mapping on 2D and 3D meshes. Random access of video provides functionalities such as pause, fast forward, and fast reverse of stored video. Different types of scalability are provided, including complexity scalability at the encoder and decoder, object scalability, spatial scalability, temporal scalability, and quality scalability. See Chapter 8 for more detail.

- **Robustness in error-prone environments:** Error resilience is supported to assist the access of images and video over a wide range of storage and transmission media. This includes the useful operation of image and video compression algorithms in error-prone environments at low bit rates (e.g., less than 64 kbit/s). There are coding tools that address both the bandwidth-limited nature and error-resilience aspects of wireless networks. See Chapter 8 for more detail.
- **Fine grain scalability (FGS):** FGS allows small quality steps by adding or deleting layers of extra information with fine granularity (quality scalability). It is useful in a number of environments, notably for streaming purposes, but also for dynamic (*statistical*) multiplexing of pre-encoded content in broadcast environments. See Chapter 8 for more detail.
- **Shape and alpha channel coding:** Efficient techniques are provided for coding of *binary* and *gray-scale* or *alpha* shape information. A binary alpha map defines whether a pixel belongs to an object; in a binary alpha map, a pixel can be *on* or *off*. A gray-scale map offers the possibility of defining a certain transparency for each pixel. See Chapter 8 for more detail.
- **Face and body animation:** The face and body animation tools specify parameters for defining, calibrating, and animating synthetic faces and bodies. The 3D models themselves are not standardized by MPEG-4; only the calibration and animation parameters are. See Chapter 9 for more detail.
- **Coding of 2D meshes:** MPEG-4 has tools for mesh-based coding and animated texture transfiguration. These tools comprise 2D Delaunay and regular mesh formalisms, with motion tracking of animated objects, motion prediction, and suspended texture transmission with dynamic meshes. There is also geometry compression for motion vectors (2D mesh compression with implicit structure and decoder reconstruction). See Chapter 9 for more detail.
- **Coding of 3D meshes:** MPEG-4 includes tools for coding 3D polygonal meshes. Polygonal meshes are widely used as a general representation of 3D objects. The underlying technologies compress the connectivity, geometry, and properties such as shading normals, colors, and texture coordinates of 3D polygonal meshes. See Chapter 9 for more detail.

Google Exhibit No. 1021
Page 041

Google Exhibit No. 1063
Page 0805

Natural video coding tools are presented in further detail in Chapter 8, and synthetic coding tools are described in Chapter 9.

2.4.3 Audio Tools

This section lists MPEG-4's tool set for audio coding. Speech and general audio coding tools are explained in more detail in Chapters 10 and 11, respectively; tools for coding of synthetic audio and speech are described in Chapter 12. The major tools standardized in MPEG-4 Audio [MPEG4-3] are

- **Speech coding:** MPEG-4 Audio supports speech coding at bit rates from 2 kbit/s up to 24 kbit/s, using two algorithms: a Code Excited Linear Prediction (CELP) coder and a parametric coder. With variable bit-rate coding, even lower bit rates can be achieved—down to an average of 1.2 kbit/s. Low delay is possible for communications applications. The parametric Harmonic Vector Excitation Coding (HVXC) coder allows the user to modify speed and pitch during playback. If the CELP coder is used, a change of the playback speed can be achieved by using an additional tool for effects processing. See Chapter 10 for more detail.
- **General audio coding:** Most of the MPEG-4 general audio coding tools are based on the filterbank coding approach as used in MPEG-2 Advanced Audio Coding (AAC) [MPEG2-7]. Building on this technology, MPEG-4 added new tools with the target of improving the compression performance and supporting new functionalities. For example, the Transform-domain Weighted Interleave Vector Quantization (TwinVQ) coder supports operation at very low bit rates (e.g., 6 kbit/s/channel). MPEG-4 is the first audio coding standard to support scalability modes (e.g., fine grain scalability with a granularity down to 1 kbit/s/ch). A special low delay coding mode is available for communication applications among others. MPEG-4 also specifies a parametric audio coder, Harmonic and Individual Lines plus Noise (HILN), that makes a decomposition of the input signal into distinct sound components, namely a harmonic tone, individual sinusoids, and a noise component. This coder can operate at bit rates down to about 4 kbit/s, provides bit-rate scalability, and can be combined with the MPEG-4 HVXC parametric speech coder. See Chapter 11 for more detail.
- **Synthesized audio coding:** Structured audio coding tools provide a powerful framework in which score-based control information can be used to control "musical instruments" described with a special language. If limited resources are available, synthetic sound can be generated using a standardized wavetable format. The popular MIDI format is also supported. See Chapter 12 for more detail.
- **Synthesized speech coding:** Text-to-Speech (TTS) tools allow a text, possibly with prosodic parameters (pitch contour, phoneme duration, and so on), to be used to generate intelligible synthetic speech. Scalable TTS cod-

Google Exhibit No. 1021
Page 042

Google Exhibit No. 1063
Page 0806

ers' bit rates range from 200 bit/s to 1.2 kbit/s. This tool also enables the prosody of the original speech to be used; lip synchronization control with phoneme information; trick mode functionality; international language and dialect support for the text; international symbol support for phonemes; support for specifying age, gender, and speech rate of the speaker; as well as support for conveying Facial Animation Parameter (FAP) bookmarks.

2.4.4 DMIF Tools

The major tools standardized in DMIF, Part 6 of the MPEG-4 standard [MPEG4-6] are

- **DMIF-Application Interface (DAI):** The DAI is a transparent application interface allowing access to multimedia content, irrespective of whether the peer is a remote interactive peer, a broadcast, or a local storage entity. The communication model underlying these APIs allows for continuous Quality of Service (QoS) monitoring, specific QoS queries, and notification of QoS violations. It also enables the simultaneous access, presentation, and synchronization of MPEG-4 content carried through different delivery technologies.
- **DMIF Signaling Protocol:** The DMIF Signaling Protocol is a generic session-level protocol designed for streaming multimedia data. The protocol is used to configure the delivery protocol stacks at the peer ends. The protocol stack may also include the FlexMux tool specified in MPEG-4 Systems [MPEG4-1]. The protocol was conceived with an eye to the future of networking technologies, and it enables features that are not readily available with current techniques. Such features include QoS and resource management and operation over a variety of networks, including heterogeneous networks.

See Chapter 7 for further details on DMIF tools.

2.4.5 Other MPEG-4 Tools

Strictly speaking, the following tools are not part of the MPEG-4 standard, but they allow the transport of MPEG-4 content on various delivery networks. See Chapter 7 for details.

- **Carriage of MPEG-4 on MPEG-2 Systems:** This extension to the Systems Part of the MPEG-2 standard [MPEG2-1] specifies how MPEG-2 Systems streams may carry individual MPEG-4 audio and visual elementary streams as well as audiovisual scenes with their associated streams. A distinction is made between MPEG-4 individual elementary streams (only MPEG-2 Systems tools are used) and an MPEG-4 audiovisual scene with its associated streams (MPEG-4 Systems stream management tools are used along with MPEG-2 Systems tools).

Google Exhibit No. 1021
Page 043

Google Exhibit No. 1063
Page 0807

- **Carriage of MPEG-4 on IP networks:** This specification of the Internet Engineering Task Force (IETF) describes a payload format for transporting MPEG-4 encoded data using the Real-Time Transfer Protocol (RTTP) [GENT01]. This specification is complemented by Part 8 of the MPEG-4 standard, "Carriage of ISO/IEC 14496 contents over IP networks" [MPEG4-8], which provides a framework for the carriage of MPEG-4 content over IP networks as well as guidelines for designing payload format specifications for the detailed mapping of MPEG-4 content into several IP-based protocols.

2.4.6 Profiles and Levels

MPEG-4 would be too unwieldy for most applications if any compliant decoder had to be able to support all of the tools listed above. Therefore, MPEG has defined profiles and levels for the MPEG-4 standard, much like it did for its predecessors MPEG-1 Audio (which has three *layers* of audio coding) and MPEG-2 Audio and Video.

Profiles define subsets of tools that can be used for a large class of applications and services. Profiles are defined for specific tool dimensions (e.g., visual or audio) and not for combinations of these dimensions. This means that application developers can choose the tools they need by picking, for each tool dimension, the profile that best suits the application. Profiles exist for these dimensions:

- Visual
- Audio
- Graphics
- Scene graph
- Object descriptors
- MPEG-J

Levels further limit the complexity of the tools within a given profile (see Appendixes A–D). Conformance points of the MPEG-4 standards are defined as a given profile at a given level. Conformance to the standard can be tested at these conformance points for the various profiling dimensions. Conformant terminals for a given profile and level will interoperate. MPEG-4 profiles and levels are further described in Chapter 13.

2.5 MPEG-4 AND OTHER MULTIMEDIA STANDARDS

The multimedia standards and solutions in place before the MPEG-4 standard was developed influenced the development of MPEG-4. Among the standards and solutions that MPEG-4 has used and referenced, the following are especially relevant:

Google Exhibit No. 1021
Page 044

Google Exhibit No. 1063
Page 0808

CHAPTER 3

Object Description and Synchronization

by Carsten Herpel

Keywords: object description, stream description, OCI, IPMP, content access, synchronization, system decoder model

An MPEG-4 presentation may consist of many audiovisual objects that are conveyed through a number of audiovisual streams to a receiver, either in a truly streaming manner or stored in a file. An important task in structuring an MPEG-4 presentation is the description of all streams, including information such as media codecs, bit rates, and so on used to encode them as well as the relations between streams. Additional information about streams may be desired for semantic description or for implementing digital rights management. Further to the stream description, means for a synchronized presentation of all streams are required, including a timing model and a buffering model for the MPEG-4 terminal. The way MPEG-4 provides all these functionalities is presented in this chapter. These tools are specified in MPEG-4 Part 1: Systems [MPEG4-1].

3.1 Object Descriptors: Entry Points to MPEG-4 Content

Part 2 [MPEG4-2] and Part 3 [MPEG4-3] of the MPEG-4 standard (corresponding to Chapters 8 to 12 in this book) specify the coding tools to generate compressed audio and visual streams. Each semantic visual or audio object may be coded into one or more such streams. An MPEG-4 presentation that consists of

Google Exhibit No. 1021
Page 045

Google Exhibit No. 1063
Page 0809

multiple such objects (and, hence, many streams) needs mechanisms to describe all the objects in the presentation. One part of this mechanism is embodied in the object descriptor framework [MPEG4-1, Clause 8]. All components of an MPEG-4 presentation are discovered recursively through object descriptors (ODs). The details of this approach are explained in this chapter.

Object description enumerates only the streams in a presentation and specifies how they relate to media objects. In order to assemble those media objects into a specific audiovisual scene, most MPEG-4 presentations also include a scene description. The scene description format, called *binary format for MPEG-4 scenes* (BIFS), is explained in detail in Chapter 4. In order to understand the present chapter, one must understand the general role of scene description and its conceptual difference from object description.

The scene description conveys the spatio-temporal layout of the media objects in the scene—that is, it captures the work of the creative content designers. Object description serves an asset management purpose; it allows locating streams and retrieving information about them. This information may be produced during authoring of the scene description, but—different from the scene description—it also may be subsequently amended or changed, for example, when media streams for a presentation are relocated to a different media server.

The basic concept of an OD is that of a container aggregating all the useful information about the corresponding object. This information is structured in a hierarchical manner through a set of subdescriptors. Each OD describes streams that are referred from a specific location in the scene description. In the case of audio streams, for example, this could be an `AudioSource` node (see Figure 3.1). The media streams that are related to this single `AudioSource` node constitute the *object* or *media object* that is described by the OD. More than one media stream can be associated with the object, as will be seen.

3.1.1 Syntactic Description Language

In order to document the syntax of the OD framework, the MPEG-4 Systems standard—as well as this chapter—uses the Syntactic Description Language (SDL). SDL is specified in [MPEG4-1, Clause 14]. This language is a convenient shorthand to express bitstream syntax of inherently object-oriented structures in a C++ kind of way. The strict syntax definition of SDL has the advantage that it enables the fully automatic generation of a code stub for the bitstream parser from the SDL code in [MPEG4-1].[1]

A variable definition in SDL typically includes a directive to parse the value for this variable from a bitstream. Specifically, parsing `length` bits off

1. The other Parts of the MPEG-4 standard continue to use a less rigid pseudo-C notation for bitstream specification. This is mostly due to the fact that SDL has been developed in the same timeframe as the other Parts of the MPEG-4 standard.

Google Exhibit No. 1021
Page 046

Google Exhibit No. 1063
Page 0810

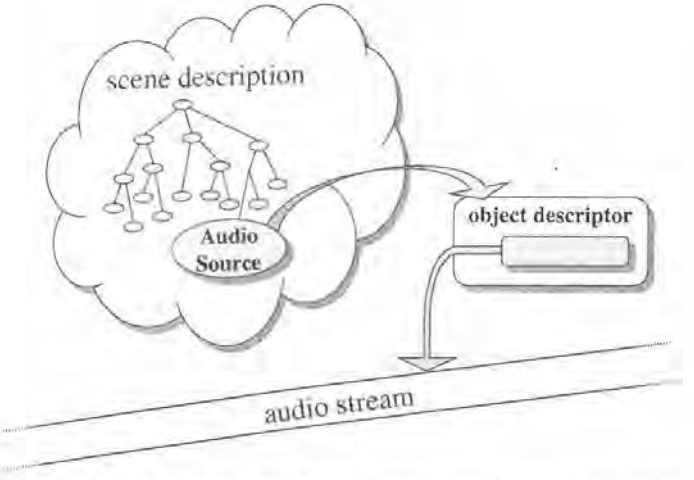


Fig. 3.1 BIFS node (`AudioSource`) pointing to a(n audio) stream through an OD.

the bitstream into a variable called `element_name` with a certain `type` is expressed as follows:

```
[aligned] type[(length)] element_name [= value];
```

where `type` is any of the data types—`bit` for arbitrary bit sequences, `double` for floating values, `int` for integer or `unsigned int` for unsigned integer. Square brackets express optional elements. For example, if `length` is missing, it means that the variable is not parsed from the bitstream but rather is a locally defined variable as in any programming language. The optional `aligned` attribute causes the reading of `length` bits to start from the next byte boundary, making the variable byte-aligned. If the variable `element_name` is assigned a `value`, this causes the parser to assert that the value read from the bitstream actually corresponds to this *value*. In the example,

```
aligned bit(32) picture_start_code=0x00000100;
```

the parser is asked to parse 32 bits from the next byte boundary and to check that this variable corresponds to the hexadecimal value 100.

Often, the number of bits for a certain syntax element is itself signaled in the bitstream. Therefore, it is permitted to use variables in the `length` field, as in the second line of the next example:

```
unsigned int(3) precision;
int(precision) DC;
```

A number of variables that constitute the data for an object may be arranged into classes:

```
class Foo {
    unsigned int(5) aVal;
```

Google Exhibit No. 1021
Page 047

Google Exhibit No. 1063
Page 0811

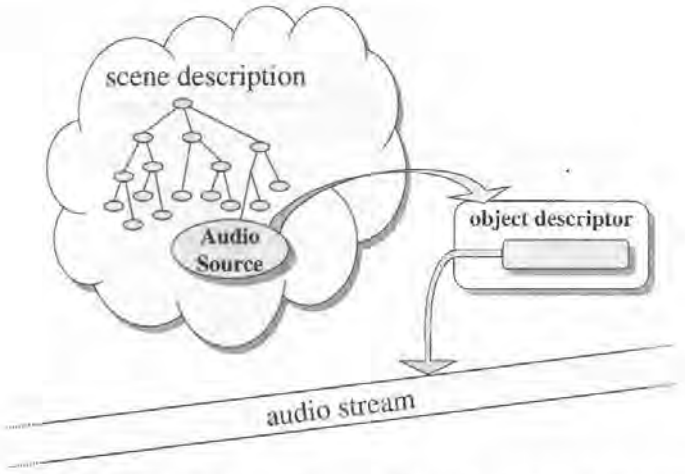


Fig. 3.1 BIFS node (`AudioSource`) pointing to a(n audio) stream through an OD.

the bitstream into a variable called `element_name` with a certain `type` is expressed as follows:

```
[aligned] type[(length)] element_name [= value];
```

where `type` is any of the data types—`bit` for arbitrary bit sequences, `double` for floating values, `int` for integer or `unsigned int` for unsigned integer. Square brackets express optional elements. For example, if `length` is missing, it means that the variable is not parsed from the bitstream but rather is a locally defined variable as in any programming language. The optional `aligned` attribute causes the reading of `length` bits to start from the next byte boundary, making the variable byte-aligned. If the variable `element_name` is assigned a `value`, this causes the parser to assert that the value read from the bitstream actually corresponds to this *value*. In the example,

```
aligned bit(32) picture_start_code=0x00000100;
```

the parser is asked to parse 32 bits from the next byte boundary and to check that this variable corresponds to the hexadecimal value 100.

Often, the number of bits for a certain syntax element is itself signaled in the bitstream. Therefore, it is permitted to use variables in the `length` field, as in the second line of the next example:

```
unsigned int(3) precision;
int(precision) DC;
```

A number of variables that constitute the data for an object may be arranged into classes:

```
class Foo {
    unsigned int(5) aVal;
```

Google Exhibit No. 1021
Page 048

Google Exhibit No. 1063
Page 0812

```
    bit(3) bVal;
}
```

Each time an instance of class `Foo` occurs in another class, a five-bit value `aVal` and a three-bit value `bVal` are parsed from the bitstream. For example, class `Bar` will parse five elements of type `Foo` (i.e., read five bytes from the bitstream) with the following code:

```
class Bar {
    Foo aFoo[5];
}
```

The array elements read into `aFoo` can be further used in the SDL code of class `Bar` with the familiar syntax `aFoo[i].aVal` and `aFoo[i].bVal`, yielding the `aVal` and `bVal` values of the i^{th} array element.

After those fairly simple definitions, it comes as no surprise that classes may inherit from other classes, as in the next example:

```
class Foobar extends Foo {
    int(8) cVal;
}
```

`Foobar` now actually has the elements `aVal`, `bVal`, and `cVal`. In other words, parsing of an instance of class `Foobar`, as in the next example,

```
class ParseAfoobar {
    Foobar aFoobar;
}
```

results in reading two bytes (5 + 3 + 8 bit) that are accessible as `aFoobar.aVal`, `aFoobar.bVal`, and `aFoobar.cVal`.

Furthermore, classes can be identified with a tag value (0×23 in the following example) that is parsed from the bitstream before reading any element within the class:

```
class ClassWithID : aligned bit(8) classTag=0x23 {
    Foobar aFoobar;
}
```

So, if an instance of `ClassWithID` is expected, as in

```
class ParseAclassWithID {
    int (8) a
    ClassWithID aCwID;
}
```

an eight-bit value other than 0×23 is read from the bitstream after reading the eight bits of `a`, and thus the parser knows that an error has occurred. Although this may be useful to know, the major use of tags is the discrimination of different classes that may occur in the bitstream at the same location. Those classes would have to derive from a common base class:

```
class ClassesWithID : aligned bit(8) classTag=0x23 .. 0x33 {
    Foobar aFoobar;
}
```

Google Exhibit No. 1021
Page 049

Google Exhibit No. 1063
Page 0813

in a manner like:

```
class FirstClass extends ClassesWithID : aligned bit(8) classTag=0x23
     {
    int(8) aaValue;
}
class SecondClass extends ClassesWithID : aligned bit(8)
     classTag=0x24 {
    int(8) bbValue;
}
```

giving unique class tags to `FirstClass` and `SecondClass` in the range of tags specified for classes that inherit from `ClassesWithID`.

Now, instances of the derived classes of `ClassesWithID` can be parsed through

```
class ParseClassesWithID {
    int (8) a
    ClassesWithID aCwID;
}
```

If a value of 0×23 is read after the eight bits of a, an instance of `FirstClass` will be read; if it is 0×24, it will be an instance of `SecondClass`; in all other cases with tag values between 0×25 and 0×33, an instance of the base class `ClassesWithID` will be read.

Base classes also can be used to organize a set of classes and provide a common name scope for their tag values. For this purpose, similar to C++, abstract classes can be defined that cannot be instantiated (i.e., they cannot be read from the bitstream) and which do not have tag values of their own. The derived classes can then populate the full range of tag values, as given by the type declaration of the tag. So, in the following example,

```
abstract class MyClassesWithID : aligned bit(8) classTag=0 {
    Foobar aFoobar;
}
class FirstClass extends MyClassesWithID : aligned bit(8)
     classTag=0x23 {
    int(8) aaValue;
}
class SecondClass extends MyClassesWithID : aligned bit(8)
     classTag=0x24 {
    int(8) bbValue;
}
class OtherClass : aligned bit(8) classTag=0x24 {
    int(8) ccValue;
}
class ParseClassesWithID {
    int (8) a;
    OtherClass aClass;
    MyClassesWithID aCwID;
}
```

Google Exhibit No. 1021
Page 050

Google Exhibit No. 1063
Page 0814

OtherClass may reuse the tag value of 0×24, as it is not derived from ClassesWithID. ParseClassesWithID parses an integer variable a and one instance of OtherClass before parsing an instance of MyClassesWithID. Here, the parser has to encounter either a value of 0×23 or 0×24 to identify a FirstClass or SecondClass instance. Otherwise, the parser would assume it has encountered an unknown derived class from ClassesWithID. Nevertheless, parsing would have to stop here with an error, because parsing of that unknown class is not possible.

Therefore, a special type of classes exists that is both expandable and can be skipped easily, as the overall length of this class is encoded. The length indication is conveyed immediately after the tag (if any) and before any parsable variable.

```
abstract expandable (228-1) class ClassesWithID : aligned bit(8)
     classTag=0 {
    Foobar aFoobar;
}
```

The derived classes of this version of ClassesWithID would have a length indication following the class tag, limiting the class size to 2^{28}–1. When an instance of such a derived class is encountered with a length bigger than what can be computed from the class definition, the parser knows that some trailing data are conveyed in that class which, possibly, it cannot parse but can ignore and continue parsing after skipping that data. When using expandable classes, the parser does not have to fail when parsing an unknown derived class out of a number of derived classes from a base class. The length indication is helpful here to skip such a class instance.

These basic[2] SDL principles outlined above are used in the subsequent sections to introduce the basic syntactic structures of the OD framework.

3.1.2 Object Description

The syntax specification of all descriptors [MPEG4-1, Clause 8.6] is derived from a common base class, called BaseDescriptor (see Table 3.1). It specifies that descriptors always are identified by a numeric tag of eight-bit length. Furthermore, it limits the length of descriptors to 2^{28}–1 bytes and, through the use of an expandable class definition, specifies that each derived descriptor

Table 3.1 Base Descriptor: The template for other descriptors

```
abstract aligned(8) expandable(2^28-1) class BaseDescriptor : bit(8) tag=0 {
    // empty. To be filled by classes extending this class.
}
```

2. Features of SDL that are not needed for the code fragments given in this book have been omitted.

Google Exhibit No. 1021
Page 051

Google Exhibit No. 1063
Page 0815

defined in the MPEG-4 specification can be extended in the future by syntax elements that are added at the end of the descriptor.

The syntax of the OD [MPEG4-1, Clause 8.6.3] presented in Table 3.2 shows that `ObjectDescriptor` inherits from `ObjectDescriptorBase` which in turn inherits from `BaseDescriptor` (Table 3.1). The latter is another empty abstract base class for different types of ODs.

For labeling, each OD has an `ObjectDescriptorID`. This ID allows the unique identification of a set of streams in an MPEG-4 presentation. Additionally, it allows the scene description stream, BIFS, to refer to a specific descriptor.

An OD may further contain either only a URL or an actual OD. If it includes only a URL, it means that the actual OD can be found at the location pointed to by the URL. This is useful for managing large assets of media, in which not all media and their descriptions are located on a single media server. Basically, URLs would allow assembling one MPEG-4 presentation from content distributed all over the globe in a manner comparable to today's compound Web pages, which assemble elements from multiple Web servers.

The content of the OD (in the absence of a URL) is a series of subdescriptors. These subdescriptors describe four elements:

- Individual elementary streams (ESs)
- Semantic information about an object (OCI[3] descriptor)
- Hooks for content access management (IPMP[4] descriptor pointer)
- A placeholder for future extension descriptors

ES description is detailed first, with the OCI and IPMP frameworks coming later in the section about semantic description and access management.

Table 3.2 OD syntax

```
class ObjectDescriptor extends ObjectDescriptorBase : bit(8)
     tag=ObjectDescrTag {
    bit(10) ObjectDescriptorID;
    bit(1) URL_Flag;
    const bit(5) reserved=0b1111.1;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    } else {
        ES_Descriptor esDescr[1 .. 255];
        OCI_Descriptor ociDescr[0 .. 255];
        IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    }
    ExtensionDescriptor extDescr[0 .. 255];
}
```

3. Object content information.

4. Intellectual property management and protection.

Google Exhibit No. 1021
Page 052

Google Exhibit No. 1063
Page 0816

3.1.3 Stream Description

Each individual stream—or, more precisely, each *elementary stream*—is described by an elementary stream descriptor (ES descriptor or ESD; see Table 3.3) specified in [MPEG4-1, Clause 8.6.5]. An ES is conceived as a flow of (possibly compressed) data that originates from a single source in the sender and terminates in a single sink at the receiver. In a simple case, a media encoder (audio or video) and the corresponding media decoder are communicating through one ES (media encoder/decoder 1 in Figure 3.2). However, a single media decoder also may consume multiple ESs concurrently, specifically in case of a scalable encoding (media encoder/decoder 2 in Figure 3.2). Formally, a set of one or more ESs that are jointly decoded to recover the media object makes up a *media stream* in the MPEG-4 context.

Each ESD provides detailed information about the associated stream, including its location, type, encoding, bit rate, buffer requirements, and decoder specific information. As a comparison, only a subset of this information could be conveyed with multipurpose Internet mail extensions (MIME) [RFC2045] or the session description protocol (SDP) [RFC2327], mechanisms often used in Internet streaming applications for stream description.

Looking at the ESD elements in detail, stream location is provided first. This is done through one of two mechanisms: It can be done solely by means of an elementary stream ID (ES_ID). In that case, it is assumed that the media

Table 3.3 ESD syntax

```
class ES_Descriptor extends BaseDescriptor : bit(8) tag=ES_DescrTag {
    bit(16) ES_ID;
    bit(1) streamDependenceFlag;
    bit(1) URL_Flag;
    bit(1) OCRstreamFlag;
    bit(5) streamPriority;
    if (streamDependenceFlag)
        bit(16) dependsOn_ES_ID;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    }
    if (OCRstreamFlag)
        bit(16) OCR_ES_Id;
    DecoderConfigDescriptor decConfigDescr;
    SLConfigDescriptor slConfigDescr;
    IPI_DescrPointer ipiPtr[0 .. 1];
    IP_IdentificationDataSet ipIDS[0 .. 255];
    IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    LanguageDescriptor langDescr[0 .. 255];
    QoS_Descriptor qosDescr[0 .. 1];
    RegistrationDescriptor regDescr[0 .. 1];
    ExtensionDescriptor extDescr[0 .. 255];
}
```

Google Exhibit No. 1021
Page 053

Google Exhibit No. 1063
Page 0817

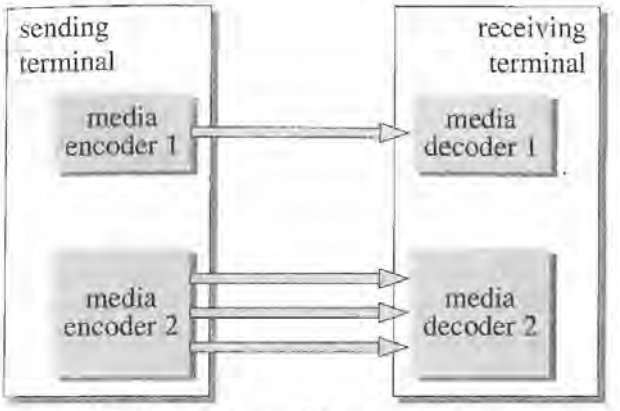


Fig. 3.2 Media encoder and decoder can be connected by one (case 1) or more (case 2) elementary streams.

server storing this stream will be able to connect this `ES_ID` to some stream resource in an application-specific way. Second, the URL field can be used. This allows locating ESs in a more versatile manner. In both cases, however, the actual description of the stream is done by the ESD at hand.

The flags after `ES_ID` signal whether the related syntax elements are present in the ES descriptor in question. The `streamPriority` element signals the relative importance of the streams within a presentation: The higher the value, the more important the stream. This information is mainly of use for a stream server or a gateway that can evaluate the stream priority values for all streams in a presentation and then group them into different transmission channels accordingly. In other words, the semantics of this field must be determined within a specific application scope.

As mentioned before, ODs and ESDs are containers for information about the associated object and ES, respectively. Therefore, the subsequent syntax elements relate to a number of different features that are introduced in the sections to come. First, the stream dependence mechanism defines how streams hierarchically depend on each other by pointing to the `ES_ID` of the stream upon which this one directly depends. The usage will be further discussed in the stream-dependence section. The `OCRstreamFlag` and `OCR_ES_Id` allow tying streams to a specific time base by pointing to the `ES_ID` of the stream that conveys this time base information. The usage will be discussed in the section on synchronization.

The remainder of the ESD is a series of subdescriptors with more specific and optional information. Details on `IPI_DescrPointer`, `IP_IdentificationDataSet`, `IPMP_DescriptorPointer`, and language descriptors are given in the IPMP and OCI sections. `SLConfigDescriptor` describes synchronization-related properties and is explained in the synchronization section. The remaining subdescriptors are covered in the following sections.

Google Exhibit No. 1021
Page 054

Google Exhibit No. 1063
Page 0818

3.1.3.1 Decoder Configuration The `DecoderConfigDescriptor` (Table 3.4) provides setup information required by the decoder to process the stream in question. First, the stream is roughly classified through the `streamType` value as visual, audio, scene description, and so on. Then, the `objectTypeIndication` gives further information on the actual coding employed, such as JPEG, MPEG-1, MPEG-2, and MPEG-4. For legacy stream types (MPEG-1, MPEG-2), profile and level information is directly included, as applicable.

The `upStream` flag is set for streams that provide upstream information from the client terminal back to the server. Attaching such a stream to an OD, in addition to a regular stream flowing from server to client, means that this object has its own back channel.

The remaining values in `DecoderConfigDescriptor` indicate the buffer size required for the decoding buffer as well as the maximum and the average bit rates that should be expected when receiving the stream.

The `DecoderSpecificInfo` descriptor serves as an opaque container from the MPEG-4 Systems point of view and carries information about the decoder configuration needed to decode the stream properly.

Finally, the `profileLevelIndicationIndexDescriptor` allows associating this stream to one or more complexity categories specified for the overall MPEG-4 presentation, as detailed in the complexity section below.

3.1.3.2 Quality of Service Descriptor Comparable to the `streamPriority` element in the ESD, the quality of service (QoS) descriptor is mostly of use for the media server and for gateways that intelligently forward streams or sets of streams. The QoS descriptor consists of either an (application-dependent) index of a predefined QoS scenario or a set of QoS qualifiers (Table 3.5). QoS qualifiers are defined as derived classes from the abstract `QoS_Qualifier` class.

Table 3.4 Decoder Configuration Descriptor syntax

```
class DecoderConfigDescriptor extends BaseDescriptor : bit(8)
      tag=DecoderConfigDescrTag {
    bit(8) objectTypeIndication;
    bit(6) streamType;
    bit(1) upStream;
    const bit(1) reserved=1;
    bit(24) bufferSizeDB;
    bit(32) maxBitrate;
    bit(32) avgBitrate;
    DecoderSpecificInfo decSpecificInfo[0 .. 1];
profileLevelIndicationIndexDescriptor profileLevelIndicationIndexDescr
      [0..255];
}
```

Google Exhibit No. 1021
Page 055

Google Exhibit No. 1063
Page 0819

Table 3.5 QoS Descriptor syntax and QoS Qualifier definition

```
class QoS_Descriptor extends BaseDescriptor : bit(8) tag=QoS_DescrTag {
    bit(8) predefined;
    if (predefined==0) {
        QoS_Qualifier qualifiers[];
    }
}

abstract aligned(8) expandable(2^28-1) class QoS_Qualifier : bit(8)
     tag=0x01..0xff {
    // empty. To be filled by classes extending this class.
}
```

A number of QoS qualifiers have been predefined to express the following:

- Maximum end-to-end delay for the stream in microseconds
- Preferred end-to-end delay for the stream in microseconds
- Allowable loss probability of any single AU as a fractional value between 0.0 and 1.0
- Maximum allowable number of consecutively lost AUs
- Maximum size of an AU in bytes
- Average size of an AU in bytes
- Maximum arrival rate of AUs in AU/second

This set of qualifiers may need application-specific extension, depending on the selection of a specific QoS framework.

3.1.3.3 Registration Descriptor Finally, a registration descriptor (Table 3.6) is provided to enable MPEG-4 users to unambiguously carry non-MPEG-4 ESs. Although MPEG does not (prefer to) open up the set of allowed ESs, it is acknowledged that applications will still have the need to convey such private streams. The registration descriptor provides a method to unambiguously identify the characteristics of these streams.

As in other such cases, ISO will issue a Call for Registration Authority, a body that will subsequently manage the space of `formatIdentifier` values and make sure that they are uniquely assigned. The second field in the descriptor may hold any additional information, as requested by the owner of a certain `formatIdentifier`.

Table 3.6 Registration Descriptor syntax

```
class RegistrationDescriptor extends BaseDescriptor : bit(8)
     tag=RegistrationDescrTag {
    bit(32) formatIdentifier;
    bit(8) additionalIdentificationInfo[sizeOfInstance-4];
}
```

Google Exhibit No. 1021
Page 056

Google Exhibit No. 1063
Page 0820

3.1.4 Stream Relationship Description

The mechanisms to describe a single ES for a media object have been introduced. Describing an object with a single ES description consists in creating an ESD and placing it within an OD. However, the OD framework has some more features that are specified in [MPEG4-1, Clause 8.7]. For example, it is possible to describe various alternative representations of the same media object.

In a simple case, a single media object is encoded in different qualities. The corresponding media streams can be generated independently, including for each of them an ESD with the necessary parameters, such as coding format and bit rate. Those ESDs are simply put together in one single OD. The client terminal will then select one of the streams and request it from the server for playback (see Figure 3.3).

In some application contexts, scalable coding is more adequate than providing a set of independent quality streams—for example, in the case of multicast distribution, in which the client cannot request a specific quality from the server. A base quality (base layer) is provided by a first ES that can be decoded by the decoder independently. Depending on the concrete scalability algorithms used by the media codec, this quality can be improved, possibly in multiple steps, corresponding to the multiple ESs (enhancement layers) that are generated. Here it is important to know in which order to pass the set of streams to the media decoder (see Figure 3.4). This is achieved with the stream-dependency mechanism of the ESD. Each stream points to the one it depends on; more precisely, it points to the `ES_ID` of that stream using the `dependsOn_ES_ID` syntax element. The client terminal in that case will select one or more of these streams for playback, starting, of course, from the base layer stream.

Both in the case of different quality alternatives for a media object and in the case of scalable coding, the receiving terminal has a choice. The terminal needs to figure out which of the streams to use, depending on its capabilities. The terminal can do this by simply evaluating all the ODs and using custom

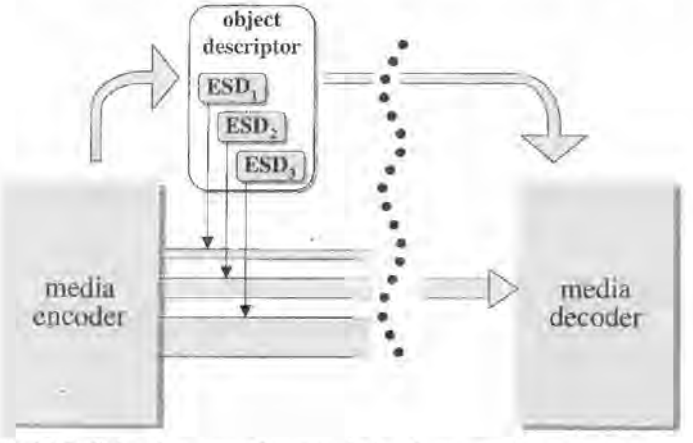


Fig. 3.3 Selecting one of a set of simulcast media streams.

Google Exhibit No. 1021
Page 057

Google Exhibit No. 1063
Page 0821

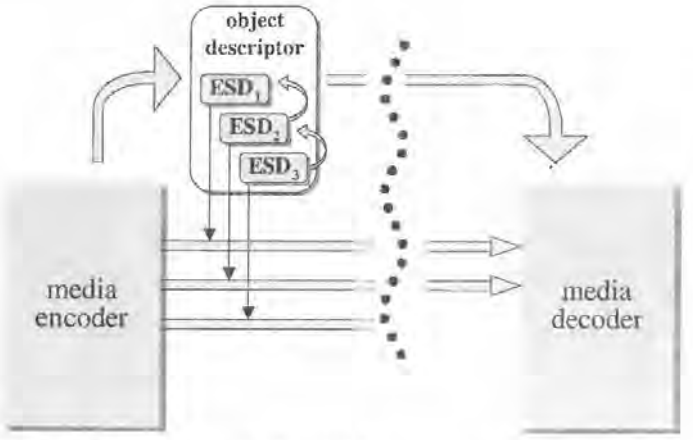


Fig. 3.4 Selecting a set of scalable streams.

rules to decide what to do. More likely, however, profile and level definitions are used for this purpose, as will be seen in the next section.

3.1.5 Content Complexity Description

The question of content complexity is important and has to be answered up front before an MPEG-4 terminal can decide whether it will attempt to decode and present that content at all. For a single media stream, it may be simple to examine the OD for that stream in all its advertised variations, and then to decide which one the terminal is capable of presenting. But now assume that the content consists of a large number of media objects embedded into a large scene. In that case, an examination of all related ODs would be quite arduous. Because of that, another mechanism has been implemented to answer the question of content complexity. It is basically a set of labels indicating profiles and levels for the content. The profile and level concepts will be addressed in detail in Chapter 13.

The profile and level indications are conveyed by means of two types of descriptors: a derivative of the OD called the *initial object descriptor* (IOD, Table 3.7), and the *extension profile level descriptor* (Table 3.8).

The IOD is identical to the regular OD, except that it features the said elements for signaling the different profiling dimensions for scene graph, audio, visual, graphics, and OD (see Chapter 13). Additionally, a flag (`includeInlineProfileLevelFlag`) signals whether these profiles and levels correspond to the sum of the complexity of the overall scene including all sub-scenes[5] (`includeInlineProfileLevelFlag=1`), or whether those complexities

5. Scene descriptions can be recursively included in each other using the `Inline` node, which will be introduced in Chapter 4.

Google Exhibit No. 1021
Page 058

Google Exhibit No. 1063
Page 0822

Table 3.7 Initial OD syntax

```
class InitialObjectDescriptor extends ObjectDescriptorBase : bit(8)
     tag=InitialObjectDescrTag {
    bit(10) ObjectDescriptorID;
    bit(1) URL_Flag;
    bit(1) includeInlineProfileLevelFlag;
    const bit(4) reserved=0b1111;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    } else {
        bit(8) ODProfileLevelIndication;
        bit(8) sceneProfileLevelIndication;
        bit(8) audioProfileLevelIndication;
        bit(8) visualProfileLevelIndication;
        bit(8) graphicsProfileLevelIndication;
        ES_Descriptor esDescr[1 .. 255];
        OCI_Descriptor ociDescr[0 .. 255];
        IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    }
    ExtensionDescriptor extDescr[0 .. 255];
}
```

Table 3.8 Extension Profile Level Descriptor syntax

```
class ExtensionProfileLevelDescriptor() extends BaseDescriptor :
    bit(8) ExtensionProfileLevelDescrTag {
    bit(8) profileLevelIndicationIndex;
    bit(8) ODProfileLevelIndication;
    bit(8) sceneProfileLevelIndication;
    bit(8) audioProfileLevelIndication;
    bit(8) visualProfileLevelIndication;
    bit(8) graphicsProfileLevelIndication;
    bit(8) MPEGJProfileLevelIndication;
}
```

are signaled by their own `InitialObjectDescriptors` or `ExtensionProfileLevelDescriptors`.

The `ExtensionProfileLevelDescriptor` is necessary because a single complexity indication for an MPEG-4 presentation may not be sufficient. As explained, ODs allow aggregating multiple representations of the same content with different characteristics. Therefore, a signaling means for the complexity of these different representations in terms of profiles and levels is required. This is achieved by attaching multiple `ExtensionProfileLevelDescriptors` to the initial OD, each describing one of these representations and uniquely labeled by a `profileLevelIndicationIndex`. Additionally, this descriptor serves to signal MPEG-J profiles (see Chapter 5) that did not exist in Version 1 of the MPEG-4 Systems standard.

Google Exhibit No. 1021
Page 059

Google Exhibit No. 1063
Page 0823

A stream that belongs to a given representation (or complexity category) may use the `profileLevelIndicationIndexDescriptor` (Table 3.9) from its `DecoderConfigDescriptor` to reference the appropriate `profileLevelIndicationIndex`.

Table 3.9 Profile Level Indication Index Descriptor syntax

```
class ProfileLevelIndicationIndexDescriptor () extends BaseDescriptor
:   bit(8) ProfileLevelIndicationIndexDescrTag {
    bit(8) profileLevelIndicationIndex;
}
```

3.1.6 Streaming ODs

The previous sections explained how media objects and media streams are defined and described through ODs and ESDs. They also detailed how to describe different encoded versions of the same media object and how to indicate the complexity of these objects. The next step is to communicate the ODs with all this information from the sender to the client terminal.

In MPEG-4 the focus is on enabling dynamic multimedia content. This led to a design in which ODs can be sent, updated, or deleted at any time during an MPEG-4 presentation. In fact, ODs are sent within their own OD stream [MPEG4-1, Clause 8.5]; that is, they are generally processed in the same way as other media streams or the scene description streams.

This MPEG-4 approach is quite different from existing mechanisms for signaling stream properties. Both SDP [RFC2327], in conjunction with the real-time streaming protocol (RTSP) [RFC2326] used in some Internet streaming applications, and the ITU-T Recommendation H.245 [H245] used for videotelephone and videoconferencing applications perform the signaling of stream properties only at the beginning of the multimedia presentation or communication.

Within the OD stream, a set of OD commands can be conveyed:

- OD update
- OD remove
- ESD update
- ESD remove
- OD execute

The OD update command conveys a set of ODs or initial ODs. Adding and updating an OD are treated the same way. For each updated OD, the terminal will check whether existing streams were removed from it or new streams have been added. Through an OD update, all the auxiliary information related to OCI and IPMP can be updated, too.

The ESD update makes it possible to convey individual ESDs and, therefore, plug an ES into an OD after that OD is already conveyed to the receiving

Google Exhibit No. 1021
Page 060

Google Exhibit No. 1063
Page 0824

terminal. It advertises at any stage during a presentation that an additional version of a media object just became available—for example, an enhanced quality version. Removing ESDs or ODs indicates future unavailability of the associated streams.

The OD execute command exists in order to let the terminal know upfront for which streams a decoder needs to be set up immediately. In the absence of such OD execute commands, the terminal may set up decoders only when the scene description actually requests the playback of some media stream. In that case, an extra delay for setting up the decoder must be expected, which will be implementation-dependent. The need for the OD execute command was established very late in the standardization process. Therefore it is found only in the second edition of the MPEG-4 Systems standard [MPEG4-1].

All these commands follow a syntax similar to that of the descriptors encountered before. They are given in Table 3.10 for completeness. All commands derive from the abstract class `BaseCommand`. As mentioned earlier, an abstract class defines a name scope for the class tags. So, `BaseCommand` defines

Table 3.10 Syntax of OD commands

```
abstract aligned(8) expandable(2^28-1) class BaseDescriptor : bit(8) tag=0 {
    // empty. To be filled by classes extending this class.
}

class ObjectDescriptorUpdate extends BaseCommand : bit(8)
     tag=ObjectDescrUpdateTag {
    ObjectDescriptorBase OD[1 .. 255];
}

class ObjectDescriptorRemove extends BaseCommand : bit(8)
     tag=ObjectDescrRemoveTag {
    bit(10) objectDescriptorId[(sizeOfInstance*8)/10];
}

class ES_DescriptorUpdate extends BaseCommand : bit(8)
     tag=ES_DescrUpdateTag {
    bit(10) objectDescriptorId;
    ES_Descriptor esDescr[1 .. 255];
}

class ES_DescriptorRemove extends BaseCommand : bit(8)
     tag=ES_DescrRemoveTag {
    bit(10) objectDescriptorId;
    aligned (8) bit(16) ES_ID[1..255];
}

class ObjectDescriptorExecute extends BaseCommand : bit(8) tag=
     ObjectDescriptorExecuteTag {
        bit(10) objectDescriptorId[(sizeOfInstance*8)/10];
}
```

Google Exhibit No. 1021
Page 061

Google Exhibit No. 1063
Page 0825

a new name scope for the command tags that is distinct from the name scope for descriptor tags.

3.1.6.1 Streaming Configuration Information in ODs In previous standards (such as MPEG-2), decoder configuration information (like coding format and coding options) is repeated regularly as part of the media ES itself in order to allow the random access to a stream, as indicated in Figure 3.5a. In case of MPEG-2, some of this configuration information is additionally repeated in the Systems layer [MPEG2-1]. In MPEG-4, it is suggested to convey the decoder configuration information within the `DecoderSpecificInfo` element (Table 3.4), avoiding any duplication of this information in the ES itself.

This approach makes it possible to adapt the delivery rules for the decoder configuration information to the delivery medium. In case of unicast delivery without the ability to randomly tune in, decoder configuration information is delivered up front and no repetition is needed. In case of a broadcast medium, the ODs can easily be repeated as often as needed within their OD stream, as indicated in Figure 3.5b. The arrows in this principal drawing indicate that it is possible to maintain a temporal link between the OD stream and the media streams through the use of time stamps.

Unfortunately, this ideal approach cannot always be followed, because the MPEG-4 toolbox approach allows the use of MPEG-4 Visual or Audio coding

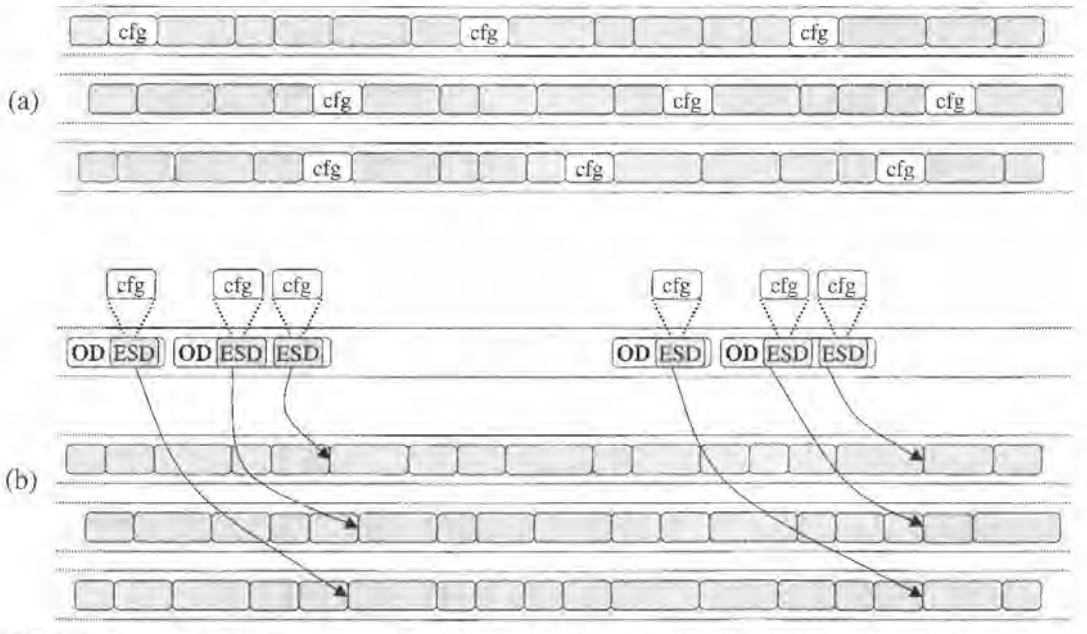


Fig. 3.5 Carriage of configuration (cfg) information (a) in each ES (possibly repeated), as in MPEG-2, and (b) in the decoder-specific info descriptor within a separate OD stream, as in MPEG-4.

Google Exhibit No. 1021
Page 062

Google Exhibit No. 1063
Page 0826

tools independently of the Systems tools, which include the ODs. In the absence of ODs, the ES syntax has to revert to the approach shown in Figure 3.5a.

3.1.7 Linking a Scene to Its Media Streams

It was mentioned before (Figure 3.1 and Figure 3.2) that ODs are the link between an individual media object as represented in the scene description and its associated media stream(s). Given that both ODs and the scene description are conveyed in streams, it becomes apparent that further rules are needed to associate an entire scene description stream to an OD stream [MPEG4-1, Clause 8.7.2].

3.1.7.1 Association of OD Streams and Scene Description Streams OD streams are usually associated with a scene description (BIFS) stream, as that stream conveys the information on how to assemble the various media streams described within the OD stream into an MPEG-4 presentation. The means to associate OD and BIFS streams is another OD. The ESDs for an OD stream and its associated scene description stream are put into one OD or IOD and the scene description stream is set to depend on its OD stream (Figure 3.6).

It is also possible to split the scene description information into more than one stream—basically, to do a scalable encoding of the scene description. The order of relevance of the various BIFS streams is again signaled through the stream dependency mechanism in the OD (Figure 3.7), as done for scalable media streams.

Splitting a single BIFS stream in this way is notably different than partitioning a scene in subscenes through `Inline` nodes (see Chapter 4). It just means that some of the scene description data (called BIFS commands) that construct a scene are sent in one stream and some in the other(s). The BIFS decoder in the receiving terminal merges the BIFS commands from those streams in the right temporal order prior to decoding. It is up to the authoring tool to use this feature in a meaningful way, for example to support error resilience for the scene description data.

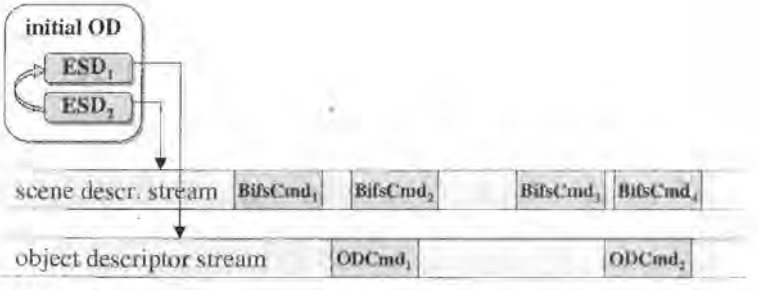


Fig. 3.6 Associating a scene description stream with its OD stream.

Google Exhibit No. 1021
Page 063

Google Exhibit No. 1063
Page 0827

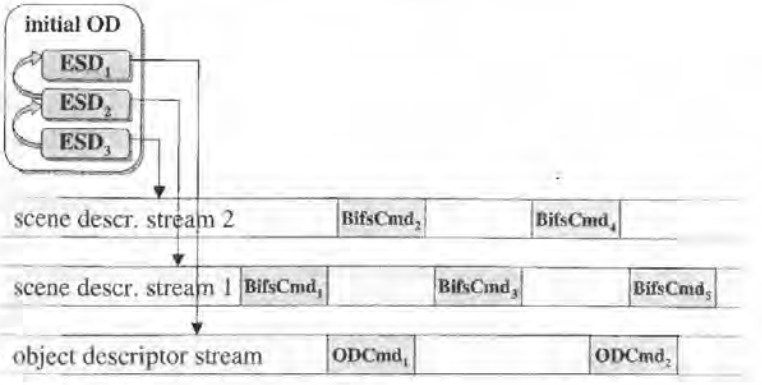


Fig. 3.7 Scalable scene description.

3.1.7.2 Association of MPEG-4 Presentations Another real-world question is how to author small MPEG-4 presentations and plug them together into bigger presentations later on. As an example, consider a main audiovisual show that is presented together with an overlaid ad banner including animated graphics and its own stamp-sized video stream (see Figure 3.8).

The authors for these two items are likely to work independently, but the overall presentation still needs to be assembled from both parts. Notably, in

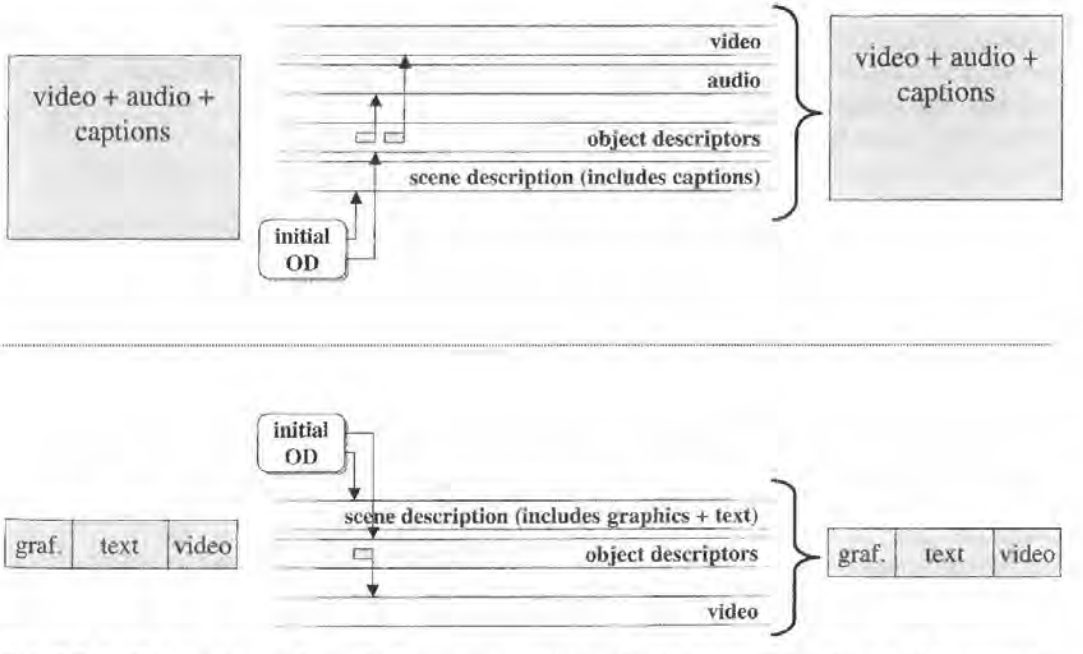


Fig. 3.8 Authoring two independent scenes: a main audiovisual program (top) and a banner-ad-like portion consisting of graphics, text, and video elements (bottom).

Google Exhibit No. 1021
Page 064

Google Exhibit No. 1063
Page 0828

MPEG-4 this must be possible without reediting either content, because in fact the assembly will take place at the receiver. Unfortunately, independent editing means that both partial presentations may have used the same numerical identifiers to mean different things.

The proper solution for such problems is the definition of name scopes for the various parts of the content. This means, similar to the name scopes for descriptor tags encountered before, those identifiers have to be unique only within a well-defined scope.

BIFS defines a mechanism to establish subscenes, called the `Inline` node (as will be further explained in Chapter 4). Each such subscene constitutes a separate name scope. This means that the same identifier values for BIFS nodes can be used in each subscene. This name scoping rule needs to be extended to the references from BIFS to OD and from OD to media streams, which are usually made through `OD_ID` and `ES_ID`.

Consequently, name scopes in the OD domain are defined the following way: Each OD that points to BIFS and OD streams opens a new name scope. In fact such an OD will be either the initial OD of a scene or an OD that is pointed to by an `Inline` node. So this translates back to the BIFS rule that each `Inline` node opens a new name scope.

For the example in Figure 3.9, this means that `OD_IDs` and `ES_IDs`, as

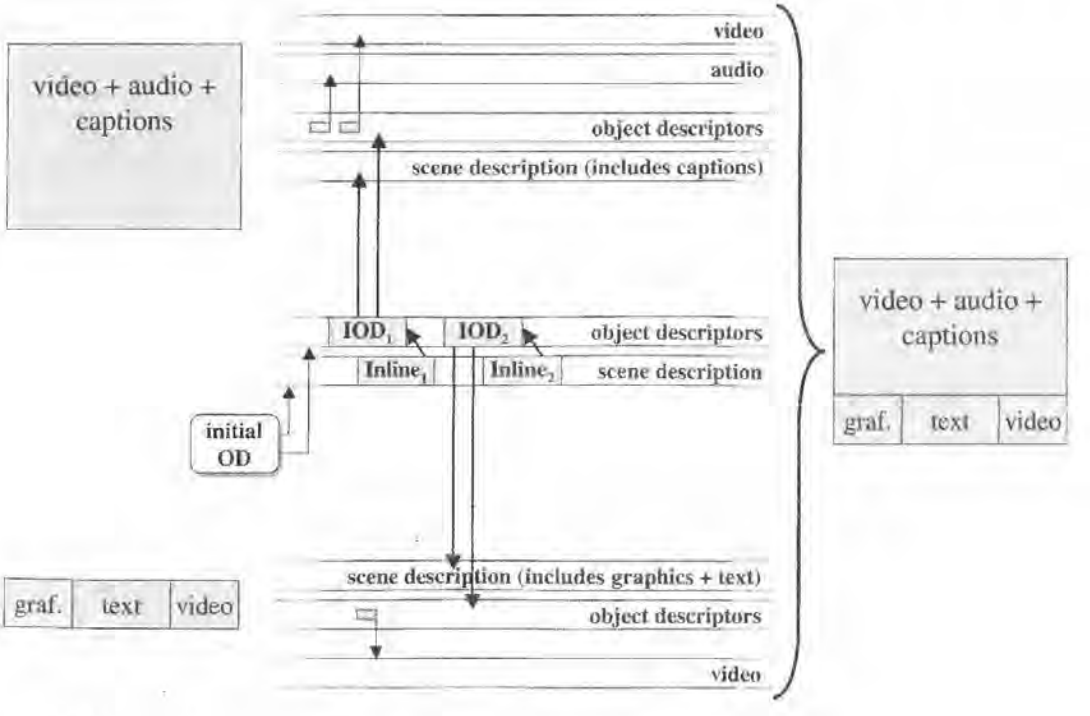


Fig. 3.9 Assembling a scene from two subscenes.

Google Exhibit No. 1021
Page 065

Google Exhibit No. 1063
Page 0829

well as nodeIDs, routeIDs, and so on in BIFS, can be assigned independently in both individual scenes. The initial ODs (see Figure 3.8) for both scenes open one unique name scope for each scene. In these conditions, there will be no conflict when the scenes are composed in a combined presentation.

The process to compose the two scenes into one is the following: Another piece of scene description has to be authored that *inlines* the other two scenes as subscenes, defining the right place on the screen for each of them, of course. The initial ODs of the two subscenes will become part of the OD stream for the composite scene, as indicated in Figure 3.9. Starting from the new initial OD, the receiver will be able to reconstruct the composite scene.

3.1.8 MPEG-4 Content Access Procedure

In the first part of this chapter, all the elements necessary to access MPEG-4 content have been introduced. Content access always starts with an initial OD (see the top left of Figure 3.10). The IOD contains pointers to at least the two

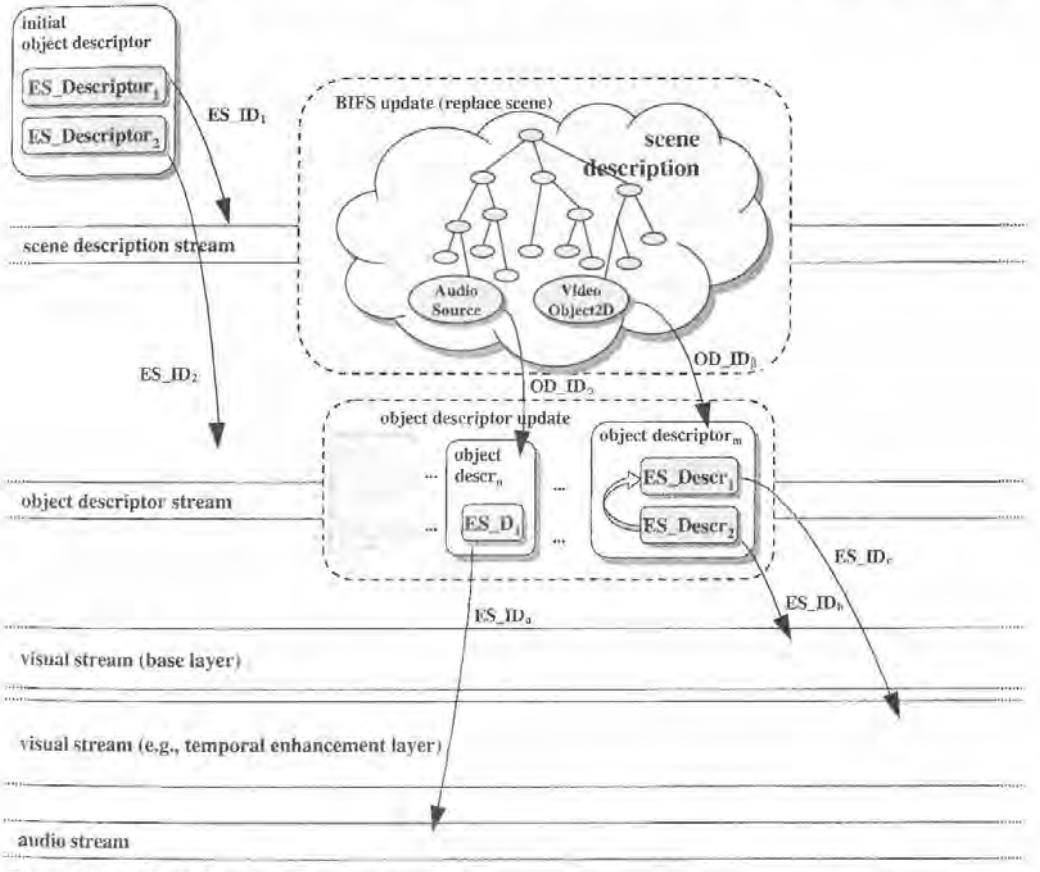


Fig. 3.10 MPEG-4 content access procedure.

Google Exhibit No. 1021
Page 066

Google Exhibit No. 1063
Page 0830

essential streams: a scene description (BIFS) stream and an OD stream. The ODs in the OD stream allow the location of the additional media streams used in the scene and have all the information needed to initialize the corresponding media decoders. The streams are then referenced from the BIFS scene through pointers to their respective ODs.

This content access procedure [MPEG4-1, Clause 8.7.3] has been formalized in the MPEG-4 standard as a series of calls to an API, called the DMIF Application Interface (DAI). This API is specified in Part 6 of the MPEG-4 standard, *Delivery Multimedia Integration Framework* [MPEG4-6]. The DAI is an abstraction layer that enables the description of the content access procedure independently of the actual transport and signaling protocols used.

MPEG-4 does not define the transport or delivery of its coded data. Therefore, depending on the concrete MPEG-4 application scenario, different means may exist to communicate the IOD to the receiver. For example, this could be achieved by publishing URLs for IODs on Web pages, by using a suitable signaling protocol such as RTSP [RFC2326] or ITU-T H.245 [H245], or by embedding the IOD in an MPEG-2 transport stream [MPEG2-1]. Some of those mechanisms will be further explored in Chapter 7.

3.2 Semantic Description and Access Management

The previous section detailed the essentials of MPEG-4 stream description necessary to enable simple content access. In addition, there are auxiliary streams and descriptors providing a little more semantic information about media objects and enabling content access management in order to prevent unauthorized content distribution and use. Two major types of auxiliary streams are defined in MPEG-4 Systems—OCI streams and IPMP streams. The intent of OCI is clear: It provides what is called *metadata*, describing objects semantically. IPMP provides hooks to control the legitimate usage of MPEG-4 content.

3.2.1 Object Content Information: Meta Information About Objects

A set of descriptors and a stream type have been defined to carry information about the media object in general: OCI descriptors and OCI streams. The following OCI descriptors are available [MPEG4-1, Clause 8.6.18]:

- `ContentClassificationDescriptor`
- `KeyWordDescriptor`
- `RatingDescriptor`
- `LanguageDescriptor`
- `ShortTextualDescriptor`
- `ExpandedTextualDescriptor`

Google Exhibit No. 1021
Page 067

Google Exhibit No. 1063
Page 0831

- ContentCreatorNameDescriptor
- ContentCreationDateDescriptor
- OCICreatorNameDescriptor
- OCICreationDateDescriptor
- SmpteCameraPositionDescriptor
- SegmentDescriptor
- MediaTimeDescriptor

Content is classified, for example, by genre. Classification schemata need to be registered with a registration authority to make sure that all implementers of MPEG-4 systems have access to this information. The same approach holds for the rating descriptor that is supposed to indicate the suitability of the content for different groups of audiences. Descriptors for keywords, short text, and expanded text are all freestyle but presumably in increasing order of complexity. Name of content creators and OCI creators as well as the creation dates for those are self-explanatory. The camera position descriptor inherited from SMPTE [S315M] allows keeping track of the camera position for that media stream. Finally, the segment descriptor and media time descriptors are additions that allow assigning a name to a temporal segment of the media stream based on the definition of a media timeline (see section on synchronization).

Any of those descriptors can be attached to an OD. This means that all the ESs referenced through this OD share the same (static) object content information. The only information that can be attributed to a single stream within one single OD is the language, because it should be possible to convey different language variants of the same content as one logical media object, as presented in Figure 3.11. This is why the `LanguageDescriptor` features as a subdescriptor of the ESD (see Table 3.3).

OCI also may be dynamically changing over time—for example, the keywords and textual descriptor. In that case, a separate OCI stream may be used to convey that information [MPEG4-1, Clause 8.4]. Each OD can refer to one OCI stream at most. Within an OCI stream, an `OCI_Event` message is used to wrap each set of descriptors that is to be conveyed jointly. The `OCI_Event` has the added functionality that the embedded set of OCI descriptors can now be associated with a temporal segment of the media stream, identified through start time and duration (see Figure 3.12). The `SmpteCameraPosition-Descriptor` is a typical example of a descriptor that mostly makes sense when conveyed in an OCI stream, as it may be assumed that camera position changes frequently over time.

Concluding, it should be mentioned that OCI was established at a time when no other MPEG metadata format was available. Today the standardization effort for the MPEG-7 multimedia description framework is well advanced [N4509]. All concepts expressed in OCI could equally well be expressed in MPEG-7. In fact, a label for MPEG-7 streams has already been reserved. So, instead of an OCI stream, an MPEG-7 stream can be associated

Google Exhibit No. 1021
Page 068

Google Exhibit No. 1063
Page 0832

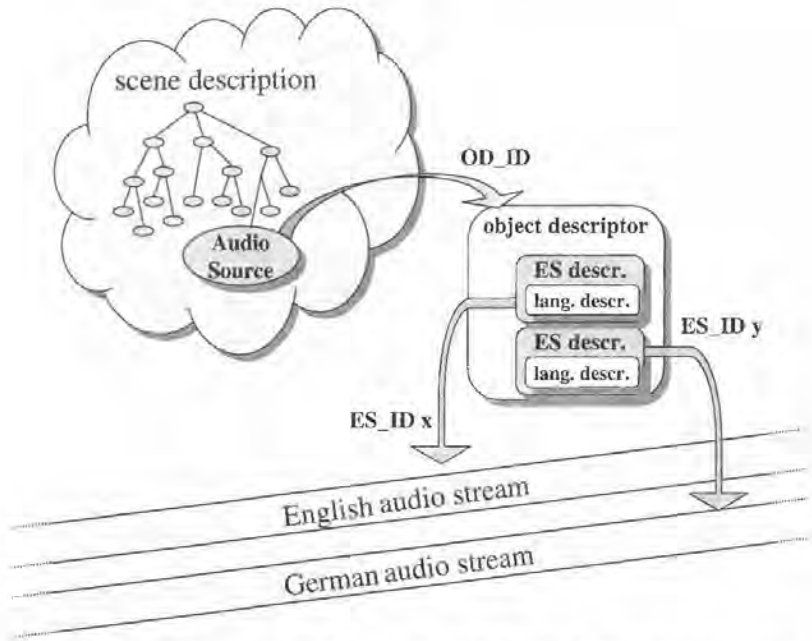


Fig. 3.11 Describing one audio object available in different languages.

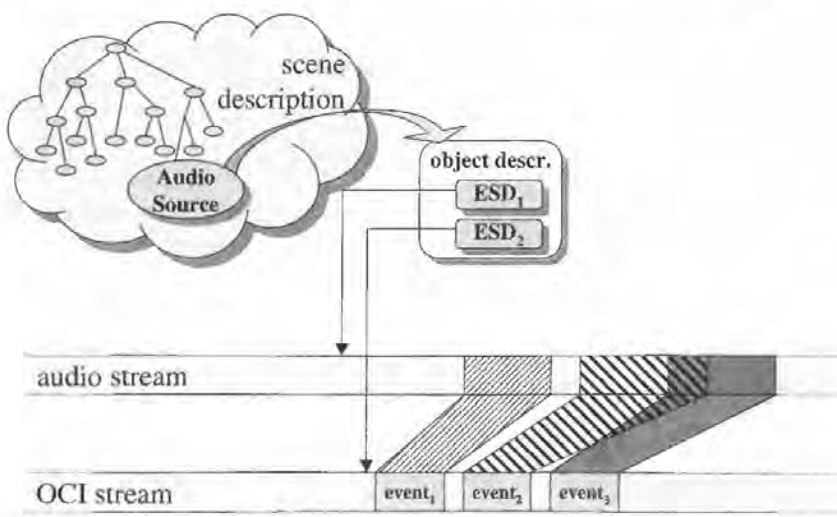


Fig. 3.12 OCI stream with various OCI events.

with an MPEG-4 media stream. It is hard to predict how OCI and MPEG-7 will coexist. It is conceivable that future MPEG-4 applications will rely more on MPEG-7 than on MPEG-4 OCI as a metadata format because of its larger capabilities, but it may also be that applications with few metadata requirements continue to use OCI.

Google Exhibit No. 1021
Page 069

Google Exhibit No. 1063
Page 0833

mined by the maximum bit rate of the associated ES, as indicated in the ES descriptor. As with the basic SDM compliance, it is the sender's task to ensure that the FlexMux buffers never overflow. Or, phrased positively, the sender knows how to interleave data coming from different streams, being sure they will arrive in time, relative to all the other streams.

In Section 7.4.3.5, it will be seen how this model is further extended to cover the whole MPEG-4 over MPEG-2 delivery stack.

Both the `FlexMuxBufferDescriptor` and the `FlexMuxTimingDescriptor` need to be conveyed *out-of-band*—that is, outside the actual FlexMux stream, and, as for the `MuxCodeTableEntries`, this will be specified in the MPEG-4 adaptation of each DMIF instance. A proposal for in-band transport of these descriptors has been made, as that would give the additional benefit of establishing the FlexMux stream as a self-supported exchange format for multiplexed MPEG-4 streams, including delivery timing, independent of any delivery layer.

A potential use for in-band descriptor delivery would be applications that need to convey the same preproduced presentations both on MPEG-2 and on IP, or even those that need to forward content arriving on one delivery stack to another one. That would be largely eased, as a necessary gateway could simply act as a media pump, retrieving data from one delivery instance and pumping it out again through the second delivery instance, driven by the time stamps embedded in the FlexMux stream and, of course, taking into account the information in the embedded descriptors.

So, the rationale for in-band transport of FlexMux descriptors is the ease of streaming MPEG-4 presentations across heterogeneous networks. A generic solution to ease the exchange of MPEG-4 presentations, not necessarily in a streaming manner, already exists and is discussed in Section 7.3.

7.3 MPEG-4 File Format

Historically, prior to the MPEG-4 standard, MPEG has not had an explicit file format. MPEG-1 and MPEG-2 content typically is exchanged as files that represent, in a sense, a stream ready to be delivered. These files usually have embedded absolute time stamps, and the data often has been fragmented with a particular transport in mind (e.g., MPEG-2 Systems transport stream). These characteristics can make random access difficult and editing or reuse of the streams hard without decoding, demultiplexing, and then rebuilding the stream after editing. In MPEG-4, no single transport protocol is *preferred*, so fragmenting the data to suit the packet size of a specific protocol was not acceptable.

The MPEG committee sought a *life-cycle* file format—one in which the files could be used when capturing media, editing it, and combining it; when serving the media as a file download or as a stream; and when exchanging partial or complete presentations. This need for a life-cycle format is not met in many simple file format designs. For example, as noted previously, the

Google Exhibit No. 1021
Page 070

Google Exhibit No. 1063
Page 0834

design approach of MPEG-2, in which a stream is simply recorded to a file, makes editing hard. A more flexible design and approach were needed.

The design approach and principles embodied in the QuickTime file format from Apple [QUICKTIME] were closest to meeting these needs. After a requirements definition phase and a call for proposals, both design principles and details from QuickTime were adopted and refined to build the *MP4 file format*.

Perhaps the most unusual aspect of the MP4 design is the separation of the media data itself—the video data, for example—from data about that media data, often called *metadata*. Metadata might include timing information, the number of bytes occupied by a video frame, the location of the file in which it is stored, its position within that file, and so on. This timing and structural data is kept separate from the media data itself, in compact tables. So, for example, neither the length in bytes of a video frame nor its time stamps are stored adjacent to that frame, but in tables in the metadata part of the file. This is what is meant when the MP4 format is described as *nonframing*.

To further facilitate editing, the timing data is relative, not absolute. This means that time stamps do not have to be rewritten when, for example, a frame is inserted or deleted. Also, the interrelationships between streams are compactly represented in the metadata. In addition, media data is stored in its *natural* or base state, not preferring any one transport protocol or system: This means, for example, video frames are stored whole, not fragmented to any fixed size.

When the data must be streamed, special instructions for the protocol, optionally stored in the file, instruct streaming servers in the process of fragmenting and time stamping the data. These instructions are known as *hints*. As a result, the MP4 file format is, on first inspection, more complex than simple single-function file formats. However, as a format, MP4 is also more powerful than single-function formats and can be used for a wide variety of purposes.

With these principles in mind, this section looks first at how time is managed, then at how storage and space are managed in the context of the MP4 file format. Then the way MPEG-4 Systems concepts and structures are handled in the file format is explained. The special support for streaming is then detailed, and the MP4 section concludes with a printout of a small example file. This printout provides an example that can be reviewed as the format is described.

7.3.1 Temporal Structure: Tracks and Streams, Time and Durations

An MPEG-4 presentation is composed of a number of streams. In MP4, each stream is stored as a separate track in a file. A *track* in a file represents a timed series of media—successive video frames, for example. Each track has a *track identifier*, which maps to the ES_ID. A track is treated as a sequence of

Google Exhibit No. 1021
Page 071

Google Exhibit No. 1063
Page 0835

samples, which are AUs, in MPEG-4 terms—the smallest entities to which timing can be attributed (see Chapter 3). So, a sample in a video track is a coded frame or VOP; in an audio track, a compressed frame of audio (not a single uncompressed audio sample); in a BIFS track, a set of instructions for one-time instant; and so on.

For each sample, the file format records the time *difference* between its time stamp and the time stamp of the next sample—its *duration*. This relative timing makes the process of editing much easier, as sections of the timeline may be moved around without disturbing their internal structure. Indeed, the file format has a facility for expressing precisely this operation. A set of structures known as an *edit list* can provide an explicit *map* of how the timeline of a single track should be mapped into the timeline of the presentation. Edit lists allow for the insertion of *empty time* and the use (and reuse) of sections of the timeline. So, a repeated piece of media can be defined once and used repeatedly by reference.

MPEG-4 video follows earlier MPEG standards and uses the concept of bidirectionally predicted frames or VOPs, called B-frames or B-VOPs. To decode a B-frame, the decoder needs three frames: the closest non-B-frame that will be displayed before it (called here F[p]), the closest non-B-frame that will be displayed after it (called here F[s]), and the B frame being decoded itself (called here F(b)). The decoder must be given these frames not in the order they are displayed but in the order that they must be decoded: Frame F(s) is given to the decoder before the B-frame F(b). To support this, and to manage decoding buffers in some terminals, MPEG separates the concepts of decoding time and composition (display) time. In the MP4 file format, frames are stored always in decoding order, and by default the decoding and composition time stamps are the same. However, an optional table can provide a positive offset from the decoding time to the composition time, when they differ. The reordering for display purposes is managed by this composition time offset.

Each track has a time scale, which defines how fast its clock ticks. Using an appropriate time scale enables precise definition of timing. For example, usually the time scale of an audio track will be chosen to match its sampling rate, to enable sample-accurate timing (e.g., a time scale of 22,050 ticks per second for 22.05 kHz sampled audio). In video, a time scale of 30,000 ticks per second, with media sample duration of 1,001 ticks, exactly defines NTSC video (often, but incorrectly, referred to as 29.97 frames per second).

In many sequences of media, some of the samples (frames) can be decoded without knowing any other samples: In video, these are known as I-frames or I-VOPs. In general, such samples provide useful random access or synchronization points. The indices of these samples are stored in a table, for each track, to enable random access and editing.

Sometimes, when seeking, it can be useful to find extra random access points. The file format provides a facility called *shadow sync* samples. These

Google Exhibit No. 1021
Page 072

Google Exhibit No. 1063
Page 0836

samples provide an alternative random access point (I-frame or sync sample) for a sample that is not a random access point.

Streams stored in the same file are normally assumed to be synchronized together. However, the file format provides explicit facilities to indicate the synchronization relationships (clock reference relationships) between streams, if that is needed. This is done by means of *track references*, which are described in more detail below.

When a file is delivered as an MPEG-4 stream, the timing is conveyed by the MPEG4 SL, or by the transport layer being chosen (such as RTP on TCP/IP networks). Because MP4 files do not record the actual time stamps (as noted previously), the time stamps must be computed. This computation proceeds in three steps:

1. If the track has an edit list, this list is used to build an overall time map of the stream times from the implicit timeline of the sample times.
2. The decoding time stamp of each AU is then computed by adding the *durations* of the preceding samples in that track; this time is then mapped according to the edit list.
3. If the track has explicit composition time information, it is represented by the offset from the decoding time computed in step 2. The composition offset must be added to the decoding time, to obtain the composition time, and then again mapped through the edit list information (if any).

This process is shown in Figure 7.10. In Figure 7.10, a simple timeline of 10 samples is shown with an edit list. The edit list contains four edits: an empty edit, then two edits each containing three samples, and finally an edit containing two samples (marked empty, 3 sample edit, and 2 sample edit, respectively). The computation of the time stamps for the second sample involves (a) the duration of the empty edit; (b) the duration of the preceding sample in the edit, which gives the decoding time for this sample; and (c) the composition offset for this sample, which, added to the decoding time, gives the composition time.

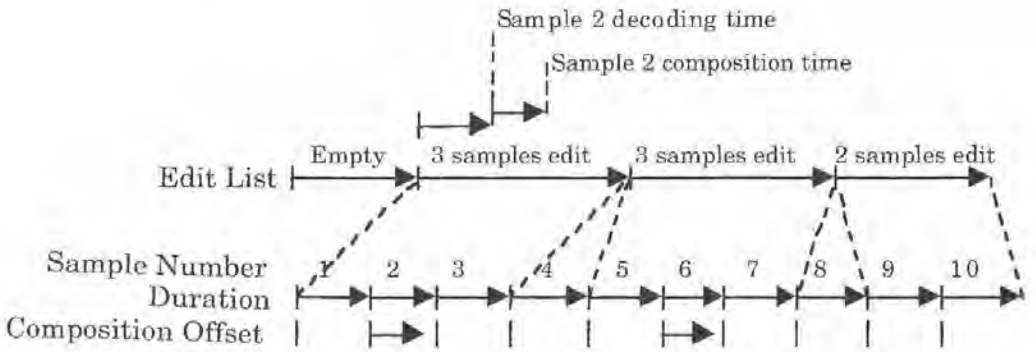


Fig. 7.10 Calculating time stamps for a sample.

Google Exhibit No. 1021
Page 073

Google Exhibit No. 1063
Page 0837

7.3.2 Physical Structure: Atoms and Containers, Offsets and Pointers

This section introduces the concepts that underpin the physical structures in the MP4 file. All data stored in MP4 files is organized in atoms. An *atom* is a container that has an explicit *type* and *length*. So, for example, the synchronization points discussed previously can be found in a table stored in each track, in an atom of type `stss`; this atom contains a table listing, in ascending order, the numbers of the samples that are random access, or sync, points.

Atoms can contain other atoms. Typically, an atom either serves as a container or has fields that define particular concepts, although this is not a general rule. Any media data (samples) are stored in a media data atom (of type `mdat`).

All the metadata in a presentation is stored in the *movie* structure, the atom of type `moov`. Unused space in a file is designated with an atom of type `free`. The three atoms `mdat`, `moov`, and `free` are generally the only three atoms that occur at the top-level of a file. Of these, only the `moov` atom has internal structure. Inside the `mdat` atom, the media data itself is stored. Clearly the content of a `free` atom is irrelevant.

The `moov` atom contains a `header` atom and then a set of tracks. The movie header (`mvhd`) gives basic information about the presentation—the date it was last edited, for example, and its overall duration. The metadata for each track is stored in an atom of type `trak`.

Like the movie header, the `trak` atom has a track header, which also has creation and edit dates, a time scale for this track, and the overall duration of the track. This `trak` atom in turn contains a series of atoms, which define the media and how it may be handled.

There are four important structures in the `trak` atom. They are the edit list, the data reference information, the handler information, and the sample table.

The *edit list* is related to timing and was discussed previously. The *data reference information* defines whether the media data used in this track is stored in the same file or in other files (referenced by `URL`). This ability to refer to media data in many files is important when composing a project: Media from a library of files may be used by reference, and only copied when a self-contained file is needed. The *handler information* provides the first level of differentiation of the tracks; the handler type identifies the general kind of information in a track, such as visual, audio, BIFS, and so on. The *sample table* provides the detailed information about each sample—AU—within the track. It is structured as a set of atoms that are formatted as tables. These atoms define both the physical location of each sample (frame of video, for example) and its timing. To keep these tables compact, a variety of techniques is used. One that is used to compact the location and size information relies on the observation that several frames from the same track are often stored contiguously, even when data from various tracks is interleaved. This run of contiguous samples is called a *chunk*. The sample-to-chunk table provides the

Google Exhibit No. 1021
Page 074

Google Exhibit No. 1063
Page 0838

mapping from sample number to chunk. The position of each chunk is recorded as a chunk offset (using 32 or 64 bits), which is measured from the beginning of the file in which the chunk resides. The length, in bytes, of each sample also is recorded, in the sample size table. Therefore, by using the data reference from the track, the chunk offset, and the sizes of the preceding samples in the same chunk, it is possible to find the data file containing the sample, which may be a file referenced by URL from the MP4 file itself, the chunk within that file, the offset of the sample within the chunk, and the size of the sample itself.

Figure 7.11 shows an example of a file that starts with the media data. In this media data, the frames of video and frames of audio are stored. There are three video chunks, containing three, two, and two samples (frames or VOPs) each. There are two audio chunks, containing two samples each. In this example, there also is some unused data. This unused data might be a video frame that has been edited and now stored elsewhere in the file, or any other kind of data that is no longer needed.

Note that because of the way this works, there is no need for the samples to use all the data in a file: There may be unreferenced data between the chunks. This can be used to skip unwanted data or the headers of a foreign file format, for example. Nor do the chunks have to occur in any specific order; data can be collected from disparate places in the files as it is needed. This means that much editing can be done by reference, simply changing tables and pointers.

The final structural concept that needs to be introduced is that of a *sample description*. A terminal wishing to display an MPEG-4 presentation needs to know how to decode each track: not only whether the data is visual or audio, but also how to set up each decoder to *understand* the coded data (e.g., whether the audio is AAC or HVXC, for example). This means knowing its corresponding audio or visual object type and decoder specific information. As with all the metadata encountered so far, this information is kept in a table. There is a set of sample descriptions in each track, and each sample is implicitly tied to exactly one of them. The name of the sample description *names* the

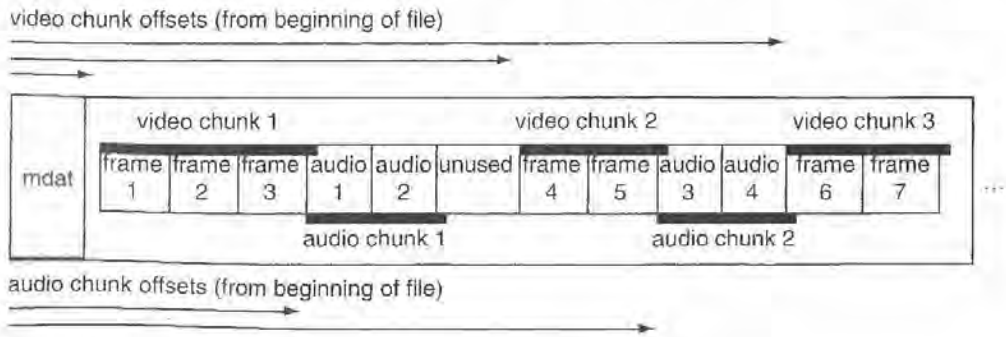


Fig. 7.11 Chunk offsets.

Google Exhibit No. 1021
Page 075

Google Exhibit No. 1063
Page 0839

media format, and the parameters to that sample description parameterize the decoder as needed.

7.3.3 MPEG-4 Systems Concepts in MP4

So, how do the general concepts introduced previously map into the concepts introduced in MPEG-4 Systems? In order to facilitate editing, the descriptors for each stream are stored in the track that contains the stream. This makes it easy to move a stream from one file to another, for example. However, when a presentation is streamed, these descriptors are delivered to the terminal in an OD stream, or in the IOD (see Chapter 3). To avoid duplicating the descriptors, a special technique is used inside MP4 files: Neither the IOD nor the OD stream stored in the file contains ES descriptors (ESDs). Instead, a reference to the appropriate tracks is placed in the ODs.

A new MP4 structural concept is used here. Each track has a table of typed references to other tracks. The OD track has track references to those tracks for which it will carry setup information. In that OD track, at the time when an ESD is needed, a structure is stored instead which refers to the track from which the ESD is needed. In the process of streaming, the reference is replaced by the actual descriptor from the referenced track.

The initial *hook* into an MPEG-4 streamed presentation is the IOD. This descriptor, too, would normally contain the ESDs for the streams that must be initially decoded. As with the OD stream, these are not stored in the IOD; instead, again, the track ID of the initial tracks is stored in the file, and these structures must be replaced by the appropriate ESDs when the file is streamed.

The other MPEG-4 streams (and, indeed, streams that are integrated with MPEG-4, such as MPEG-7) follow these structures without needing any specific support from the file format. For example, BIFS streams—both animation and update—use these structures, including the concept of random access, or sync, samples. A random access point in BIFS performs a `Replace Scene` command.

7.3.4 MPEG-4 Track Types and Storage

In the MP4 file format, determining how to decode a track involves two levels of decision. First, what basic kind of track is it: audio, visual, and so on? And then, precisely how are the samples coded: what codec—MPEG-4 object type—and with what parameters and settings (decoder-specific configuration)?

The first decision is made by examining the *handler type;* along with the handler type, there is a type-specific header atom within the track structures for each kind of track. The code points shown in Table 7.5 are used for the handler type and the type-specific header atom.

The last line in Table 7.5 is included for completeness; clock reference streams are not usually stored within the file. Instead, if a particular stream-

Google Exhibit No. 1021
Page 076

Google Exhibit No. 1063
Page 0840

Table 7.5 MP4 handler and header atom types

Stream type	Handler type	Header atom type
VisualStream	vide	vmhd
AudioStream	soun	smhd
ObjectDescriptorStream	odsm	nmhd
SceneDescriptionStream	sdsm	nmhd
MPEG7Stream	m7sm	nmhd
ObjectContentInfoStream	ocsm	nmhd
IPMP Stream	ipsm	nmhd
MPEG-J Stream	mjsm	nmhd
Hint Track	hint	hmhd
ClockReferenceStream	crsm	nmhd

ing transport requires a clock reference stream, it is constructed when a file is prepared for streaming, or by the streaming server itself.

7.3.5 Hinting

The penultimate line in Table 7.5 declares hint tracks. These are not MPEG-4 media tracks at all; instead, they are the key to understanding how efficient streaming may be done from a file in which the media data has been stored without regard to the needs of a particular streaming protocol.

Streaming media data involves sending media to a client system *just in time*, using small amounts of buffering. If packets get lost in transit, the client usually has to carry on; even if a retransmission is requested, there is no assurance it will arrive in time. The way in which the decoder can cope with this loss is critically dependent on the way the media data was originally placed into packets. For example, if a media sample (such as a video frame) is split arbitrarily into packets, any one of those packets being lost will usually make the rest of the frame, or perhaps the entire frame, unusable, and packets that did arrive will have to be discarded.

However, this problem may be alleviated by packetizing the media data carefully. If appropriate *boundary* points are found in the media, then the client often can recover all the data from the packets that successfully arrived. This can make a major difference to the user experience. For example, if a stream divides each video frame into six packets, and there is 10% loss, almost half the frames will lose at least one packet ($1 - 0.9^6$). If all those frames are discarded, the user experience is of very degraded video (approximately 50%

Google Exhibit No. 1021
Page 077

Google Exhibit No. 1063
Page 0841

missing). If, instead, the parts of those frames that arrive are displayed, the user experience is of the full frame rate, with 10% degradation.

The problem in finding these boundary points is twofold: Scanning the media data to find these boundaries is time consuming, and the way to do it requires the knowledge of the coding system being used—format parsing. If a media server must execute these rules as it streams, it will slow down; and if new rules are introduced, then every server must be updated.

Hint tracks provide a bridge between the media data and the packet data. In a media track, a sample is an audio frame—a media sample. In contrast, a hint sample tells the server how to make a packet or set of packets. Hint samples reference the media tracks they are packetizing and contain instructions, such as "insert these 4 header bytes into the packet," "copy 200 bytes from offset 43 in sample 267 of my media track," and so on. By following these instructions, the server can rapidly build valid packets without knowing either the coding system or the packetization rules for that coding system. Special authoring tools called *hinters* generate these hint tracks, after the editing process is complete. Hinters are specific both to the format of the media data and to the requirements of a specific transport protocol. The format-specific knowledge required has been moved from the server to the hinter. This means that the hinter is at liberty to compute the boundaries carefully and perform other optimizations, transformations, or corrections that are needed. So it would normally be the hinter that constructs any required clock reference stream, for example. And likewise the hinter would transform the OD track into an OD stream, replacing track references by the required ESDs from those tracks.

Of course, servers always have the freedom to know the parsing rules and packetize on the fly; this is why they are called hints—hints that the server is free to ignore.

Hint tracks are structured like media tracks. Just as media tracks must cope with differing coding systems, so hint tracks must manage multiple streaming protocols. Each server must find the hint tracks for the protocol it uses, and ignore the rest.

As a result, the same file contains both the original media—unpacketized, in its native format—and support for potentially multiple streaming formats. Indeed, by careful use of the *data references*, various MP4 files can be built that use the same media database in different ways. A hinted file can always be taken from a streaming server, the hint tracks deleted, and then edited, rehinted, and replaced on the server. This is part of the life-cycle management that MP4 offers.

The format of the hint samples is, of course, protocol-specific. Because this parallels the way that the format of media samples is codec-specific, the same structure is used. The sample description both names the protocol (and hence defines the format of the hints) and provides any necessary parameters to the server.

Google Exhibit No. 1021
Page 078

Google Exhibit No. 1063
Page 0842

At the time of writing, the format of hint tracks for the RTP protocol is being standardized. There is also a proposed format of hint tracks for MPEG-2 Systems transport streams.

7.3.6 Atoms

At this point, all the basic MP4 concepts are in place: how time and space are handled, how MPEG-4 structures are mapped into the MP4 file format, and the basic concept of hinting. This section presents the atoms that make up an MP4 file.

All atoms start with a 4-byte `size`, and a 4-byte `atom type`. The type is usually four printable characters (this makes debugging and inspection of files much easier). The `size` is the length of the entire atom in bytes, including this 8-byte header. Two special values of the `size` field are relevant: 0 means that this atom extends to the end of the file, and 1 means that a 64-bit `size` field immediately follows the `type` field. This allows for an occasional very large atom.

Many atoms are defined to have a `version` and `flags` field, of 1 and 3 bytes respectively, following the `type`. These are called *full atoms* in the specification. A few atoms have more than one version, and use the `version` field to indicate which variant is present. The `flags` field is used by some atoms for purposes specific to that atom.

7.3.6.1 Top-Level Atoms Three kinds of atoms occur at the top-level of MP4 files (not contained in any other atom):

1. **`moov`:** The movie atom `moov` is the container for all the metadata in the file. A movie atom is mandatory in an MP4 file; it can be anywhere in the top-level sequence of atoms. The `moov` atom is a pure container atom; it has no fields of its own, and it contains other atoms.
2. **`mdat`:** If there is media data in this file, it is stored in an `mdat` (media data) atom.
3. **`free` or `skip`:** If there is spare space, that is in atoms of type `free` or `skip`.

7.3.6.2 Inside the Movie Atom A movie atom `moov` contains these atoms:

- A movie header atom `mvhd`
- Optionally, an IOD atom `iods`
- A track atom `trak` for each track in the presentation

The movie header has bookkeeping fields for the creation and modification time of the movie itself. It also records the duration of the presentation (the duration of its longest track, considering edits), and defines the time scale (in ticks per second) for the movie and track duration values. Finally, it has a `helper` field that suggests the next free track identifier. The track atom is a

Google Exhibit No. 1021
Page 079

Google Exhibit No. 1063
Page 0843

CHAPTER 8

Natural Video Coding

by Michael Wollborn, Iole Moccagatta, and Ulrich Benzler

Keywords: object-based video coding, motion compensated hybrid video coding, shape coding, error resilience, scalability, texture coding

The coding of natural video content, including still images, has been under investigation in research and industry for a long time. But what does it mean, to *code* video content? In early times, the only goal was to represent a video scene, be it moving or not, in a format allowing transmission or storage. And, because storage capacity and bandwidth were and still are restricted resources, the main and ultimate objective of video coding for a long time was *compression*. In this sense, the coding of video content aimed mainly at representing it using as few data as possible, at a certain visual quality, or vice versa, to maximize the visual quality at a given bit rate.

This coding paradigm is reflected in the early video coding standards, such as H.261 [H261] and MPEG-1 Video [MPEG1-2]. Of course, they also included application-driven functionalities such as random access, scalability, and others; however, the idea of improving compression was always the driving force behind all the activities. At the same time, several research activities [e.g., Kunt85, MuHO89, SaTM95] investigated and proposed a different approach than the well-established *coding of rectangular frames*—namely, the so-called *object-based* or *region-based* video coding. Although one objective of these proposals was higher compression, the principal ideas opened a much broader scope of possible functionalities and applications. Influenced by both

Google Exhibit No. 1021
Page 080

Google Exhibit No. 1063
Page 0844

the well-established video-compression standards and the new ideas of coding (not the rectangular window to the visual world, but its content), the MPEG-4 natural video coding activities began in 1995 with the aim of combining the benefits of both approaches: high compression over a broad range of bit rates and innovative content-based functionalities.

This chapter gives an overview of the MPEG-4 natural video coding tools [MPEG4-2]. However, it does not describe the complete standard in all details—that would require a book of its own. Therefore, the emphasis is on the general concepts and on those tools that are newer, compared to already existing technology and standards. For more details on specific topics, the reader is referred to the references.

8.1 General Overview

In this section, an overview of the MPEG-4 natural video coding tools is given. The functionalities and respective application scenarios are described briefly, followed by a short description of the principal MPEG-4 video coding approach.

8.1.1 Functionalities and Application Scenarios

The MPEG-4 Visual standard [MPEG4-2] comprises tools for coding natural visual sources—that is, sequences of images (video)—as well as still images (visual texture). Whereas the video coding is based on the well-known motion-compensated hybrid DCT coding scheme, the visual texture coding (VTC) is based on wavelet transform and zero-tree coding.

In addition to previous standards that addressed only coding efficiency, the MPEG-4 video coding tools provide several additional functionalities (see also Chapter 1): coding of arbitrarily shaped objects; efficient compression of video sequences and still images over a wide range of bit rates; spatial, temporal, and quality scalability; and robust transmission in error-prone environments.

These advanced features greatly aid content creators in generating rich video and thus multimedia content. In particular, the ability to *code objects of arbitrary shape and size* goes beyond the scope of all previous video and image coding standards. It allows content creators to overcome the limits of frame-based content coding and opens the doors for a multitude of innovative applications in the future.

The ability to efficiently compress still textures and videos over a wide range of bit rates using the MPEG-4 tool set greatly simplifies the content creator's job by providing one-stop shopping opportunity for nearly lossless to lossy compression while delivering the best quality for the bit rate required by the application.

Scalable video coding is of interest for various applications. For mobile communication, a hierarchical transmission system can be used, in which the

Google Exhibit No. 1021
Page 081

Google Exhibit No. 1063
Page 0845

base layer signal is especially well protected against errors in the transmission channel by applying robust forward-error correction to the base layer bitstream. For video server applications, scalable coding allows users to access the video at different bit rates, without the need for multiple coding and storage of separate video bitstreams in different resolutions or qualities.

To enable the transmission over wireless and wired networks with severe error conditions, the MPEG-4 Visual standard has adopted *error-resilience coding* tools. Depending on the transmission channel, these tools can replace or complement channel coding to improve quality of service (QoS) [M4575].

Finally, the MPEG-4 VTC provides an efficient tool for the compression of still images, based on the wavelet transform and zero-tree coding. It can be combined with the MPEG-4 shape coding tools and thus provides, with this respect, the same features as the video coding tools. The combination of VTC with the natural video coding tools allows the coding of scenes comprising both moving and still images by means of a single standard.

8.1.2 Basic Principles

One important principle in ISO/MPEG standardization activities is that the encoder is not standardized—only the bitstream it produces and the procedure for decoding it (i.e., the decoder). The reason is that for interoperability, it is not necessary to standardize the encoder. As a positive consequence, there are several degrees of freedom and optimization left for building the encoder. This gives one the possibility of improving the performance of the coding algorithm by introducing new (non-normative) techniques while the standard is already fixed, and it also leaves room for competition in building better encoders. This principle holds true also for MPEG-4 natural video coding. Therefore, in the remainder of this chapter, mostly decoding procedures will be described. Only when it is essential for the understanding of the technical issues will encoding algorithms be explained.

As mentioned in the introduction, MPEG-4 natural video coding tools comprise a whole tool-set for different applications and use cases. Moreover, these tools can be combined to code different sources of natural video. Not all of the tools are required for every application, and in some cases, it may be preferable if a decoder not support certain tools (e.g., for reasons of complexity or computational load). On the other hand, for reasons of interoperability, a decoder should at least support a clearly defined set of tools. Thus, for the MPEG standards, *profiles* are defined. For each profile, the tools to be supported are specified, and each decoder that claims to be compliant to a certain profile must support all of these tools. Further, for each profile different *levels* are specified. A level sets certain complexity bounds—for example, for the required memory, number of objects, or the bit rate that can be handled by respective decoders. In order to define a conformance point for an MPEG decoder, in addition to the profile the level must be specified. This means that

Google Exhibit No. 1021
Page 082

Google Exhibit No. 1063
Page 0846

for compliant decoders a profile and a level must be defined. More details about MPEG-4 conformance and profiling can be found in Chapter 13.

In Figure 8.1, the complete set of decoders described in the MPEG-4 Visual standard [MPEG4-2] is shown. The decoders printed in gray (face and body animation decoding and [2D and 3D] mesh decoding) are not described in this chapter, but in Chapter 9. Also the scene composition is not explained here, but in Chapter 4. All of the other elements shown in Figure 8.1 (except for the DeMUX, which is a simple demultiplexer) are explained in detail in the following sections.

Analyzing Figure 8.1, several MPEG-4 natural video coding features can be highlighted. First, it is possible to code both still images (referred to as visual textures) and video sequences with the standard's tool-set. Moreover, taking only the two blocks Motion Decoding and Compensation and Texture Decoding results in the structure of the well-known (frame-based) motion-compensated hybrid coder—actually, it is even a block-based one. Adding Shape Decoding to the tool-set converts the entire MPEG-4 Visual standard into an object-based video coding solution. This means that the coded content is no longer restricted to be rectangular, but can be of arbitrary shape, which is coded by means of the MPEG-4 shape coding tools. In this mode, the shape is used also for decoding of motion and texture and for motion compensation.

It may seem that the usage of shape, and thus the possibility for content-based functionalities, is the only innovation of MPEG-4. However, this is not the case; the *rectangular* or *frame-based* parts of the video coding algorithms also have been greatly improved—for example, with increased compression

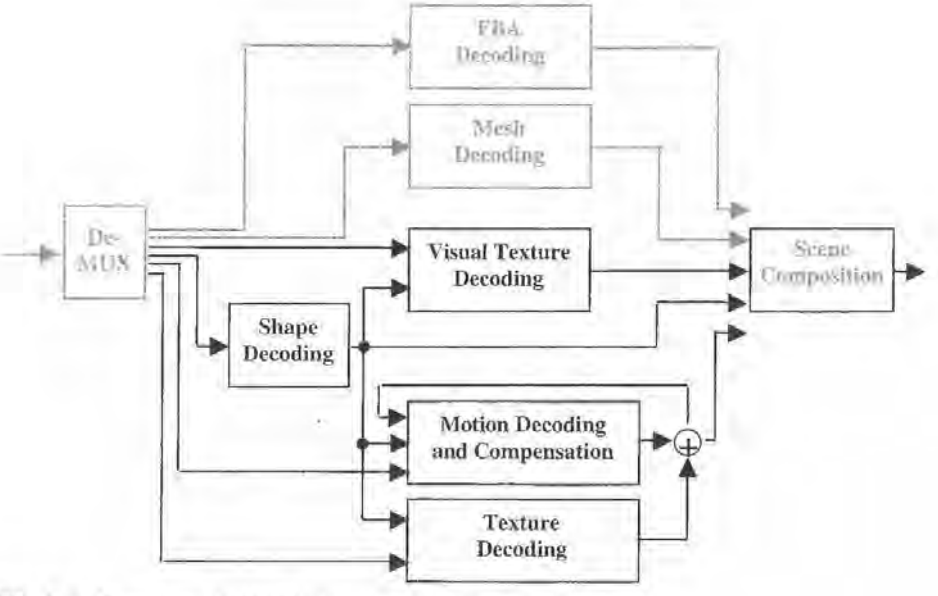


Fig. 8.1 Overview of MPEG-4 natural video decoding tools.

Google Exhibit No. 1021
Page 083

Google Exhibit No. 1063
Page 0847

efficiency and error resilience. This will be described in detail in the remaining sections. Nevertheless, the object-based approach is a major issue in MPEG-4, and therefore the basic ideas and principles are described in the following section.

First, it has to be noted that even in the object-based mode, the coding of the images of a video sequence in MPEG-4 is done blockwise. This means that each image is subdivided into square blocks, whose content is then coded. In order to understand this principle, some terminology for MPEG-4 video coding is introduced and explained, using the simple exemplary scene shown in Figure 8.2.

In MPEG-4, each scene to be coded can be composed of one or several *visual objects* (VOs). In case of video sequences, the VO is of type `video` and is defined as a sequence of *video object planes* (VOPs). In the following, a VO of type `video` is also referred to as *video object*. Each VOP is an instance of the video object at a certain time instance, t. This means that a video object is composed of a sequence of VOPs, much as an image sequence is composed of a sequence of images. Using that analogy, the VOP corresponds to a single image, and the video object corresponds to an image sequence. The main difference is that a VOP can be either rectangular or of arbitrary shape. In the latter case, it is defined not only by its Y, U, and V components but also by its shape.[1] The representation and coding of the shape will be described in detail later.

The complete hierarchy of an MPEG-4 video bitstream is shown in Figure 8.3. The top level is a visual sequence (VS) with several VOs. Each VO of type `video` comprises one or (in case of scalable video coding) more *video object layers* (VOLs). Each layer corresponds to a certain spatial resolution or image quality, as will be explained in Section 8.4. Further, a set of successive VOPs

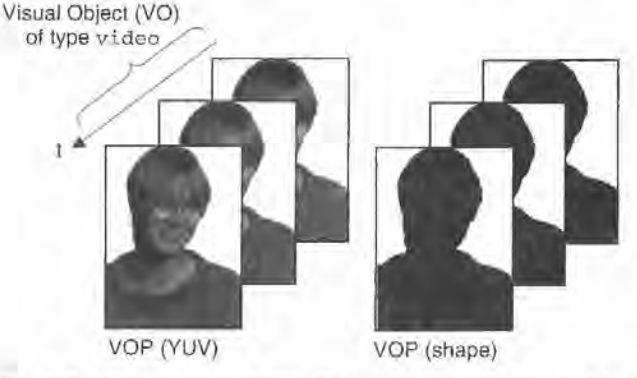


Fig. 8.2 Visual object (VO) of type `video`, represented by a sequence of video object planes (VOPs), which are composed of Y, U, and V texture matrixes and a shape matrix in case of an arbitrarily shaped VO, as shown. Only the nontransparent pels, marked black in the shape matrix, belong to the VO.

1. In other contexts, the VO shape matrix is also referred as the *alpha plane*.

Google Exhibit No. 1021
Page 084

Google Exhibit No. 1063
Page 0848

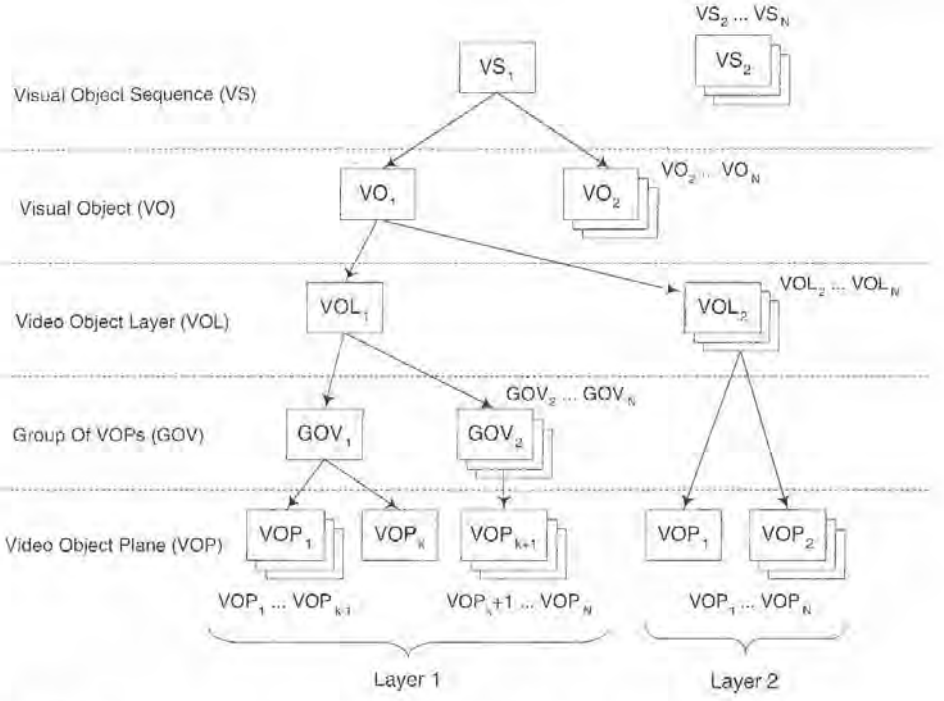


Fig. 8.3 Hierarchical structure of the MPEG-4 video bitstream.

can be clustered in order to form a *group of VOPs* (GOV). The GOV carries header information, which is useful for random access and resynchronization (e.g., in case of channel errors). The lowest level of the hierarchy is the VOP, as shown in Figure 8.3.

Just as previous coding standards do not specify how a video sequence is captured and even encoded, MPEG-4 does not standardize how a video object (and in particular the arbitrary shape information) is generated. The applied segmentation algorithm may be fully automatic (e.g., as described in [SalP94, NeCR97, MecW98]) or existing blue screen technology may be used. Also, user-assisted systems (as described in [AOWM98, MCMM99, KJKL99]) can be applied to generate arbitrarily shaped video content of high visual quality.

For coding purposes, each VOP is processed blockwise. Therefore, the smallest surrounding rectangle of the VOP is built, the *VOP bounding box*. This bounding box is extracted from the scene for each time instance and used to represent the corresponding VOP by means of its texture and shape information (see Figure 8.4).

For coding, the bounding box is divided into square blocks of size 16×16 luminance pels (and corresponding chrominances), the *macroblocks* (MBs). Therefore, the VOP bounding box is extended to match horizontal and vertical

Google Exhibit No. 1021
Page 085

Google Exhibit No. 1063
Page 0849

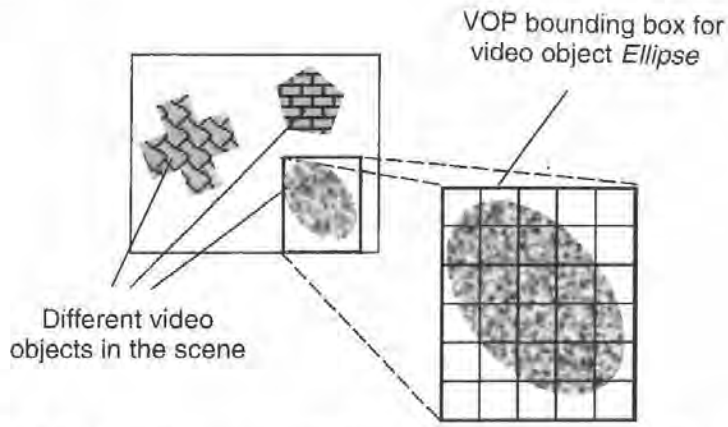


Fig. 8.4 Exemplary scene with three different video objects (*Cross*, *Pentagon*, and *Ellipse*); enlarged is the VOP bounding box for the video object `Ellipse`, divided into macroblocks of size 16×16 pels.

size of multiples of 16 pels (as the smallest surrounding rectangle to the object does not necessarily have multiples of 16 pels dimensions). Three different types of MBs are distinguished in the VOP bounding box, as shown in Figure 8.5.

1. **Transparent:** MBs that are completely outside the VOP. For these MBs, there is no YUV data to be coded; due to the shape information, the receiver knows that these MBs are *transparent* (i.e., not visible in the decoded scene).
2. **Opaque:** MBs that are completely inside the VOP. These MBs are processed as in the well-known block-based hybrid coders—that is, they are either *intra* coded using their YUV values or *inter* coded applying motion-compensated prediction and coding of the prediction error. The processing of these blocks is described in Section 8.2.

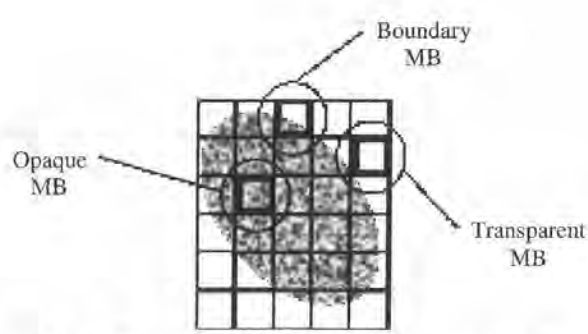


Fig. 8.5 MB types within the VOP boundary box.

3. **Boundary:** MBs that are at the boundary of the VOP. For processing these boundary MBs, specific tools for coding of arbitrarily shaped objects have been developed. These tools are presented in Section 8.3.

In addition to the two sections mentioned previously, which describe the MPEG-4 coding of rectangular video objects and of arbitrarily shaped video objects, there are sections describing specific MPEG-4 video coding tools with more detail. In Section 8.4, scalable video coding is explained. In particular, those parts of the scalable coding that are new compared to the state-of-the-art scalable video coding known, for example, from MPEG-2 Video [MPEG2-2], are emphasized. In Section 8.5, specific tools for interlaced video coding, error-resilient coding, dynamic resolution coding, sprite coding, and high-quality texture coding are described. Finally, Section 8.6 describes the MPEG-4 visual texture coding tool, which is an efficient wavelet-based tool for coding of rectangular as well as arbitrarily shaped still images.

8.2 Coding of Rectangular Video Objects

In MPEG-4 video coding, two main coding modes are distinguished: the coding of rectangular video objects and the coding of arbitrarily shaped video objects. In this section, the coding of rectangular video objects is described.

In principle, a rectangular video object is identical to a *frame* that is used in well-known video coding standards such as MPEG-1 and MPEG-2 [MPEG1-2, MPEG2-2, HaPN96], H.261 and H.263 [H261, H263, Rijk96]. Also, the coding principle of rectangular MPEG-4 video coding is the same as in these standards—namely, motion-compensated block-based hybrid coding. However, some significant improvements have been introduced by applying new tools or improving known algorithms. Therefore, this section describes the root coding algorithm only briefly, and concentrates on the new tools and algorithms introduced by MPEG-4 Visual [MPEG4-2].

8.2.1 Overview

For coding of rectangular video objects, only motion and texture information is transmitted to the decoder. Shape information is not required, and therefore in this case, no shape coding and decoding are necessary. In this respect, the MPEG-4 video coding solution is a block-based motion-compensated hybrid coder, as shown in the block diagram of Figure 8.6.

Figure 8.6 shows a block diagram of the decoding process for MPEG-4 video. Because no shape information is required for rectangular video objects, the corresponding modules in the block diagram are depicted in gray. The remaining blocks are those for motion decoding and compensation and for texture decoding. Compared with known block-based hybrid coding solutions, all

Google Exhibit No. 1021
Page 087

Google Exhibit No. 1063
Page 0851

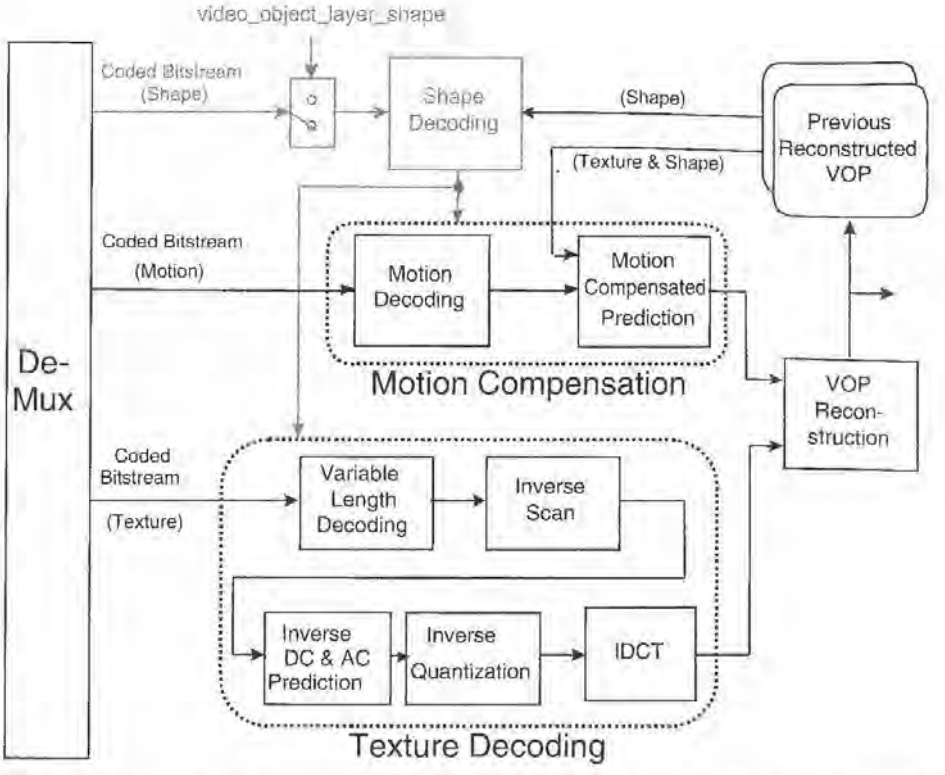


Fig. 8.6 MPEG-4 video decoding block diagram (shape decoding is in gray, as it is not required for rectangular video objects).

of these blocks have been significantly improved by introducing new or enhanced tools and algorithms.

Because the block-based hybrid coding principle is well known and copiously described in the literature (see, e.g., [CFHM71, Giro87, MuPG85, NetH88]), only a short description of the overall coding technique is given here. Each image of a video sequence is considered as one (rectangular) VOP. The coding of each VOP is performed blockwise—that is, each VOP is divided into macroblocks of 16×16 luminance pels size (with the corresponding chrominances, which depend on the colour subsampling format). Two major coding modes are distinguished:

1. ***Intra* mode:** The luminance (Y) and chrominance (U,V) values for each MB are coded independently of previous or future VOPs.
2. ***Inter* mode:** The difference of the MB luminance and chrominance values with respect to prediction values obtained from previous or future VOPs is coded.

Google Exhibit No. 1021
Page 088

Google Exhibit No. 1063
Page 0852

The advantage of the inter mode is that, in general, the images in a video sequence do not change their content completely between two images (or VOPs, speaking in MPEG-4 terminology). Therefore, not the complete image but only the differences to the previous or future images (or combinations of them), referred to as the *reference image*, need to be transmitted to the decoder. Because the decoder knows the reference image(s), it can reconstruct the current image by adding the received difference signal. This principle is also referred to as *predictive coding*—that is, the encoder first calculates prediction values and then transmits only the *prediction error*.

In motion-compensated block-based hybrid coding, this prediction takes into account the motion within the captured scene and is therefore called *motion-compensated prediction*. The motion information is calculated and coded as side information, so that the decoder can carry out the motion-compensated prediction. The representation of the motion information depends on the specific coding algorithm. In MPEG-4, two motion modes are defined at the MB level:

1. ***1MV mode:*** One motion vector (MV) is transmitted for the complete macroblock (i.e., for each 16×16 luminance pels block).
2. ***4MV mode:*** The macroblock is subdivided into four blocks of 8×8 luminance pels, and one MV is transmitted for each of these blocks (i.e., in total, four MVs for the complete MB).

A VOP can be coded using one of two prediction modes: For *predicted VOPs* (P-VOPs), the reference image is the previously decoded intra coded VOP (I-VOP) or P-VOP, whereas for *bidirectionally predicted VOPs* (B-VOPs) both the previous and the subsequent I- or P-VOP can be used for prediction.

The computation of the motion information is referred to as *motion estimation*. Because it is an encoder issue, the standard does not specify how to carry out this step, as however it is done, it does not impact interoperability (providing that the syntax and semantics are respected); however, exemplary algorithms can be found in an informative annex of MPEG-4 Visual [MPEG4-2] and in the informative part of the MPEG-4 Reference Software [MPEG4-5]. Some specifically optimized algorithms can also be found in the MPEG-4 Optimised Reference Software [N4554].

For the coding of the texture (i.e., the luminance and chrominance information), the same algorithms are used for the intra and inter modes. The difference is that in the intra mode actual texture values are coded, whereas in the inter mode the coded values represent the prediction error (i.e., difference values). The texture coding is based on a 2D 8×8 *discrete cosine transform* (DCT; see, e.g., [NetH88, MPEG4-2]), with subsequent quantization and entropy coding of the transform coefficients. In addition, predictive coding is used for some of the transform coefficients, referred to as AC/DC prediction.

Google Exhibit No. 1021
Page 089

Google Exhibit No. 1063
Page 0853

In the following sections, the motion-compensation tools and algorithms introduced by the MPEG-4 video standard are described. Then, the improvements regarding the texture coding are explained in detail.

8.2.2 New Motion-Compensation Tools

In this section, new and improved tools and algorithms for motion compensation in block-based hybrid coding, introduced by the MPEG-4 Visual standard [MPEG4-2], are described. Therefore, first the temporal prediction structure is described, as it is essential for the understanding of the following sections. Then, the following tools and algorithms are explained:

- **Quarter-pel motion compensation:** Algorithm for motion compensation using MVs with an increased resolution of one-quarter pel instead of only half- or full-pel resolution used in previous standards such as H.261, H.263, MPEG-1 Video, and MPEG-2 Video. This allows a significantly improved prediction and thus decreases the prediction error.
- **Global motion compensation:** A single set of motion parameters is coded for the complete VOP, representing the VOP global motion. These parameters can be used alternatively to the (local) MVs of an MB for motion compensation. This tool is particularly important for sequences with a large portion of global motion (e.g., caused by a moving camera) and for nontranslational motion like zoom or rotation.
- **Direct mode in bidirectional prediction:** This is an improvement of the bidirectional motion-compensated prediction, which uses the MVs of neighboring P-VOPs in order to decrease the bit rate required for coding the B-VOP MVs. It is a generalization of the "PB frames" introduced by H.263 [H263].

Those readers who are more familiar with the MPEG-4 Visual standard may miss one tool in this list—namely, the *overlapped block motion compensation* (OBMC). Because at the time of writing OBMC is not included in any of the MPEG-4 Visual object types and thus cannot be used by any MPEG-4–conformant application up to now, this tool is not described in this book. The interested reader is instead referred to the literature [MPEG4-2, OrcS94].

8.2.2.1 Temporal Prediction Structure In order to understand the algorithms and terminology of motion-compensated prediction, it is important to clarify the temporal prediction structure. For the MPEG-4 video case, this structure is depicted in Figure 8.7.

As shown in Figure 8.7, an I-VOP does not refer to any other VOP (has no reference VOP), as it is completely encoded using the intra coding mode. P-VOPs refer to the so-called *forward reference*, which is defined as the most recently decoded I- or P-VOP in the past for which information has been trans-

Google Exhibit No. 1021
Page 090

Google Exhibit No. 1063
Page 0854

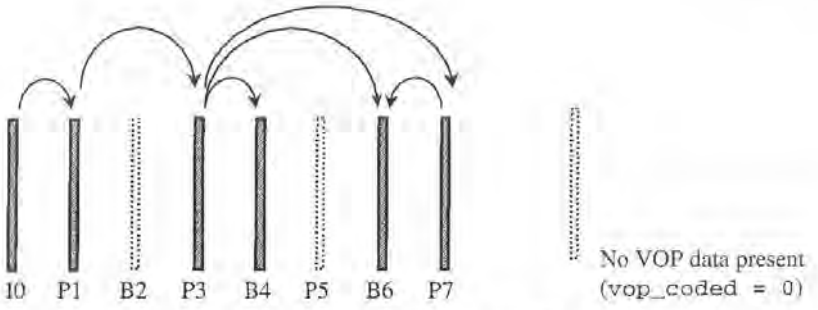


Fig. 8.7 Temporal prediction structure of MPEG-4 motion-compensated hybrid coding. Intra VOPs are marked as Ix, predictive VOPs as Px, and bidirectionally predicted VOPs as Bx. The arrows describe the direction of the prediction. The number x denotes the display order of the VOPs, not their order of transmission [MPEG4-2].

mitted (signaled by the condition[2] vop_coded=1). Finally, B-VOPs may refer to the forward reference (as just defined) as well as to the *backward reference*, which is defined as the most recently decoded I- or P-VOP in the future. In the case that no information has been transmitted that can be used as backward reference (i.e., vop_coded=0 for the backward reference), only the forward reference is used for the prediction of the B-VOP.

In Figure 8.7, the *display order* of the VOPs is shown. It should be noted that this order is not identical to the *transmission order*. As the B-VOPs may make reference to an I- or P-VOP that is to be displayed *in the future* with respect to their own display time, these backward-reference VOPs must be transmitted before the corresponding B-VOPs. For the example shown in Figure 8.7, the transmission order would be I0 P1 P3 B2 P5 B4 P7 B6. Because in that example no information is transmitted for VOPs B2 and P5 (i.e., for these VOPs vop_coded=0), VOPs B6 and P7 make reference to VOP P3. The omission of VOP B2 has no influence on the referencing structure, as B-VOPs are not used as reference for motion-compensated prediction.

8.2.2.2 Quarter-Pel Motion Compensation The MVs that are used for the motion-compensated prediction must be transmitted to the decoder. Therefore, the resolution of the MVs must be limited to a finite number. In the early motion-compensation algorithms (e.g., H.261 [H261]), only full-pel MVs were used—that is, the components of an MV were restricted to an integer number. Because the real motion between two successive images of a sequence is, in general, not exactly described by steps of one pel, this restricted resolution leads to errors in the motion-compensated prediction and thus to an increased prediction error.

2. The parameter vop_coded is a syntax element that is transmitted at VOP level; if it is set to zero, this signals that no more information is transmitted for this VOP—that is, the VOP is "empty" (no VOP data present).

Google Exhibit No. 1021
Page 091

Google Exhibit No. 1063
Page 0855


Edited by Fernando Pereira and Touradj Ebrahimi
THE MPEG-4
TK
6680.5
.P47
2002

ultimedia

THE MPEG-4 BOOK

Edited by Fernando Pereira and Touradj Ebrahimi

The most complete, focused guide to MPEG-4—the breakthrough standard for interactive multimedia

MPEG-4 represents a breakthrough in multimedia, delivering not just outstanding compression but also a fully interactive user experience. In ***The MPEG-4 Book***, two leaders of the MPEG-4 standards community offer a comprehensive, targeted guide to the MPEG-4 standard and its use in cutting-edge applications. Fernando Pereira and Touradj Ebrahimi, together with a unique collection of key MPEG experts, demonstrate how MPEG-4 addresses tomorrow's multimedia applications more successfully than any previous standard. They review every element of the standard to offer you a book that covers:

- Synthetic and natural audio and video object coding, description, and synchronization
- BIFS—the MPEG-4 language for scene description and interaction
- The extensible MPEG-4 textual format XMT
- Transport and delivery of MPEG-4 content
- MPEG-J: using Java™ classes within MPEG-4 content
- A complete overview of MPEG-4 profiles and levels
- Verification tests

The authors also walk through the MPEG-4 Systems Reference Software—offering powerful real-world insights for every product developer, software professional, engineer, and researcher involved with MPEG-4 and state-of-the-art multimedia delivery.

The MPEG-4 Book is part of the new IMSC Press Multimedia Series from the Integrated Media Systems Center at the University of Southern California, a federally funded center specializing in cutting-edge multimedia research.

About the Editors

FERNANDO PEREIRA, Associate Professor at Instituto Superior Técnico, Lisboa, Portugal, and researcher at Instituto de Telecomunicações, is area editor of *Signal Processing: Image Communication Journal* and Associate Editor of *IEEE Transactions of Circuits and Systems for Video Technology*. Pereira has chaired many ad hoc groups in the context of the MPEG-4 and MPEG-7 standards' development and received an ISO certificate for his contributions to the MPEG-4 standard. His current interests include video analysis, processing, coding, description, and interactive multimedia services for Internet and mobile environments.

TOURADJ EBRAHIMI, Professor of Image Processing at EPFL's Signal Processing Lab in Switzerland, has been nominated for two ISO certificates for his contributions to MPEG-4 and JPEG 2000 standards. He leads the Swiss delegation to MPEG and JPEG and is associate editor of *IEEE Transactions on Image Processing*, area editor of *Image Communication Journal of EURASIP*, and associate editor of *Optical Engineering*. His interests include image processing, coding, and visual information security. He holds a dozen patents.

IMSC Press Multimedia Series/Andrew Tescher, Series Editor

imsc

ISBN 0-13-061621-4

90000

PRENTICE HALL

Google Exhibit No. 1021
Page 093

Google Exhibit No. 1063
Page 0857

Ex. 1021A

LIBRARY | CATALOG

1 of 1

BOOK

The MPEG-4 book

Full Record MARC Tags

Personal name

Pereira, F. (Fernando Manuel Bernardo)

Main title

The MPEG-4 book / Fernando Pereira, Touradj Ebrahimi.

Published/Created

Upper Saddle River, NJ : Prentice Hall PTR, c2002.

Request this Item | LC Find It

More Information

LCCN Permalink	https://lccn.loc.gov/2002070433
Description	xxxvi, 849 p. : ill. ; 25 cm.
ISBN	0130616214
LC classification	TK6680.5 .P47 2002
Related names	Ebrahimi, Touradj.
LC Subjects	MPEG (Video coding standard)

Google Exhibit No. 1063
Page 0859

Video compression--Standards.

Browse by shelf order	TK6680.5
Notes	Includes bibliographical references and index.
Series	IMSC Press multimedia series Prentice Hall IMSC Press multimedia series.
LCCN	2002070433
Dewey class no.	621.388
Other system no.	(OCoLC)ocm49805496
Type of material	Book

Item Availability

CALL NUMBER	TK6680.5 .P47 2002 CABIN BRANCH Copy 1
Request in	Jefferson or Adams Building Reading Rooms - STORED OFFSITE
Status	Not Charged

CALL NUMBER	TK6680.5 .P47 2002 CABIN BRANCH Copy 2
Request in	Jefferson or Adams Building Reading Rooms - STORED OFFSITE
Status	Not Charged

Google Exhibit No. 1063
Page 0860

LIBRARY CATALOG

1 of 1

BOOK

The MPEG-4 book

Full Record MARC Tags

000		01269cam a22003134a 4500
001		12769684
005		20160907112553.0
008		020509s2002 njua b 001 0 eng
906	__	|a 7 |b cbc |c orignew |d 1 |e ocip |f 20 |g y-gencatlg
925	1_	|a acquire |b 2 shelf copies |x policy default
925	0_	|a acquire |b 2 shelf copies |x KW, 2016-09-07
955	__	|a oc16 2002-05-09 to ASCD |c jg14 2002-05-13 |d jg12 2002-05-14 |e jg07 2002-05-14 to Dewey |a aa07 2002-05-15 |a ps06 2002-11-07 bk rec'd, to CIP ver. |f ov06 2002-11-12 CIP ver to BCCD; |a jg07 2002-12-20 to BCCD, copy 2
010	__	|a 2002070433
020	__	|a 0130616214
035	__	|a (OCoLC)ocm49805496
040	__	|a DLC |c DLC |d C#P |d DLC
042	__	|a pcc
050	00	|a TK6680.5 |b .P47 2002
082	00	|a 621.388 |2 21
100	1_	|a Pereira, F. |q (Fernando Manuel Bernardo)
245	14	|a The MPEG-4 book / |c Fernando Pereira, Touradj Ebrahimi.
260	__	|a Upper Saddle River, NJ : |b Prentice Hall PTR, |c c2002.
300	__	|a xxxvi, 849 p. : |b ill. ; |c 25 cm.
490	1_	|a MSC Press multimedia series
504	__	|a Includes bibliographical references and index.

Google Exhibit No. 1063
Page 0861

650	_0	|a MPEG (Video coding standard)
650	_0	|a Video compression |x Standards.
700	1_	|a Ebrahimi, Touradj.
830	_0	|a Prentice Hall IMSC Press multimedia series.

Request this Item | LC Find It

Item Availability

CALL NUMBER	TK6680.5 .P47 2002 CABIN BRANCH Copy 1
Request in	Jefferson or Adams Building Reading Rooms - STORED OFFSITE
Status	Not Charged

CALL NUMBER	TK6680.5 .P47 2002 CABIN BRANCH Copy 2
Request in	Jefferson or Adams Building Reading Rooms - STORED OFFSITE
Status	Not Charged

Google Exhibit No. 1063
Page 0862

Ex. 1021B

Copyright
United States Copyright Office

Try the Copyright Public Records System (CPRS) pilot with enhanced search features and filters.

Help | Search | History | Titles | Start Over

Public Catalog

Copyright Catalog (1978 to present) at DC4
Search Request: Left Anchored Title = MPEG-4 Book
Search Results: Displaying 1 of 1 entries

previous next

Labeled View

The MPEG-4 book / Fernando Pereira, Touradj Ebrahimi, IMSC Press.

Type of Work: Text
Registration Number / Date: TX0005617873 / 2002-09-18
Title: The MPEG-4 book / Fernando Pereira, Touradj Ebrahimi, IMSC Press.
Imprint: Upper Saddle River, NJ : Prentice Hall PTR, c2002.
Description: 849 p.
Series: IMSC Press multimedia series
Copyright Claimant: Pearson Education, Inc.
Date of Creation: 2001
Date of Publication: 2002-07-10
Authorship on Application: Integrated Media Systems Center, USC, employer for hire.
Other Title: IMSC Press
Names: Pereira, Fernando
Ebrahimi, Touradj
Pearson Education, Inc.
Integrated Media Systems Center
University of Southern California (Los Angeles) School of Engineering. Integrated Media Systems Center

previous next

Save, Print and Email (Help Page)
Select Download Format Full Record Format for Print/Save
Enter your email address: Email

Help Search History Titles Start Over

Contact Us | Request Copies | Get a Search Estimate | Frequently Asked Questions (FAQs) about Copyright | Copyright Office Home Page | Library of Congress Home Page

Google Exhibit No. 1063
Page 0864

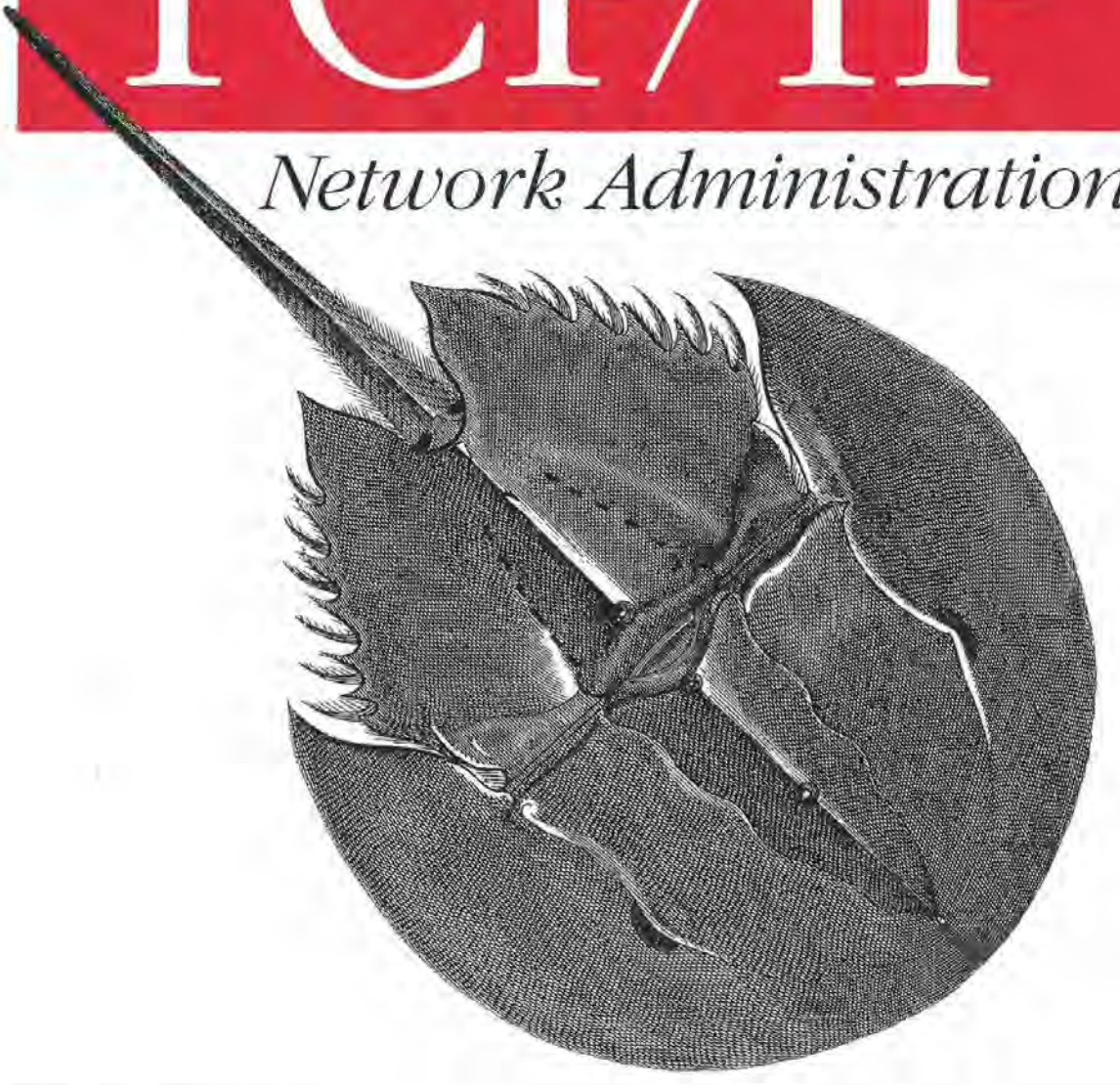
Help for Windows NT System Administra

Prep for MCSE

Windows NT

TCP/IP

Network Administration



O'REILLY®

Craig Hunt & Robert Bruce Thompson

Google Exhibit No. 1022
Page 001

Windows NT TCP/IP Network Administration

Craig Hunt and Robert Bruce Thompson

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

Google Exhibit No. 1022
Page 002

Google Exhibit No. 1063
Page 0866

Windows NT TCP/IP Network Administration
by Craig Hunt and Robert Bruce Thompson

Copyright © 1998 O'Reilly & Associates, Inc. All rights reserved.
Printed in the United States of America.

Published by O'Reilly and Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

Editor: Robert Denn

Production Editor: Paula Carroll

Production Service: *TIPS* Technical Publishing

Printing History:

October 1998: First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly & Associates, Inc. The association between the horseshoe crab and the topic of TCP/IP is a trademark of O'Reilly and Associates.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.



This book is printed on acid-free paper with 85% recycled content, 15% post-consumer waste. O'Reilly & Associates is committed to using paper with the highest recycled content available consistent with high quality.

ISBN: 1-56592-377-4 [9/99]

Google Exhibit No. 1022
Page 003

Google Exhibit No. 1063
Page 0867

Table of Contents

Preface ..... *xi*

1. *Overview of TCP/IP* *1*
   - TCP/IP and the Internet *2*
   - A Data Communications Model *4*
   - TCP/IP Protocol Architecture *7*
   - Network Access Layer *9*
   - Internet Layer *10*
   - Transport Layer *16*
   - Application Layer *20*
   - Summary *21*

2. *Delivering the Data* *23*
   - Addressing, Routing, and Multiplexing *23*
   - The IP Address *25*
   - Subnets *31*
   - The Routing Table *34*
   - Address Resolution *37*
   - Protocols, Ports, and Sockets *38*
   - Summary *44*

3. *Network Services* *46*
   - Names and Addresses *47*
   - The Host Table *49*
   - LMHOSTS *51*
   - Domain Name System *52*

Google Exhibit No. 1022
Page 004

Google Exhibit No. 1063
Page 0868

Windows Internet Name Service 59
Mail Services 60
Dynamic Host Configuration Protocol 69
Summary 71

4. Getting Started 73
To Connect or Not to Connect 74
Basic Information 76
Planning Routing 83
Planning Naming Service 86
Other Services 90
Informing the Users 91
Summary 92

5. Installing TCP/IP 94
Windows NT Networking 94
Configuring the TCP/IP Protocol 107
Summary 119

6. Using Dynamic Host Configuration Protocol 121
Why DHCP Is Needed 123
How DHCP Works 125
Planning for DHCP 133
Installing the DHCP Server Service 139
Installing and Configuring the DHCP Relay Agent 141
Managing DHCP with DHCP Manager 142
Maintaining and Troubleshooting DHCP Server 154
Summary 156

7. Using Windows Internet Name Service 158
Why WINS Is Needed 159
How WINS Works 163
Installing the WINS Server Service 175
Managing WINS with WINS Manager 176
Troubleshooting WINS 192
Summary 195

8. Configuring DNS Name System 197
The Microsoft DNS Server 199
Planning Your DNS Service 200

Google Exhibit No. 1022
Page 005

Google Exhibit No. 1063
Page 0869

Installing the DNS Server 202
The DNS Manager 202
Creating a New Zone 203
Building the DNS Database 207
The DNS Files 209
Using nslookup 218
Summary 222

9. Microsoft Routing and Remote Access Service 223
Routing and Remote Access Service Features 224
Installing Microsoft Routing and RAS 230
Administering Microsoft Routing and RAS 241
Using Dial-Up Networking (DUN) 266
Summary 276

10. Internet Information Server (IIS) 278
IIS Components 279
Installing IIS 281
Configuring IIS 292
Managing IIS 340
Summary 347

11. Troubleshooting TCP/IP 348
Approaching a Problem 348
Diagnostic Tools 351
Testing Basic Connectivity 353
Troubleshooting Network Access 356
Checking Routing 363
Checking Name Service 367
Analyzing Protocol Problems 371
Protocol Case Study 375
Simple Network Management Protocol 378
Summary 380

12. Network Security 382
Security Planning 383
Firewalls 388
Encryption 392
User Authentication 394
Software Security 401

Google Exhibit No. 1022
Page 006

Google Exhibit No. 1063
Page 0870

Security Monitoring 403
File Security 407
Words to the Wise 410
Summary 411

13. Information Resources 412
The World Wide Web 412
Anonymous FTP 416
Retrieving RFCs 416
Mailing Lists 418
Newsgroups 420
The White Pages 424
Summary 427

A. PPP Scripting Languages 429

B. DNS Resource Records 443

C. Microsoft DHCP Option Support 458

D. Routing Protocols 465

Index 475

Google Exhibit No. 1022
Page 007

Google Exhibit No. 1063
Page 0871

10

In this chapter:
- *IIS Components*
- *Installing IIS*
- *Configuring IIS*
- *Managing IIS*
- *Summary*

Internet Information Server (IIS)

Microsoft jumps into anything they do with both feet. When Microsoft belatedly noticed the Internet phenomenon, they were caught with their collective pants down. Without any products to address the needs of this exploding new market segment, they believed themselves to be in danger of losing market share to the new guys on the block—read Netscape. As anyone familiar with Microsoft might guess, they pulled out all the stops to get competitive products to market as soon as possible. On the client side, they bought rights to a third-party web browser, renamed it Internet Explorer, and turned it into a world-class browser in record time. On the server side, they created a multi-purpose product called Internet Information Server, or IIS for short.

IIS has grown in Internet time. Microsoft cobbled together a quick and dirty IIS 1.0 and released it as an update to Windows NT Server 3.51. It wasn't much more than a placeholder, but at least it got the ball rolling. When Windows NT Server 4.0 was released, it bundled IIS 2.0, a significant upgrade that filled most of the gaping holes and killed most of the bugs in IIS 1.0. Still, the major third-party web server vendors like Netscape and O'Reilly weren't too worried—IIS2 couldn't compete with their feature sets.

When Microsoft shipped IIS3, the third party vendors started to sweat. Microsoft was giving away a product that compared favorably in features and performance with their expensive commercial products. Even worse, Microsoft started to ship free supporting components, like Index Server, that made IIS a complete server-side Internet solution.

At this point, the way things are headed is pretty clear. The release of IIS4 as a part of the Windows NT 4.0 Option Pack, which shipped in Q1 1998, may well be the final nail in the coffin of the commercial web server vendors. The majority of

Google Exhibit No. 1022
Page 008

Google Exhibit No. 1063
Page 0872

Internet web servers are now running either IIS or the free Apache server. When you consider private Intranet servers, the proportion running IIS is even higher.

The Web, FTP, and email are where the action is on the Internet today. Depending on whose figures you believe, something close to 99 percent of the current traffic on the Internet is web-, ftp-, or email-related. As IP telephony becomes more popular, that may change, but for now at least it is true. IIS4 provides an excellent no-cost server-side solution for all three of these core Internet services.

NOTE To understate the case, IIS4 is a major application, with supporting applications that are major applications in themselves. Covering IIS thoroughly would require an entire book—or several books. Accordingly, this chapter focuses on getting started with IIS—the essentials of installing and configuring the web, FTP, and SMTP components of IIS at the overview level. For a book-length treatment of this topic, see any of the numerous books devoted to it.

IIS Components

The IIS4 distribution, which is contained within the Windows NT 4.0 Option pack, contains the following major functional IIS components:

World Wide Web (WWW) Service
: A standards-compliant HTTP (web) server.

File Transfer Protocol (FTP) Service
: A standards-compliant FTP server.

Simple Mail Transfer Protocol (SMTP) Service
: A standards-compliant SMTP relay server.

Network News Transport Protocol (NNTP) Service
: A standards-compliant NNTP (news) server. The NNTP Service is not installed by default, and we do not cover it in this chapter.

The IIS distribution also includes the following management tools:

Internet Service Manager (ISM)
: The main management tool for IIS and the other applications included in the Windows NT 4.0 Option Pack. You can use ISM to manage IIS on the local IIS server, from another computer on the network, or across the Internet. ISM is a snap-in for the Microsoft Management Console (MMC).

Google Exhibit No. 1022
Page 009

Google Exhibit No. 1063
Page 0873

NOTE MMC is a Windows NT 5.0 component that provides an application framework for small dedicated utilities—called snap-ins—that perform management functions for a particular application. For example, you use MMC to manage IIS by loading the ISM snap-in into MMC. Windows NT 5.0 will include snap-ins that replace the patchwork quilt of individual management tools, for example, User Manager for Domains and Server Manager, used by Windows NT 4.0. Like Routing and RAS, MMC is a Windows NT 5.0 technology that Microsoft has released as an upgrade to Windows NT 4.0.

Internet Service Manager (HTML)
A browser-based management tool that provides most ISM functions. You can use ISM (HTML) to manage IIS on the local IIS server, from another computer on the network, or across the Internet.

In addition, the Windows NT 4.0 Option Pack distribution includes the following components which are not a part of IIS, but provide supporting functionality:

Certificate Server
Manages the digital certificates used by IIS to provide server and client authentication with the SSL (Secure Sockets Layer) and PCT (Private Communication Technology) protocols.

FrontPage 98 Server Extensions
Server-based extensions that allow FrontPage clients and IIS to operate as a client-server pair.

Internet Connection Services for RAS
Supplements and extends basic RAS with tools that provide a customizable dialer, centralized phone book administration, and RADIUS support.

Microsoft Data Access Components 1.5
Includes ActiveX Data Objects (ADO), Remote Data Service (RDS), the Microsoft OLE DB Provider for ODBC, and Open Database Connectivity (ODBC) components that can be used to provide client/server applications with access to data on remote servers.

Microsoft Index Server
Indexes documents on an IIS web site, and allows client browsers to search the web site by using an HTML query form.

Microsoft Management Console (MMC)
Provides the framework for the application-specific management tools for Option Pack components, including IIS4, Index Server, and Transaction Server. MMC is a general-purpose template manager that uses programs called snap-ins to manage specific components. For example, the Internet Service

Google Exhibit No. 1022
Page 010

Google Exhibit No. 1063
Page 0874

Manager (ISM) application is actually a snap-in that runs under the MMC framework. Just think of MMC as the socket wrench and the snap-ins as the sockets. In addition to the MMC-based management tools that can be used locally or across the network, IIS4 and Index Server both include HTML-based tools that allow administrators to use a limited subset of management functions from a web browser.

Microsoft Message Queue (MSMQ)
: Allows applications to communicate asynchronously with other applications by sending and receiving messages.

Microsoft Script Debugger
: Debugger for ASP pages that contain VBScript, JScript, or Java scripts. You can also use Microsoft Script Debugger to debug scripts written in other scripting languages, for example, Perl or REXX, that support host-independent debugging.

Microsoft Site Server Express
: A functionally limited version of the commercial Microsoft Site Server product that provides site and usage analysis, the Posting Acceptor, and the Web Publishing Wizard.

Microsoft Transaction Server (MTS)
: A component-based transaction processing system that you can use to develop, deploy, and manage server-based applications. MTS provides a runtime environment and GUI-based administration tool for managing applications.

Windows Scripting Host (WSH)
: Allows you to perform common administrative tasks from the command line. WSH is a language-independent scripting host for 32-bit Windows platforms. Presently, WSH uses only the VBScript and JScript scripting engines supplied by Microsoft. However, it is likely that other companies will soon provide Active-X scripting engines for other scripting languages like Perl and REXX.

Installing IIS

Installing IIS4 is a relatively straightforward (albeit time consuming) multi-step process. After you obtain the IIS4 distribution files and prepare your server, you install Internet Explorer 4.01 or higher and then run IIS4 Setup. Because installing IIS4 may require unloading services that may be needed by users and because it requires at least two system restarts, you may want to schedule the installation outside normal working hours unless you are installing IIS4 on a dedicated server. The remainder of this section details the steps necessary to install IIS4.

Google Exhibit No. 1022
Page 011

Google Exhibit No. 1063
Page 0875

Configuring IIS

The bulk of this chapter is contained in the following sections, which detail how to configure IIS once you've installed it. This material concentrates on the three major services—the WWW Service, the FTP Service, and the SMTP Service—that are installed when you run IIS Setup using the Typical option. Although the NNTP Service is not covered, configuring it is very similar to configuring the other services.

Using Internet Service Manager (ISM)

Internet Service Manager (ISM) is the main management tool for IIS. ISM runs as a snap-in under the Microsoft Management Console (MMC). To start ISM choose Start → Programs → Windows NT 4.0 Option Pack → Microsoft Internet Information Server → Internet Service Manager. MMC runs and automatically loads the ISM snap-in.

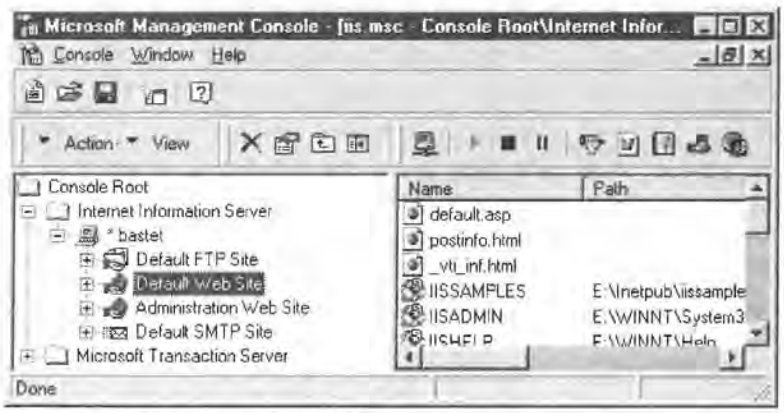


Figure 10-2. The Internet Service Manager

Which toolbar icons and menu choices are available in ISM depend on the object currently selected. For example, Figure 10-2 shows ISM with the Default Web Site object selected. Because we have a web site selected, the toolbar includes three VCR-like icons that you can use to start, stop, or pause the web site. The toolbar also contains icons for Key Manager, Performance Monitor, Event Viewer, Server Manager, and User Manager for Domains, all of which are commonly used when managing a web site. In addition to context-sensitive toolbars, ISM provides the standard context-sensitive menus that you can access by right-clicking an object. We'll use ISM in the following sections to configure settings for the various services installed.

Google Exhibit No. 1022
Page 012

Google Exhibit No. 1063
Page 0876

Configuring Computer Properties

Use the Internet Information Server page of the <computer-name> Properties dialog to configure the global IIS properties for the selected computer. To display this dialog, shown in Figure 10-3, log on as an administrator, start Internet Service Manager, right-click the computer name in MMC to display the context-sensitive menu, and choose Properties.

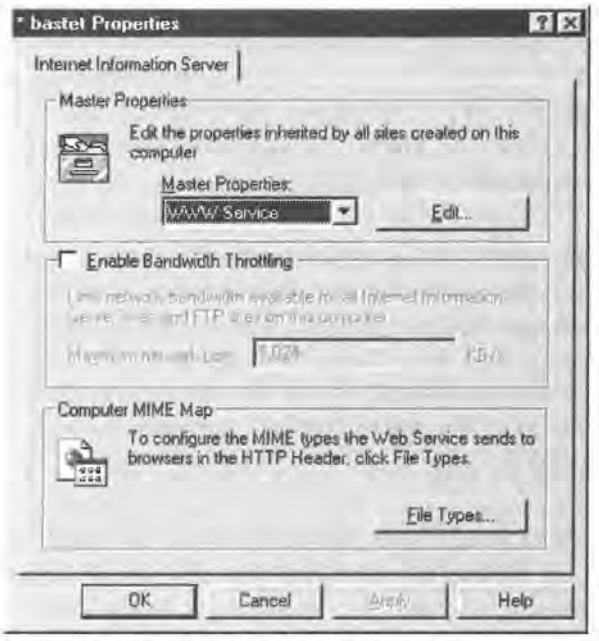


Figure 10-3. Configuring global IIS properties

You use this dialog to set default properties for all Web and FTP sites, control the bandwidth used by all web sites, and set MIME configuration for all sites, as follows:

Master Properties section

Use the Master Properties drop-down list to select a service type—either WWW Service or FTP Service—and click Edit to display the tabbed properties dialog for that service. Using these tabbed properties dialogs is described in the following sections. The settings you make here are the default values that will be used when you create a new Web or FTP site. If these settings are different from those of an existing site, you are given the opportunity to replace the current settings with the new master property settings or to continue to use the existing settings. You can also override these settings later for any particular site. List-formatted properties—for example, access lists, custom error

Google Exhibit No. 1022
Page 013

Google Exhibit No. 1063
Page 0877

messages, or filters—are treated as single properties. Changes are not merged with existing list data, but are overwritten by the new data.

Enable Bandwidth Throttling section

Use this section to limit the bandwidth available to all WWW sites on the selected computer. To do so, mark the Enable Bandwidth Throttling checkbox and enter a value in the Maximum network use box. You would typically use this option to limit the availability of bandwidth to the WWW Service on a computer that also provides other bandwidth-consuming services, for example, mail or RAS. On such a computer, Microsoft recommends initially allocating 50 percent of the available bandwidth, and adjusting this value up or down as needed. For example, on a T1 (1.544 Mbps) interface, you might initially set throttling to a value of 772 Kbps. Note that bandwidth throttling controls bandwidth use only by static HTML files.

Computer MIME Map section

Use this section to configure MIME (Multipurpose Internet Mail Extensions) mappings, which specify the file types that the WWW Service returns to browsers. To display the registered file types installed with Windows NT, click File Types to display the File Types dialog. Highlight a registered file type in the Registered file types pane to display details about that file type in the File type details section at the bottom of the dialog. You can also take the following actions in the File Types dialog:

Add a MIME mapping

Click New Type to display the File Type dialog, and enter values in the Associated Extension: and Content Type (MIME): boxes. For the latter box, use the syntax *<mime-type>/<file-extension>*, for example, *image/bmp*.

Remove a MIME mapping

Highlight a registered file type and click Remove.

Edit a MIME mapping

Highlight a registered file type, click Edit to display the File Type dialog, and make the necessary changes to the Associated Extension and/or Content Type (MIME): boxes.

Changing MIME mapping in the master property sheet for the computer changes those properties for all sites, and for the computer itself. You can instead modify MIME mappings for a particular site, which displays only the changed mappings. If you subsequently change the MIME mappings in the master property sheet, the custom mappings for individual sites are overwritten by the new master property sheet mappings.

Google Exhibit No. 1022
Page 014

Google Exhibit No. 1063
Page 0878

The pane lists the status of each ISAPI filter (loaded, unloaded or disabled), that filter's name, and the priority assigned to the filter in the DLL. The Details section displays the same information, and adds the name of the executable file that contains the filter. You can modify the ISAPI filter configuration for the web site as follows:

Add an ISAPI filter
: Click Add to display the Filter Properties dialog. Enter the Filter Name and Executable, or browse for the executable that contains the filter.

Remove an ISAPI filter
: Highlight the filter in the pane and click Remove.

Edit an ISAPI filter
: Highlight the filter in the pane and click Edit to display the Filter Properties dialog.

Enable or Disable an ISAPI filter
: Highlight the filter in the pane and click Enable to enable a filter that is currently disabled, or click Disable to disable a filter that is currently enabled.

Change the order in which ISAPI filters are loaded
: ISAPI filters are loaded in the order that they appear in this pane. To change this order, highlight a filter in the pane and use the up arrow or down arrow to rearrange the list.

ISAPI filters are processed according to the following rules:

- All web sites inherit the ISAPI filters configured in Master Properties for the computer, which are not displayed for the individual web sites.
- When you add filters to an individual web site, those filters are merged with the default filters for the site that are configured in Master Properties for the computer.
- If several filters are registered for the same event, they are called sequentially, as follows:
  - A filter with a higher priority is always processed before any filter with a lower priority.
  - If two filters have the same priority, the global filter set in Master Properties is processed before the local filter defined for the individual web site.
  - If two filters at the same level (global or local) have the same priority, they are processed in the order that they were loaded, which is determined by the order in which they appear in the pane.

Configuring Home Directory properties

Use the Home Directory page of the Web Site Properties dialog to change the location of or modify the properties for the home directory for the web site.

Google Exhibit No. 1022
Page 015

Google Exhibit No. 1063
Page 0879

Installing IIS creates the web root directory that you specify, ordinarily *\Inetpub\wwwroot*. You can use this dialog to change that location. This is a dynamic dialog, whose contents vary according to the option button you select in the top section, as follows:

A directory located on this computer
: Select this option, which is the default, to use a web root directory on the local computer where you installed IIS. Use this page to specify the location, access permissions, and content control for the local directory, as well as application settings. Uses standard drive/directory syntax, for example *C:\web\htmlfiles*.

A share located on another computer
: Select this option to use a web root directory located on a network share. The options and settings are similar to those available when the web root directory is on the local computer. Uses UNC syntax in the form *\\servername\sharename*, for example, *\\bastet\htmlfiles*.

A redirection to a URL
: When a browser requests the original URL associated with the web site, it is automatically redirected to the URL specified here. You can map requests to another web site by using the fully qualified URL for that web site, for example, *http://www.oreilly.com*. Alternatively, you can map requests to a virtual directory by using a virtual path, for example, */htmlfiles*. You can use redirection variables and wildcards in the target URL to control how the source URL is mapped to the target URL.

These options are described in the following sections.

A directory located on this computer

Use the Home Directory page of the Web Site Properties dialog shown in Figure 10-6 to change the location of or modify the properties for the home directory for the web site, if that home directory is located on the computer where IIS is installed.

Set the following options:

Local Path
: Type the drive and directory for the local path where the home directory resides on the local computer, or browse for the location.

Access Permissions
: Use these checkboxes to specify what type of access is allowed to the home directory. If the home directory is located on an NTFS volume, these settings should match the NTFS permissions for that directory. If the settings do not match, the most restrictive settings are used. For example, if you grant Write

Google Exhibit No. 1022
Page 016

Google Exhibit No. 1063
Page 0880

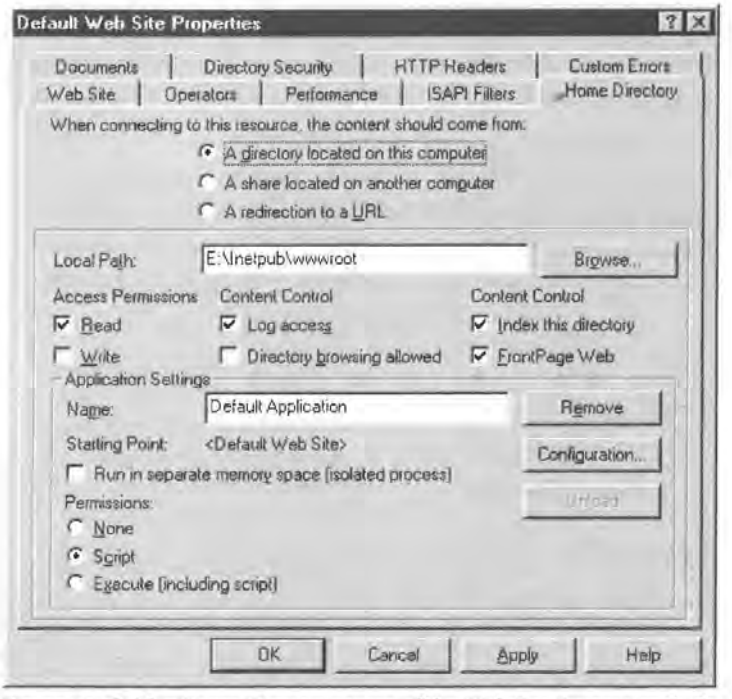


Figure 10-6. The Home Directory page of the Web Site Properties dialog

permission in this dialog, but a particular user has the No Access NTFS permission for the directory, that user will be unable to access the directory.

Read

Mark this checkbox, which is marked by default, to assign Read permission if you want clients to be able to read or download the files contained in the home directory. The WWW Service returns an error if a client attempts to retrieve a file located in a directory for which Read permission has not been granted. Grant Read permission only for directories that contain public information like HTML pages and images. Do not grant Read permission for directories that contain system files, scripts (CGI and ISAPI), and other material that you do not want clients to be able to retrieve.

Write

Mark this checkbox, which is cleared by default, to assign Write permission if you want clients to be able to upload files to the web site, or to change content in files that are write-enabled. Only clients who are using a web browser that supports that HTTP 1.1 put feature can write files.

Google Exhibit No. 1022
Page 017

Google Exhibit No. 1063
Page 0881

Content Control

Use these checkboxes to control the behavior of the WWW service for the home directory, as follows.

Log access

Mark this checkbox, which is marked by default, to cause the WWW service to write a log file entry for each visit to this directory. For this setting to be effective, logging must be enabled for the web site in the Web Site properties page, which it is by default.

Directory browsing allowed

Mark this checkbox, which is cleared by default, to automatically generate a hypertext directory listing. When this setting is enabled, if a client enters this directory without specifying a file name, and if no default document exists in the directory, the client receives the hypertext directory list, which enables him to navigate the directory tree. Only actual directories are included in the hypertext list. In order to change to a virtual directory, a client must explicitly enter the alias or URL or select a link on another page that points to that virtual directory. This option is disabled by default because enabling it allows a user to navigate the directory structure—a potential security risk.

Index this directory

Mark this checkbox, which is marked by default, to cause Index Server to index the contents of this directory when it does a full-text index of the web site.

FrontPage Web

Mark this checkbox, which is marked by default, to create a FrontPage web for this web site. Clear the checkbox to delete the FrontPage web for this site.

Application Settings

IIS defines an application as all directories and files contained within a directory that is marked as an application starting point, until another application starting point is encountered. If you define the home directory for the web site as an application starting point, all directories (physical and virtual) within that site can participate.

Create/Remove

To define a directory as an application starting point, click Create and name the application. To remove the home directory from the application, click Remove.

Name

Type the name of the application, which appears within the property sheets for any directory contained in the application.

Google Exhibit No. 1022
Page 018

Google Exhibit No. 1063
Page 0882

Configuration

Click this button to display the tabbed Application Configuration dialog, which you can use to set ISAPI caching, define application mappings, set application configuration options, and set application debugging options.

Run in separate memory space (isolated process)

Mark this checkbox, which is cleared by default, to run the application as a separate process. Choosing this option protects all other processes—including the WWW Service itself—from being affected if this application crashes and burns. The only downside to marking this checkbox is that it causes the application to consume more memory.

Permissions

Mark one of the following option buttons to specify whether or not applications can run in the directory or to limit the types of applications that can run.

None

Select this option button to prevent any programs or scripts from running in this directory.

Script

Select this option button, which is enabled by default, to allow applications mapped to a script engine to run even though Execute permission is not granted. Set this option button for directories that contain scripts, for example, CGI, IDC, or ASP scripts. Granting Script permission is safer than granting execute permission because it limits the types of programs than can run.

Execute (including script)

Select this option button to allow any application to run in the directory, including applications that are mapped to script engines and Windows NT executables.

A share located on another computer

Use the Home Directory page of the Web Site Properties dialog to change the location of or modify the properties for the home directory for the web site, if that home directory is located on a network share rather than on the computer where IIS is installed. All settings on this page (and the appearance of the page itself) are identical to those described in the preceding section, with the following exceptions:

Network Directory

Enter the location of the home directory on the network share, using UNC format, for example, *servername**sharename*.

Google Exhibit No. 1022
Page 019

Google Exhibit No. 1063
Page 0883

Connect As

Click this button to display the Network Directory Security Credentials dialog, and enter the username and password you want to use to connect to this share.

A redirection to a URL

Use the Home Directory page of the Web Site Properties dialog to redirect requests for the source URL to a URL you specify. Set the following options:

Redirect to

Specify a valid URL to which requests will be redirected, as follows:

- To map requests to another web site, enter the fully qualified URL of that web site. For example, if this web site is *www.ttgnet.com* and you want to redirect requests to *www.oreilly.com*, enter *http://www.oreilly.com* here.
- To map requests to a virtual directory, enter the virtual path here. For example, to redirect requests to the virtual directory *wwwtemp*, enter */wwwtemp* here.
- To map requests selectively, you may enter a URL here that uses redirection variables and wildcards, as described in the online IIS documentation.

The client will be sent to

Mark one of the checkboxes below, as follows:

The exact URL entered above

Mark this checkbox, which is cleared by default, to redirect a virtual directory to the target URL without adding any part of the original URL. For example, if you want to redirect requests to the virtual directory */wwwtemp* to the HTML page *index.html*, enter *index.html* in the Redirect to: box and mark this checkbox.

A directory below this one

Mark this checkbox, which is cleared by default, to redirect a parent directory to a child directory. For example, if you want to redirect the home directory (/) to a subdirectory named */wwwtemp*, enter */wwwtemp* in the Redirect to: box and mark this checkbox. If this checkbox is not marked, the WWW Service will always map the parent directory to itself.

A permanent redirection for this resource

Mark this checkbox, which is cleared by default, to permanently redirect one resource to another. Ordinarily, redirects are temporary. When this checkbox is marked, the WWW Service returns a 301 Permanent Redirect message to the requesting browser, which can use that message to permanently alter a stored URL like a bookmark or favorite.

Google Exhibit No. 1022
Page 020

Google Exhibit No. 1063
Page 0884

Configuring Documents properties

Use the Documents page of the Web Site Properties dialog to configure default documents and document footers. You can configure Documents properties at the web site, directory, and virtual directory levels, as follows:

Enable Default Document

Mark this checkbox, which is marked by default, to cause the web site to return a default document to a web browser that does not request a specific file name. The default document may be the home page for the directory, or the index page described earlier that allows the user to navigate the directory structure. You can specify multiple default documents, and the web site will return the first listed document it locates to the requesting browser. Use the Add and Remove buttons to modify the list of documents shown in the pane, and the up and down arrow keys to set the priority for each, with the top-most document having the highest priority.

Enable Document Footer

Mark this checkbox, which is cleared by default, and enter the full path and file name of an HTML-formatted page fragment in the box if you want the web site to append that HTML fragment as a footer to every page it delivers. This fragment should contain only the HTML tags necessary to format and display the footer content. The most common use of this feature is to append relatively static information, for example, copyright notices, to the bottom of each page delivered by the web site.

Configuring Directory Security properties

Use the Directory Security page of the Web Site Properties dialog, shown in Figure 10-7, to configure security settings for the web site.

Set the following options:

Anonymous Access and Authentication Control section

To allow anonymous users to access public information on the web site while permitting only designated users to access restricted content, you can configure the web site to authenticate users. To do so, click Edit to display the Authentication Methods dialog, shown in Figure 10-8.

Configure the following options to control access based on existing Windows NT user accounts and existing NTFS permissions on files and directories:

Allow Anonymous Access

Mark this checkbox, which is marked by default, if you want to allow users to connect to the server anonymously with restricted permissions (the normal state of affairs for a web server). When a user connects anonymously, the web server logs the user on with a valid Windows NT user account, which by default is named IUSR_<computer-name>, in this case

Google Exhibit No. 1022
Page 021

Google Exhibit No. 1063
Page 0885

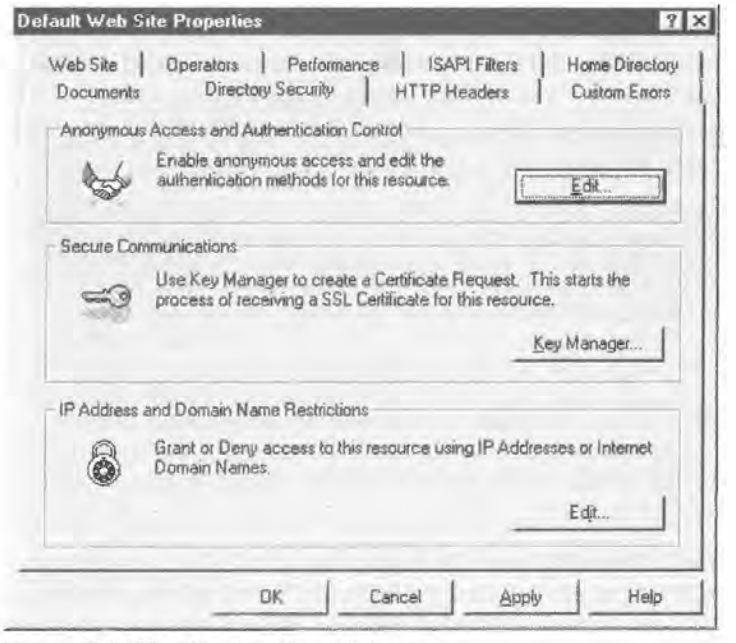


Figure 10-7. The Directory Security page

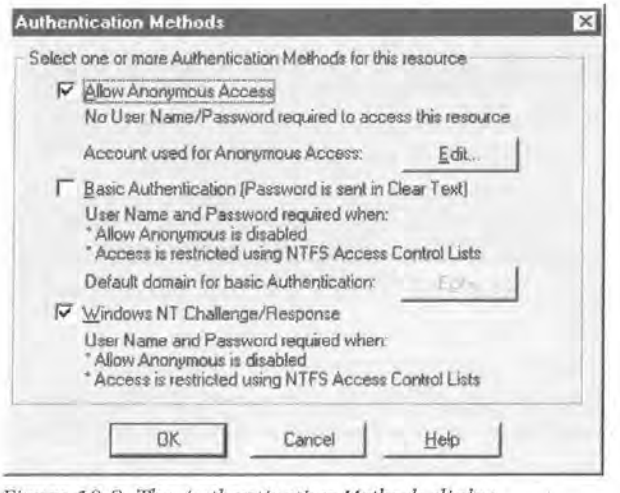


Figure 10-8. The Authentication Methods dialog

IUSR_BASTET. Windows NT creates this user account in User Manager for Domains and Internet Service Manager when IIS is installed, and assigns it the limited permissions needed by anonymous users. To change this anonymous username, click Edit to display the Anonymous User Account

Google Exhibit No. 1022
Page 022

Google Exhibit No. 1063
Page 0886

dialog, and make the necessary changes. If you create a new anonymous user account, you must make sure that the UMD and ISM password settings are identical. Mark the Enable Automatic Password Synchronization checkbox to ensure that this occurs automatically. If you create a new anonymous account, make sure to grant it the "Log on locally" user right, which is required for anonymous access to the web site.

WARNING You can use ISM to enable password synchronization between ISM and a Windows NT user account located either on the local computer that runs IIS or on a computer elsewhere on the network. However, enabling password synchronization for a non-local user account may result in all anonymous access to the web site or FTP site being denied. If the web or FTP site depends on anonymous users accessing resources located on the network (rather than on the local IIS computer), you cannot use the password synchronization feature. Instead, specify a valid password for the anonymous user.

Basic Authentication (Password is sent in Clear Text)

Mark this checkbox, which is cleared by default, to enable Basic Authentication for the web site as a fallback method. The web site uses Basic Authentication only if anonymous access has been disabled, or with anonymous access enabled, if a user attempts to access content that is forbidden to anonymous users. When either of these conditions obtains, the user's web browser displays a dialog box to prompt the user to enter a valid Windows NT username and password, which is transmitted in unencrypted form, a potential security risk. ISM displays a warning dialog to that effect when you mark this checkbox. Users who log in with Basic Authentication must provide the logon domain in addition to the user name and password. By default, IIS uses the local IIS domain if the user does not enter a logon domain. You can change the default logon domain to other than the local IIS domain by clicking Edit and entering another domain name.

Windows NT Challenge/Response

Mark this checkbox, which is marked by default, to enable Windows NT Challenge/Response authentication, a more secure authentication method than the Basic Authentication described above. The web site uses Windows NT Challenge/Response Authentication only if anonymous access has been disabled, or with anonymous access enabled, if a user attempts to access content that is forbidden to anonymous users. When either of these conditions obtains, the user's web browser (which must be Microsoft Internet Explorer 2.0 or higher) displays a dialog box to prompt the user to enter a valid Windows NT username and password. During

Google Exhibit No. 1022
Page 023

Google Exhibit No. 1063
Page 0887

this authentication process, the web server encrypts the exchanges required to authenticate the user.

Secure Communications section

You cannot use the secure communications features of the WWW Service until you have applied for, received, and installed a server certificate, and bound that certificate to the server key pair. Until you have taken these actions, the button in this section is labeled Key Manager. Clicking it displays Key Manager, which you use to create an SSL certificate request for this server. Once that certificate is installed and configured, the button in this section is relabeled Edit. Clicking it allows you to configure the SSL and PCT options on the server.

IP Address and Domain Name Restrictions section

You can use this section to deny all but specified IP addresses and/or domain names access to the web site. For example, if you have created a private web site that is intended for use only by your employees, you might use this feature to deny access to all users but those who are members of your own DNS domain. Alternatively, you can elect to grant all but specified IP addresses and/or domain names access to your web site. For example, if for some reason you decide that you don't want any user from the *aol.com* domain to access your web site, you can choose to deny that domain. To configure permit/deny settings, click Edit to display the IP Address and Domain Name Restrictions dialog, shown in Figure 10-9.



Figure 10-9. The IP Address and Domain Name Restrictions dialog

Begin by marking one of the option buttons at the top of the dialog to specify the default access. Mark the Granted Access option button if you want to grant access to all computers except those you enter in the list. Mark the Denied Access option button if you want to deny access to all computers except those you enter in the

Google Exhibit No. 1022
Page 024

Google Exhibit No. 1063
Page 0888

list. Once you have specified the default access method, you may take the following actions to modify the list of computers that are granted or denied permission to access the site.

Add

Click this button to display the Deny Access On dialog (if you marked the Granted Access option button for default access) or Grant Access On dialog (if you marked the Denied Access option button for default access). Under Type, mark one of the following option buttons:

Single Computer

To display an IP address entry field in dotted decimal format. ISM enters a line item in the list that grants (or denies, as appropriate) access to this single computer.

Group of Computers

To display a Network ID entry box and a Subnet Mask entry box, both formatted as dotted decimal. Enter the network ID and subnet mask to grant or deny access to the specified subnet.

Domain Name

To display an entry box for domain name. Note that filtering by domain name is a very resource intensive option, because it requires a reverse DNS lookup for each operation.

Remove

Highlight the line item in the pane and click Remove to remove that grant or deny filter.

Edit

Highlight the line item in the pane and click Edit to change the properties for that grant or deny filter.

Configuring HTTP Headers properties

Use the HTTP Headers page of the Web Site Properties dialog, shown in Figure 10-10, to configure the values returned by the web site in the HTML header and to set content expiration properties.

Set the following options:

Enable Content Expiration section

Mark this checkbox, which is cleared by default, and set expiration options to control the display of time-sensitive content, for example, seasonal material or time-limited offers. The WWW Service embeds content-expiration data in content page HTTP headers. The client browser compares its current date with the expiration date supplied with the content to determine whether to display the

Google Exhibit No. 1022
Page 025

Google Exhibit No. 1063
Page 0889

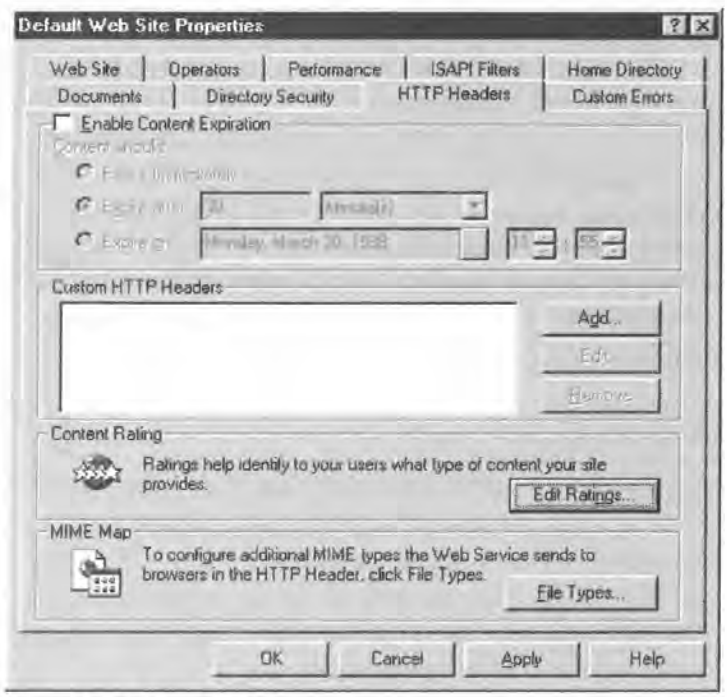


Figure 10-10. The HTTP Headers page

cached page or to request an updated page from the web server. If you enable content expiration, select one of the following option buttons:

Expire Immediately

Select this option button if you never want the browser to display a cached page. It will instead display the page only once, and then request an updated page on subsequent accesses.

Expire after

Select this option button, enter a number, and use the drop-down list to select minutes, hours, or days to specify that the browser may display this page from cache for the specified time period before requesting an updated page. The duration defaults to 30 minutes.

Expire on

Select this option button and enter a date and time to specify that the browser may display this page from cache until the specified date and time. The default value is 10 days since ISM was last started.

Custom HTTP Headers section

Use this section to enter custom headers that the web server will send to client browsers. To add a custom header, click Add to display the Add/Edit Custom Header dialog, and enter values for Custom Header Name and Custom

Google Exhibit No. 1022
Page 026

Google Exhibit No. 1063
Page 0890

Header Value. To edit an existing custom header, highlight the custom header, click Edit to display the Add/Edit Custom Header dialog, and alter the values for Custom Header Name and Custom Header Value as necessary. To remove a custom header, highlight it and click Remove.

Content Rating section

IIS can embed content rating data in content page HTTP headers. This information can be read by some web browsers, for example, Internet Explorer 3.0 or higher, and used to suppress display of pages with potentially objectionable content, according to the preferences set by the browser user. To assign ratings to your content, take the following steps:

1. Click Edit Ratings to display the Content Ratings dialog.
2. Click Ratings Questionnaire to download the current version of the ratings questionnaire from the RSAC web site.
3. After completing the questionnaire, click the Ratings tab to display the Ratings page.
4. Mark the Enable Ratings for this resource checkbox.
5. In the Category pane, expand the RSACi tree, choose each component—Violence, Sex, Nudity, Language—in sequence, and assign it the appropriate rating based on your answers to the questionnaire.
6. In the Optional Information section, enter the email address of the person who rated the content and click Date to enter a date when the rating information expires.

MIME Map section

To change MIME mappings for this web site, click File Types to display the File Types dialog, and use this dialog to add, edit, or remove registered file types as described earlier in the section about configuring Master Properties. The mappings you modify here apply only to this web site. If you set MIME mappings in Master Properties, you can still modify the mappings on a site by site basis. However, when you reapply Master Properties, the individual per-site MIME mappings are overwritten by those contained in Master Properties.

Configuring Custom Errors properties

Use the Custom Errors page of the Web Site Properties dialog to configure the type of error message that the WWW Service returns to the client browser when an HTTP error occurs. For each HTTP 1.1 error, you can select among the following options:

Default

Display the default HTTP 1.1 error message, for example, "404—Object Not Found". If you set an HTTP 1.1 error to use the default message, the contents of that error message are displayed in the Error Message pane. By default, IIS

Google Exhibit No. 1022
Page 027

Google Exhibit No. 1063
Page 0891

displays default HTTP 1.1 error messages for 403;14 - Forbidden—Directory Listing Denied; 500—Internal Server Error; 501—Not Implemented; and 502—Bad Gateway.

File

Display the contents of an HTML-formatted file. For most HTTP 1.1 errors, Microsoft supplies a more informative friendly version to replace most default error messages. Each of these messages is contained in an HTML-formatted file that is named with the HTTP 1.1 error number. You can use these replacement error message files as is, modify them to suit you, or replace them with your own versions. If you set an HTTP 1.1 error to use a custom message from a file, the path and filename are displayed in the Error Message pane.

URL

Display the contents of the specified URL, which must be located on a local server. At first glance, this option appears to duplicate the functionality of the File option. However, it is intended for a different purpose. If you use static HTML files for error messages, always use the File option. Use the URL option only if you are using a custom ASP or ISAPI application to return dynamic error messages. If you do this, the application must set the HTTP header status appropriate for the error. Otherwise, the server will always return HTTP 1.1 status 200—OK. If you set an HTTP 1.1 error to use a custom message from a URL, the URL is displayed in the Error Message pane.

You can configure custom error messages at the web site, directory (physical or virtual), and file level. When you configure custom error messages, all lower entities inherit that list, which replaces the existing list rather than being merged with it.

To configure the error message for an HTTP 1.1 error, highlight the error in the Error Message pane, and click Edit Properties to display the Error Mapping Properties dialog, shown in Figure 10-11. Use the Message Type drop-down list to select Default, File, or URL. If you select Default, there are no further options to choose. If you select File, enter the path and filename of the custom error file in the File box, or click Browse to locate it. If you select URL, type the full URL in the URL box. After you complete your edits, click on OK to save the change and return to the Custom Errors page.

Creating Additional Web Sites

IIS can support multiple web sites on one computer. Each web site must be uniquely identified by a combination of IP address, host header name, and/or TCP port number, as follows:

IP Address

By far the cleanest method is to assign a unique static IP address to each web site and create a DNS entry for each. For example, we could assign the web

Google Exhibit No. 1022
Page 028

Google Exhibit No. 1063
Page 0892



Figure 10-11. Configuring the error messages that will be returned by the server to clients when HTTP errors occur

site *www.ttgnet.com* the IP address 204.238.30.165 and the web site *support.ttgnet.com* the IP address 204.238.30.166, create CNAME records for each, and use the drop-down IP Address list on the Web Site page of the Web Site Properties dialog to associate each IP address with its corresponding web site. There is no ambiguity when a client browser attempts to access one of these sites, because the resolver stub in the browser uses the web site name to resolve a unique IP address for the web site.

Host header name

If you cannot assign a unique IP address to each web site—for example, if your ISP has given you only one static IP address—you can use the same IP address to support multiple web sites by assigning each web site a unique host header name. For example, *www.ttgnet.com* and *support.ttgnet.com* could both use the IP address 204.238.30.165. When a client browser requests either site, the request is directed to the same IP address, but IIS uses the host name requested by the client to determine which web site content to deliver. This method works with modern browsers, but fails with older browsers unless you take the steps described in the IIS documentation under "Supporting Host Header Names in Older Browsers". To configure multiple web sites using host header names, use the Advanced button on the Web Site page of the Web Site Properties dialog to associate each web site with its corresponding host header name.

TCP Port number

As a last resort, you can define multiple unique web sites on an IIS server by assigning each web site the same IP address, but with a unique TCP Port number for each site. We say as a last resort, because every web browser on the planet assumes that a web site will respond to TCP Port 80, the Well-Known Address for HTTP. For a client browser to access a web site that uses a TCP Port number other than 80, the browser must explicitly enter the port number with the URL.

Google Exhibit No. 1022
Page 029

Google Exhibit No. 1063
Page 0893

After you have decided how to identify the new site uniquely, take the following steps to create it:

1. Start ISM and highlight the IIS computer in the left pane.
2. Select Action → New → Web Site to invoke the New Web Site Wizard.
3. Enter a name for the new web site in the Web Site Description box and click Next. Note that the name you assign here is only the display name used by ISM. It need not be the same as the DNS name for the web site.
4. Use the Select the IP Address to use for this Web Site drop-down list to choose the IP address for the site, and enter the TCP Port number in the TCP Port this Web Site should use (Default: 80) box. If you have installed a server certificate and want to use SSL, enter the SSL port. Click Next.
5. Enter (or browse for) the home directory path for this web site. By default, the "Allow anonymous access to this web site" checkbox is marked. Clear it if you want to disable anonymous access and require anyone who wants to use the site to provide a valid username and password. Click Next.
6. Set the access permissions for the new web site. By default, only the "Allow Read Access" and "Allow Script Access" checkboxes are marked. The "Allow Execute Access (includes Script Access)", "Allow Write Access", and "Allow Directory Browsing" checkboxes are cleared. Once you have configured the access permissions, click Finish to continue.
7. ISM creates the web site and installs it in the tree for the computer. By default, the new web site is stopped. To start it, right click the new web site name in ISM to display the context sensitive menu and choose Start.
8. Use the Web Site Properties dialog for the new web site to make any configuration changes necessary.
9. Create a DNS CNAME entry for the new web site.

Creating an FTP Site

Like setting up a web site, setting up an FTP site with IIS is a two-step process. You first design your FTP directory structure, create the directory tree, and populate it with the files you want available for download. Once you have done so, you configure the properties for the FTP site that will make those files accessible to users. To create an FTP site based on the Default FTP Site that was installed when you installed IIS, take the following steps:

1. If you accepted the defaults when installing IIS, the Default FTP Site home directory is the *\Inetpub\Ftproot* directory on the drive where you installed IIS. If the FTP site will have only a few files, you can copy them all to this

Google Exhibit No. 1022
Page 030

Google Exhibit No. 1063
Page 0894

directory. If you will make many files available for download, create a subdirectory structure under the home directory and use it to organize the files.

2. Create a CNAME DNS entry to alias ftp to the computer name that runs IIS. For example, we installed IIS on the computer *bastet.ttgnet.com*. To allow DNS users to resolve the FTP site as *ftp.ttgnet.com*, we added a DNS entry to alias ftp.ttgnet.com to bastet.ttgnet.com. When a user enters ftp.ttgnet.com in his browser or FTP client, DNS resolves the request to the computer *bastet.ttgnet.com*.

3. Modify the Default FTP Site Properties dialog as necessary. Configuring FTP site properties is done with a tabbed dialog similar to that used to configure web site properties. It includes the following pages:

   FTP Site
   Configure identification, connection limits, and logging options for the FTP site.

   Security Accounts
   Configure anonymous connection options and specify which users and groups can perform routine administrative functions for the FTP site.

   Messages
   Configure the messages that FTP users will see when they connect to and exit the FTP site, and the message that the FTP server returns to clients who attempt to connect to it when the maximum number of concurrent connections has been exceeded.

   Home Directory
   Configure the location of or modify the properties for the home directory for the FTP site, and specify the type of directory listing—Unix or MS-DOS—that will be returned to FTP clients.

   Directory Security
   Configure TCP/IP access restrictions for the FTP site, to grant or deny access to particular IP addresses, subnets, or domain names.

Installing IIS4 using Typical installs a Default FTP Site. You can use the procedures described in the following sections to modify the properties for this site, as well as for other FTP sites you create. Changes you make to them affect only the individual FTP site you are changing. You can also these same procedures to modify the Master Properties for the FTP Service, described earlier. Changes you make to the Master Properties for the FTP Service are used as default settings when you create a new FTP site, and may optionally be used to update (and overwrite) the properties for existing FTP sites.

You can designate one FTP site as the default FTP site for IIS 3.0 administration. Any software that creates virtual directories will use the designated default FTP site

Google Exhibit No. 1022
Page 031

Google Exhibit No. 1063
Page 0895

as the target. To designate a default FTP site, right-click the computer name in MMC to display the context-sensitive menu and choose Properties to display the <computer-name> Properties dialog. In the Master Properties section, use the drop-down list to select FTP Service, and click Edit to display the FTP Service Master Properties for <computer-name> tabbed dialog. Click the IIS 3.0 Admin tab, and use the Default FTP Site drop-down list to select the default FTP site.

To change properties for an FTP site, right-click that web site in MMC to display the context-sensitive menu, and then choose Properties to display the tabbed <FTP-site-name> FTP Site Properties dialog.

Configuring FTP Site properties

Use the FTP Site page of the FTP Site Properties dialog to configure identification, connection limits, and logging options for the FTP site, and to view and manage connected users, as follows:

- FTP Site Identification section

  Description

  The name of the FTP site. This is what appears in Internet Service Manager tree view.

  IP Address

  Use the drop-down list to select an IP address to associate with the FTP site. Only IP addresses that have been assigned to this computer (using the Control Panel Network tool) appear in this list. If you accept the default setting for this field, All Unassigned, this web site responds to all IP addresses assigned to this computer that are not assigned to other sites.

  TCP Port

  Specifies the TCP port upon which the FTP Service responds to queries, by default port 21. You can specify any other TCP port number here, but clients must explicitly use that port number to access the FTP server. TCP Port number is required. If you change the TCP Port, you must restart the server before the change takes effect.

- Connections section

  Unlimited

  Mark this option button to allow unlimited simultaneous connections to the FTP site.

  Limited To

  Mark this option button, which is marked by default, and enter a value for connections to limit the number of simultaneous connections to the FTP site. By default, the concurrent connection limit is set to 100,000.

Google Exhibit No. 1022
Page 032

Google Exhibit No. 1063
Page 0896

Connection Timeout

Specify, in seconds, how long the FTP server waits before disconnecting an inactive user. The default value is set to 900 seconds.

- Enable Logging section

Enable Logging

Mark this checkbox, which is marked by default, to cause IIS to log FTP user activities.

Active log format

If you enable logging, use the drop-down list to select one of the following log file formats. Once you have selected a format, click Properties to display a property sheet specific to that format, where you can set various logging options, including log time period, log file location, maximum log file size, items to be logged, and so on.

Microsoft IIS Log File Format

An ASCII log file format with predefined headings.

ODBC Logging

A fixed format logged to an ODBC-aware database, for example, Access or SQL Server. You must create the database yourself before data can be written to it. Using this format allows you to use the more powerful data analysis and reporting tools available with the database application.

W3C Extended Log File Format

The default log file format. W3C is a configurable ASCII format. By default, Time, Client IP Address, Method, URI Stem, and HTTP Status are logged. You can select among numerous other data types to be logged.

You can also use this page to view and manage active FTP sessions. To do so, click Current Sessions to display the FTP User Sessions dialog, shown in Figure 10-12. This dialog displays the total number of users currently connected to the FTP server and, for each connected user, the identity of the user (typically, that user's email address), the IP address from which the user is connected, and the elapsed duration of the connection.

This dialog presents a snapshot of the current sessions as of the time you display it rather than updating sessions dynamically. Use this dialog to perform the following activities:

- To update the current sessions list, click Refresh.
- To disconnect one user session, highlight that session and click Disconnect. ISM displays a warning message to notify you that you are about to terminate the session. Click Yes to terminate the session. When you terminate a session, the connected user does not receive any preliminary warning.

Google Exhibit No. 1022
Page 033

Google Exhibit No. 1063
Page 0897



Figure 10-12. The FTP User Sessions dialog.

- To disconnect all user sessions, click Disconnect All. ISM displays a warning message to notify you that you are about to terminate all user sessions. Click Yes to terminate the sessions immediately.

Configuring Security Accounts properties

Use the Security Accounts page of the FTP Site Properties dialog, shown in Figure 10-13, to configure anonymous access options and to specify which users and groups can perform routine administrative functions for the FTP site.

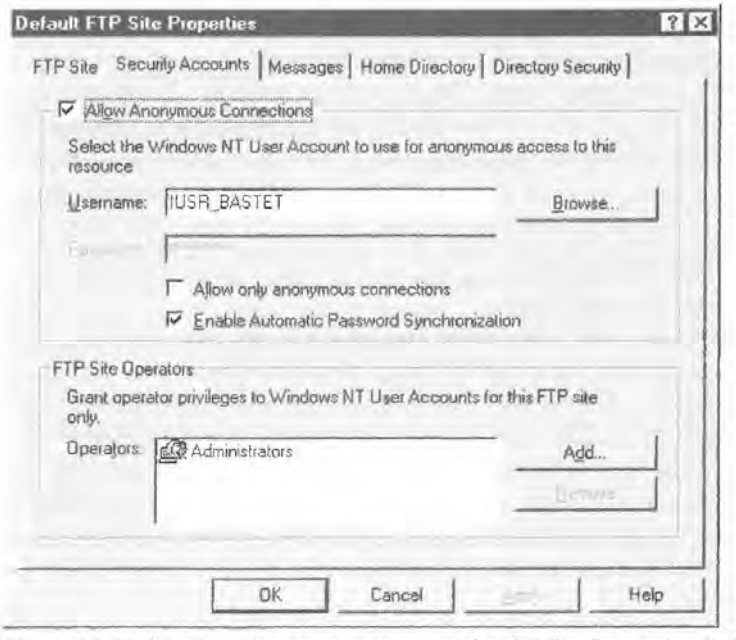


Figure 10-13. The Security Accounts page of the FTP Site Properties dialog

Google Exhibit No. 1022
Page 034

Google Exhibit No. 1063
Page 0898

Configure the following settings:

- Allow Anonymous Connections section—Mark this checkbox, which is marked by default, to allow FTP clients to log on to the FTP site by entering anonymous as their username and their email address as the password. The FTP Service then uses the username and password described below as the Windows NT logon account to grant access to resources on the FTP server. Note that this username and password is only used internally by Windows NT to grant access to shared resources. The anonymous user does not enter them.

  Username

  When you install IIS, Windows NT automatically creates a user account for anonymous logons in User Manager for Domains and in Internet Service Manager, and assigns the same random password to each of them. By default, this account is given the name IUSR_<computer-name>, for example, IUSR_BASTET. When an anonymous user connects to the FTP server, the FTP Service uses this account to access shared Windows NT resources and to determine the permissions available to the anonymous user. To substitute an existing Windows NT user account as the anonymous logon account, click Browse to display the Select Windows NT User Account dialog and select the account you want to use for anonymous logons. This account must have the "Log on locally" user right, which normal user accounts do not have by default.

  Password

  This field contains the password for the Windows NT user account used to grant access to anonymous users. This field must contain a password. You cannot use a blank password. This field is inactive by default because the Enable Automatic Password Synchronization checkbox described below is enabled by default. If you clear that checkbox, this field activates, allowing you to change the password in ISM. If you do this, you must also change the password in UMD to the same password for this account.

  Allow only anonymous connections

  Mark this checkbox, which is cleared by default, if you want to allow only anonymous connections to the FTP server. When this checkbox is cleared, users who connect anonymously are granted only the limited permissions associated with the anonymous account, while users who provide a valid Windows NT username and password are granted the higher level of permissions associated with their own user accounts. By marking this checkbox, you limit all users to connecting as user anonymous at the limited privilege level assigned to anonymous logons.

Google Exhibit No. 1022
Page 035

Google Exhibit No. 1063
Page 0899

Enable Automatic Password Synchronization

If you configure a different Windows NT user account to be used for anonymous logons, you must ensure that the password for this account is identical in both User Manager for Domains and Internet Service Manager. Mark this checkbox to cause ISM to automatically update the ISM password for the anonymous user account when that password is changed in UMD. This checkbox is marked by default if the account used for anonymous logons exists in ISM but not in UMD when that account is created.

- FTP Site Operators section—By default, the Administrators group is assigned operator status. However, you can assign any user or group as an operator. Assign and remove operators as follows:

  Add an operator

  Click on Add to display the Add Users and Groups dialog. Use the List Names From drop-down list to specify the domain from which the account is to be added. The Names pane lists the available users and groups. Double-click a user or group name to add it to the Add Names pane, and then click on OK. ISM adds the selected user or group to the Operators pane.

  Remove an operator

  Highlight the user or group name in the Operators pane and click Remove.

Granting operator status to a user or group grants privileges only for this FTP site. Note that only members of the Windows NT Administrator group can manage virtual FTP sites. Adding a user or group to the Operators pane does not enable them to manage virtual FTP sites.

Configuring Messages properties

Use the Messages page of the FTP Site Properties dialog to configure the messages that FTP clients will see when they visit your site. You can create or modify the following messages, all of which are blank by default, simply by typing the message into the appropriate box:

Welcome

Enter the message you want FTP clients to see when they first connect to your FTP site. It's a good idea to include the name of the site with the greeting. Nowadays, many sites use this message to display copyright information, legal disclaimers, and a warning that unauthorized access or use of the site is prohibited.

Exit

Enter the message you want FTP clients to see when they log off the server. Traditionally, you thank them for visiting the site and invite them to return soon—kind of like the Beverly Hillbillies.

Google Exhibit No. 1022
Page 036

Google Exhibit No. 1063
Page 0900

Maximum Connections

If you restrict concurrent connections, enter the message you want FTP clients to see when they attempt to log on to the server when all available connections are already in use.

Configuring Home Directory properties

Use the Home Directory page of the FTP Site Properties dialog, shown in Figure 10-14, to change the location of or modify the properties for the home directory for the FTP site. Installing IIS creates the FTP root directory that you specify, ordinarily *\Inetpub\ftproot*. You can use this dialog to change that location. This is a dynamic dialog, whose contents vary according to the option button you select in the top section, as follows:

A directory located on this computer

Select this option, which is the default, to use an FTP root directory on the local computer where you installed IIS. Use this page to specify the location, access permissions, and directory listing style for the local directory. Uses standard drive/directory syntax, for example, *C:\web\ftpfiles*.

A share located on another computer

Select this option to use an FTP root directory located on a network share, using UNC syntax in the form \\servername\sharename, for example, *bastet\ftpfiles*. When you enter the sharename, click Connect As to provide the username and password that will be used to access that share. The other settings are identical to those available when the FTP root directory is on the local computer.

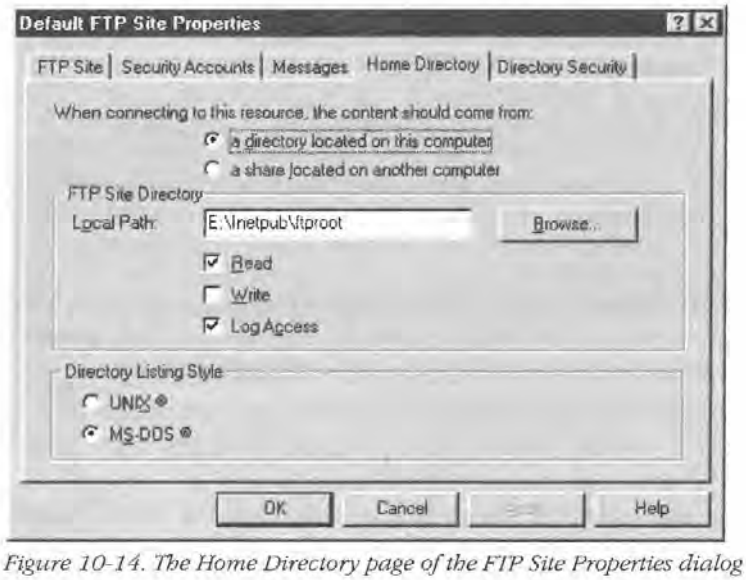


Figure 10-14. The Home Directory page of the FTP Site Properties dialog

Google Exhibit No. 1022
Page 037

Google Exhibit No. 1063
Page 0901

Set the following options:

FTP Site Directory section

Local Path / Network Share

Type the drive and directory for the local path where the home directory resides on the local computer, or browse for the location. If you've chosen to locate the FTP root directory on a network share, enter the network share name here in UNC form, and click Connect As to enter the username and password for the account that will be used to access the share.

Read

Mark this checkbox, which is marked by default, to assign Read permission if you want FTP clients to be able to read or download the files contained in the home directory. This checkbox must be marked for FTP to be functional.

Write

Mark this checkbox, which is cleared by default, to assign Write permission if you want FTP clients to be able to upload files to the FTP site. Set this permission only for directories that are intended to receive uploaded files.

Log access

Mark this checkbox, which is marked by default, to cause the FTP Service to write a log file entry for each visit to this directory. For this setting to be effective, logging must be enabled for the FTP site in the FTP Site properties page, which it is by default.

Directory Listing Style section

Specify the directory listing style—Unix or MS-DOS—that the FTP Service will use when returning directory listings to FTP clients. Most modern FTP clients can accept either style, but some clients choke on MS-DOS style listings, so select Unix for maximum compatibility.

Configuring Directory Security properties

Use the Directory Security page of the FTP Site Properties dialog, shown in Figure 10-15, to grant or deny all but specified IP addresses and/or domain names access to the FTP site. For example, if you have created a private FTP site that is intended for use only by your employees, you might use this feature to deny access to all users but those who are members of your own DNS domain. Alternatively, you can elect to grant all but specified IP addresses and/or domain names access to your FTP site. For example, if a particular computer is downloading an excessive number of files from your FTP site, you can deny FTP access to that computer by IP address.

Google Exhibit No. 1022
Page 038

Google Exhibit No. 1063
Page 0902

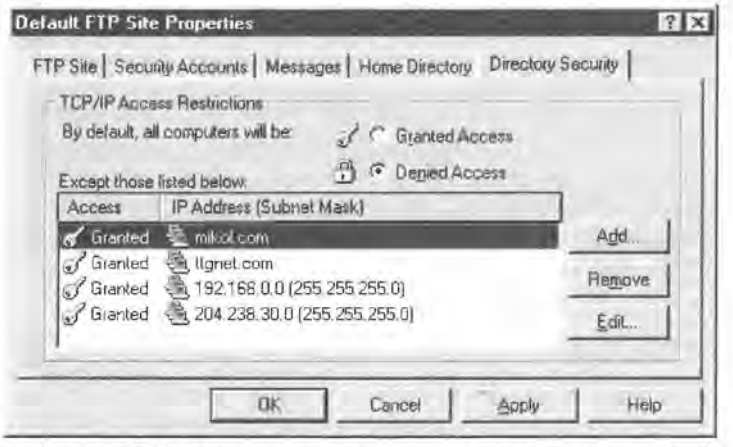


Figure 10-15. The Directory Security page of the FTP Site Properties dialog

Begin by marking one of the option buttons at the top of the dialog to specify the default access. Mark the Granted Access option button if you want to grant access to all computers except those you enter in the list. For a public FTP site, this is the proper choice. Mark the Denied Access option button if you want to deny access to all computers except those you enter in the list. For a private FTP site, this is the usual choice. Once you have specified the default access method, you may take the following actions to modify the list of computers that are granted or denied permission to access the site.

Add

Click this button to display the Deny Access On dialog (if you marked the Granted Access option button for default access) or Grant Access On dialog (if you marked the Denied Access option button for default access). Under Type, mark one of the following option buttons:

Single Computer

To display an IP address entry field in dotted decimal format. ISM enters a line item in the list that grants (or denies, as appropriate) access to this single computer.

Group of Computers

To display a Network ID entry box and a Subnet Mask entry box, both formatted as dotted decimal. Enter the network ID and subnet mask to grant or deny access to the specified subnet.

Domain Name

To display an entry box for domain name. Note that restricting access by domain name is a very resource intensive option, because it requires a reverse DNS lookup for each connection.

Google Exhibit No. 1022
Page 039

Google Exhibit No. 1063
Page 0903

Remove
: Highlight the line item in the pane and click Remove to remove that grant or deny filter.

Edit
: Highlight the line item in the pane and click Edit to change the properties for that grant or deny filter.

Creating Additional FTP Sites

IIS can support multiple FTP sites on one computer. Each FTP site must be uniquely identified by either IP address or TCP port number, as follows:

IP Address
: If possible, assign a unique static IP address to each FTP site and create a DNS entry for each. For example, we could assign the FTP site *ftp.ttgnet.com* the IP address 172.16.10.167 and the FTP site *files.ttgnet.com* the IP address 172.16.10.168, create CNAME records for each, and use the drop-down IP Address list on the FTP Site page of the FTP Site Properties dialog to associate each IP address with its corresponding FTP site. There is no ambiguity when an FTP client attempts to access one of these sites, because the resolver stub in the FTP client uses the FTP site name to resolve a unique IP address for the FTP site.

TCP Port number
: If you cannot assign a unique IP address to the new FTP site—for example, if your ISP has given you only one IP address—you can define multiple unique FTP sites on an IIS server by assigning each FTP site the same IP address, but with a unique TCP Port number for each site. Use this method only if you must, because FTP clients assume that an FTP server will respond to TCP Port 21, the Well-Known Address for FTP. For an FTP client to access an FTP site using a nonstandard port number, that client must explicitly specify the port number to establish the connection.

After you have decided how to identify the new site uniquely, take the following steps to create it:

1. Start ISM and highlight the IIS computer in the left pane.
2. Choose Action → New → FTP Site to invoke the New FTP Site Wizard.
3. Enter a name for the new FTP site in the FTP Site Description box and click Next. Note that the name you assign here is only the display name used by ISM. It need not be the same as the DNS name for the FTP site.
4. Use the Select the IP Address to use for this FTP Site drop-down list to choose the IP address for the site, and enter the TCP Port number in the TCP Port this FTP Site should use (Default: 21) box. Click Next to continue.

Google Exhibit No. 1022
Page 040

Google Exhibit No. 1063
Page 0904

5. Enter (or browse for) the home directory path for this FTP site. Click Next.
6. Set the access permissions for the new FTP site. By default, the Allow Read Access checkbox is marked and the Allow Write Access checkbox is cleared. Accept or modify the default access permissions and click Finish to continue.
7. ISM creates the FTP site and installs it in the tree for the computer. By default, the new FTP site is stopped. To start it, right click the new FTP site name in ISM to display the context sensitive menu and choose Start.
8. Use the FTP Site Properties dialog for the new FTP site to make any configuration changes necessary.
9. Create a DNS CNAME entry for the new FTP site.

Creating an SMTP Site

Installing IIS4 automatically installs the SMTP Site for the Microsoft SMTP Service. You cannot create additional SMTP sites, and you cannot delete the existing SMTP site. There is no SMTP Service selection in Master Properties, because the Default SMTP Site is the one and only SMTP site for the computer. You configure the STMP Site using the SMTP Site Properties tabbed dialog, which includes the following pages:

SMTP Site
: Configure identification, connection options, and logging options for the SMTP site.

Operators
: Specify which users and groups can perform routine administrative functions for the SMTP site.

Messages
: Limit maximum message and session sizes, set the maximum number of outbound messages per connection and the maximum number of recipients per message, and specify what actions the SMTP Service takes when a message cannot be delivered.

Delivery
: Configure message queue options, domain options, smart host settings, and outbound security.

Directory Security
: Configure settings for anonymous access and authentication, secure communications, IP address and domain name restrictions, and SMTP relay restrictions.

To change properties for the SMTP site, right-click the site entry in MMC to display the context-sensitive menu, and then choose Properties to display the tabbed

Google Exhibit No. 1022
Page 041

Google Exhibit No. 1063
Page 0905

13

Information Resources

In this chapter:
- *The World Wide Web*
- *Anonymous FTP*
- *Retrieving RFCs*
- *Mailing Lists*
- *Newsgroups*
- *The White Pages*
- *Summary*

Now that our network is configured, debugged, and secure, how will we use it? The classic applications Telnet, FTP, and mail, are still popular. But increasingly the network is used not merely as a delivery link between two hosts, but as a path to information resources. Information servers, file repositories, databases, and information directories are available throughout the Internet.

An important application for these information resources is to keep you up-to-date about the latest developments in Windows NT and in TCP/IP network technologies. Computer technology changes very rapidly. The things that you read about in this book will surely change over time.* Online information can help you keep your knowledge current.

This chapter explores various ways to avail yourself of this storehouse of information. We look at how information is retrieved from network servers, and some tools that make it easier to locate that information.

The World Wide Web

The primary method used to retrieve network information is the World Wide Web. The Web is an interlinked network of hypertext servers based on the *Hypertext Transfer Protocol* (HTTP) that runs on top of TCP/IP. The Web is accessed via a *browser*, which is a program that provides a consistent graphical interface to the user. All of the popular browsers, including Microsoft Internet Explorer, are modeled after the original browser developed at the National Center for Supercomputer Applications (NCSA), and therefore share a common look and feel.

* We maintain a web site at *www.itgnet.com/nticp.html* that contains online updates for this book and other useful information.

Google Exhibit No. 1022
Page 042

Google Exhibit No. 1063
Page 0906

Windows NT systems ship with a built-in browser. The Microsoft Internet Explorer is bundled with the operating system. Other browsers are available on the Internet. The Netscape browser is available at *http://www.netscape.com*. It can be downloaded and quickly installed. Netscape is a popular browser, but Internet Explorer is probably the most popular browser for Windows NT.

You can, of course, also set up your Windows NT Server as a Web server. In this chapter we only look at browsing other services. To see how to set up your own service, refer to Chapter 10, *Internet Information Server (IIS)*.

Obtaining information from hypertext Web pages is a common use for a browser. Use yours to keep up with the most current network information. Figure 13-1 shows a network administrator checking online security information.

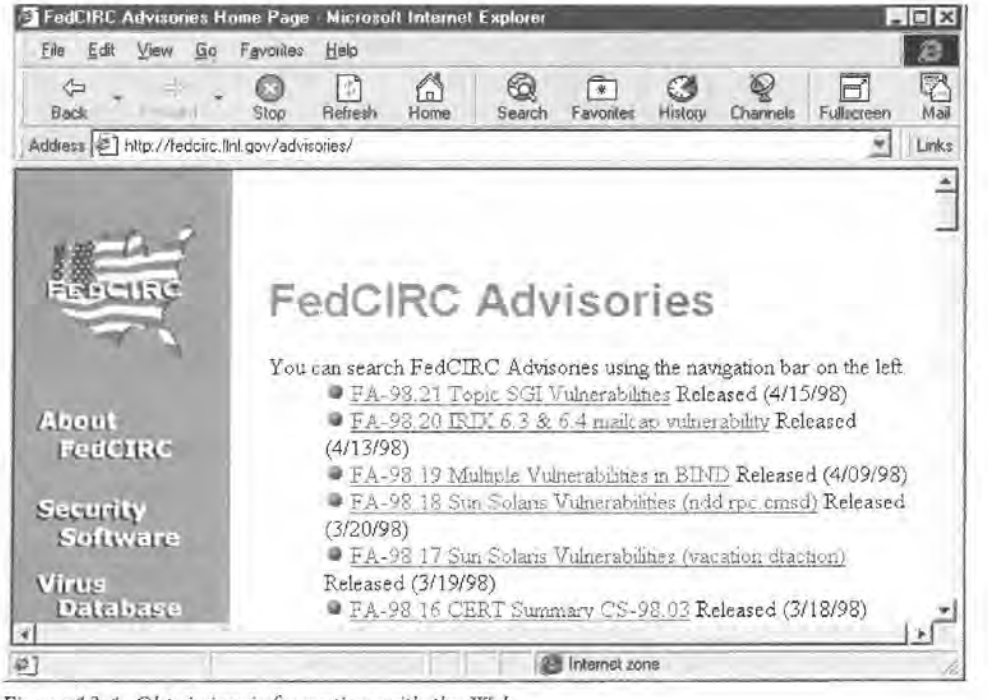


Figure 13-1. Obtaining information with the Web

The Address field near the top of the sample screen is the location of the Web page we are reading. On some other browsers this field is labeled URL, Location, or Netsite, but in all cases it performs the same function: it holds the path to the information resource. In the example the location is *http://fedcirc.llnl.gov/advisories/*. The Address is a *universal resource locator* (URL). It is a standard way of defining a network resource and it has a specific structure:

service://server/path/file

Google Exhibit No. 1022
Page 043

Google Exhibit No. 1063
Page 0907

In the sample URL, *http* is the service; *fedcirc.llnl.gov* is the server; and *advisories* is the path to the resource contained on that server. No file name is provided, so the default file, usually *index.html*, is used. This tells the browser to locate a host with the domain name *fedcirc.llnl.gov*, and to ask it for the hypertext information located in the default file in the *advisories* path.

Hypertext is not the only type of information that can be retrieved by a browser. The browser is intended to provide a consistent interface to various types of network resources. HTTP is only one of the services that can be specified in a URL.

A web browser can be used to view local hypertext files. Increasingly this is how documentation is delivered. Figure 13-2 shows an NT administrator reading the mailsrv documentation.



Figure 13-2. Reading documentation with a browser

The URL in Figure 13-2 is *file://c:\ntreskit\mailsrv\mailsrv.htm*. The service is *file*, which means that the resource is to be read via the standard file system. No server name is provided so the Internet Explorer looks for the resource on the local host. The path is *c:\ntreskit\mailsrv*, and the file is *mailsrv.htm*.

Google Exhibit No. 1022
Page 044

Google Exhibit No. 1063
Page 0908

The browser is also often used to download files, a role that is similar to FTP. Figure 13-3 shows an NT administrator using a browser to download new Windows NT modem drivers.

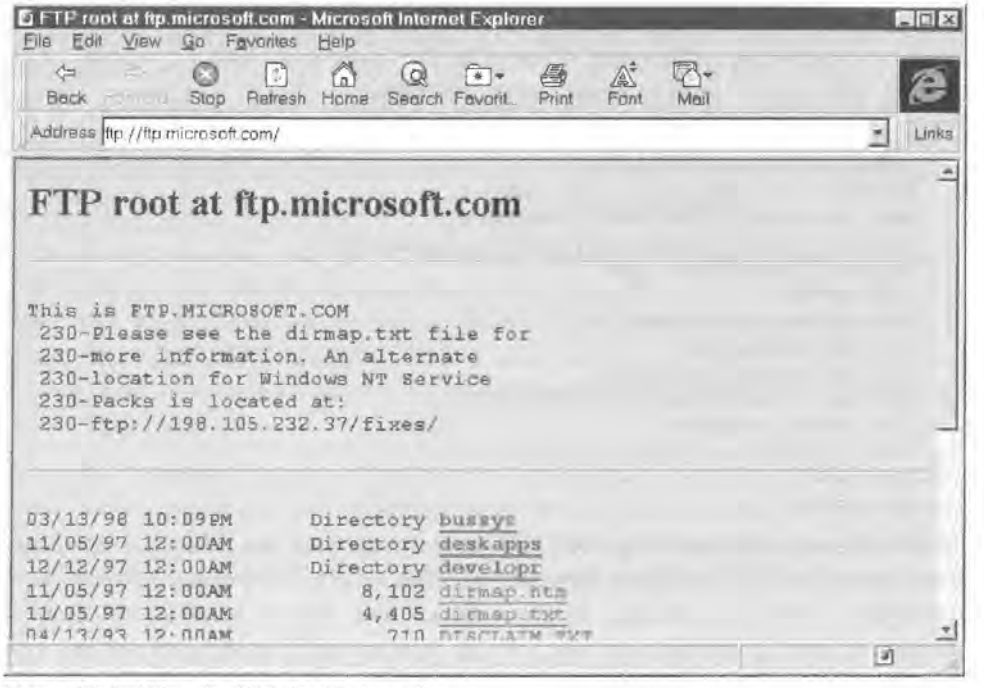


Figure 13-3. Using the Web for file transfer

The URL in Figure 13-3 shows HTTP being used to retrieve a file. FTP can also be the service used to access a resource through the browser. For example, *ftp://ftp.microsoft.com* connects to the anonymous FTP server at Microsoft. Files can be downloaded via HTTP or FTP.

Reading important announcements and documentation and downloading files are probably the most common uses a network administrator has for a Web browser. There are, however, many other things that can be done with a browser and the huge number of resources available on the network. A detailed discussion of browsers and the Web is beyond the scope of this book. See *The Whole Internet User's Guide and Catalog*, by Ed Krol, for a full treatment of these subjects.

The browser provides a consistent interface to a variety of network services. But it is not the only way, or necessarily the best way, to access all of these service. In particular, it may not be the fastest or most efficient way to download a file. An alternative is to invoke ftp directly.

Google Exhibit No. 1022
Page 045

Google Exhibit No. 1063
Page 0909

Windows NT TCP/IP
Network Administration
Hunt & Thompson

Windows NT/Networking

O'REILLY®

Windows NT TCP/IP Network Administration



Windows NT TCP/IP Network Administration is a complete guide to setting up and running a TCP/IP network on Windows NT. Windows NT and TCP/IP have long had a close association, and this is the first book to focus exclusively on NT networking with TCP/IP. It starts with the fundamentals—what the protocols do and how they work, how addresses and routing move data through the network, and how to set up a network connection. Beyond that, all the important networking services provided as part of Windows NT—including IIS, RRAS, DNS, WINS, and DHCP—are presented in detail. This book is the NT administrator's indispensable guide.

Contents include:

- Overview
- Delivering the data
- Network services
- Getting started
- Installing TCP/IP
- Using Dynamic Host Configuration Protocol
- Using Windows Internet Name Service
- Using Domain Name System
- Microsoft RRAS
- Internet Information Service
- Troubleshooting TCP/IP
- Network security
- Information resources
- Appendixes on the PPP scripting languages DNS resource records, Microsoft DHCP option support, and routing protocols

Craig Hunt is an expert on TCP/IP and the author of the bestseller *TCP/IP Network Administration.*

Robert Bruce Thompson is an expert on Windows NT and the author of several books on NT including the recently released *Windows NT Server 4.0 for NetWare Administrators.*

ISBN 1-56592-377-4
US $34.95
CAN $51.95
90000
9 781565 923775
6 36920 92377 0

Visit O'Reilly on the Web at *www.oreilly.com*

RepKover.

Printed on Recycled Paper

Google Exhibit No. 1022
Page 047

Google Exhibit No. 1063
Page 0911

Ex. 1022A

LIBRARY | CATALOG

1 of 1

BOOK

Windows NT TCP/IP network administration

Full Record MARC Tags

Personal name

Hunt, Craig.

Main title

Windows NT TCP/IP network administration / Craig Hunt and Robert Bruce Thompson.

Edition

1st ed.

Published/Created

Beijing ; Sebastopol, CA : O'Reilly, 1998.

Request this Item | LC Find It

Links

Links

Contributor biographical information http://www.loc.gov/catdir/enhancements/fy0915/2001274264-b.html
Publisher description http://www.loc.gov/catdir/enhancements/fy0915/2001274264-d.html
Table of contents only http://www.loc.gov/catdir/enhancements/

Request this Item | LC Find It

Google Exhibit No. 1063
Page 0913

More Information

LCCN Permalink	https://lccn.loc.gov/2001274264
Description	xvi, 487 p. : ill. ; 24 cm.
ISBN	1565923774
LC classification	TK5105.585 .H864 1998
Variant title	Windows NT TCP IP network administration
Related names	Thompson, Robert Bruce.
LC Subjects	Microsoft Windows NT. TCP/IP (Computer network protocol) Computer networks--Management.
Browse by shelf order	TK5105.585
Notes	"Help for Windows NT system administrators"--Cover. Includes index.
LCCN	2001274264
Dewey class no.	005.7/13769
Other system no.	(OCoLC)ocm40235371
Type of material	Book

Item Availability

CALL NUMBER	TK5105.585 .H864 1998 Copy 1
Request in	Jefferson or Adams Building Reading Rooms

Request this Item | LC Find It

Google Exhibit No. 1063
Page 0914

LIBRARY CATALOG

1 of 1

BOOK

Windows NT TCP/IP network administration

Full Record MARC Tags

000		01513cam a2200361 a 4500
001		12286344
005		20100707072139.0
008		010123s1998 cc a 001 0 eng d
906	__	|a 7 |b cbc |c copycat |d 2 |e ncip |f 20 |g y-gencatlg
925	0_	|a acquire |b 2 shelf copy |x policy default
955	__	|a to ASCD ps11 01-23-01; jf00 01-26-2001;jf05 (esc) 02/02/01; jf01 to sl 2-5-01; jf25 to Dewey 02-28-01; aa11 03-14-01
010	__	|a 2001274264
020	__	|a 1565923774
035	__	|a (OCoLC)ocm40235371
040	__	|a CUS |c CUS |d NMP |d DLC
042	__	|a pcc pycat
050	00	|a TK5105.585 |b .H864 1998
082	00	|a 005.7/13769 |2 21
100	1_	|a Hunt, Craig,
245	10	|a Windows NT TCP/IP network administration / |c Craig Hunt and Robert Bruce Thompson.
246	3_	|a Windows NT TCP IP network administration
250	__	|a 1st ed.
260	__	|a Beijing ; |a Sebastopol, CA : |b O'Reilly, |c 1998.
300	__	|a xvi, 487 p. : |b ill. ; |c 24 cm.
500	__	|a "Help for Windows NT system administrators"--Cover.

Google Exhibit No. 1063
Page 0915

500	__	|a Includes index.
650	_0	|a TCP/IP (Computer network protocol)
630	00	|a Microsoft Windows NT.
650	_0	|a Computer networks |x Management.
700	1_	|a Thompson, Robert Bruce.
856	42	|3 Contributor biographical information |u http://www.loc.gov/catdir/enhancements/fy0915/2001274264-b.html
856	42	|3 Publisher description |u http://www.loc.gov/catdir/enhancements/fy0915/2001274264-d.html
856	41	|3 Table of contents only |u http://www.loc.gov/catdir/enhancements/fy1009/2001274264-t.html

Request this Item | LC Find It

Item Availability

CALL NUMBER	TK5105.585 .H864 1998 Copy 1
Request in	Jefferson or Adams Building Reading Rooms
Status	Not Charged

Google Exhibit No. 1063
Page 0916

Ex. 1022B

Copyright
United States Copyright Office

Try the Copyright Public Records System (CPRS) pilot with enhanced search features and filters.

Help | Search | History | Titles | Start Over

Public Catalog

Copyright Catalog (1978 to present) at DC4
Search Request: Left Anchored Title = Windows NT TCP/IP Network Admin
Search Results: Displaying 5 of 5 entries

previous next

Labeled View

Windows NT TCP/IP network administration / Craig Hunt and Robert Bruce...

Type of Work: Text
Registration Number / Date: TX0005076938 / 1999-08-30
Title: Windows NT TCP/IP network administration / Craig Hunt and Robert Bruce Thompson.
Edition: 1st ed.
Imprint: Sebastopol, CA : O'Reilly, c1998.
Description: 487 p.
Copyright Claimant: O'Reilly & Associates
Date of Creation: 1998
Date of Publication: 1998-10-13
Names: Hunt, Craig
Thompson, Robert Bruce
O'Reilly & Associates

previous next

Save, Print and Email (Help Page)		
Select Download Format	Full Record	Format for Print/Save
Enter your email address:		Email

Help Search History Titles Start Over

Contact Us | Request Copies | Get a Search Estimate | Frequently Asked Questions (FAQs) about Copyright | Copyright Office Home Page | Library of Congress Home Page

Google Exhibit No. 1063
Page 0918

Volume 1

INTERNETWORKING with TCP/IP

PRINCIPLES, PROTOCOLS, AND ARCHITECTURES

FOURTH EDITION



DOUGLAS E. COMER

Google Exhibit No. 1023
Page 001

Google Exhibit No. 1063
Page 0919

Internetworking With TCP/IP

Vol I:

Principles, Protocols, and Architecture

Fourth Edition

DOUGLAS E. COMER

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

PRENTICE HALL
Upper Saddle River, New Jersey 07458

Google Exhibit No. 1023
Page 002

Google Exhibit No. 1063
Page 0920

Library of Congress Cataloging-in-Publication Data

Comer, Douglas
Internetworking with TCP/IP / Douglas E. Comer. — 4th ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-018380-6
1. Principles, protocols, and architecture. 2. Client/server computing. 3. Internetworking (Telecommunications) I. Title

Publisher: *Alan Apt*
Project Manager: *Ana Arias Terry*
Editorial Assistant: *Toni Holm*
Vice-president and director of production and manufacturing, ESM: *David W. Riccardi*
Vice-president and editorial director of ECS: *Marcia Horton*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Editorial/production supervision: *Irwin Zucker*
Art Director: *Heather Scott*
Assistant to Art Director: *John Christiana*
Manufacturing Buyer: *Pat Brown*
Marketing manager: *Danny Hoyt*



©2000, 1995 Prentice Hall
Prentice-Hall, Inc.
Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale. The publisher offers discounts on this book when ordered in bulk quantities.
For more information, contact Corporate Sales Department, Phone: 800-382-3419;
Fax: 201-236-7141; E-mail: corpsales@prenhall.com
Or write: Prentice Hall PTR, Corp. Sales Dept., One Lake Street, Upper Saddle River, NJ 07458.

UNIX is a registered trademark of UNIX System Laboratories, Incorporated
proNET-10 is a trademark of Proteon Corporation
LSI 11 is a trademark of Digital Equipment Corporation
Microsoft Windows is a trademark of Microsoft Corporation
EUI-64 is trademark of the Institute of Electrical and Electronics Engineers (IEEE)

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America
10 9 8 7 6 5 4 3 2

ISBN 0-13-018380-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte Ltd
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Google Exhibit No. 1023
Page 003

Google Exhibit No. 1063
Page 0921

Library of Congress Cataloging-in-Publication Data

Comer, Douglas
Internetworking with TCP/IP / Douglas E. Comer. -- 4th ed
p. cm.
Includes bibliographical references and index.
ISBN 0-13-018380-6
1. Principles, protocols, and architecture. 2. Client/server computing. 3. Internetworking (Telecommunications) I. Title

Publisher: *Alan Apt*
Project Manager: *Ana Arias Terry*
Editorial Assistant: *Toni Holm*
Vice-president and director of production and manufacturing, ESM: *David W. Riccardi*
Vice-president and editorial director of ECS: *Marcia Horton*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Editorial/production supervision: *Irwin Zucker*
Art Director: *Heather Scott*
Assistant to Art Director: *John Christiana*
Manufacturing Buyer: *Pat Brown*
Marketing manager: *Danny Hoyt*



©2000, 1995 Prentice Hall
Prentice-Hall, Inc.
Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale. The publisher offers discounts on this book when ordered in bulk quantities.
For more information, contact Corporate Sales Department, Phone: 800-382-3419;
Fax: 201-236-7141; E-mail: corpsales@prenhall.com
Or write: Prentice Hall PTR, Corp. Sales Dept., One Lake Street, Upper Saddle River, NJ 07458.

UNIX is a registered trademark of UNIX System Laboratories, Incorporated
proNET-10 is a trademark of Proteon Corporation
LSI 11 is a trademark of Digital Equipment Corporation
Microsoft Windows is a trademark of Microsoft Corporation
EUI-64 is trademark of the Institute of Electrical and Electronics Engineers (IEEE)

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America
10 9 8 7 6 5 4 3 2

ISBN 0-13-018380-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte Ltd
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Google Exhibit No. 1023
Page 004

Google Exhibit No. 1063
Page 0922

About The Author

Dr. Douglas Comer is an internationally recognized expert on TCP/IP protocols and the Internet. One of the researchers who contributed to the Internet as it was being formed in the late 1970s and 1980s, he was a member of the Internet Architecture Board, the group responsible for guiding the Internet's development. He was also chairman of the CSNET technical committee and a member of the CSNET executive committee.

Comer consults for companies on the design and implementation of networks, and gives professional seminars on TCP/IP and internetworking to both technical and nontechnical audiences around the world. His operating system, Xinu, and implementation of TCP/IP protocols are documented in his books, and used in commercial products.

Comer is a professor of computer science at Purdue University, where he teaches courses and does research on computer networking, internetworking, and operating systems. In addition to writing a series of best-selling technical books, he serves as the North American editor of the journal *Software — Practice and Experience*. Comer is a Fellow of the ACM.

Additional information can be found at:

www.cs.purdue.edu/people/comer

Google Exhibit No. 1023
Page 005

Google Exhibit No. 1063
Page 0923

Contents

Foreword xxiii

Preface xxvii

Chapter 1 Introduction And Overview 1

1.1 The Motivation For Internetworking 1
1.2 The TCP/IP Internet 2
1.3 Internet Services 3
1.4 History And Scope Of The Internet 6
1.5 The Internet Architecture Board 8
1.6 The IAB Reorganization 9
1.7 The Internet Society 11
1.8 Internet Request For Comments 11
1.9 Internet Protocols And Standardization 12
1.10 Future Growth And Technology 12
1.11 Organization Of The Text 13
1.12 Summary 14

Chapter 2 Review Of Underlying Network Technologies 17

2.1 Introduction 17
2.2 Two Approaches To Network Communication 18
2.3 Wide Area And Local Area Networks 19
2.4 Ethernet Technology 20
2.5 Fiber Distributed Data Interconnect (FDDI) 33
2.6 Asynchronous Transfer Mode 37
2.7 WAN Technologies: ARPANET 38
2.8 National Science Foundation Networking 40

Google Exhibit No. 1023
Page 006

Google Exhibit No. 1063
Page 0924

2.9 *ANSNET* 44
2.10 *A Very High Speed Backbone (vBNS)* 45
2.11 *Other Technologies Over Which TCP/IP Has Been Used* 46
2.12 *Summary And Conclusion* 50

Chapter 3 Internetworking Concept And Architectural Model 53

3.1 *Introduction* 53
3.2 *Application-Level Interconnection* 53
3.3 *Network-Level Interconnection* 54
3.4 *Properties Of The Internet* 55
3.5 *Internet Architecture* 56
3.6 *Interconnection Through IP Routers* 56
3.7 *The User's View* 58
3.8 *All Networks Are Equal* 58
3.9 *The Unanswered Questions* 59
3.10 *Summary* 60

Chapter 4 Classful Internet Addresses 63

4.1 *Introduction* 63
4.2 *Universal Identifiers* 63
4.3 *The Original Classful Addressing Scheme* 64
4.4 *Addresses Specify Network Connections* 65
4.5 *Network And Directed Broadcast Addresses* 65
4.6 *Limited Broadcast* 66
4.7 *Interpreting Zero To Mean "This"* 67
4.8 *Subnet And Supernet Extensions* 67
4.9 *IP Multicast Addresses* 68
4.10 *Weaknesses In Internet Addressing* 68
4.11 *Dotted Decimal Notation* 69
4.12 *Loopback Address* 70
4.13 *Summary Of Special Address Conventions* 70
4.14 *Internet Addressing Authority* 71
4.15 *Reserved Address Prefixes* 72
4.16 *An Example* 72
4.17 *Network Byte Order* 74
4.18 *Summary* 75

Google Exhibit No. 1023
Page 007

Google Exhibit No. 1063
Page 0925

Chapter 5 Mapping Internet Addresses To Physical Addresses (ARP) 77

5.1 *Introduction* 77
5.2 *The Address Resolution Problem* 77
5.3 *Two Types Of Physical Addresses* 78
5.4 *Resolution Through Direct Mapping* 78
5.5 *Resolution Through Dynamic Binding* 79
5.6 *The Address Resolution Cache* 80
5.7 *ARP Cache Timeout* 81
5.8 *ARP Refinements* 82
5.9 *Relationship Of ARP To Other Protocols* 82
5.10 *ARP Implementation* 82
5.11 *ARP Encapsulation And Identification* 84
5.12 *ARP Protocol Format* 84
5.13 *Summary* 86

Chapter 6 Determining An Internet Address At Startup (RARP) 89

6.1 *Introduction* 89
6.2 *Reverse Address Resolution Protocol (RARP)* 90
6.3 *Timing RARP Transactions* 92
6.4 *Primary And Backup RARP Servers* 92
6.5 *Summary* 93

Chapter 7 Internet Protocol: Connectionless Datagram Delivery 95

7.1 *Introduction* 95
7.2 *A Virtual Network* 95
7.3 *Internet Architecture And Philosophy* 96
7.4 *The Conceptual Service Organization* 96
7.5 *Connectionless Delivery System* 97
7.6 *Purpose Of The Internet Protocol* 97
7.7 *The Internet Datagram* 97
7.8 *Internet Datagram Options* 107
7.9 *Summary* 113

Chapter 8 Internet Protocol: Routing IP Datagrams 115

8.1 *Introduction* 115
8.2 *Routing In An Internet* 115
8.3 *Direct And Indirect Delivery* 117

Google Exhibit No. 1023
Page 008

Google Exhibit No. 1063
Page 0926

8.4 *Table-Driven IP Routing* 119
8.5 *Next-Hop Routing* 119
8.6 *Default Routes* 121
8.7 *Host-Specific Routes* 121
8.8 *The IP Routing Algorithm* 121
8.9 *Routing With IP Addresses* 122
8.10 *Handling Incoming Datagrams* 124
8.11 *Establishing Routing Tables* 125
8.12 *Summary* 125

Chapter 9 Internet Protocol: Error And Control Messages (ICMP) 129

9.1 *Introduction* 129
9.2 *The Internet Control Message Protocol* 129
9.3 *Error Reporting vs. Error Correction* 130
9.4 *ICMP Message Delivery* 131
9.5 *ICMP Message Format* 132
9.6 *Testing Destination Reachability And Status (Ping)* 133
9.7 *Echo Request And Reply Message Format* 134
9.8 *Reports Of Unreachable Destinations* 134
9.9 *Congestion And Datagram Flow Control* 136
9.10 *Source Quench Format* 136
9.11 *Route Change Requests From Routers* 137
9.12 *Detecting Circular Or Excessively Long Routes* 139
9.13 *Reporting Other Problems* 140
9.14 *Clock Synchronization And Transit Time Estimation* 140
9.15 *Information Request And Reply Messages* 142
9.16 *Obtaining A Subnet Mask* 142
9.17 *Router Discovery* 143
9.18 *Router Solicitation* 144
9.19 *Summary* 145

Chapter 10 Classless And Subnet Address Extensions (CIDR) 147

10.1 *Introduction* 147
10.2 *Review Of Relevant Facts* 147
10.3 *Minimizing Network Numbers* 148
10.4 *Transparent Routers* 149
10.5 *Proxy ARP* 150
10.6 *Subnet Addressing* 152
10.7 *Flexibility In Subnet Address Assignment* 154
10.8 *Variable-Length Subnets* 155

Google Exhibit No. 1023
Page 009

Google Exhibit No. 1063
Page 0927

10.9 Implementation Of Subnets With Masks 156
10.10 Subnet Mask Representation 157
10.11 Routing In The Presence Of Subnets 158
10.12 The Subnet Routing Algorithm 159
10.13 A Unified Routing Algorithm 160
10.14 Maintenance Of Subnet Masks 161
10.15 Broadcasting To Subnets 161
10.16 Anonymous Point-To-Point Networks 162
10.17 Classless Addressing (Supernetting) 164
10.18 The Effect Of Supernetting On Routing 165
10.19 CIDR Address Blocks And Bit Masks 165
10.20 Address Blocks And CIDR Notation 166
10.21 A Classless Addressing Example 167
10.22 Data Structures And Algorithms For Classless Lookup 167
10.23 Longest-Match Routing And Mixtures Of Route Types 170
10.24 CIDR Blocks Reserved For Private Networks 172
10.25 Summary 173

Chapter 11 Protocol Layering **177**

11.1 Introduction 177
11.2 The Need For Multiple Protocols 177
11.3 The Conceptual Layers Of Protocol Software 178
11.4 Functionality Of The Layers 181
11.5 X.25 And Its Relation To The ISO Model 182
11.6 Differences Between ISO And Internet Layering 185
11.7 The Protocol Layering Principle 187
11.8 Layering In The Presence Of Network Substructure 189
11.9 Two Important Boundaries In The TCP/IP Model 191
11.10 The Disadvantage Of Layering 192
11.11 The Basic Idea Behind Multiplexing And Demultiplexing 192
11.12 Summary 194

Chapter 12 User Datagram Protocol (UDP) **197**

12.1 Introduction 197
12.2 Identifying The Ultimate Destination 197
12.3 The User Datagram Protocol 198
12.4 Format Of UDP Messages 199
12.5 UDP Pseudo-Header 200
12.6 UDP Encapsulation And Protocol Layering 201
12.7 Layering And The UDP Checksum Computation 203

Google Exhibit No. 1023
Page 010

Google Exhibit No. 1063
Page 0928

12.8 UDP Multiplexing, Demultiplexing, And Ports 203
12.9 Reserved And Available UDP Port Numbers 204
12.10 Summary 206

Chapter 13 Reliable Stream Transport Service (TCP) **209**

13.1 Introduction 209
13.2 The Need For Stream Delivery 209
13.3 Properties Of The Reliable Delivery Service 210
13.4 Providing Reliability 211
13.5 The Idea Behind Sliding Windows 213
13.6 The Transmission Control Protocol 215
13.7 Ports, Connections, And Endpoints 216
13.8 Passive And Active Opens 218
13.9 Segments, Streams, And Sequence Numbers 219
13.10 Variable Window Size And Flow Control 220
13.11 TCP Segment Format 221
13.12 Out Of Band Data 222
13.13 Maximum Segment Size Option 223
13.14 TCP Checksum Computation 224
13.15 Acknowledgements And Retransmission 225
13.16 Timeout And Retransmission 226
13.17 Accurate Measurement Of Round Trip Samples 228
13.18 Karn's Algorithm And Timer Backoff 229
13.19 Responding To High Variance In Delay 230
13.20 Response To Congestion 232
13.21 Congestion, Tail Drop, And TCP 234
13.22 Random Early Discard (RED) 235
13.23 Establishing A TCP Connection 237
13.24 Initial Sequence Numbers 239
13.25 Closing a TCP Connection 239
13.26 TCP Connection Reset 241
13.27 TCP State Machine 241
13.28 Forcing Data Delivery 243
13.29 Reserved TCP Port Numbers 243
13.30 TCP Performance 243
13.31 Silly Window Syndrome And Small Packets 245
13.32 Avoiding Silly Window Syndrome 246
13.33 Summary 249

Google Exhibit No. 1023
Page 011

Google Exhibit No. 1063
Page 0929

Chapter 14 Routing: Cores, Peers, And Algorithms 253

14.1 Introduction 253
14.2 The Origin Of Routing Tables 254
14.3 Routing With Partial Information 255
14.4 Original Internet Architecture And Cores 256
14.5 Core Routers 257
14.6 Beyond The Core Architecture To Peer Backbones 260
14.7 Automatic Route Propagation 262
14.8 Distance Vector (Bellman-Ford) Routing 262
14.9 Gateway-To-Gateway Protocol (GGP) 264
14.10 Distance Factoring 265
14.11 Reliability And Routing Protocols 265
14.12 Link-State (SPF) Routing 266
14.13 Summary 267

Chapter 15 Routing: Exterior Gateway Protocols And Autonomous Systems (BGP) 269

15.1 Introduction 269
15.2 Adding Complexity To The Architectural Model 269
15.3 Determining A Practical Limit On Group Size 270
15.4 A Fundamental Idea: Extra Hops 271
15.5 Hidden Networks 273
15.6 Autonomous System Concept 274
15.7 From A Core To Independent Autonomous Systems 275
15.8 An Exterior Gateway Protocol 276
15.9 BGP Characteristics 277
15.10 BGP Functionality And Message Types 278
15.11 BGP Message Header 278
15.12 BGP OPEN Message 279
15.13 BGP UPDATE Message 280
15.14 Compressed Mask-Address Pairs 281
15.15 BGP Path Attributes 282
15.16 BGP KEEPALIVE Message 283
15.17 Information From The Receiver's Perspective 284
15.18 The Key Restriction Of Exterior Gateway Protocols 285
15.19 The Internet Routing Arbiter System 287
15.20 BGP NOTIFICATION Message 288
15.21 Decentralization Of Internet Architecture 289
15.22 Summary 290

Google Exhibit No. 1023
Page 012

Google Exhibit No. 1063
Page 0930

Chapter 16 Routing: In An Autonomous System (RIP, OSPF, HELLO) 293

16.1 Introduction 293
16.2 Static Vs. Dynamic Interior Routes 293
16.3 Routing Information Protocol (RIP) 296
16.4 The Hello Protocol 305
16.5 Delay Metrics And Oscillation 305
16.6 Combining RIP, Hello, And BGP 307
16.7 Inter-Autonomous System Routing 307
16.8 Gated: Inter-Autonomous System Communication 308
16.9 The Open SPF Protocol (OSPF) 308
16.10 Routing With Partial Information 315
16.11 Summary 315

Chapter 17 Internet Multicasting 319

17.1 Introduction 319
17.2 Hardware Broadcast 319
17.3 Hardware Origins Of Multicast 320
17.4 Ethernet Multicast 321
17.5 IP Multicast 321
17.6 The Conceptual Pieces 322
17.7 IP Multicast Addresses 323
17.8 Multicast Address Semantics 325
17.9 Mapping IP Multicast To Ethernet Multicast 325
17.10 Hosts And Multicast Delivery 326
17.11 Multicast Scope 326
17.12 Extending Host Software To Handle Multicasting 327
17.13 Internet Group Management Protocol 328
17.14 IGMP Implementation 328
17.15 Group Membership State Transitions 329
17.16 IGMP Message Format 331
17.17 Multicast Forwarding And Routing Information 332
17.18 Basic Multicast Routing Paradigms 334
17.19 Consequences Of TRPF 335
17.20 Multicast Trees 337
17.21 The Essence Of Multicast Routing 338
17.22 Reverse Path Multicasting 338
17.23 Distance Vector Multicast Routing Protocol 339
17.24 The Mrouted Program 340
17.25 Alternative Protocols 343
17.26 Core Based Trees (CBT) 343
17.27 Protocol Independent Multicast (PIM) 344

Google Exhibit No. 1023
Page 013

Google Exhibit No. 1063
Page 0931

17.28 Multicast Extensions To OSPF (MOSPF) 347
17.29 Reliable Multicast And ACK Implosions 347
17.30 Summary 349

Chapter 18 TCP/IP Over ATM Networks **353**

18.1 Introduction 353
18.2 ATM Hardware 354
18.3 Large ATM Networks 354
18.4 The Logical View Of An ATM Network 355
18.5 The Two ATM Connection Paradigms 356
18.6 Paths, Circuits, And Identifiers 357
18.7 ATM Cell Transport 358
18.8 ATM Adaptation Layers 358
18.9 ATM Adaptation Layer 5 360
18.10 AAL5 Convergence, Segmentation, And Reassembly 361
18.11 Datagram Encapsulation And IP MTU Size 361
18.12 Packet Type And Multiplexing 362
18.13 IP Address Binding In An ATM Network 363
18.14 Logical IP Subnet Concept 364
18.15 Connection Management 365
18.16 Address Binding Within An LIS 366
18.17 ATMARP Packet Format 366
18.18 Using ATMARP Packets To Determine An Address 369
18.19 Obtaining Entries For A Server Database 370
18.20 Timing Out ATMARP Information In A Server 370
18.21 Timing Out ATMARP Information In A Host Or Router 371
18.22 IP Switching Technologies 371
18.23 Switch Operation 372
18.24 Optimized IP Forwarding 372
18.25 Classification, Flows, And Higher Layer Switching 373
18.26 Applicability Of Switching Technology 374
18.27 Summary 374

Chapter 19 Mobile IP **377**

19.1 Introduction 377
19.2 Mobility, Routing, and Addressing 377
19.3 Mobile IP Characteristics 378
19.4 Overview Of Mobile IP Operation 378
19.5 Mobile Addressing Details 379
19.6 Foreign Agent Discovery 380

Google Exhibit No. 1023
Page 014

Google Exhibit No. 1063
Page 0932

19.7 *Agent Registration* 381
19.8 *Registration Message Format* 381
19.9 *Communication With A Foreign Agent* 383
19.10 *Datagram Transmission And Reception* 383
19.11 *The Two-Crossing Problem* 384
19.12 *Communication With Computers On the Home Network* 385
19.13 *Summary* 386

Chapter 20 Private Network Interconnection (NAT, VPN) 389

20.1 *Introduction* 389
20.2 *Private And Hybrid Networks* 389
20.3 *A Virtual Private Network (VPN)* 390
20.4 *VPN Addressing And Routing* 392
20.5 *A VPN With Private Addresses* 393
20.6 *Network Address Translation (NAT)* 394
20.7 *NAT Translation Table Creation* 395
20.8 *Multi-Address NAT* 396
20.9 *Port-Mapped NAT* 396
20.10 *Interaction Between NAT And ICMP* 398
20.11 *Interaction Between NAT And Applications* 398
20.12 *Conceptual Address Domains* 399
20.13 *Slirp And Masquerade* 399
20.14 *Summary* 400

Chapter 21 Client-Server Model Of Interaction 403

21.1 *Introduction* 403
21.2 *The Client-Server Model* 403
21.3 *A Simple Example: UDP Echo Server* 404
21.4 *Time And Date Service* 406
21.5 *The Complexity of Servers* 407
21.6 *RARP Server* 408
21.7 *Alternatives To The Client-Server Model* 409
21.8 *Summary* 410

Chapter 22 The Socket Interface 413

22.1 *Introduction* 413
22.2 *The UNIX I/O Paradigm And Network I/O* 414
22.3 *Adding Network I/O to UNIX* 414

Google Exhibit No. 1023
Page 015

Google Exhibit No. 1063
Page 0933

22.4 *The Socket Abstraction* 415
22.5 *Creating A Socket* 415
22.6 *Socket Inheritance And Termination* 416
22.7 *Specifying A Local Address* 417
22.8 *Connecting Sockets To Destination Addresses* 418
22.9 *Sending Data Through A Socket* 419
22.10 *Receiving Data Through A Socket* 421
22.11 *Obtaining Local And Remote Socket Addresses* 422
22.12 *Obtaining And Setting Socket Options* 423
22.13 *Specifying A Queue Length For A Server* 424
22.14 *How A Server Accepts Connections* 424
22.15 *Servers That Handle Multiple Services* 425
22.16 *Obtaining And Setting Host Names* 426
22.17 *Obtaining And Setting The Internal Host Domain* 427
22.18 *Socket Library Calls* 427
22.19 *Network Byte Order Conversion Routines* 428
22.20 *IP Address Manipulation Routines* 429
22.21 *Accessing The Domain Name System* 431
22.22 *Obtaining Information About Hosts* 432
22.23 *Obtaining Information About Networks* 433
22.24 *Obtaining Information About Protocols* 434
22.25 *Obtaining Information About Network Services* 434
22.26 *An Example Client* 435
22.27 *An Example Server* 437
22.28 *Summary* 440

Chapter 23 Bootstrap And Autoconfiguration (BOOTP, DHCP) 443

23.1 *Introduction* 443
23.2 *The Need For An Alternative To RARP* 444
23.3 *Using IP To Determine An IP Address* 444
23.4 *The BOOTP Retransmission Policy* 445
23.5 *The BOOTP Message Format* 446
23.6 *The Two-Step Bootstrap Procedure* 447
23.7 *Vendor-Specific Field* 448
23.8 *The Need For Dynamic Configuration* 448
23.9 *Dynamic Host Configuration* 450
23.10 *Dynamic IP Address Assignment* 450
23.11 *Obtaining Multiple Addresses* 451
23.12 *Address Acquisition States* 452
23.13 *Early Lease Termination* 452
23.14 *Lease Renewal States* 454
23.15 *DHCP Message Format* 455

Google Exhibit No. 1023
Page 016

Google Exhibit No. 1063
Page 0934

23.16 DHCP Options And Message Type 456
23.17 Option Overload 457
23.18 DHCP And Domain Names 457
23.19 Summary 458

Chapter 24 The Domain Name System (DNS) 461

24.1 Introduction 461
24.2 Names For Machines 462
24.3 Flat Namespace 462
24.4 Hierarchical Names 463
24.5 Delegation Of Authority For Names 464
24.6 Subset Authority 464
24.7 Internet Domain Names 465
24.8 Official And Unofficial Internet Domain Names 466
24.9 Named Items And Syntax Of Names 468
24.10 Mapping Domain Names To Addresses 469
24.11 Domain Name Resolution 471
24.12 Efficient Translation 472
24.13 Caching: The Key To Efficiency 473
24.14 Domain Server Message Format 474
24.15 Compressed Name Format 477
24.16 Abbreviation Of Domain Names 477
24.17 Inverse Mappings 478
24.18 Pointer Queries 479
24.19 Object Types And Resource Record Contents 479
24.20 Obtaining Authority For A Subdomain 480
24.21 Summary 481

Chapter 25 Applications: Remote Login (TELNET, Rlogin) 485

25.1 Introduction 485
25.2 Remote Interactive Computing 485
25.3 TELNET Protocol 486
25.4 Accommodating Heterogeneity 488
25.5 Passing Commands That Control The Remote Side 490
25.6 Forcing The Server To Read A Control Function 492
25.7 TELNET Options 492
25.8 TELNET Option Negotiation 493
25.9 Rlogin (BSD UNIX) 494
25.10 Summary 495

Google Exhibit No. 1023
Page 017

Google Exhibit No. 1063
Page 0935

Chapter 26 Applications: File Transfer And Access (FTP, TFTP, NFS) 497

26.1 Introduction 497
26.2 File Access And Transfer 497
26.3 On-line Shared Access 498
26.4 Sharing By File Transfer 499
26.5 FTP: The Major TCP/IP File Transfer Protocol 499
26.6 FTP Features 500
26.7 FTP Process Model 500
26.8 TCP Port Number Assignment 502
26.9 The User's View Of FTP 502
26.10 An Example Anonymous FTP Session 504
26.11 TFTP 505
26.12 NFS 507
26.13 NFS Implementation 507
26.14 Remote Procedure Call (RPC) 508
26.15 Summary 509

Chapter 27 Applications: Electronic Mail (SMTP, POP, IMAP, MIME) 511

27.1 Introduction 511
27.2 Electronic Mail 511
27.3 Mailbox Names And Aliases 513
27.4 Alias Expansion And Mail Forwarding 513
27.5 The Relationship Of Internetworking And Mail 514
27.6 TCP/IP Standards For Electronic Mail Service 516
27.7 Electronic Mail Addresses 516
27.8 Pseudo Domain Addresses 518
27.9 Simple Mail Transfer Protocol (SMTP) 518
27.10 Mail Retrieval And Mailbox Manipulation Protocols 521
27.11 The MIME Extension For Non-ASCII Data 522
27.12 MIME Multipart Messages 523
27.13 Summary 524

Chapter 28 Applications: World Wide Web (HTTP) 527

28.1 Introduction 527
28.2 Importance Of The Web 527
28.3 Architectural Components 528
28.4 Uniform Resource Locators 528
28.5 An Example Document 529
28.6 Hypertext Transfer Protocol 530

Google Exhibit No. 1023
Page 018

Google Exhibit No. 1063
Page 0936

28.7 HTTP GET Request 530
28.8 Error Messages 531
28.9 Persistent Connections And Lengths 532
28.10 Data Length And Program Output 532
28.11 Length Encoding And Headers 533
28.12 Negotiation 534
28.13 Conditional Requests 535
28.14 Support For Proxy Servers 535
28.15 Caching 536
28.16 Summary 537

Chapter 29 Applications: Voice And Video Over IP (RTP) 539

29.1 Introduction 539
29.2 Audio Clips And Encoding Standards 539
29.3 Audio And Video Transmission And Reproduction 540
29.4 Jitter And Playback Delay 541
29.5 Real-Time Transport Protocol (RTP) 542
29.6 Streams, Mixing, And Multicasting 543
29.7 RTP Encapsulation 544
29.8 RTP Control Protocol (RTCP) 544
29.9 RTCP Operation 545
29.10 IP Telephony And Signaling 546
29.11 Resource Reservation And Quality Of Service 548
29.12 QoS, Utilization, And Capacity 549
29.13 RSVP 549
29.14 COPS 550
29.15 Summary 551

Chapter 30 Applications: Internet Management (SNMP) 553

30.1 Introduction 553
30.2 The Level Of Management Protocols 553
30.3 Architectural Model 554
30.4 Protocol Framework 556
30.5 Examples of MIB Variables 557
30.6 The Structure Of Management Information 558
30.7 Formal Definitions Using ASN.1 559
30.8 Structure And Representation Of MIB Object Names 559
30.9 Simple Network Management Protocol 564
30.10 SNMP Message Format 566
30.11 Example Encoded SNMP Message 569

Google Exhibit No. 1023
Page 019

Google Exhibit No. 1063
Page 0937

30.12 New Features In SNMPv3 572
30.13 Summary 572

Chapter 31 Summary Of Protocol Dependencies **575**

31.1 Introduction 575
31.2 Protocol Dependencies 575
31.3 The Hourglass Model 577
31.4 Application Program Access 578
31.5 Summary 579

Chapter 32 Internet Security And Firewall Design (IPsec) **581**

32.1 Introduction 581
32.2 Protecting Resources 582
32.3 Information Policy 583
32.4 Internet Security 583
32.5 IP Security (IPsec) 584
32.6 IPsec Authentication Header 584
32.7 Security Association 585
32.8 IPsec Encapsulating Security Payload 586
32.9 Authentication And Mutable Header Fields 587
32.10 IPsec Tunneling 588
32.11 Required Security Algorithms 588
32.12 Secure Sockets 589
32.13 Firewalls And Internet Access 589
32.14 Multiple Connections And Weakest Links 589
32.15 Firewall Implementation 590
32.16 Packet-Level Filters 590
32.17 Security And Packet Filter Specification 591
32.18 The Consequence Of Restricted Access For Clients 592
32.19 Proxy Access Through A Firewall 592
32.20 The Details Of Firewall Architecture 593
32.21 Stub Network 594
32.22 An Alternative Firewall Implementation 595
32.23 Monitoring And Logging 596
32.24 Summary 596

Google Exhibit No. 1023
Page 020

Google Exhibit No. 1063
Page 0938

Chapter 33 The Future Of TCP/IP (IPv6) 599

33.1 Introduction 599
33.2 Why Change? 600
33.3 New Policies 600
33.4 Motivation For Changing IPv4 600
33.5 The Road To A New Version Of IP 601
33.6 The Name Of The Next IP 602
33.7 Features Of IPv6 602
33.8 General Form Of An IPv6 Datagram 603
33.9 IPv6 Base Header Format 603
33.10 IPv6 Extension Headers 605
33.11 Parsing An IPv6 Datagram 606
33.12 IPv6 Fragmentation And Reassembly 607
33.13 The Consequence Of End-To-End Fragmentation 607
33.14 IPv6 Source Routing 608
33.15 IPv6 Options 609
33.16 Size Of The IPv6 Address Space 610
33.17 IPv6 Colon Hexadecimal Notation 610
33.18 Three Basic IPv6 Address Types 612
33.19 The Duality Of Broadcast And Multicast 612
33.20 An Engineering Choice And Simulated Broadcast 613
33.21 Proposed IPv6 Address Space Assignment 613
33.22 Embedded IPv4 Addresses And Transition 614
33.23 Unspecified And Loopback Addresses 616
33.24 Unicast Address Hierarchy 616
33.25 Aggregatable Global Unicast Address Structure 617
33.26 Interface Identifiers 618
33.27 Additional Hierarchy 619
33.28 Local Addresses 619
33.29 Autoconfiguration And Renumbering 620
33.30 Summary 620

Appendix 1 A Guide To RFCs 623

Appendix 2 Glossary Of Internetworking Terms And Abbreviations 673

Bibliography 721

Index 729

Google Exhibit No. 1023
Page 021

Google Exhibit No. 1063
Page 0939

28

Applications: World Wide Web (HTTP)

28.1 Introduction

This chapter continues the discussion of applications that use TCP/IP technology by focusing on the application that has had the most impact: the *World Wide Web* (*WWW*). After a brief overview of concepts, the chapter examines the primary protocol used to transfer a Web page from a server to a Web browser. The discussion covers caching as well as the basic transfer mechanism.

28.2 Importance Of The Web

During the early history of the Internet, FTP data transfers accounted for approximately one third of Internet traffic, more than any other application. From its inception in the early 1990s, however, the Web had a much higher growth rate. By 1995, Web traffic overtook FTP to become the largest consumer of Internet backbone bandwidth, and has remained the leader ever since. By 2000, Web traffic completely overshadowed other applications.

Although traffic is easy to measure and cite, the impact of the Web cannot be understood from such statistics. More people know about and use the Web than any other Internet application. Most companies have Web sites and on-line catalogs; references to the Web appear in advertising. In fact, for many users, the Internet and the Web are indistinguishable.

Google Exhibit No. 1023
Page 022

Google Exhibit No. 1063
Page 0940

28.3 Architectural Components

Conceptually, the Web consists of a large set of documents, called *Web pages*, that are accessible over the Internet. Each Web page is classified as a *hypermedia* document. The suffix *media* is used to indicate that a document can contain items other than text (e.g., graphics images); the prefix *hyper* is used because a document can contain *selectable links* that refer to other, related documents.

Two main building blocks are used to implement the Web on top of the global Internet. A *Web browser* consists of an application program that a user invokes to access and display a Web page. The browser becomes a client that contacts the appropriate *Web server* to obtain a copy of the specified page. Because a given server can manage more than one Web page, a browser must specify the exact page when making a request.

The data representation standard used for a Web page depends on its contents. For example, standard graphics representations such as *Graphics Interchange Format* (*GIF*) or *Joint Picture Encoding Group* (*JPEG*) can be used for a page that contains a single graphics image. Pages that contain a mixture of text and other items are represented using *HyperText Markup Language* (*HTML*). An HTML document consists of a file that contains text along with embedded commands, called *tags*, that give guidelines for display. A tag is enclosed in less-than and greater-than symbols; some tags come in pairs that apply to all items between the pair. For example, the two commands *<CENTER>* and *</CENTER>* cause items between them to be centered in the browser's window.

28.4 Uniform Resource Locators

Each Web page is assigned a unique name that is used to identify it. The name, which is called a *Uniform Resource Locator* (*URL*)†, begins with a specification of the *scheme* used to access the item. In effect, the scheme specifies the transfer protocol; the format of the remainder of the URL depends on the scheme. For example, a URL that follows the *http scheme* has the following form‡:

http: // *hostname* [: *port*] / *path* [; *parameters*] [? *query*]

where brackets denote an optional item. For now, it is sufficient to understand that the *hostname* string specifies the domain name or IP address of the computer on which the server for the item operates, *:port* is an optional protocol port number needed only in cases where the server does not use the well-known port (*80*), *path* is a string that identifies one particular document on the server, *;parameters* is an optional string that specifies additional parameters supplied by the client, and *?query* is an optional string used when the browser sends a question. A user is unlikely to ever see or use the optional parts directly. Instead, URLs that a user enters contain only a *hostname* and *path*. For example, the URL:

†A URL is a specific type of the more general *Uniform Resource Identifier* (*URI*).

‡Some of the literature refers to the initial string, *http:*, as a *pragma*.

Google Exhibit No. 1023
Page 023

Google Exhibit No. 1063
Page 0941

http://www.cs.purdue.edu/people/comer/

specifies the author's Web page. The server operates on computer *www.cs.purdue.edu*, and the document is named */people/comer/*.

The protocol standards distinguish between the *absolute* form of a URL illustrated above, and a *relative* form. A relative URL, which is seldom seen by a user, is only meaningful when the server has already been determined. Relative URLs are useful once communication has been established with a specific server. For example, when communicating with server *www.cs.purdue.edu*, only the string */people/comer/* is needed to specify the document named by the absolute URL above. We can summarize.

> *Each Web page is assigned a unique identifier known as a Uniform Resource Locator (URL). The absolute form of a URL contains a full specification; a relative form that omits the address of the server is only useful when the server is implicitly known.*

28.5 An Example Document

In principle, Web access is straightforward. All access originates with a URL — a user either enters a URL via the keyboard or selects an item which provides the browser with a URL. The browser parses the URL, extracts the information, and uses it to obtain a copy of the requested page. Because the format of the URL depends on the scheme, the browser begins by extracting the scheme specification, and then uses the scheme to determine how to parse the rest of the URL.

An example will illustrate how a URL is produced from a *selectable link* in a document. In fact, a document contains a pair of values for each link: an item to be displayed on the screen and a URL to follow if the user selects the item. In HTML, the pair of tags <*A*> and </*A*> are known as an *anchor*. The anchor defines a link; a URL is added to the first tag, and items to be displayed are placed between the two tags. The browser stores the URL internally, and follows it when the user selects the link. For example, the following HTML document contains a selectable link:

```
<HTML>
  The author of this text is
  <A HREF="http://www.cs.purdue.edu/people/comer">
  Douglas Comer.</A>
</HTML>
```

When the document is displayed, a single line of text appears on the screen:

The author of this text is Douglas Comer.

Google Exhibit No. 1023
Page 024

Google Exhibit No. 1063
Page 0942

The browser underlines the phrase *Douglas Comer* to indicate that it corresponds to a selectable link. Internally, of course, the browser stores the URL from the <A> tag, which it follows when the user selects the link.

28.6 Hypertext Transfer Protocol

The protocol used for communication between a browser and a Web server or between intermediate machines and Web servers is known as the *HyperText Transfer Protocol (HTTP)*. HTTP has the following set of characteristics:

Application Level. HTTP operates at the application level. It assumes a reliable, connection-oriented transport protocol such as TCP, but does not provide reliability or retransmission itself.

Request/Response. Once a transport session has been established, one side (usually a browser) must send an HTTP request to which the other side responds.

Stateless. Each HTTP request is self-contained; the server does not keep a history of previous requests or previous sessions.

Bi-Directional Transfer. In most cases, a browser requests a Web page, and the server transfers a copy to the browser. HTTP also allows transfer from a browser to a server (e.g., when a user submits a so-called "form").

Capability Negotiation. HTTP allows browsers and servers to negotiate details such as the character set to be used during transfers. A sender can specify the capabilities it offers and a receiver can specify the capabilities it accepts.

Support For Caching. To improve response time, a browser caches a copy of each Web page it retrieves. If a user requests a page again, HTTP allows the browser to interrogate the server to determine whether the contents of the page has changed since the copy was cached.

Support For Intermediaries. HTTP allows a machine along the path between a browser and a server to act as a *proxy server* that caches Web pages and answers a browser's request from its cache.

28.7 HTTP GET Request

In the simplest case, a browser contacts a Web server directly to obtain a page. The browser begins with a URL, extracts the hostname section, uses DNS to map the name into an equivalent IP address, and uses the IP address to form a TCP connection

Google Exhibit No. 1023
Page 025

Google Exhibit No. 1063
Page 0943

Volume 1
COMER
Fourth Edition
INTERNETWORKING WITH
TCP/IP
Principles, Protocols, and Architectures
PRENTICE HALL

Fourth Edition

INTERNETWORKING with TCP/IP VOLUME I

PRINCIPLES, PROTOCOLS, AND ARCHITECTURES

DOUGLAS E. COMER

"Comer continues to prove himself the Baedeker of the Internet Protocols with this fine 4th edition."
-Vint Cerf, Senior Vice President, MCI WorldCom, "Father of the Internet"

"Doug's deep insight into the Internet, coupled with his talent for teaching, yields an updated classic that once again leads the pack." *-Paul Mockapetris, Inventor of the Domain Name System*

"There are many TCP/IP books on the shelves today, but Doug Comer's "Internetworking with TCP/IP" is the one that comes *off* the shelf for accessible and authoritative answers to questions about Internet technology."
-Lyman Chapin, Chief Scientist, BBN Technologies

The TCP/IP "bible"—now completely updated to reflect all the latest technologies! Leading Internet expert Douglas Comer has updated his classic, superb introduction to TCP/IP-based internetworking that is renowned for its clarity and accessibility in explaining internetworking and routing. Discover how the basic TCP/IP technology has survived and evolved over two decades of exponential growth, and understand the TCP/IP protocols and technical advances. This edition explains emerging technologies such as Mobile IP, Virtual Private Networks, resource reservation with RSVP, and IPv6. Comer reveals how to master TCP/IP and how the Internet works. The reader is required to have a modest background in the fundamentals of computer systems, but does not need sophisticated mathematics. As with previous editions, this edition provides an introduction to physical networks and then shows how they are combined to form an internet. It states design principles clearly, and discusses motivations and consequences.

THIS NEW EDITION OF VOLUME 1:

- Explains how voice and video are sent over IP internets and how IP Telephony operates.
- Describes Mobile IP (a technology that allows a computer to move from one network to another without changing its IP address).
- Discusses IP security and the security standard, IPsec.
- Revises the discussion of IPv6 to incorporate the latest changes.
- Shows how to interconnect private intranets and the global Internet using Virtual Private Network (VPN) and Network Address Translation (NAT) technologies.
- Expands the description of IP multicasting to cover multicast routing protocols.
- Presents the Differentiated Services (DiffServe) scheme for classes of services as well as Path MTU discovery and routing for anonymous serial networks.
- Explains Random Early Discard (RED), which is now recommended for routers.
- Updates the coverage of all protocols to the latest versions, including RIP, IGMP, and SNMP.

Douglas E. Comer is a Professor of Computer Science at Purdue University and a Fellow of the ACM. He is the author of many best-selling books: the three-volume series, *Internetworking with TCP/IP, Computer Networks and Internets, Second Edition,* and *The Internet Book.* Comer served as chairman of the DARPA Distributed Systems Architecture Board and the CSNET Technical Committees, and is a former member of the Internet Architecture Board (IAB).

ISBN 0-13-018380-6

90000

Prentice Hall
Upper Saddle River, NJ 07458
http://www.prenhall.com

Google Exhibit No. 1023
Page 027

Google Exhibit No. 1063
Page 0945

Ex. 1023A

LIBRARY | CATALOG

1 of 1

BOOK

Internetworking with TCP/IP

Full Record MARC Tags

Personal name
Comer, Douglas

Main title
Internetworking with TCP/IP / Douglas E. Comer.

Edition
4th ed.

Published/Created
Upper Saddle River, N.J.: Prentice Hall, 2000-

Request this item | LC Find It

More Information

LCCN Permalink	https://lccn.loc.gov/00027439
Description	v. <1- > : ill. ; 25 cm.
ISBN	0130183806 (v. 1)
LC classification	TK5105.585 .C66 2000

Google Exhibit No. 1063
Page 0947

Incomplete contents	v. 1. Principles, protocols, and architecture
LC Subjects	TCP/IP (Computer network protocol) Client/server computing. Internetworking (Telecommunication)
Browse by shelf order	TK5105.585
Notes	Includes bibliographical references (v. 1, p. [721]-722) and index.
LCCN	00027439
Dewey class no.	005.2
Type of material	Book

Item Availability

CALL NUMBER	TK5105.585 .C66 2000
Request in	Jefferson or Adams Building Reading Rooms
Status	c.2 v. 1 2000 In Process 04-20-2000

CALL NUMBER	TK5105.58 .C66 2000
Request in	Jefferson or Adams Building Reading Rooms
Status	Not Charged

Google Exhibit No. 1063
Page 0948

LIBRARY CATALOG

1 of 1

BOOK

Internetworking with TCP/IP

Full Record MARC Tags

000		01127cam a22002894a 4500
001		11923781
005		20030527145300.0
008		000228m20009999njua b 001 0 eng
906	__	|a 7 |b cbc |c orignew |d 1 |e ocip |f 20 |g y-gencatlg
925	0_	|a acquire |b 2 shelf copies |x policy default
955	__	|a to ASCD pb18 02-28-00; jg12 03-02-00; jg08; to Dewey 03-02-00; aa07 03-02-00; v.1 rec'd, to CIP ver. ps16 04-20-00; CIP ver. jb08 05-03-00 to jf00; jf12 to BCCD 08-31-00
010	__	|a 00027439
020	__	|a 0130183806 (v.1)
040	__	|a DLC |c DLC |d DLC
042	__	|a pcc
050	00	|a TK5105.585 |b .C66 2000
082	00	|a 005.2 |2 21
100	1_	|a Comer, Douglas.
245	10	|a nternetworking with TCP/IP / |c Douglas E. Comer.
250	__	|a 4th ed.
260	__	|a Upper Saddle River, N.J. : |b Prentice Hall, |c 2000-
300	__	|a v. <1- > : |b ill. ; |c 25 cm.
504	__	|a ncludes bibliographical references (v. 1, p. [721]-722) and index.
505	1_	|a v. 1. Principles, protocols, and architecture.
650	_0	|a TCP/IP (Computer network protocol)
650	_0	|a Client/server computing.

Google Exhibit No. 1063
Page 0949

650 _0 |a Internetworking (Telecommunication)

Request this Item | LC Find It

Item Availability

CALL NUMBER	TK5105.585 .C66 2000
Request in	Jefferson or Adams Building Reading Rooms
Status	c.2 v. 1 2000 In Process 04-20-2000

CALL NUMBER	TK5105.58 .C66 2000
Request in	Jefferson or Adams Building Reading Rooms
Status	Not Charged

Google Exhibit No. 1063
Page 0950

Ex. 1023B

Copyright
United States Copyright Office

Try the Copyright Public Records System (CPRS) pilot with enhanced search features and filters.

Help | Search | History | Titles | Start Over

Public Catalog

Copyright Catalog (1978 to present) at DC4
Search Request: Left Anchored Title = Internetworking with TCP/IP
Search Results: Displaying 13 of 22 entries

previous | next

Labeled View

Internetworking with TCP/IP : vol. 1, principles, protocols, and...

Type of Work: Text
Registration Number / Date: TX0005160814 / 2000-03-15
Title: Internetworking with TCP/IP : vol. 1, principles, protocols, and architecture / Douglas E. Comer.
Edition: 4th ed.
Imprint: Upper Saddle River, NJ : Prentice Hall, c2000.
Description: 750 p.
Copyright Claimant: Prentice-Hall, Inc.
Date of Creation: 1999
Date of Publication: 2000-01-18
Previous Registration: Prev. reg. 1995, TX 4-030-639.
Basis of Claim: New Matter: updating & additions.
Variant title: Internetworking with TCP/IP : vol. 1, principles, protocols, and architecture
Names: Comer, Douglas E.
Prentice-Hall, Inc.

previous | next

Save, Print and Email (Help Page)		
Select Download Format	Full Record	Format for Print/Save
Enter your email address:		Email

Help | Search | History | Titles | Start Over

Contact Us | Request Copies | Get a Search Estimate | Frequently Asked Questions (FAQs) about Copyright | Copyright Office Home Page | Library of Congress Home Page

Google Exhibit No. 1063
Page 0952

H.264/AVC in Wireless Environments

Thomas Stockhammer, Miska M. Hannuksela, and Thomas Wiegand

***Abstract*—Video transmission in wireless environments is a challenging task calling for high-compression efficiency as well as a network friendly design. Both have been major goals of the H.264/AVC standardization effort addressing "conversational" (i.e., video telephony) and "nonconversational" (i.e., storage, broadcast, or streaming) applications. The video compression performance of the H.264/AVC video coding layer typically provides a significant improvement. The network-friendly design goal of H.264/AVC is addressed via the network abstraction layer that has been developed to transport the coded video data over any existing and future networks including wireless systems. The main objective of this paper is to provide an overview over the tools which are likely to be used in wireless environments and discusses the most challenging application, wireless conversational services in greater detail. Appropriate justifications for the application of different tools based on experimental results are presented.**

***Index Terms*—Error concealment, error-resilient video coding, H.264/AVC, multiple reference frames, rate-distortion optimization, video coding standards, wireless video transmission.**

I. Introduction

SINCE 1997, the ITU-T's Video Coding Experts Group (VCEG) has been working on a new video coding standard with the internal denomination H.26L. In late 2001, the Moving Picture Expert Group (MPEG) and VCEG decided to work together as a Joint Video Team (JVT), and to create a single technical design called H.264/AVC for a forthcoming ITU-T Recommendation H.264/AVC and for a new part of the MPEG-4 standard called AVC [1][1], [2]. Since the meeting in November 2002, the technical specification is frozen and the standard text and software have been finalized. The primary goals of H.264/AVC are *improved coding efficiency* and *improved network adaptation*. The syntax of H.264/AVC typically permits a significant reduction in bit rate [3] compared to all previous standards such as ITU-T Rec. H.263 [4] and ISO/IEC JTC 1 MPEG-4 [5] at the same quality level.

The demand for fast and location-independent access to multimedia services offered on today's Internet is steadily increasing. Hence, most current and future cellular networks, like GSM-GPRS, UMTS, or CDMA-2000, contain a variety of packet-oriented transmission modes allowing transport of practically any type of IP-based traffic to and from mobile terminals, thus providing users with a simple and flexible transport interface. The third generation partnership project (3GPP) has selected several multimedia codecs for the inclusion into its multimedia specifications [6]. To provide basic video service in the first release of the 3G wireless systems, the well-established and almost identical baseline H.263 and the MPEG-4 visual simple profile have been integrated. The choice was based on the manageable complexity of the encoding and decoding process, as well as on the maturity and simplicity of the design.

However, due to the likely business models in emerging wireless systems in which the end-user's costs are proportional to the transmitted data volume and also due to limited resources bandwidth and transmission power, compression efficiency is the main target for wireless video and multimedia applications. This makes H.264/AVC coding an attractive candidate for all wireless applications including multimedia messaging services (MMS), packet-switched streaming services (PSS), and conversational applications. However, to allow transmission in different environments, not only is coding efficiency relevant, but also seamless and easy integration of the coded video into all current and possible future protocol and multiplex architectures. In addition, for conversational applications the video codec's support of enhanced error-resilience features is of major importance. This has also been taken into account in the standardization of this codec.

This paper is structured as follows. Section II introduces applications and transmission characteristics for wireless video applications. The transport of H.264/AVC video is briefly discussed and common test conditions for mobile video transmission are presented. Section III provides an overview over the H.264/AVC video coding standard from the perspective of wireless video applications. We categorize features according to their applicability in different video services. Section IV discusses the most challenging application in terms of delay constraints and error resilience, namely wireless conversational applications. A system description and problem formulation is followed by providing several alternatives on the system design using H.264/AVC as well as the combination of several modes. Section V provides experimental results for selected system concepts based on the common test conditions.

II. Video in Mobile Networks

A. Overview: Applications and Constraints

Video transmission for mobile terminals is likely to be a major application in emerging 3G systems and may be a key factor in their success. The video-capable display on mobile devices paves the road to several new applications. Three

Manuscript received April 9, 2002; revised May 10, 2003.

T. Stockhammer is with the Institute for Communications Engineering (LNT), Munich University of Technology (TUM), 80290 Munich, Germany (e-mail: stockhammer@ei.tum.de).

M. M. Hannuksela is with the Nokia Corporation, 33721 Tampere, Finland (e-mail: miska.hannuksela@nokia.com).

T. Wiegand is with the Fraunhofer-Institute for Telecommunications—Heinrich-Hertz-Institute Einsteinufer 37, 10587 Berlin, Germany (e-mail: wiegand@hhi.de).

Digital Object Identifier 10.1109/TCSVT.2003.815167

[1]All referenced standard documents can be accessed via anonymous ftp at ftp://standard.pictel.com/video_site, ftp://ftp.imtc-files.org/jvt-experts, ftp://ftp.ietf.org/, or ftp://www.3gpp.org/Specs/archive.

1051-8215/03$17.00 © 2003 IEEE

Google Exhibit No. 1027
Page 001

Google Exhibit No. 1063
Page 0953

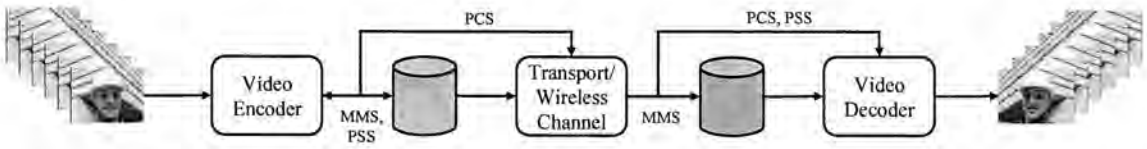


Fig. 1. Wireless video application MMS, PSS, and PCS: differentiation by real-time or offline processing for encoding, transmission, and decoding.

major service categories were identified in the H.264/AVC standardization process [7].

1) Circuit-switched [8] and packet-switched conversational services (PCS) [9] for video telephony and conferencing.
2) Live or pre-recorded video packet-switched streaming services (PSS) [10].
3) Video in multimedia messaging services (MMS) [11].

Although new services such as multimedia broadcast/multicast services (MBMS) [12] are planned for future releases of wireless networks, we restrict ourselves to single receiver applications. Mobile devices are hand-held and constrained in processing power and storage capacity. Therefore, a mobile video codec design must minimize terminal complexity while remaining consistent with the efficiency and robustness goals of the design. As complexity issues are discussed elsewhere in this issue [13], [14], we restrict ourselves to transmission constraints and properties.

The transmission requirements for the three identified applications can be distinguished with respect to requested data rate, the maximum allowed end-to-end delay, and the maximum delay jitter. This results in different system architectures for each of these applications. A simplified illustration is provided in Fig. 1. As MMS does not include any real-time constraints, encoding, transport, and decoding are completely separated. The recorded video signal is offline encoded and locally stored. The transmission is started using the stored signal at any time. The decoding process at the receiver is in general not started until the completion of the download. In PSS applications, the user typically requests pre-coded sequences, which are stored at a server. Whereas encoding and transmission are separated, decoding and display is started during transmission to minimize the initial delay and memory usage in mobile devices. Finally, in conversational services, the end-to-end delay has to be minimized to avoid any perceptual disturbances and to maintain synchronicity of audio and video. Therefore, encoding, transmission and decoding is performed simultaneously in real-time and, moreover, in both directions. These different ancillary conditions permit and require different strategies in encoding, transport, decoding, as well as in the underlying network and control architecture.

In general, the available bandwidth and therefore the bit-rate over the radio link are limited and the costs for a user are expected to be proportional to the reserved bit rate or the number of transmitted bits over the radio link. Thus, low bit rates are likely to be typical, and *compression efficiency* is the main requirement for a video coding standard to be successful in a mobile environment. This makes H.264/AVC a prime candidate for the use in wireless systems, because of its superior compression efficiency [3].

In addition, the mobile environment is characterized by harsh transmission conditions in terms of attenuation, shadowing, fading, and multi-user interference, which result in time- and location-varying channel conditions. The frequency of the channel variations highly depends on the environment, the user topology, the velocity of the mobile user, and the carrier frequency of the signal. For sufficiently long code words averaging over channel statistics is possible and transmission strategies can be used that are based on the long-term averages of fading states and the ergodic behavior of the channel. Many highly sophisticated radio link features such as broadband access, diversity techniques, space-time coding, multiple antenna systems, fast power control, interleaving, and forward error correction (FEC) by Turbo codes are used in 3G systems to reduce variations in channel conditions. However, only for fast-moving users and relatively large tolerated maximum delay can these advanced techniques provide a negligible bit error and radio block loss rate. Usually, some amount of residual errors has to be tolerated for low-delay applications due to the nonergodic behavior of the channel and the imperfectness of the applied signal processing. Therefore, in addition to high compression efficiency and reasonable complexity, a video coding standard to be applicable for conversational services in wireless environments has to be error resilient.

In addition, it is worth noting at this point that new directions in the design of wireless systems do not necessarily attempt to minimize the error rates in the system, but to maximize the throughput. This is especially appealing for services with relaxed delay constraints such as PSS and MMS. The nonergodic behavior of the channel is exploited such that in case of good channel states significantly higher data rate is supported than in bad channel states. In addition, reliable link layer protocols with persistent Automatic Repeat reQuest (ARQ) are usually used to guarantee error-free delivery. For example, in the high-speed downlink packet access (HSDPA) concept [15] ARQ, adaptive modulation schemes, and multiuser scheduling taking into account the channel states are combined to significantly enhance the throughput in wireless systems.

3G wireless transmission stacks usually consist of two different bearer types, dedicated and shared channels. Whereas in dedicated channels one user gets assigned a fixed data rate for the entire transmission interval, shared channels allow a dynamic bit-rate allocation similar to ATM or GSM GPRS. HSDPA will be an extension of the shared channel concept on the air interface. Except for MMS all streaming and conversational applications are assumed to use dedicated channels in the initial phase of 3G wireless systems due to their almost constant bit-rate behavior. In modern system designs, an application can request one of many different quality-of-service (QoS) classes. QoS classes contain parameters like a maximum error rate,

Google Exhibit No. 1027
Page 002

Google Exhibit No. 1063
Page 0954

TABLE I
QoS SERVICE CLASSES IN PACKET RADIO SYSTEMS [15]

Traffic Class	Fundamental Characteristics	Typical Examples
Conversational	Preserve time relation between information entities of the stream Conversational pattern (stringent and low delay)	Voice and video telephony, Video conferencing
Streaming	Preserve time relation (variation) between information entities of the stream	Streaming multimedia (video, audio, etc.)
Interactive	Request response pattern, Preserve data integrity	Web browsing, network games
Background	Destination is not expecting the data within a certain time Preserve data integrity	Background download of e-mails, files, etc.

maximum delay, and a guaranteed maximum data rate. Furthermore, according to [16], applications are usually divided into different service classes: conversational, streaming, interactive, and background traffic. Characteristics and typical examples are shown in Table I.

B. Transport of H.264/AVC Video in Wireless Systems

According to Fig. 2, H.264/AVC distinguishes between two different conceptual layers, the video coding layer (VCL) and the network abstraction layer (NAL). Both the VCL and the NAL are part of the H.264/AVC standard. The VCL specifies an efficient representation for the coded video signal. The NAL of H.264/AVC defines the interface between the video codec itself and the outside world. It operates on NAL units which give support for the packet-based approach of most existing networks. At the NAL decoder interface, it is assumed that the NAL units are delivered in decoding order and that packets are either received correctly, are lost, or an error flag in the NAL unit header can be raised if the payload contains bit errors. The latter feature is not part of the standard as the flag can be used for different purposes. However, it provides a way to signal an error indication through the entire network. Additionally, interface specifications are required for different transport protocols that will be specified by the responsible standardization bodies. The exact transport and encapsulation of NAL units for different transport systems, such as H.320 [17], MPEG-2 Systems [18], and RTP/IP [19], are also outside the scope of the H.264/AVC standardization. The NAL decoder interface is normatively defined in the standard, whereas the interface between the VCL and the NAL is conceptual and helps in describing and separating the tasks of the VCL and the NAL.

For real-time video services over 3G mobile networks, two protocol stacks are of major interest. 3GPP has specified a multimedia telephony service for circuit-switched channels [8] based on ITU-T Recommendation H.324M. For IP-based packet-switched communication, 3GPP has chosen to use SIP and SDP for call control [20] and RTP for media transport [9]. In other words, the IP-based protocol stack as presented in [21] will be used in packet-switched 3G mobile services. While the H.324 and the RTP/UDP/IP stacks have different roots and a completely different switching philosophy, the loss and delay effects on the media data when transmitting over wireless dedicated channels are very similar.

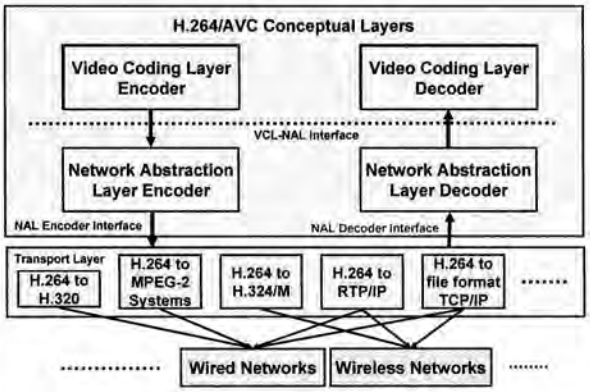


Fig. 2. H.264/AVC standard in transport environment.

H.324 [22] was primarily established by the ITU-T for low-bit-rate circuit-switched modem links and error prone extensions for mobile circuit switched low-bit-rate conversational services have been added. H.324M officially known as ITU-T Rec. H.324 Annex C, allows transmission over low, moderate, and high bit-error rate circuit switched links. 3GPP adopted H.324M including an error robust extension of the multiplexing protocol H.223 known as H.223 Annex B as the protocol used for circuit-switched video communication. This multiplexing protocol includes two layers: an error-resilient packet-based multiplex layer and an adaptation layer featuring common error detection capabilities, such as sequence numbering and cyclic redundancy checks (CRSs). Therefore, it is very similar to the RTP/UDP/IP stack (see [21]).

For packet-switched services, 3GPP/3GPP2 agreed on an IP-based stack. Fig. 3 shows a typical packetization of a NAL unit encapsulated in RTP/UDP/IP [19] through the 3GPP2 user plane protocol stack. After robust header compression (RoHC) [23] this IP/UDP/RTP packet is encapsulated into one packet data convergence protocol/point-to-point protocol (PDCP/PPP) packet that becomes a radio link control (RLC)-service data unit (SDU). The RLC protocol can operate in three modes: 1) transparent; 2) unacknowledged; and 3) acknowledged mode [26]. The RLC protocol provides segmentation and retransmission services for both users and control data. The transparent and unacknowledged mode RLC entities are defined to be

Google Exhibit No. 1027
Page 003

Google Exhibit No. 1063
Page 0955

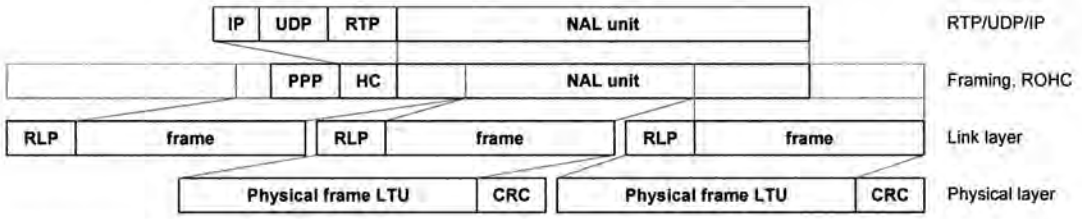


Fig. 3. Packetization through 3GPP2 user plane protocol stack.

TABLE II
BIT-ERROR PATTERNS

No.	Bit rate	Length	BER	Mobile Speed	Application
1	64 kbit/s	60 s	9.3e-3	3 km/h	Streaming
2	64 kbit/s	60 s	2.9e-3	3 km/h	Streaming
3	64 kbit/s	180 s	5.1e-4	3 km/h	Conversational
4	64 kbit/s	180 s	1.7e-4	50km/h	Conversational
5	128 kbit/s	180 s	5.0e-4	3 km/h	Conversational
6	128 kbit/s	180 s	2.0e-4	50km/h	Conversational

unidirectional and acknowledged mode entities are described as bi-directional. For all RLC modes, CRC error detection is performed on the physical layer and the result of the CRC check is delivered to the RLC together with the actual data. In the transparent mode no protocol overhead is added to higher layer data. Erroneous protocol data units (PDUs) can be discarded or marked erroneous. In the unacknowledged mode, no retransmission protocol is in use and data delivery is not guaranteed. Received erroneous data is either marked or discarded depending on the configuration. In the acknowledged mode, an automatic repeat request mechanism is used for backward error correction.

As video packets are of varying length by nature, the length of RLC-SDU's varies as well. If an RLC-SDU is larger than an RLC-PDU, the SDU is segmented into several PDUs. In the used, unacknowledged, and acknowledged modes, the flow of variable-size RLC-SDUs is continuous to avoid padding of bits as necessary for the transparent mode. In unacknowledged mode, if any of the RLC-PDUs containing data from a certain RLC-SDU have not been received correctly, the RLC-SDU is typically discarded. In acknowledged mode, the RLC/radio link protocol (RLP) layer can perform re-transmissions.

Additionally, both protocol stacks H.324 and RTP/IP/UDP use reliable setup and control protocols, H.245 and SIP, respectively. Hence, it can be assumed that a small amount of control information can be transported out-of-band in a reliable way. The resulting properties of real-time low-delay video transmission are therefore very similar in both cases. Packets are transmitted over underlying transports protocols and channels, which provide framing, encapsulation, error detection, and reliable setup. We focus hereafter on the RTP/IP-based transmission over wireless channels [21].

C. Common Test Conditions for Wireless Video

In the H.264/AVC standardization process, the importance of mobile video transmission has been recognized by adopting appropriate common test conditions for 3G mobile transmission for circuit switched conversational services based on H.324M [24] and for packet-switched conversational and streaming services [25]. These test conditions permit the selection of appropriate coding features, testing and evaluating error-resilience features, as well as meaningful anchor results. In this paper, we focus on the IP-based test conditions. The common test conditions define six test-case combinations for packet-switched conversational services as well as packet-switched streaming services over 3G mobile networks. Additionally, the test conditions include simplified offline 3GPP/3GPP2 simulation software, programming interfaces and evaluation criteria. Radio channel conditions are simulated with bit-error patterns, which were generated from simulated mobile radio channel conditions. The bit-error patterns are captured above the physical layer and below the RLC/RLP layer and, therefore, they are used as the physical layer simulation in practice. The properties bit rate, length, bit-error rate, and the mobile speed of the bit-error patterns are presented in Table II.

The bit errors in the files are statistically dependent, as channel coding and decoding included in 3G systems produces burst errors. This has been taken into account by evaluating the bit-error pattern files in the following. Patterns 1 and 2 are mostly suited to be used in video streaming applications, where RLP/RLC layer re-transmissions can correct many of the frame losses. The applied channel-coding scheme is a Turbo code scheme and power control targeting throughput maximization rather than error minimization. Patterns 1 and 2 are unrealistic for conversational service, as an acceptable quality cannot be achieved with such high error rates without retransmissions.

Patterns 3–6 are meant to simulate a more reliable, lower error-rate bearer that is required in conversational applications. Assuming a random byte starting position within the file the packet error probability $p_e(r)$ depending of the length of the packet r in bytes can be determined. These error probabilities $p_e(r)$ for all bit-error patterns are shown in Fig. 4. It is obvious

Google Exhibit No. 1027
Page 004

Google Exhibit No. 1063
Page 0956

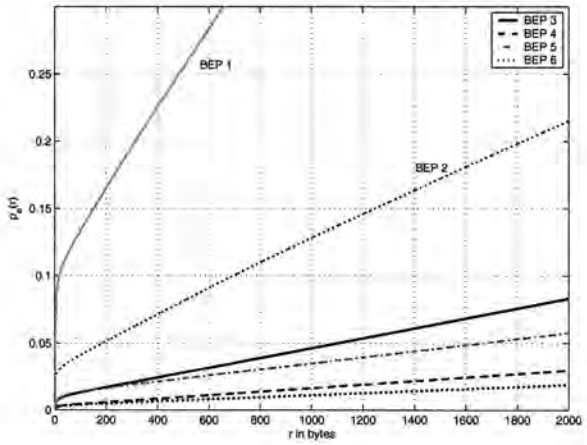


Fig. 4. Packet-loss probability $p_e(r)$ over packet length r in bytes for bit-error patterns according to [25].

that the error rate increases significantly with increasing packet length. For bit-error patterns (BEP1 and 2), the loss rates are not acceptable as already short packets, e.g., 500 bytes, have loss probabilities up to 25%. The high error rates require retransmission on the link layer.

The patterns useful for conversational services provide very decent error characteristics for reasonable packet lengths. The loss probability for reasonable packet sizes up to 1000 bytes is below 5%. This means that typical fixed Internet packet loss probabilities (compare [21]) are not exceeded. Note that with higher speed (50 km/h), the channel tends to be "more ergodic" than in case of the walking user (3 km/h). Therefore, the error rates are usually higher for slowly moving users than for fast-moving users.

During the standardization process, it was agreed in the very beginning that the standard should include error-resilience features for IP-based wired and wireless transmission. Usually two kinds of errors are present in today's transmission systems: bit inversion errors or packet losses. However, all relevant multiplexing protocols like H.223 and UDP/IP and almost all underlying mobile systems include packet loss and bit-error detection capabilities applying sequence numbering and block check sequences, respectively. Therefore, it can be assumed that a vast majority of erroneous transmission packets can be detected. Moreover, even if packets were detected to contain bit errors, decoding could be attempted. Some research has been conducted in this area and, in a few scenarios, gains have been reported for still image transmission [27]. However, in the standardization of H.264/AVC for the development of the reference software, bit-erroneous packets have been considered as being discarded by the receiver for the following reasons.

1) Processing of bit-erroneous packets is likely to be possible in a receiving mobile device only, as gateways and receivers having fixed network connection typically drop erroneous packets.
2) Joint source-channel decoding, such as trellis-based decoding of variable-length codes or iterative source and channel decoding might be applied. However, these techniques have not yet shown significant improvements for video decoding.
3) The decoding based on lost packets serves as a lower bound on the performance in bit-error-prone environments and, therefore, provides a valid benchmark of the performance of H.264/AVC in error-prone environment.
4) Finally, handling of bit errors generally complicates the implementation of decoder software significantly. As the test model software for H.264 was developed for multiple purposes, only simple but meaningful network interfaces have been integrated.

III. H.264/AVC—An Efficient and Flexible Video Coding Toolbox

A. Compression Efficiency and Encoder Flexibility

The features for compression efficiency are discussed elsewhere in this issue, we will only briefly present the key features of the standard, for more details we refer to [2]. Although the design of the H.264/AVC codec basically follows the hybrid design (motion compensation with lossy coding of residual signal) of prior video coding standards such as MPEG-2, H.263, and MPEG-4, it contains many new features that enable it to achieve a significant improvement in terms of compression efficiency. This is the main reason why H.264/AVC will be very attractive for use in wireless environments with the costly resource bit rate. The main features for significantly increased coding efficiency are multiframe motion-compensated prediction, adaptive block size for motion compensation, generalized B-pictures concepts, quartel-pel motion accuracy, intra coding utilizing prediction in the spatial domain, in-loop deblocking filters, and efficient entropy-coding methods.

The normative part of a video coding standard in general only consists of the appropriate definition of the order and semantics of the syntax elements and the decoding of error-free bit streams. This allows a significant flexibility at the encoder, which can, on the one hand, be exploited for pure compression efficiency, and on the other hand, several included features in the standard can be selected by the encoder for other purposes such as error resilience, random access, etc. A typical encoder with the main encoding options is shown in Fig. 5.

The encoding options relevant for wireless transmission are highlighted. The recorded video data is preprocessed by appropriate spatial and temporal preprocessing such that the data rates and displays in a wireless environment are well-matched. For the quantization of transform coefficients, H.264 coding uses scalar quantization. The quantizers are arranged in a way that there is an increase of approximately 12.5% from one quantization parameter (QP) to the next. The quantized transform coefficients are converted into coding symbols and all syntax elements of a macroblock (MB) including the coding symbols are conveyed by entropy coding methods. A MB can always be coded in one of several intra modes with and without prediction, as well as various efficient inter modes. Each motion-compensated mode corresponds to a specific partition of the MB into fixed-size blocks used for motion description, and up to 16 motion vectors may be transmitted for a MB. In addition, for each MB, a different reference frame can be selected. Finally, a NAL

Google Exhibit No. 1027
Page 005

Google Exhibit No. 1063
Page 0957

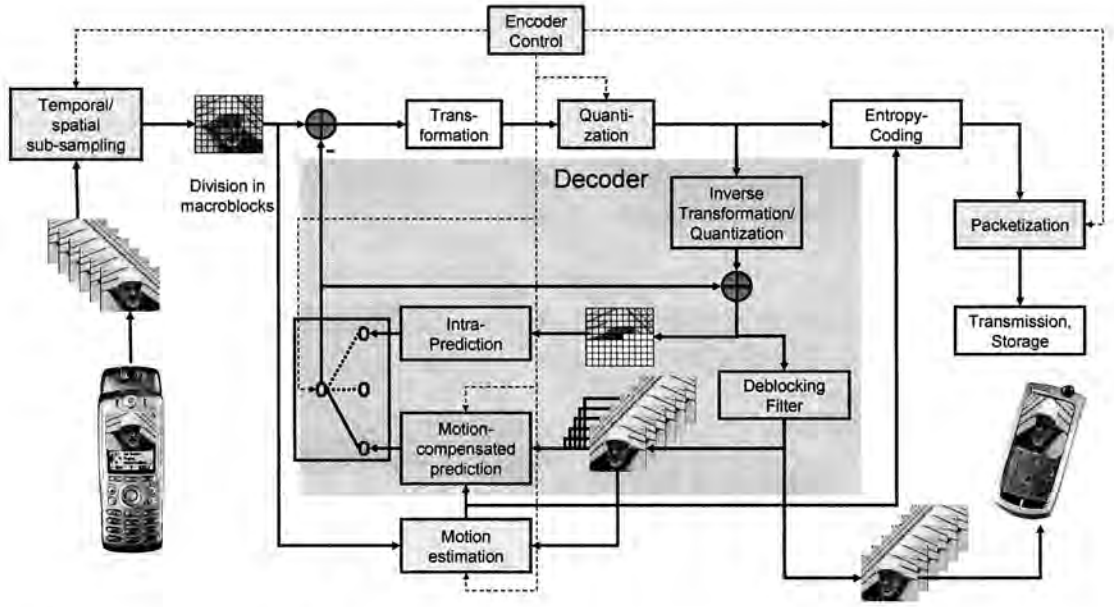


Fig. 5. H.264/AVC encoder realization with coding options.

unit in single-slice mode consists of the coded MBs of an entire frame or a subset of frame. More details on the packetization as well as on the appropriate selection of the presented different modes are discussed in Section IV.

B. Features for Multimedia Messaging and Wireless Packet-Based Streaming

In addition to pure compression efficiency features, additional tools for different purposes have been included in H.264/AVC. We will highlight those features with application to wireless video transmission. Because of the strict separation of encoding, transmission, and decoding, the main issue for MMS is compression efficiency. Other helpful features include the insertion of regular intra frames with instantaneous decoder refresh (IDR) for random access and fast forward. The rate control is typically applied such that video quality is almost constant over the sequence, regardless of the scene complexity except for constraints from the hypothetical reference decoder (HRD) [28]. If time, memory capabilities, and battery power permit, even several encoding passes for optimized rate-distortion (R-D) performance would be possible. On the link layer, reliable transmission strategies as known for data transmission such as file download are used.

Due to on-line transmission and decoding, streaming applications involve more technical challenges than MMS. Usually, pre-encoded data is requested by the user, which inherently does not allow an adaptation to the transmission conditions such as bit rate or error rate in the encoding process. However, the receiver usually buffers the received data and starts play-back after a few seconds. Once starting playback, a continuous presentation of the sequence should be guaranteed. As wireless channels usually show ergodic behavior within a window of a few seconds, reliable transmission schemes can be applied on the link layer, especially when the channel is known at the transmitter or retransmissions for erroneous link layer packets can be used as for example in the acknowledged mode. Slow variance due to distance, shadowing, or varying multiuser topology in the supported cell with renewed resource allocation transform the wireless channel in a slowly varying variable-bit-rate channel. With an appropriate setting of the initial delay and receiver buffer a certain quality of service can be guaranteed [28].

Furthermore, general channel-adaptive streaming technologies, which allow reacting to variable bit-rate channels, have gained significant interest recently. According to [30], these techniques can be grouped into three different categories. Firstly, *adaptive media playout* [31] is a new technique that allows a streaming media client, without the involvement of the server, to control the rate at which data is consumed by the playout process. Therefore, the probability of decoder buffer underflows and overflows can be reduced, but still noticeable artifacts in the displayed video occur. A second technology for a streaming media system is proposed, which makes decisions that govern how to allocate transmission resources among packets. Recent work [32] provides a flexible framework to allow *R-D optimized packet scheduling*. Finally, it is shown that this R-D-optimized transmission can be supported, if media streams are pre-encoded with appropriate packet dependencies, possibly adapted to the channel (*channel-adaptive packet dependency control*) [33].

The latter techniques are supported by H.264/AVC by various means. As the streaming server is in general aware of the current channel bit rate, the transmitter can decide to send one of several pre-encoded versions of the same content taking into account the expected channel behavior. If the channel rate fluctuates only in a small range, frame dropping of nonreference frames might be sufficient resulting in well-known temporal scalability.

Google Exhibit No. 1027
Page 006

Google Exhibit No. 1063
Page 0958

Switching of versions can be applied at I-frames that are also indicated as instantaneous decoder refresh (IDR) pictures to compensate large scale variations of the channel rate. In addition, H.264/AVC supports efficient version switching with the introduction of synchronization-predictive (SP) pictures. For more details on SP pictures, see [35]. Note that quality scalable video coding methods such as MPEG-4 fine-grain scalability (FGS) [34] are not supported by H.264/AVC and such extensions of H.264/AVC are currently not planned.

C. Features for Wireless Conversational Services—Error Resilience

A necessary requirement for conversational services is a low end-to-end delay being less that 250 ms. This delay constraint has two main impacts on the video transmitted over wireless bearer services with constant bit rate. Firstly, features have to be provided which allow adapting the bit-rate such that over a short window a constant bit-rate can be maintained. In addition, usually only temporally backward references in motion compensation are used in conversational applications, since prediction from future frames would introduce additional delay. Secondly, within the round-trip time of the communication, the channel usually shows nonergodic behavior and transmission errors cannot be avoided. Whereas the first issue can be solved by adapting the QP appropriately, the second issue requires error-resiliency tools in the video codec itself. More exactly, an error-resilient video coding standard suitable for conversational wireless services has to provide features to combat two problems: on the one hand, it is necessary to minimize the visual effect of errors within one frame. On the other hand, as errors cannot be avoided, the well-known problem of spatio-temporal error propagation in hybrid video coding has to be limited. We will present all error-resilience features included in the H.264/AVC standard and provide further details on the exact application in Section IV.

Packet loss probability and the visual degradation from packet losses can be reduced by introducing slice-structured coding. A slice is a sequence of MBs and provides spatially distinct resynchronization points within the video data for a single frame. No intra-frame prediction takes place across slice boundaries. With that, packet loss probability can be reduced if slices and, therefore, transmission packets are relatively small, since the probability of a bit-error hitting a short packet is generally lower than for large packets (see, e.g., Fig. 4). Moreover, short packets reduce the amount of lost information and, hence, the error is limited and error concealment methods can be applied successfully. However, the loss of intra-frame prediction and the increased overhead associated with decreasing slice sizes adversely affect coding performance and requires additional overhead per slice. Especially for mobile transmission, where the packet size clearly affects loss probability, a careful selection of the packet size is necessary. H.264/AVC specifies several enhanced concepts to reduce the artifacts caused by packet losses within one frame. Slices can be grouped by the use of aggregation packets into one packet and, therefore, concepts such as group-of-block (GOB) and *slice interleaving* [37], [38] are possible. This does not reduce the coding overhead in the VCL, but the costly RTP overhead of up to 40 bytes per packet can be avoided.

A more advanced and generalized concept is provided by a feature that has been called by the proponents *flexible MB ordering* (FMO) [39]. FMO permits the specification of different patterns for the mapping of MBs to slices including checkerboard-like patterns, sub-pictures within a picture (e.g., splitting a CIF picture into four QCIF pictures), or a dispersed mapping of MBs to slices. FMO is especially powerful in conjunction with appropriate error concealment when the samples of a missing slice are surrounded by many samples of correctly decoded slices. For more details on FMO, see [21].

Another error-resilience feature in H.264/AVC is *data partitioning*, which can also reduce visual artifacts resulting from packet losses, especially if prioritization or unequal error protection is provided by the network. For more details on the data-partitioning mode, we refer to [21]. In general, any kind of *forward error protection* (FEC) in combination with interleaving for packet lossy channels can be applied. A simple solution is provided by RFC2733 [40], more advanced schemes have been evaluated in many papers, e.g., [41], [42]. However, in the following, we do not consider FEC schemes in the transport layer as this requires a reasonable number of packets per codeword.

Despite all these techniques, packet losses and resulting reference frame mismatches between encoder and decoder are usually not avoidable. Then, the effects of spatio-temporal error propagation are, in general, severe. The impairment caused by transmission errors decays over time to some extent. However, the leakage in standardized video decoders, such as H.264/AVC, is not very strong, and quick recovery can only be achieved when image regions are encoded in intra mode, i.e., without reference to a previously coded frame. Completely intra-coded frames are usually not inserted in real-time and conversational video applications as the instantaneous bit rate and the resulting delay is increased significantly. Instead, H.264/AVC allows encoding of single MBs for regions that cannot be predicted efficiently as it is also known from other standards. In H.264/AVC, the efficient intra prediction can be constrained to intra MBs only to avoid error propagation from inter-coded MBs to refreshing intra-coded MBs. Another feature in H.264/AVC is the possibility to select the reference frame from the multiframe buffer. Both features have mainly been introduced for improved coding efficiency, but they can efficiently be used to limit the error propagation. Conservative approaches transmit a number of intra-coded MBs anticipating transmission errors. In this situation, the selection of intra-coded MBs can be done either randomly or preferably in a certain update pattern. For details and early work on this subject, see [43]–[45]. Multiple reference frames can also be used to limit the error propagation, for example in *video redundancy coding* schemes (see, e.g., [46]). In addition, a method known from H.263 under the acronym *redundant slices* will be supported in JVT coding. This will allow sending the same slice predicted from different reference frames which provides the decoder the possibility to predict this slice from error-free reference areas. Finally, multiple reference frames can be successfully combined with a feedback channel, which will be discussed in detail among others in Section IV.

Google Exhibit No. 1027
Page 007

Google Exhibit No. 1063
Page 0959

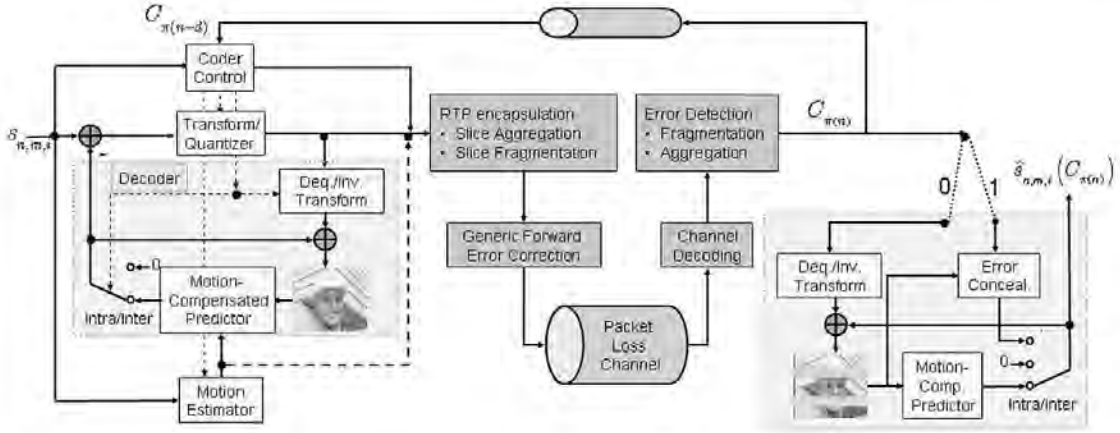


Fig. 6. H.264/AVC in IP-based packet-lossy environment with RTP encapsulation, generic forward error correction, delayed feedback information, and error concealment.

IV. Using H.264/AVC in Wireless Conversational Services

A. Problem Formulation and System Setup

The various error-resilience tools as described in the previous section provide a significant amount of freedom and flexibility for the implementation of H.264/AVC in wireless conversational services. A main criterion for the performance of the system is the appropriate selection of one or several of the mentioned tools together with the exact parameters, e.g., the position and number of intra MBs.

We will discuss in the following channel behavior and error concealment by formalizing the notation for sample representation. The investigated video transmission system is shown in Fig. 6. H.264/AVC video encoding is based on a sequential encoding of frames denoted with the index $n = 1, \ldots, N$ with N being the total number of frames to be encoded. In most existing video coding standards including H.264/AVC, within each frame video encoding is typically based on sequential encoding of MBs (except for FMO) denoted by index $m = 1, \ldots, M$, where M specifies the total number of MBs in one frame and depends on the spatial resolution of the video sequence. The encoding process creates slices by grouping a certain number of MBs. Picture number n and start MB address m_j are binary coded in the slice header. The coded representation of a slice is the payload of a NAL unit. The RTP payload specification specifies simple encapsulation of NAL units. In addition, several NAL units can be combined into one aggregation packet or one NAL unit can be fragmented into several transport packets [19], [21].

For notational convenience, let us define the number of transmission packets to transmit all frames up to n as $\pi(n)$. With that, we can define the packet loss or channel behavior c as a binary sequence $\{0, 1\}^{\pi(n)}$ indicating whether a slice is lost (indicated by 1) or correctly received (indicated by 0). Obviously, if a NAL unit with the encapsulated slice is lost, all MBs contained by this slice are lost. It can be assumed that the decoder is aware of any lost packet as discussed previously. The channel-loss sequence is random and, therefore, we denote it as $C_{\pi(n)}$, where the statistics are in general unknown to the encoder. According to Fig. 6, in addition to the forward link it possible that a low-bit-rate reliable back-channel from the decoder to the encoder is available which allows reporting a d-frame delayed version $C_{\pi(n-d)}$ of the observed channel behavior at the decoder to the encoder. In RTP/IP environments, this is usually based on RTCP messages, and in wireless environments, internal protocols might be used.

The decoder processes the received sequence of packets. Whereas correctly received packets are decoded as usual for the lost packet, an error-concealment algorithm has to be invoked. The reconstructed sample $s_{n,m,i}$ at position i in MB m and frame n depends on the channel behavior and on the decoder error concealment. In inter-coding mode, i.e., when motion-compensated prediction (MCP) is utilized, the loss of information in one frame has a considerable impact on the quality of the following frames, if the concealed image content is referenced for MCP. Because errors remain visible for a longer period of time, the resulting artifacts are particularly annoying. Therefore, due to the motion-compensation process and the resulting error propagation, the reconstructed image depends not only on the lost packets for the current frame, but in general on the entire channel-loss sequence $C_{\pi(n)}$. We denote this dependency by $\hat{s}_{n,m,i}(o, C_{\pi(n)})$

In the following, we will discuss appropriate extensions of the encoder, the decoder, and the transmission environment, which are either necessary or at least beneficial to enhance the quality of the transmitted video.

B. Decoder Extensions—Loss Detection and Error Concealment

The H.264 standard defines how a decoder reacts to an error-free bit stream. In addition, a decoder implementation has also to deal with transmission errors. As we discussed earlier, it is assumed that bit errors are detected by the lower layer

Google Exhibit No. 1027
Page 008

Google Exhibit No. 1063
Page 0960

entities and any remaining transmission error results in a packet loss. Therefore, we address the reaction of the decoder to slice losses. Two major issues are important for an error-resilient decoder: a robust video decoder has to detect transmission errors, and appropriate concealment on detected errors has to be applied. In this section, we present how the H.264 test model decoder meets the goal of error resiliency.

The error detection capabilities of the test model decoder are based on two assumptions about the error detection operation of the underlying system. First, bit-erroneous slices are discarded prior to passing the slices to the test model decoder. Second, received data is buffered in a way that the correct decoding order is recovered. In other words, the test model decoder expects noncorrupted slices in a correct decoding order. Temporal and spatial localization of lost packets is left to the decoder. In particular, the decoder has to detect if an entire picture or one or more slices of a picture were lost. Losses of entire pictures are detected using frame numbers associated with each frame and carried in slice headers. A frame number n is incremented by one for each coded and transmitted frame that is further used for motion compensation. These frames are herein referred to as reference frames. For disposable nonreference pictures, such as conventional B-pictures, the frame number is incremented relative to the value in the most recent reference frame that precedes the disposable picture in the bit-stream order.

The decoder generates a slice structure for each received single slice packet and forwards it to the VCL decoder which maintains a state machine that keeps track of the expected frame number n_e. Moreover, the decoder maintains a loss indication for each MB within the current picture. At the beginning of a new picture, the binary map is reset to indicate that all MBs were lost. If the frame number n of the next slice to be decoded equals n_e, the decoder decodes the slice and updates the binary map. If n is greater than n_e, it is deduced that all the received slices of the previous picture have been decoded. Then, the binary map is investigated, and if the picture is not fully covered by correctly decoded MBs, a slice loss is inferred and losses are concealed as presented in the following. Moreover, if n is greater than $n_e + 1$, the decoder infers a loss of pictures and inserts concealed pictures to the reference picture buffer as if the lost pictures were decoded. The concealment is accomplished by copying the previous decoded picture. Finally, the decoder resets the binary map and decodes the next slice of the next picture.

Error concealment is a nonnormative feature in the H.264 test model. The target for the selected error concealment is to provide a basic level of error resiliency for the decoder. Any error-robust coding scheme proposed for H.264 should be compared against the H.264 test model equipped with the selected error concealment algorithms. Two well-known concealment algorithms, weighted pixel value averaging for intra pictures [47] and boundary-matching-based motion vector recovery for inter pictures [48], were tailored for H.264 as summarized below and described in details in [49] and [50].

Weighted pixel value averaging operates as follows. If a MB has not been received, it is concealed from the pixel values of spatially adjacent MBs. If a lost MB has at least two correctly decoded neighboring MBs, only these neighbors are used in the concealment process. Otherwise, previously concealed neighboring MBs take part in the process, too. Each pixel value in a MB to be concealed is formed as a weighted sum of the closest boundary pixels of the selected adjacent MBs. The weight associated with each boundary pixel is relative to the inverse distance between the pixel to be concealed and the boundary pixel. The performance of the intra concealment method is shown in Fig. 7



Fig. 7. Intra frame error concealment.

In the motion vector recovery algorithm, the motion activity of the correctly received slices of the current picture is investigated first. If the average length of a motion vector component is smaller than a pre-defined threshold (currently 1/4 pixels), all the lost slices are copied from co-located positions in the reference frame. Otherwise, motion-compensated error concealment is used, and the motion vectors of the lost MBs are predicted as described in the following paragraphs. The image is scanned MB-column-wise from left and right edges to the center of the image. Consecutive lost MBs in a column are concealed starting from top and bottom of the lost area toward to center of the area. This processing order is used to ensure that lost MBs at the center of an image are concealed using as many neighboring concealed MBs as possible.

Each 8×8 luminance block of a MB to be concealed is handled separately. If a block has spatially adjacent blocks whose motion vectors are correctly received, these motion vectors and their reference pictures are used to obtain a candidate prediction block. If all the motion vectors in the adjacent blocks are lost, neighboring concealed motion vectors are used similarly. In addition, the spatially co-located block from the previous frame is always one of the candidates. The candidate prediction block whose boundary matching error is the smallest is chosen. The boundary matching error is defined as the sum of the pixel-wise absolute differences of the adjacent luminance pixels in the concealed block and its decoded or concealed neighbor blocks. The lost prediction error block is not concealed. The performance of the inter-frame concealment is shown in Fig. 8.

C. Encoder Extensions

Let us now consider algorithms and rules for the appropriate selection of different encoding parameters according to the presentation in Fig. 5 for wireless conversational services. First, the rate control has to guarantee that the delay jitter is as small as possible. A good choice is to adapt for each frame to obtain almost constant encoding, but keep the QP within one frame constant. In the case that the highest QP cannot achieve the required bit rate, frame dropping is introduced; otherwise, the frame rate

Google Exhibit No. 1027
Page 009

Google Exhibit No. 1063
Page 0961



Fig. 8. Inter-frame error concealment.

is kept constant. Obviously, this results in varying quality depending of the complexity of the scene. Low-motion scenes with little complexity have higher quality than for example frames at the beginning of a sequence or at a scene cut.

As discussed in Section III, H.264/AVC provides a large flexibility on the selection of appropriate coding modes. The concept of selecting appropriate coding options in optimized encoder designs for many video coding standards is based on R-D optimization algorithms [51],[52]. The two cost terms "rate" and "distortion" are linearly combined and the mode is selected such that the total cost is minimized. This can be formalized by defining the set of selectable coding options for one MB as $\mathcal{O}$. Assuming the introduced distortion when encoding with a certain mode o as $D(o)$ and the corresponding rate as $R(o)$, the rate-constrained mode decision selects the coding option o^* such that the Lagrangian cost functional is minimized, i.e.,

$$o^* = \arg\min_{o \in \mathcal{O}} (D(o) + \lambda R(o)) \tag{1}$$

with λ being the Lagrange parameter for appropriate weighting of rate and distortion. In the H.264/AVC test model, the Lagrangian mode selection is used for motion vector search as well as MB mode and reference frame selection. Then, for the distortion $D(o)$, the sum of squared sample differences (SSD) or the sum of absolute sample difference (SAD) are used, and for the rate $R(o)$, the number of bits for the encoding is used. The selection of the Lagrange parameter depends on the selected QPs (for details, see [53]). In addition, the mode selection might be extended by selecting the appropriate QP for each MB, as the QP can be changed at least in small ranges for each MB. This is not considered in the following.

As discussed in Section III-C, the tools for increased error resilience, in particular those to limit error propagation, do not significantly differ from those used for compression efficiency. Features like multiframe prediction or intra MB coding are not exclusively error-resilience tools. They are also used to increase coding efficiency in error-free environments providing a trade-off that is left to the encoder. This also means that bad decisions at the encoder can lead to poor results in coding efficiency or error resiliency or both. Therefore, the selection of the coding mode according to (1) is modified taking into account the influence of the random lossy channel. In the case of error-prone transmission, the distortion in (1) is replaced with the expected decoder distortion when encoding with mode o. Assuming that the encoder can access information about the channel statistics $C_{\pi(n)}$, the encoder can get an estimate of the expected decoder distortion $D_{n,m}(o, C_{\pi(n)})$ when encoding MB m in frame n at the decoder by the expected distortion as

$$D_{n,m}\left(o, C_{\pi(n)}\right) = \sum_{i=1}^{I} E_{C_{\pi(n)}} \left| s_{n,m,i} - \hat{s}_{n,m,i}\left(o, C_{\pi(n)}\right) \right|^2 \tag{2}$$

where the expectation is over the random process characterizing the channel $C_{\pi(n)}$. With the expected distortion measure, the mode selection for lossy channels is identical to that in (1) except for modified distortion term according to (2). However, the same parameter λ [54] and the same set of possible coding options $\mathcal{O}$ are used.

This leaves the problem of computing the expected decoder distortion at the encoder which depends on the coding mode o, the channel statistics $C_{\pi(n)}$, and the applied error concealment in the decoder. The estimate of the expected sample distortion in packet loss environment has been addressed in several publications. For example, in [58], [55], or [56], methods to estimate the distortion introduced due the transmission errors and the resulting drift are presented. A similar approach has recently been proposed within the H.264/AVC standardization process which attempts to measure the drift noise between encoder and decoder [57]. In suboptimal approaches [55]–[57], the quantization noise and the distortion introduced by the transmission errors are linearly combined. The encoder keeps track of an estimated sample distortion and, therefore, requires additional complexity in encoder, which is dependent on the actual method chosen.

The most recognized out of these methods, called recursive optimal per-sample estimate (ROPE) algorithm [58], provides an estimation by keeping track of the first- and second-order moment of $\hat{s}_{n,m,i}, E\left\{\hat{s}_{n,m,i}(o, C_{\pi(n)})\right\}$, and $E\left\{\hat{s}^2_{n,m,i}(o, C_{\pi(n)})\right\}$, respectively. For H.263-like encoding ROPE can provide an exact estimate of the expected decoder distortion. As two moments for each sample have to be tracked in the encoder, the added complexity of ROPE is approximately twice the complexity of the decoder. However, the extension of the ROPE algorithm to H.264/AVC is not straightforward. The in-loop deblocking filter, the fractional sample motion accuracy, the complex intra prediction and the advanced error concealment require taking into account the expectation of products of samples at different positions to obtain an accurate estimation which makes the ROPE either infeasible or inaccurate in this case.

Therefore, a powerful yet complex method has been introduced into the H.264/AVC test model to estimate the expected decoder distortion [54]. Let us assume that we have K copies of the random variable channel behavior at the encoder, denoted as $C_{\pi(n)}(k)$. Additionally, assume that the set of random variables $C_{\pi(n)}(k), k = 1, \cdots, K$ are *identically* and *independently* distributed (*i.i.d.*). Then, as $K \to \infty$, it follows by the strong law of large numbers that

$$\frac{1}{K}\sum_{k=1}^{K} \left| s_{n,m,i} - \hat{s}_{n,m,i}\left(C_{\pi(n)}(k)\right) \right|^2 = E_{C_{\pi(n)}} \left| s_{n,m,i} - \hat{s}_{n,m,i}\left(C_{\pi(n)}\right) \right|^2 \tag{3}$$

Google Exhibit No. 1027
Page 010

Google Exhibit No. 1063
Page 0962

holds with probability 1. An interpretation of the left-hand side leads to a simple solution of the previously stated problem to estimate the expected sample distortion. In the encoder, K copies of the random variable channel behavior and the decoder are operated. The reconstruction of the sample value depends on the channel behavior $C_{\pi(n)}(k)$ and the decoder including error concealment. The K copies of channel and decoder pairs in the encoder operate independently. Therefore, the expected distortion at the decoder can be estimated accurately in the encoder if K is chosen large enough. However, the added complexity in the encoder is obviously at least K times the decoder complexity. For details on the implementation as well as comparison of suboptimal modes based on ROPE, we refer to [54].

It was later demonstrated in [59] and [60] that a technique known as isolated regions, which combines periodical intra MB coding and limitation of inter prediction, can provide competitive simulation results compared to the error-robust MB mode selection of H.264 test model, and the best simulation results were obtained when the two MB mode-selection algorithms were combined.

The packet size is usually upper bounded by a MTU size of the underlying network. However, if bit errors cause entire packet losses, it might be an advantage to reduce the packet size. Conservative approaches limit the packet or (in our simulation environment) slice size to a fixed number of MBs, e.g., a line of MBs is transmitted in one packet. However, this makes especially important image parts very susceptible to packet losses as for example suddenly appearing objects need intra coding or high motion areas require an extensive motion vector rate. A more suitable, but still conservative, approach is the introduction of slice boundaries such that slices have almost the same length in number of bytes. This makes all packets similarly susceptible to bit errors. This method has been proven to work quite well already for MPEG-4 and can also be applied with the flexible encoding framework of H.264/AVC.

D. Exploiting Feedback in H.264/AVC

Multiple reference frames used in error-prone environment can most successfully be combined with a feedback channel. Due to the bidirectional nature of conversational applications, it is common that the encoder has knowledge of the experienced channel at the decoder, usually with a small delay. In our framework this can be expressed by the knowledge of a d-frame delayed version of the random channel $C_{\pi(n-d)}$ at the encoder. This characteristic can be conveyed from the decoder to the encoder by acknowledging correctly received slices (ACK), sending a not-acknowledge message (NAK) for missing slices or both types of messages. In general, it can be assumed that the reverse channel is error-free and the overhead is negligible. In common data transmission applications, the lost packets would obviously be re-transmitted. However, in a low-delay environment, this is not feasible, but the observed channel characteristic are still useful at the encoder even if the erroneous frame has already been decoded and concealed. The support of these techniques is out of the focus of the standardization of H.264, and a full support of the presented concepts might not be possible with current transport and control protocols. However, it seems worth to mention and discuss these concepts in here in detail as it shows the flexibility of H.264 as well as it provides motivation for inclusion of feedback in system designs.

In previous standards and transport environments similar approaches have already been discussed, usually limited by the reduced syntax capabilities of the video standard. A simple yet powerful approach suitable for video codecs using just one reference frame such as MPEG-2, H.261, or H.263 version 1 has been introduced in [61] and [62] under the name *error tracking*. When receiving a NAK on parts of frame $n-d$ or the entire frame $n-d$, the encoder attempts to track the error to obtain an estimate of the quality of frame $n-1$, which serves as reference for frame n. Appropriate actions after having tracked the error are for example presented in [56], and [61]–[64]. Note that with this concept, error propagation in frame n is only removed if frames $n-d+1,\ldots,n-1$ have been received at the decoder without any error.

A technique addressing the problem of continuing error propagation has been introduced, among others, in [65]–[67] under the acronym *NEWPRED*. Based on these early nonstandard compliant solutions in H.263 Annex N [68], a reference picture selection (RPS) for each GOB is specified such that the NEWPRED technique can be applied. RPS can be operated in two different modes. In the negative acknowledgment mode (NAM), the encoder only alters its operation in the case of reception of a NAK. Then, the encoder attempts to use an intact reference frame for the erroneous GOBs. To completely eliminate error propagation, this mode has to be combined with independent segment decoding (ISD) according to Annex R of H.263 [68]. In the positive acknowledgment mode (PAM), the encoder is only allowed to reference confirmed GOBs. If no GOBs are available to be referenced, intra coding has to be applied. NEWPRED allows to completely eliminate error propagation in frame n, even if additional errors have occurred in frames $n-d+1,\ldots,n-1$.

The flexibility provided in H.263 Annex U [68] and especially H.264/AVC to select the MB mode and reference frames on MB or subMB basis allows incorporating NEWPRED in a straightforward manner [56]. Therefore, let us define three different states of transmitted packets at the encoder: ACK, NAK, and outstanding acknowledgment (OAK). Then, based on the exploitation of these messages in the encoder, we discuss three modes which can elegantly be integrated into the R-D optimized mode selection according to (1).

1) Feedback Mode 1: Restricting Reference Areas to ACK: In this case, the reference area is restricted to frames or slices which have been acknowledged. This can be formalized in the context of (1) by applying the encoding distortion $D(o)$ in (1), but altering the accessible coding options $\mathcal{O}$ such that only acknowledged areas can be used for reference. In addition, if no reference area is available, or if no satisfying match is found in the allowed area, intra coding is applied. Although in the presentation of a single frame an error might be visible, error propagation and reference frame mismatch between encoder can be completely avoided independent of the error concealment applied at the decoder if correctly decoded samples are not altered. However, in the used test model JM1.7, the deblocking filter operation in the motion-compensation

Google Exhibit No. 1027
Page 011

Google Exhibit No. 1063
Page 0963

loop is applied over slice boundaries, which restricts the area to be referenced significantly, or a complete removal of encoder and decoder mismatch is not possible. Although the influence of this mismatch is, in general, negligible, in the final design of H.264/AVC the deblocking filter can adaptively be switched on and off at slice boundaries to allow mismatch-free operation.

2) Feedback Mode 2: Referencing ACK and Error Concealed NAK: In this case, the reference area is again restricted; however, in addition to acknowledged areas, also the areas which the decoder signaled as lost can be referenced. Therefore, the reference frames in the encoders multi-frame frame buffer are updated with reception of each ACK and NAK by applying the identical error concealment as the decoder applies. This is obviously very critical, as the error concealment is—for good reasons—nonnormative in common video coding standards including H.264/AVC. Only if the encoder is aware of the decoder's error concealment by any external means, this mode can provide benefits compared to feedback mode 1. In the context of the mode decision in (1), we have again a restricted set of coding modes $\mathcal{O}$ and, in addition, the encoding distortion is replaced by the deterministic decoder distortion.

3) Feedback Mode 3: Unrestricted Reference Areas With Expected Distortion Updating: In [56] and [58], techniques have been proposed which allow combining the error-resilient MB mode selection with feedback information. In this case, the expected decoder distortion computation is altered such that, for all packets $1, \ldots, \pi(n-d)$, the channel is deterministic at the encoder, and the expected distortion is computed for the packets with status OAK. In our case, packets containing MBs in frames $\pi(n-d)+1, \ldots, \pi(n)$ are random. The set of selectable coding options $\mathcal{O}$ is not altered compared to pure coding efficiency mode selection. This method is especially beneficial compared to mode 1 and 2, if the feedback is significantly delayed. In the case of the multiple decoder approach, this can be integrated by applying feedback mode 2 not only to the reference frames, but also to all decoders in the encoder. In combination with ROPE, however, the complexity of this method increases since the moments of the previous d frames have to be re-tracked [58].

V. Selected Simulation Results for Different System Concepts

A. Simulation Conditions and Evaluation Criteria

In the following, we will present simulation results based on the test conditions that show the influence of the selection of different error-resilience features for the quality of the decoded images. For all following tests the H.264/AVC test model software version JM1.7 is used. Note that in the final standard [1], the zig-zag scanning and run-length coding is replaced by context-adaptive variable length codes (CVLC). All UVLCs are replaced by CVLC, which are adapted to the statistics of different syntax elements. In addition, quantizer values have been shifted. However, all changes from JM1.7 to the final standard are of little relevance to the presented results and conclusions in this paper.

The reported PSNR is the arithmetic mean over the decoded luminance PSNR over all frames of the encoded sequence and over 100 transmission and decoding runs. The 100 starting positions for the error patterns have been selected such that they are almost equally distributed over the error pattern. For all comparable results, the same starting positions and, therefore, the same channel statistics have been applied. In addition, we present results for the cumulative distribution of the decoded luminance PSNR for each frame, i.e., the likelihood that the PSNR of the frames of the sequence is smaller than the value on the x axis. This shows the variance in the decoded quality. It is assumed that the high-level syntax parameters have been transmitted in advance and out-of-band applying a reliable setup protocol. The NAL overhead, the RTP/UDP/IP overhead after RoHC, and the link layer overhead is taken into account in the bit-rate constraints according to [25].

For the following simulations, we concentrate on test case 5 from [25], which includes the QCIF test sequence "Foreman" (30 Hz, 300 frames) at a constant frame rate of 7.5 Hz at bit-error pattern 3 according to Table II, i.e., a mobile user at 3 km/h and maximum bit-rate 64 kbit/s. This is the most critical case in terms of error probability, additional test results will be made available online[2]. For the following tests, entropy coding based on the UVLC and only one reference frame has been applied, if not stated otherwise. The encoded sequences are I-P-P-P... sequences; B-pictures are excluded due to the unacceptable delay involved in the encoding process. Note that due to the repeated decoding of an encoded file, every 75th frame is an I-frame, i.e., an I-frame occurs every 10 s. Constrained intra has been used to avoid error propagation from inter MBs to intra MBs. For all encoding runs, R-D optimized mode selection and motion vector selection according to (1) have been used. The distortion $D(o)$ and the set of coding modes $\mathcal{O}$ is appropriately selected according to the applied features. In the case of using the expected decoder distortion, the number of decoders operated in the encoder has been fixed to $K = 100$. Unless stated otherwise, the error concealment in the decoder is based on the advanced error concealment as presented in Section IV-B, whereas the multiple decoders in the encoder always apply previous frame concealment. This reduces encoding complexity significantly and results only in negligible performance losses for the investigated cases.

B. Low-Delay Rate Control

Version JM1.7 of the H.264/AVC test model encoder which is used in the experiments does not include a rate control to achieve a constant bit rate for wireless conversational services. Moreover, the rate control introduced in later versions of H.264/AVC test model encoder is not suitable for constant rate encoding. For this reason we have added a rate control which provides an almost constant bit-rate encoding for each frame by adapting the QP for each frame appropriately. Therefore, before we investigate the error-resilience features in H.264/AVC, we will first focus on the effect of bit-rate control that is necessary for constant bit-rate conversational applications. Fig. 9 shows the cumulative distribution of the encoding PSNR for the applied rate control and a fixed QP such that both encoded files result in the

[2]For additional simulation results, we refer to http://www.lnt.ei.tum.de/~stockhammer.

Google Exhibit No. 1027
Page 012

Google Exhibit No. 1063
Page 0964

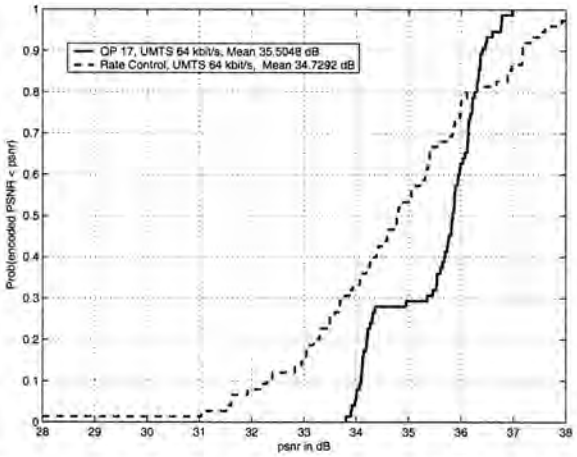


Fig. 9. Cumulative distribution of encoding luminance PSNR for each frame for constant QP 17 and constant bit rate such that rate constraint of 64 kbit/s for UMTS is fulfilled.

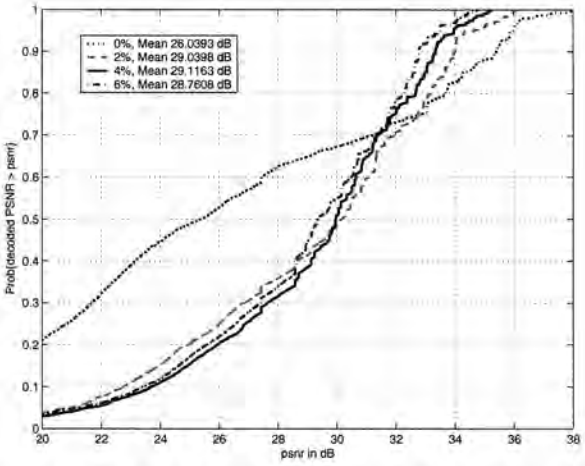


Fig. 10. Cumulative distribution of decoded PSNR for different NAL unit erasure rates for the estimation of the expected distortion in the encoder.

same total bit-rate, and, additionally the file can be transmitted over the 64 kbit/s link including NAL unit, packet, and link layer overhead. However, the fixed QP results in an extremely varying data rate and, therefore, the introduced delay jitter is not acceptable. In the encoding process, no error-resilience tools have been used, i.e., one frame is encapsulated into one NAL unit, for $D(o)$ the encoding distortion is applied, and the set of coding modes $\mathcal{O}$ is unrestricted.

The average PSNR for the rate control is about 0.8 dB below the PSNR for the fixed QP. In addition, the probability for low encoded PSNR is significantly higher as for complex scenes the QP has to be adapted appropriately. Therefore, it can be seen that the rate control necessary for conversational applications involves an inherent decrease in quality if the average PSNR is the measure of interest.

C. $R - E\{D\}$-Optimized MB Mode Selection

In this section we investigate the performance of the system in case that the channel statistics are taken into account into the selection of the coding options in the encoder. For this purpose we replace the encoding distortion $D(o)$ in (1) by the expected decoder distortion assuming a channel producing independent packet losses with probability p. As we use the mapping of one frame to one transport packet and apply the strict rate control producing almost a constant number of bytes for each encoded frame, the size of each frame results in roughly 1000 bytes. The corresponding loss rate for packets of this size for bit-error pattern 3 is approximately 4%–5% according to Fig. 4. Fig. 10 shows the cumulative distribution of decoded PSNR for different NAL unit erasure rates $p = \{0, 0.02, 0.04, 0.06\}$ for the estimation of the expected distortion in the encoder. Looking at the results for the average PSNR, it can be seen that the introduction of loss-aware R-D optimization ($p > 0$) increases the decoded quality significantly compared to the results with pure encoding distortion ($p = 0$). The average PSNR increases by at least 3 dB when compared to the to pure R-D optimization. The advantage of $R-E\{D\}$ optimization is even more evident when looking at the cumulative distribution of the different strategies. Whereas in the case of no error resilience, the probability of bad frames (frames below 22 dB) is at an unacceptable ratio of about 30%, for loss-aware coding, this is reduced significantly to less than 8%. It is also obvious that if the expected error rate matches the experienced error rate on the channel, the performance is optimal (see $p = 4\%$). However, it can also be seen that a mismatch in the expected error rate in the encoder does not have significant influence. The performance of $p = 2\%$ and $p = 6\%$ is only slightly inferior to the matching expected error rate. Therefore, a rough estimation of the expected decoder distortion at the encoder seems to be good enough for a good mode selection. Note the significant loss in average PSNR for wireless transmission compared to the error-free transmission according to Fig. 9 of more than 5 dB.

D. Slices and Error Concealment

The introduction of slices in the encoding has two beneficial aspects when transmitting over wireless channels, but adversely affects the coding efficiency due to increased packet overhead and reduced prediction within one frame, as e.g., motion vector prediction and spatial intra prediction is not allowed over slice boundaries. The two positive effects with the introduction of slices are the reduced error probability of shorter packets (see Fig. 4) and, the re-synchronization possibility within one frame. The latter technique allows restarting the decoding process at each slice and, in addition, it allows applying advanced error concealment as for example presented in Section IV-B. However, for packet-lossy transmission over the Internet, the introduction of slices does not, in general, provide gains in the decoded quality [69] as long as the slice size is below the MTU size, as the loss of a packet is then independent of its length. This is different for wireless transmission: Fig. 11 shows the average decoded PSNR for different number of packets per frame N_p and MB mode decision with encoding distortion. The experimental conditions are similar to the above section otherwise. For a given number of packets per frame, the size of each packet is selected such that the packets roughly have the same number of bytes. This makes their susceptibility to bit errors almost identical.

Google Exhibit No. 1027
Page 013

Google Exhibit No. 1063
Page 0965

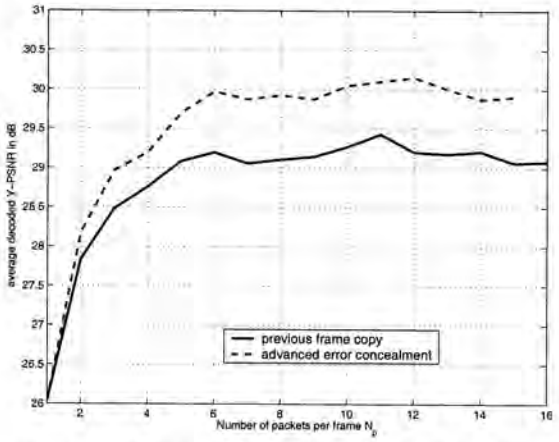


Fig. 11. Average decoded PSNR for different number of packets per frame and different error-concealment schemes; mode selection with encoding distortion.

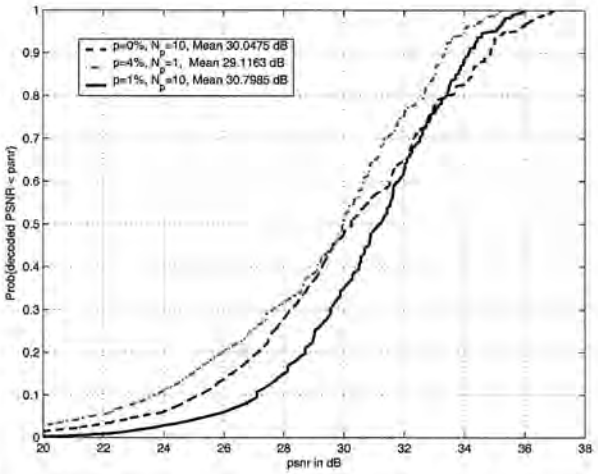


Fig. 12. Cumulative distribution of decoded PSNR for different error-resilience strategies: $R - E\{D\}$-optimized intra updates with and without slice structuring and delay $d = 2$ for different assumed loss probabilities p.

The average decoded PSNR is shown for previous frame copy error concealment and advanced error concealment according to Section IV-B. The benefit of introducing slice structuring in the encoding process is obvious from the results. Compared to the "one frame—one packet" mode indicated by $N_p = 1$ the introduction of shorter packets increases the decoded quality significantly for both error concealment methods. For about $N_p = 6$, the curve flattens out and decreases again for higher $N_p > 12$ due to increasing packet overhead and the reduced compression efficiency. Although a clear maximum cannot be determined, for the wireless environment according to our test conditions, a reasonable number of packets per frame is about 10. The resulting packet size in this case is in the range of 100 bytes. However, note that for the simulations, RoHC was applied, which reduces the typical IP/UDP/RTP overhead from 40 bytes to about 3 bytes and, therefore, the packetization overhead is less significant. The benefits of the advanced error concealment are also obvious from Fig. 11. As can be seen, the gains for advanced error concealment increase with increasing number of packets, as better concealment is possible due to increased number of neighboring MBs in case of losing a single slice.

For the experiments in Fig. 11, no explicit means to reduce the error propagation have been used. However, we can obviously combine the MB mode selection based on the expected decoder distortion with the slice structured coding. As indicated, the loss rate for decreased packet size decreases compared to the single packet for one frame. The loss rate can be estimated by dividing the approximated number of bytes for each frame, roughly 1000 bytes, by the number of packets. The loss probability can then again be estimated with Fig. 4 using the resulting average packet length. The combination of slice-structured coding and adaptive intra MB updates has been investigated and a comparison with the best cases of the previous two system designs is provided in Fig. 12 based on the cumulative distribution of the decoded PSNR. For the slice-structured coding with encoding distortion ($p = 0\%$) the number of packets is selected as $N_p = 10$. For the expected decoder distortion without slice-structuring ($N_p = 1$), the adapted loss-rate $p = 4\%$ is chosen. Finally, for the combination of slice-structured coding and channel-adaptive intra updates, the number of packets per frame is selected as $N_p = 10$ and, therefore, the appropriate loss probability to compute the expected decoder distortion according to Fig. 4 is about $p = 1\%$.

The average decoded PSNR indicates that an optimized combination of both error-resilience schemes outperforms each of the presented error-resilience schemes significantly. The result that slice-structured coding is superior to intra updates cannot be generalized for all sequences. The repeated I-frame insertion and the camera pan in the test sequence "Foreman" results in a significant amount of intra information, even if only the encoding distortion is chosen in the mode selection. This might change for different or longer sequences with less intra information. From the cumulative distribution, it can be observed that the probability for bad frames below 22 dB in PSNR is almost vanishing for the combined mode. The presented results indicate that slices in combination with advanced error concealment significantly outperform the single packet for one frame approach. However, the loss compared to error-free transmission is still about 4 dB in average PSNR.

From the results, it can be conjectured that for wireless transmission as investigated in this case, other approaches, which reduce the artifacts within one frame, might provide additional benefits. This includes concepts such as FMO, slice interleaving, or even generic forward-error correction in combination with a NAL unit fragmentation scheme as recently introduced in the draft RTP payload specification for H.264/AVC [19]. Also, data partitioning with appropriate unequal error protection might enhance the quality of the decoded video. In addition, it is conjectured from the results that a better adaptation of the link layer error protection scheme with appropriate interleaving could increase the overall system performance. These features are currently investigated and are subject of future work.

E. Exploiting Feedback in Video Encoding

Finally, we investigate a system which exploits multiple reference frames and network feedback. We restrict our simula-

Google Exhibit No. 1027
Page 014

Google Exhibit No. 1063
Page 0966

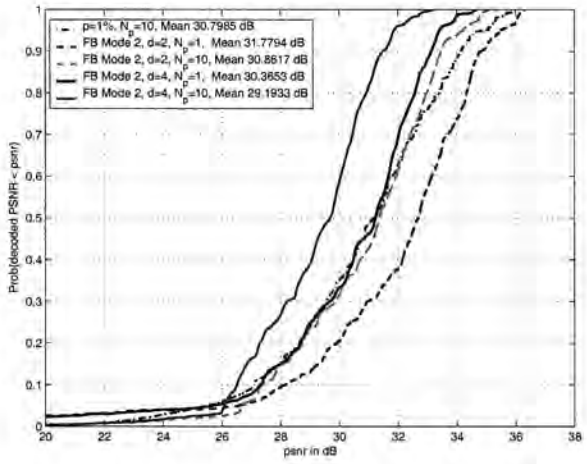


Fig. 13. Cumulative distribution of decoded PSNR for different error-resilience strategies: R-D-optimized intra updates with slice structuring, and feedback mode 2 with and without slice structuring for delay $d = 2$ and $d = 4$.

tion results to feedback mode 2 according to the presentation in Section IV-D with advanced error concealment. There are two reasons for this. On the one hand, the deblocking and mismatch problem present for JM1.7 can be avoided. On the other hand, it has been shown previously that feedback mode 1 and feedback mode 2 result in almost identical performance [70]. Therefore, we chose feedback mode 2 due to the simpler implementation, at least in our simulation environment. Feedback mode 3 is excluded as, on the one hand, the complexity of this mode is rather high and, on the other hand, as we operate with low feedback delays, the expected benefits are only marginal. In contrast to the previous simulations, we use five reference frames for the feedback mode. Fig. 13 shows the cumulative distribution of decoded PSNR for different error-resilience strategies: R-D-optimized intra updates with slice structuring, as well as feedback mode 2 with and without slice structuring for delay $d = 2$ and $d = 4$ frames, which corresponds to a round-trip time of about 250 and 500 ms, respectively.

Let us focus on the delay $d = 2$ case first. The results indicate that the optimized intra mode with slice structuring and MB mode selection with expected decoder distortion and feedback mode 2 perform very similar based on the cumulative distribution and the average decoded PSNR. The feedback mode might still be beneficial in this case as the complex estimation of the decoder distortion is not necessary for the feedback case. However, much more interesting is the case with feedback and no slice structuring. In contrast to the case without feedback (see Fig. 12), the renouncement of slice-structured coding provides a significantly higher average decoded PSNR. Initially, this is obviously surprising, as packet loss rate is still much lower when several slices are used and also the visual effects for a decoded frame when losing a single slice should be lower than in case of losing an entire frame. The first effect can indeed be observed from the cumulative distribution. The probability of bad frames (PSNR below 22 dB) is higher for $N_p = 1$ than for $N_p = 10$. However, in the case of no errors, the increased coding efficiency when not using slices provides many frames with significantly higher PSNR than for slice structuring. As we avoid error propagation, the correctly received frames are really error-free, which is not the case if we use intra updates. Therefore, if feedback information is available and several completely lost frames are tolerable, it is better to use no slice structured coding than harming the compression efficiency. For increased feedback delay $d = 4$, the curves are shifted to the left compared to feedback delay 2. However, the $d = 4$ and $N_p = 1$ performs almost as well as the best case without feedback. Therefore, in the case of available feedback, this very simple system without considering expected decoder distortion and slice structuring and just relying on multiple reference frames outperforms many highly sophisticated error-resilience schemes as long as the delay of the feedback is reasonable. The combination of these methods according to feedback mode 3 is currently investigated and should allow adaptively selecting the best methods, however, with significantly increased encoding complexity.

VI. Conclusions

H.264/AVC promises some significant advances of the state-of-the-art of standardized video coding in mobile applications. In addition to excellent coding efficiency, the design of H.264/AVC also takes into account network adaptation providing large flexibility for its use in wireless applications. The tools provided in H.264/AVC for error resilience do not necessarily differ from the compression efficiency features such as intra MBs or multiple reference frames. However, in the case of error-prone transmission, the selection methods have to be changed by using the expected decoder distortion or by restricting the set of accessible coding options. In experimental results based on common test conditions, it has been shown that in case without any feedback, several slices in combination with channel-adaptive R-D optimized mode selection is a promising approach. In this case, further investigation with advanced error-resilience tools such as flexible MB ordering, data partitioning, and generic forward error correction, might provide benefits. However, in the case of available feedback, the application of multiple reference frames to exclude error propagation without slice structuring provides excellent results.

Acknowledgment

The authors would like to thank T. Oelbaum, D. Kontopodis, Y.-K. Wang, V. Varsa, and A. Hourunranta for implementing and testing parts of the algorithms, V. Varsa and G. Liebl for providing the test conditions and software simulator, S. Wenger, N. Färber, K. Stuhlmüller, E. Steinbach, and B. Girod for useful discussions, and JVT for the collaborative work and the technically outstanding discussions and contributions.

References

[1] "Final committee draft: Editor's proposed revisions," in *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG*, T. Wiegand, Ed., Feb. 2003, JVT-F100.

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, July 2003.

Google Exhibit No. 1027
Page 015

Google Exhibit No. 1063
Page 0967

[3] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 688–703, July 2003.
[4] "Video Coding for Low Bitrate Communication, Version 1," ITU-T, ITU-T Recommendation H.263, 1995.
[5] "Coding of Audio-Visual Objects—Part 2: Visual," ISO/IEC JTC1, ISO/IEC 14 496-2 (MPEG-4 visual version 1), 1999.
[6] "Multimedia Messaging Service (MMS); Media Formats and Codecs," 3GPP Technical Specification 3GPP TR 26.140.
[7] S. Wenger, M. Hannuksela, and T. Stockhammer, "Identified H.26L Applications," ITU-T SG 16, Doc. VCEG-L34, Eibsee, Germany, 2001.
[8] "Codec for Circuit Switched Multimedia Telephony Service; General Description," 3GPP Technical Specification 3GPP TR 26.110.
[9] "Packet Switched Conversational Multimedia Applications; Default Codecs," 3GPP Technical Specification 3GPP TR 26.235.
[10] "Transparent End-to-End Packet Switched Streaming Service (PSS); RTP Usage Model," 3GPP Technical Specification 3GPP TR 26.937.
[11] "Multimedia Messaging Service (MMS); Media Formats and Codecs," 3GPP Technical Specification 3GPP TR 26.140.
[12] "Multimedia Broadcast/Multicast Services," 3GPP Technical Specification 3GPP TR 29.846.
[13] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 704–716, July 2003.
[14] V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen, "Complexity of optimized H.264/AVC video decoder implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 717–725, July 2003.
[15] "High Speed Downlink Packet Access (HSDPA); Overall UTRAN Description," 3GPP Technical Specification 3GPP TR 25.855.
[16] H. Holma and A. Toskala, Eds., *WCDMA for UMTS: Radio Access For Third Generation Mobile Communications*. New York: Wiley, 2000.
[17] *Narrow-Band Visual Telephone Systems and Terminal Equipment, Rev. 4*, ITU-T Recommendation H.320, 1999.
[18] "Generic Coding of Moving Pictures and Associated Audio Information," ISO/IEC International Standard 13 818, 1994.
[19] S. Wenger, T. Stockhammer, and M. M. Hannuksela, "RTP payload format for H.264 video," in *Internet Draft, Work in Progress*, Mar. 2003, Draft-wenger-avt-rtp-h264-01.txt.
[20] "3rd GPP; Technical Specification Group Core Network; IP Multimedia Call Control Protocol Based on SIP and SDP," 3GPP Technical Specification 3GPP TS 24.229.
[21] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 645–656, July 2003.
[22] D. Lindberg, "The H.324 multimedia communication standard," *IEEE Commun. Mag.*, vol. 34, pp. 46–51, Dec. 1996.
[23] H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "RObust header compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," in RFC 3095, July 2001.
[24] "Common conditions for video performance evaluation in H.324/M error-prone systems, VCEG (SG16/Q15)," in Ninth Meeting, Redbank, NJ, Oct. 1999, ITU-T Q15-I-60.
[25] G. Roth, R. Sjöberg, G. Liebl, T. Stockhammer, V. Varsa, and M. Karczewicz, "Common Test Conditions for RTP/IP Over 3GPP/3GPP2,", Austin, TX, ITU-T SG16 Doc. VCEG-M77, 2001.
[26] "Radio Link Control (RLC) Protocol Specification," 3GPP Technical Specification 3GPP TS 25.322.
[27] S. S. Hemami, "Robust image transmission using resynchronizing variable-length codes and error concealment," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 927–939, June 2000.
[28] J. Ribas-Corbera, P. A. Chou, and S. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 674–687, July 2003.
[29] H. Jenkac, T. Stockhammer, and G. Kuhn, "Streaming media in variable bit-rate environments," presented at the Packet Video Workshop, Nantes, France, Apr..
[30] B. Girod, M. Kalman, Y. J. Liang, and R. Zhang, "Advances in video channel-adaptive streaming," in ICIP 2002, Rochester, NY, Sept. 2002.
[31] E. G. Steinbach, N. Färber, and B. Girod, "Adaptive play-out for law latency video streaming," presented at the ICIP 2001, Thessaloniki, Greece, Oct. 2001.
[32] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, submitted for publication.
[33] Y. J. Liang and B. Girod, "Rate-distortion optimized low—Latency video streaming using channel-adaptive bitstream assembly," presented at the ICME2002, Lausanne, Switzerland, Aug. 2002.
[34] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, pp. 53–68, Mar. 2001.
[35] M. Karczewicz and R. Kurçeren, "The SP and SI frames design for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 637–644, July 2003.
[36] V. Varsa, M. M. Hannuksela, and Y. Wang, "Non-Normative Error Concealment Algorithms," ITU-T VCEG-N62, 2001.
[37] S. Wenger and G. Cotê, "Using RFC 2429 and H.263+ at low to medium bit-rates for low-latency applications," presented at the Proc. Packet Video Workshop, New York, NY, Apr. 1999.
[38] V. Varsa and M. Karczewicz, "Slice interleaving in compressed video packetization," presented at the Packet Video Workshop 2000, Forte Village, Italy, May 2000.
[39] S. Wenger and M. Horowitz, "Flexible MB ordering—A new error resilience tool for IP-based video," presented at the IWDC 2002, Capri, Italy, Sept. 2002.
[40] J. Rosenberg and H. Schulzrine, "An RTP payload format for generic forward error correction," in RFC 2733, Dec. 1999.
[41] G. Carle and E. W. Biersack, "Survey of error recovery techniques for IP-based audio-visual multicast applications," *IEEE Network Mag.*, vol. 11, pp. 2–14, Nov. 1997.
[42] U. Horn, K. Stuhlmüller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *IEEE Trans. Image Processing*, vol. 15, pp. 77–94, Sept. 1999.
[43] Q. F. Zhu and L. Kerofsky, "Joint source coding, transport processing, and error concealment for H.323-based packet video," *Proc. SPIE VCIP*, vol. 3653, pp. 52–62, Jan. 1999.
[44] P. Haskell and D. Messerschmitt, "Resynchronization of motion-compensated video affected by ATM cell loss," *Proc. IEEE ICASSP*, vol. 3, pp. 545–548, 1992.
[45] J. Liao and J. Villasenor, "Adaptive intra update for video coding over noisy channels," *Proc. ICIP*, vol. 3, pp. 763–766, Oct. 1996.
[46] S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error resilience support in H.263+," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 867–877, Nov. 1998.
[47] P. Salama, N. B. Shroff, and E. J. Delp, "Error concealment in encoded video," in *Image Recovery Techniques for Image Compression Applications*. Norwell, MA: Kluwer, 1998.
[48] W. M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," in *Proc. ICASSP*, vol. 5, Mar. 1993, pp. 417–420.
[49] V. Varsa, M. M. Hannuksela, and Y.-K. Wang, "Non-Normative Error Concealment Algorithms," ITU-T VCEG-N62, 2001.
[50] Y.-K. Wang, M. M. Hannuksela, V. Varsa, A. Hourunranta, and M. Gabbouj, "The error concealment feature in the H.26L test model," in *Proc. ICIP*, vol. 2, Sept. 2002, pp. 729–732.
[51] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.
[52] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Mag.*, vol. 15, pp. 74–90, Nov. 1998.
[53] T. Wiegand and B. Girod, "Lagrangian multiplier selection in hybrid video coder control," presented at the Proc. ICIP 2001, Oct. 2001.
[54] T. Stockhammer, D. Kontopodis, and T. Wiegand, "Rate-distortion optimization for H.26L video coding in packet loss environment," presented at the Packet Video Workshop 2002, Pittsburgh, PA, Apr. 2002.
[55] G. Cote, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error-prone networks," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 952–965, Dec. 2000.
[56] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod, "Error-resilient video transmission using long-term memory motion-compensated prediction," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1050-–1050-, Dec. 2000.
[57] C. W. Kim, D. W. Kang, and I. S. Kwang, "High-complexity mode decision for error prone environment," in JVT-C101, May 2002.
[58] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 966–976, 2000.
[59] Y.-K. Wang and M. M. Hannuksela, "Error-robust video coding using isolated regions," in JVT-C073, May 2002.
[60] Y.-K. Wang, M. M. Hannuksela, and M. Gabbouj, "Error-robust inter/intra mode selection using isolated regions," in Proc. Int. Packet Video Workshop 2003, Nantes, France, Apr. 2003.

Google Exhibit No. 1027
Page 016

Google Exhibit No. 1063
Page 0968

[61] E. Steinbach, N. Färber, and B. Girod, "Standard compatible extension of H.263 for robust video transmission in mobile environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 872–881, Dec. 1997.
[62] B. Girod and N. Färber, "Feedback-based error control for mobile video transmission," *Proc. IEEE*, vol. 97, pp. 1707–1723, Oct. 1999.
[63] W. Wada, "Selective recovery of video packet losses using error concealment," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 807–814, June 1989.
[64] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," *Proc. IEEE*, vol. 86, pp. 974–997, May 1998.
[65] T. Nakai and Y. Tomita, "Core Experiments on Feedback Channel Operation for H.263+," ITU-T SG15 LBC96-308, 1996.
[66] S. Fukunaga, T. Nakai, and H. Inoue, "Error resilient video coding by dynamic replacing of reference pictures," in *Proc. IEEE GLOBECOM*, vol. 3, Nov. 1996, pp. 1503–1508.
[67] Y. Tomita, T. Kimura, and T. Ichikawa, "Error resilient modified inter-frame coding system for limited reference picture memories," presented at the Picture Coding Symposium, Berlin, Germany, Sept. 1997.
[68] "Video Coding for Low Bit-Rate Communication, Version 1," ITU-T Recommendation H.263, 1995.
[69] T. Stockhammer, T. Wiegand, T. Oelbaum, and F. Obermeier, "Video coding and transport layer techniques for H.264-based transmission over packet-lossy networks," presented at the ICIP, Barcelona, Spain, Sept. 2003.
[70] T. Stockhammer and S. Wenger, "Standard-compliant enhancement of JVT coded video for transmission over fixed and wireless IP," presented at the IWDC 2002, Capri, Italy, Sept. 2002.



Thomas Stockhammer received the Diplom.-Ing. degree in electrical engineering in 1996 from the Munich University of Technology (TUM), Munich, Germany, where he is currently working toward the Dr.-Ing. degree in the area of source and video transmission over mobile and packet-lossy channels.

In 1996, he visited Rensselear Polytechnic Institute (RPI), Troy, NY to perform his diploma thesis in the area of combined source channel coding for video and coding theory. There he began research in video transmission and combined source and channel coding. In 2000, he was Visiting Researcher in the Information Coding Laboratory, University of San Diego at California (UCSD). Since then, he has published several conference and journal papers and holds several patents. He regularly participates and contributes to different standardization activities, e.g., ITU-T H.324, H.264, ISO/IEC MPEG, JVT, and IETF. He acts as a member of several technical program committees, as Reviewer for different journals, and as an Evaluator for the European Commission. His research interests include joint source and channel coding, video transmission, system design, rate-distortion optimization, information theory, and mobile communications.



Miska M. Hannuksela received the M.S. degree in engineering from Tampere University of Technology, Tampere, Finland, in 1997.

He is currently a Research Manager in the Visual Communications Laboratory, Nokia Research Center, Tampere, Finland. From 1996 to 1999, he was a Research Engineer with Nokia Research Center in the area of mobile video communications. From 2000 to 2003, he was a Project Team Leader and a specialist in various mobile multimedia research and product projects in Nokia Mobile Phones. He has been an active participant in the ITU-T Video Coding Experts Group since 1999 and in the Joint Video Team of ITU-T and ISO/IEC since its foundation in 2001. He has co-authored more than 80 technical contributions to these standardization groups. His research interests include video error resilience, scalable video coding, and video communication systems.



Thomas Wiegand received the Dr.-Ing. degree from the University of Erlangen-Nuremberg, Erlangen-Nuremberg, Germany, in 2000 and the Dipl.-Ing. degree in electrical engineering from the Technical University of Hamburg-Harburg, Hamburg-Harburg, Germany, in 1995.

He is the Head of the Image Communication Group in the Image Processing Department, Fraunhofer–Institute for Telecommunications – Heinrich Hertz Institute (HHI), Berlin, Germany. During 1997 to 1998, he was a Visiting Researcher at Stanford University, Stanford, CA, and served as a Consultant to 8x8, Inc., Santa Clara, CA. From 1993 to 1994, he was a Visiting Researcher at Kobe University, Kobe, Japan. In 1995, he was a Visiting Scholar at the University of California at Santa Barbara, where he began his research on video compression and transmission. Since then, he has published several conference and journal papers on the subject and has contributed successfully to the ITU-T Video Coding Experts Group (ITU-T SG16 Q.6—VCEG)/ISO/IEC Moving Pictures Experts Group (ISO/IEC JTC1/SC29/WG11—MPEG)/Joint Video Team (JVT) standardization efforts and holds various international patents in this field. He has been appointed as the Associated Rapporteur of the ITU-T VCEG (October 2000), the Associated Rapporteur/Co-Chair of the JVT that has been created by ITU-T VCEG and ISO/IEC MPEG for finalization of the H.264/AVC video coding standard (December 2001), and the Editor of the H.264/AVC video coding standard (February 2002).

Google Exhibit No. 1027
Page 017

Google Exhibit No. 1063
Page 0969

Ex. 1027A

LIBRARY OF CONGRESS
0 011 601 020 3

IEEE
TRANSACTIONS
ON CIRCUITS
AND SYSTEMS
FOR VIDEO
TECHNOLOGY

13

2003

TK 6630
.A1 I37

IEEE TRANSACTIONS ON

CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY

A PUBLICATION OF THE IEEE CIRCUITS AND SYSTEMS SOCIETY

WWW.IEEE-CAS.ORG

JULY 2003 VOLUME 13 NUMBER 7 ITCTEM (ISSN 1051-8215)

SPECIAL ISSUE ON THE H.264/AVC VIDEO CODING STANDARD

GUEST EDITORIAL

Introduction to the Special Issue on the H.264/AVC Video Coding Standard . . . *A. Luthra, G. J. Sullivan, and T. Wiegand* 557

PAPERS

Overview of the H.264/AVC Video Coding Standard . . . . . *T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra* 560
Motion- and Aliasing-Compensated Prediction for Hybrid Video Coding . . . . . *T. Wedi and H. G. Musmann* 577
Generalized B Pictures and the Draft H.264/AVC Video-Compression Standard . . . . . *M. Flierl and B. Girod* 587
Low-Complexity Transform and Quantization in H.264/AVC . . . . . *H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky* 598
Variable Block-Size Transforms for H.264/AVC . . . . . *M. Wien* 604
Adaptive Deblocking Filter . . . . . *P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz* 614
Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard . . . . . *D. Marpe, H. Schwarz, and T. Wiegand* 620
The SP- and SI-Frames Design for H.264/AVC . . . . . *M. Karczewicz and R. Kurceren* 637
H.264/AVC Over IP . . . . . *S. Wenger* 645
H.264/AVC in Wireless Environments . . . . . *T. Stockhammer, M. M. Hannuksela, and T. Wiegand* 657
A Generalized Hypothetical Reference Decoder for H.264/AVC . . . *J. Ribas-Corbera, P. A. Chou, and S. L. Regunathan* 674
Rate-Constrained Coder Control and Comparison of Video Coding Standards . . . . . *T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan* 688
H.264/AVC Baseline Profile Decoder Complexity Analysis . . . . . *M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro* 704
Complexity of Optimized H.26L Video Decoder Implementation . . . *V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen* 717

COMMENTS AND CORRECTIONS

Corrections to "Genetic Algorithm Optimization of Multidimensional Grayscale Soft Morphological Filters With Applications in Film Archive Restoration . . . . . *M. S. Hamid, N. R. Harvey, and S. Marshall* 726

CALLS FOR PAPERS

ISCAS 2004 727
Special Issue on Integrated Multimedia Platforms 728

Google Exhibit No. 1063
Page 0972

INFORMATION FOR AUTHORS

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY covers all aspects of visual information technology, including but not limited to video A/D and D/A, display technology, image analysis and processing, video signal characterization and representation, image and video compression techniques, signal processing, multidimensional filters and transforms, analog video signal processing, neural networks for video applications, nonlinear video signal processing, video archiving and retrieval, computer vision, video transport, network video, streaming video, high-speed real-time circuits, synthetic video, object-based representation, MPEG technologies, VLSI architecture and implementation for video technology, multiprocessor systems—hardware and software, video systems architecture, video quality assessment, and error resilient video coding. The TRANSACTIONS consists of two sections.

1) *Transactions Papers*—This section is intended for self-contained articles that provide new and significant contributions to the field covered by the TRANSACTIONS, with coherent coverage of theoretical background, technical contributions, practical applications, and experimental results.
2) *Transactions Letters*—This section is intended for short articles that describe new and significant contributions to the field covered by the TRANSACTIONS, to be published in the most expeditious manner. This category includes: i) disclosure of potentially important ideas with convincing and encouraging results; ii) a condensed version of a submitted Transactions Paper; and iii) updates and correspondence on previously published papers.

Submission of a contribution is taken to manifest the fact that the submission has not been submitted, accepted, published, or copyrighted elsewhere. However, manuscripts submitted to or accepted by this Transactions may also be submitted for presentation at the IEEE International Symposium on Circuits and Systems or other conferences sponsored by the IEEE Circuits and Systems Society.

A. Submission of a Transactions Paper

1) Only electronic submissions of Transactions Papers via e-mail will be accepted. To submit a Transactions Paper, send an e-mail with subject "Transactions Paper" to the address: submission_tcsvt@hhi.de with the manuscript contained as an attachment strictly in PDF format, readable using Adobe Acrobat Reader. The body of the e-mail as well as the manuscript itself should contain the following information: Manuscript Type, Transactions Paper, Color illustrations: Yes/No, Author Name(s), Author Institution(s), Author E-mail(s), Paper Title, Abstract, Key Words, and any comments to the Editor-in-Chief.
2) Manuscript should state the significance of the problem in the Introduction.
3) The total length of a manuscript should not exceed ten (10) pages when printed using IEEE two-column format including figures (approximately 30 double-spaced typewritten pages of 8 1/2 × 11). The style files for LaTeX and Word can be downloaded from http://www.ieee.org/organizations/pubs/authors.html. The final submission format must be PDF.
4) Originals for the illustrations should be ready to be submitted immediately upon acceptance of the Transactions Paper.
5) See instructions on the web site for completing the IEEE copyright form for the manuscript.
6) If the manuscript has been presented, published, or submitted for publication elsewhere, please so inform the Editor-in-Chief in the comment form. Our primary objective is to publish technical material not available elsewhere, but on occasion we publish papers of unusual merit which have appeared or will appear before other audiences.

B. Submission of a Transactions Letter

1) Only electronic submissions of Transactions Letters via e-mail will be accepted. To submit a Transactions Letter, send an e-mail with subject "Transactions Letter" to the address: submission_tcsvt@hhi.de with the manuscript contained as an attachment strictly in PDF format, readable using Adobe Acrobat Reader. The body of the e-mail as well as the manuscript itself should contain the following information: Manuscript Type, Transactions Letter, Color illustrations: Yes/No, Author Name(s), Author Institution(s), Author E-mail(s), Paper Title, Abstract, Key Words, and any comments to the Editor-in-Chief.
2) Manuscript should state the significance of the problem in the Introduction.
3) The total length of a manuscript should not exceed five (5) pages when printed using IEEE two-column format including figures (approximately 15 double-spaced typewritten pages of 8 1/2 x 11). The style files for LaTeX and Words can be downloaded from http://www.ieee.org/organizations/pubs/authors.html. The final submission format must be PDF.
4) Originals for the illustrations should be ready to be submitted immediately upon acceptance of the Transactions Letter.
5) See instructions on the web site for completing the IEEE copyright form for the manuscript.
6) If the manuscript has been presented, published, or submitted for publication elsewhere, please so inform the Editor-in-Chief in the comment form. Our primary objective is to publish technical material not available elsewhere, but on occasion we publish papers of unusual merit which have appeared or will appear before other audiences.

C. Style for Manuscript

1) The manuscript should be typewritten using double space.
2) Provide an informative abstract of 100 to 250 words at the beginning of the manuscript.
3) Provide a separate double-spaced page listing all footnotes. Acknowledgment of financial support is often placed at the end of the footnote.
4) References should appear in a separate reference section at the end of the paper, with items referred to by numerals in square brackets, e.g., [12]. References should be complete and in the IEEE style. Style for papers: author, first initial followed by last name, title, volume, inclusive page numbers, month, and year. Style for books: author, title, location and publisher, year, chapter or page numbers (if desired). See any recent issues for examples.
5) Provide a separate page listing all figure captions, in proper style for the typesetter, e.g., "Fig. 6. The final wide-band amplifier design."
6) For further information, consult *Information for IEEE Transactions and Journal Authors*, available online at http://www.ieee.org/organizations/pubs/transactions/information.htm, or by e-mail to trans@ieee.org.

D. Style for Illustrations

1) Originals for illustrations should be sharp, noise-free, and of good contrast. We regret that we cannot provide drafting or art service.
2) Line drawings should be in India ink on drafting cloth, paper, or board. Use only 8 1/2 × 11 size sheets to simplify handling of the manuscript.
3) On graphs, show only the coordinate axes, or at most the major grid lines, to avoid a dense hard-thread result.
4) All lettering should be large enough to permit legible reduction of the figure to column width, perhaps as much as 4:1. Typing on figures is not acceptable.
5) Photographs should be glossy prints, of good contrast and gradation, and any reasonable size.
6) Number each original on the back, or at the bottom of the front, and also indicate the author's name.
7) Note C-5 above. Captions lettered on figures will be blocked out in production, in favor of typeset captions.
8) Electronic art, submited by disk, email, or CD, must be in tif, ps, or eps format. (This includes author photos.)

E. Page and Color Photo Charges

After a manuscript has been accepted for publication, the author's institution will be approached with a request to pay a voluntary charge of $110 per page to cover part of the cost of publication. If the voluntary page charge is not honored, IEEE will charge a mandatory page charge of $175 per page for each page in excess of ten (10) for a TRANSACTIONS PAPER or in excess of five (5) for a TRANSACTIONS LETTER. The author will receive 100 free reprints (without cover) only if the voluntary page charge is honored. Detailed instructions will accompany the page proofs. Color photos judged to be essential to show the results of a paper may be printed in color. The cost of printing color photos will be shared by the author(s) and the Society.

F. Copyright

It is the policy of the IEEE to own the copyright to the technical contributions it publishes on behalf of the interests of the IEEE, its authors, and their employers and to facilitate the appropriate reuse of this material by others. To comply with the U.S. Copyright Law, authors are required to sign an IEEE Copyright Form before publication. This form returns to authors and their employers full rights to reuse their material for their own purposes. Authors must submit a signed copy of this form with their manuscript.

G. Special Issues

Special issues of this TRANSACTIONS may be proposed to the Editor-in-Chief by any individual when a new, unique, exciting, and/or timely subject area is identified.

Google Exhibit No. 1063
Page 0973

IEEE CIRCUITS AND SYSTEMS SOCIETY

The Circuits and Systems Society is an association of IEEE members with professional interest in the field of circuits and systems theory. All members of the IEEE are eligible for membership in the Society and will receive this TRANSACTIONS upon payment of the annual Society membership fee of $12.00, plus an annual subscription fee of $18.00. For information on joining, write to the IEEE at the address below. *Member copies of Transactions/Journals are for personal use only.*

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY

Editor-in-Chief
THOMAS SIKORA
Inst. for Telecommunications
Communication Systems Dept.
Technical Univ. Berlin
Sekr. EN-1, Einsteinufer 17
10587 Berlin, Germany
Phone: +1 49 30 314 25799
Fax: +1 49 30 314 22514
E-mail: sikora@nue.tu-berlin.de
http://www.nue.tu-berlin.de

Past Editor-in-Chief
WEIPING LI
WebCast Technologies, Inc.
1153 Bordeaux Dr., Ste. 205
Sunnyvale, CA 94089
Phone: +1 408 541 4150
Fax: +1 408 745 0249
E-mail: li@webcasttech.com

Associate Editors

ISHFAQ AHMAD
Dept. Comput. Sci. & Eng.
Univ. Texas at Arlington
248 D/E Nedderman Hall
416 Yates St., Box 19015
Arlington, TX 76019-0015
iahmad@cse.uta.edu

KIYOHARU AIZAWA
Dept. Elect. Eng.
Univ. Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113 Japan
aizawa@ee.t.u-tokyo.ac.jp

OSAMA K. AL-SHAYKH
PacketVideo Technologies
10350 Science Ctr. Dr., Ste. 140
San Diego, CA 92121
osama@packetvideo.com

JOHN F. ARNOLD
Schl. Elect. Eng.
Univ. College, UNSW
Australian Defence Force Academy
Canberra ACT 2600, Australia
j.arnold@adfa.edu.au

RAJARATHNAM CHANDRAMOULI
Dept. Elect. Comput. Eng.
207 Burchard Building
Stevens Inst. Technol.
Hoboken, NJ 07030
rchandr1@stevens-tech.edu

CHANG WEN CHEN
Dept. Elect. Comput. Eng.
Florida Inst. of Technol.
Melbourne, FL 32091
cchen@fit.edu

LIANG-GEE CHEN
Dept. Elect. Eng.
National Taiwan Univ.
Taipei, Taiwan 10764 R.O.C.
lgchen@cc.ee.ntu.edu.tw

SHERMAN CHEN
Broadband Systems Engineering
Broadcom Corp.
16215 Alton Parkway
P.O. Box 57013
Irvine, CA 92619-7013
schen@broadcom.com

LEONARDO CHIARIGLIONE
Multimedia & Video Services
CSELT
Via G. Reiss Romoli, 274
Torino I-10148 Italy
leonardo.chiariglione@cselt.it

HAMID GHARAVI
NIST
U.S. Dept. Commerce
100 Bureau Dr., Stop 8920
Gaithersburg, MD 20899-8920
gharavi@antd.nist.gov

JENQ-NENG HWANG
Dept. Elect. Eng.
Univ. Washington
Seattle, WA 98195
hwang@ee.washington.edu

EBROUL IZQUIERDO
Elect. Eng. Dept.
Queen Mary, Univ. of London
London E1 4NS, UK
ebroul.izquierdo@elec.qmul.ac.uk

ROB KOENEN
InterTrust Technologies Corp.
4750 Patrick Henry Dr.
Santa Clara, CA 95054
rob.koenen@ieee.org

ALEX KOT
Schl. Elect. Electron. Eng.
Nanyang Technol. Univ.
S1, Nanyang Ave.
Singapore 639798
eackot@ntu.edu.sg

S. Y. KUNG
Dept. Elect. Eng.
Princeton Univ.
Princeton, NJ 08544
kung@princeton.edu

MURAT KUNT
Dept. Elect. Eng.
Swiss Federal Inst. Technol.
Lausanne CH-1015 Switzerland
kunt@epfl.ch

ROSA LANCINI
CEFRIEL
Via Fucini, 2
Milano 20133 Italy
rosa@cefriel.it

SANG UK LEE
School Elect. Eng.
Seoul National Univ.
Kwanak, P.O. Box 34
Seoul 151-742 Korea
sanguk@sting.snu.ac.kr

JENS-RAINER OHM
Inst. Commun. Eng.
Aachen Univ. Technol.
Melatener Str. 23
Aachen D-52072, Germany
ohm@ient.rwth-aachen.de

LEVENT ONURAL
Elect. & Electron. Eng. Dept.
Bilkent Univ.
TR-06533 Ankara, Turkey
l.onural@ieee.org

SETHURAMAN PANCHANATHAN
Dept. Comput. Sci. Eng.
Arizona State Univ.
Tempe, AZ 85287-5406
panch@asu.edu

FERNANDO PEREIRA
Dept. Elect. Comput. Eng.
Instituto Superior Tecnico
Av. Rovisco Pais
Lisboa Codex 1096 Portugal
fp@lx.it.pt

ERIC PETAJAN
face2face animation, inc.
2 Kent Place Blvd.
Summit, NJ 07901
eric@f2f-inc.com

ATUL PURI
Apple Computer
2 Infinite Loops
MS 302-3KS
Cupertino, CA 95014
apuri@apple.com
apuri@ieee.org

HARRY SHUM
Microsoft Research Asia
3/F Beijing Sigma Center
Beijing 100080, China
hshum@microsoft.com

ROBERT L. STEVENSON
Dept. Elect. Eng.
Univ. Notre Dame
Notre Dame, IN 46556
Stevenson.1@nd.edu

MICHAEL G. STRINTZIS
Elect. Comput. Eng. Dept.
Univ. Thessaloniki
Thessaloniki 540 06 Greece
strintzi@eng.auth.gr

HUIFANG SUN
Mitsubishi Electric Research Labs
558 Central Ave.
New Providence, NJ 07974
hsun@merl.com

ALI TABATABAI
Sony Electronics
3300 Zanker Rd., MD SJ2C4
San Jose CA 95134
ali.tabatabai@am.sony.com

QI TIAN
Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613
tian@lit.org.sg

KOU-HU TZOU
Divio, Inc.
441 DeGuigne Drive
Sunnyvale, CA 94085
ktzou@divio.com

HIROSHI WATANABE
Visual Commun. Lab. 922A
NTT Human Interface Labs.
1-1 Hikarinooka
Yokosuka 239 Japan
hiroshi@giti.waseda.ac.jp

ZIXIANG XIONG
Dept. Elect. Eng.
Texas A&M Univ.
College Station, TX 77843
zx@lena.tamu.edu

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Officers

MICHAEL S. ADLER, *President*
ARTHUR W. WINSTON, *President-Elect*
LEVENT ONURAL, *Secretary*
PEDRO A. RAY, *Treasurer*
RAYMOND D. FINDLAY, *Past President*
JAMES M. TIEN, *Vice President, Educational Activities*
MICHAEL R. LIGHTNER, *Vice President, Publication Services and Products*
W. CLEON ANDERSON, *Vice President, Regional Activities*
GERALD H. PETERSON, *President, IEEE Standards Association*
LEAH H. JAMIESON, *Vice President, Technical Activities*
JAMES V. LEONARD, *President, IEEE-USA*
CARY Y. YANG, *Director, Division I—Circuits and Devices*

Executive Staff

DANIEL J. SENESE, *Executive Director*
DONALD CURTIS, *Human Resources*
ANTHONY DURNIAK, *Publications Activities*
JUDITH GORMAN, *Standards Activities*
CECELIA JANKOWSKI, *Regional Activities*
BARBARA COBURN STOLER, *Educational Activities*
MATTHEW LOEB, *Corporate Strategy & Communications*
RICHARD D. SCHWARTZ, *Business Administration*
W. THOMAS SUTTLE, *IEEE-USA*
MARY WARD-CALLAN, *Technical Activities*
JOHN WITSKEN, *Information Technology*

IEEE Periodicals
Transactions/Journals Department
Staff Director: FRAN ZAPPULLA
Editorial Director: DAWN MELLEY *Production Director*: ROBERT SMREK
Managing Editor: MONA MITTRA *Senior Editor*: JILL A. SINAY

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (ISSN 1051-8215) is published monthly by the Institute of Electrical and Electronics Engineers, Inc. Responsibility for the contents rests upon the authors and not upon the IEEE, the Society/Council, or its members. **IEEE Corporate Office:** 3 Park Avenue, 17th Floor, New York, NY 10016-5997. **IEEE Operations Center:** 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. **NJ Telephone:** +1 732 981 0060. **Price/Publication Information:** Individual copies: IEEE Members $10.00 (first copy only), nonmembers $20.00 per copy. (Note: Add $4.00 postage and handling charge to any order from $1.00 to $50.00, including prepaid orders.) Member and nonmember subscription prices available upon request. Available in microfiche and microfilm. **Copyright and Reprint Permissions:** Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. For all other copying, reprint, or republication permission, write to Copyrights and Permissions Department, IEEE Publications Administration, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. Copyright © 2003 by The Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Periodicals Postage Paid at New York, NY and at additional mailing offices. **Postmaster:** Send address changes to IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. GST Registration No. 125634188. Printed in U.S.A.

LIBRARY OF CONGRESS NOV 0 8 2003

Introduction to the Special Issue on the H.264/AVC Video Coding Standard

OVER THE LAST one and a half decades, digital video and audio compression technologies have fundamentally changed the way we create, communicate, and consume audio-visual information. They have not only transformed existing applications and services like the distribution of entertainment video to the home but also spawned brand new industries and services like video-conferencing, direct-to-home satellite distribution, digital video recording, video-on-demand services, high-definition TV, video on mobile devices, streaming video, etc.

Since the early 1990s, when the technology was in its infancy, international video coding standards such as H.261, MPEG-1, MPEG-2/H.262, H.263, and MPEG-4 (Part 2) have been powerful engines behind the commercial success of digital video compression. They have played pivotal roles in establishing the technology via providing interoperability among products developed by different manufacturers and at the same time allowing flexibility for ingenuity in optimizing and molding the technology to fit a given application and making the cost-performance tradeoffs well suited to particular requirements. They have provided much needed assurance to the content creators that their content will run everywhere and that they do not have to create and manage multiple copies of the same content to match the products of different manufacturers. They have permitted the economy of scale to allow steep reduction in cost for masses to be able to afford the technology. They have nurtured open interactions among experts from different companies to spurt exponential growth in innovation to keep pace with the implementation technology and the needs of the applications.

The subject of this Special Issue is the newest entry in the line of international video coding standards, referred to formally as ITU-T Recommendation H.264 and ISO/IEC MPEG-4 (Part 10) Advanced Video Coding (or referred to in short as H.264/AVC). H.264/AVC is the powerful and state-of-the-art video compression standard recently developed by the ITU-T/ISO/IEC Joint Video Team (JVT) consisting of experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG). As has been the case with past video coding standards, its design represents a delicate balance between coding gain, implementation complexity, and costs based on state of VLSI (ASICs and Microprocessors) design technology. It has brought the coding algorithm technology in synch with the VLSI technology that is available today or will be available in near future. Some algorithmic features that could not be considered in the past due to implementation complexity could now be accepted. New techniques developed since the time of the prior designs could be incorporated. Some features that provided some increase in coding gain but were too expensive to implement were not included. In the process, the H.264/AVC design emerged with an improvement in coding efficiency typically by a factor of two over MPEG-2—the most widely used video coding standard today—while keeping the cost within the acceptable range. Similar quality gains were shown relative to other prior designs for a very broad range of applications. Above all, this project has proven that the predictions of the demise of video compression algorithm innovations were little premature and that they are continuing to flourish vigorously.

This Special Issue opens with an overview paper about the H.264/AVC standard by Wiegand *et al.* As the title suggests, it provides a concise description of the most important tools used in the standard.

We then dive deeper into the algorithm. We start with two papers on inter-picture prediction and processing. Wedi and Musmann provide good insight into the workings of fractional-sample motion-compensated prediction in their paper "Motion and Aliasing Compensated Prediction for Hybrid Video Coding". Flierl and Girod describe how and why B-pictures work and some ways the B picture concept has been extended in H.264/AVC in their paper "Generalized B Pictures and the Draft H.264/AVC Video Compression Standard".

The next two papers focus on the transforms used and considered in the standardization process. The first of these, "Low-Complexity Transform and Quantization in H.264/AVC" by Malvar *et al.*, describes the integer DCT-like transform and quantization scaling methods adopted in the standard. This transform was designed with particular attention paid to the fact that real-number-based transform specification causes mismatches between encoders and decoders, and can be expensive to implement in products that do not have high end microprocessor or ASIC. The second paper, "Variable Block Size Transforms for H.264/AVC" by Wien, describes the effort spent by JVT on adapting the transform size to the motion estimation block size. This technology was considered to be very interesting but not fully mature or sufficiently clearly beneficial to coding efficiency, and a bit too difficult to implement for inclusion in the standard at this stage.

Deblocking filtering has been one of the centers of great discussions among the experts and has divided the video compression community for a long time. In H.264/AVC, years of past experience in this field have culminated into the development of a new and powerful design. In their paper "Adaptive Deblocking Filter" List *et al.* explain how the filter adapts itself based on the quantization parameter, different compression modes, and motion in a scene to be able to differentiate compression artifacts from the content of the scene. The result has been a filter design that has reasonable implementation complexity and provides a substantial benefit to both subjective and objective video quality.

Digital Object Identifier 10.1109/TCSVT.2003.815169

1051-8215/03$17.00 © 2003 IEEE

Google Exhibit No. 1063
Page 0975

H.264/AVC in Wireless Environments

Thomas Stockhammer, Miska M. Hannuksela, and Thomas Wiegand

Abstract—**Video transmission in wireless environments is a challenging task calling for high-compression efficiency as well as a network friendly design. Both have been major goals of the H.264/AVC standardization effort addressing "conversational" (i.e., video telephony) and "nonconversational" (i.e., storage, broadcast, or streaming) applications. The video compression performance of the H.264/AVC video coding layer typically provides a significant improvement. The network-friendly design goal of H.264/AVC is addressed via the network abstraction layer that has been developed to transport the coded video data over any existing and future networks including wireless systems. The main objective of this paper is to provide an overview over the tools which are likely to be used in wireless environments and discusses the most challenging application, wireless conversational services in greater detail. Appropriate justifications for the application of different tools based on experimental results are presented.**

Index Terms—**Error concealment, error-resilient video coding, H.264/AVC, multiple reference frames, rate-distortion optimization, video coding standards, wireless video transmission.**

I. Introduction

SINCE 1997, the ITU-T's Video Coding Experts Group (VCEG) has been working on a new video coding standard with the internal denomination H.26L. In late 2001, the Moving Picture Expert Group (MPEG) and VCEG decided to work together as a Joint Video Team (JVT), and to create a single technical design called H.264/AVC for a forthcoming ITU-T Recommendation H.264/AVC and for a new part of the MPEG-4 standard called AVC [1][1], [2]. Since the meeting in November 2002, the technical specification is frozen and the standard text and software have been finalized. The primary goals of H.264/AVC are *improved coding efficiency* and *improved network adaptation*. The syntax of H.264/AVC typically permits a significant reduction in bit rate [3] compared to all previous standards such as ITU-T Rec. H.263 [4] and ISO/IEC JTC 1 MPEG-4 [5] at the same quality level.

The demand for fast and location-independent access to multimedia services offered on today's Internet is steadily increasing. Hence, most current and future cellular networks, like GSM-GPRS, UMTS, or CDMA-2000, contain a variety of packet-oriented transmission modes allowing transport of practically any type of IP-based traffic to and from mobile terminals, thus providing users with a simple and flexible transport interface. The third generation partnership project (3GPP) has selected several multimedia codecs for the inclusion into its multimedia specifications [6]. To provide basic video service in the first release of the 3G wireless systems, the well-established and almost identical baseline H.263 and the MPEG-4 visual simple profile have been integrated. The choice was based on the manageable complexity of the encoding and decoding process, as well as on the maturity and simplicity of the design.

However, due to the likely business models in emerging wireless systems in which the end-user's costs are proportional to the transmitted data volume and also due to limited resources bandwidth and transmission power, compression efficiency is the main target for wireless video and multimedia applications. This makes H.264/AVC coding an attractive candidate for all wireless applications including multimedia messaging services (MMS), packet-switched streaming services (PSS), and conversational applications. However, to allow transmission in different environments, not only is coding efficiency relevant, but also seamless and easy integration of the coded video into all current and possible future protocol and multiplex architectures. In addition, for conversational applications the video codec's support of enhanced error-resilience features is of major importance. This has also been taken into account in the standardization of this codec.

This paper is structured as follows. Section II introduces applications and transmission characteristics for wireless video applications. The transport of H.264/AVC video is briefly discussed and common test conditions for mobile video transmission are presented. Section III provides an overview over the H.264/AVC video coding standard from the perspective of wireless video applications. We categorize features according to their applicability in different video services. Section IV discusses the most challenging application in terms of delay constraints and error resilience, namely wireless conversational applications. A system description and problem formulation is followed by providing several alternatives on the system design using H.264/AVC as well as the combination of several modes. Section V provides experimental results for selected system concepts based on the common test conditions.

II. Video in Mobile Networks

A. Overview: Applications and Constraints

Video transmission for mobile terminals is likely to be a major application in emerging 3G systems and may be a key factor in their success. The video-capable display on mobile devices paves the road to several new applications. Three

Manuscript received April 9, 2002; revised May 10, 2003.

T. Stockhammer is with the Institute for Communications Engineering (LNT), Munich University of Technology (TUM), 80290 Munich, Germany (e-mail: stockhammer@ei.tum.de).

M. M. Hannuksela is with the Nokia Corporation, 33721 Tampere, Finland (e-mail: miska.hannuksela@nokia.com).

T. Wiegand is with the Fraunhofer-Institute for Telecommunications—Heinrich-Hertz-Institute Einsteinufer 37, 10587 Berlin, Germany (e-mail: wiegand@hhi.de).

Digital Object Identifier 10.1109/TCSVT.2003.815167

[1]All referenced standard documents can be accessed via anonymous ftp at ftp://standard.pictel.com/video_site, ftp://ftp.imtc-files.org/jvt-experts, ftp://ftp.ietf.org/, or ftp://www.3gpp.org/Specs/archive.

1051-8215/03$17.00 © 2003 IEEE

This material may be protected by Copyright law (Title 17 U.S. Code)

Google Exhibit No. 1063
Page 0976

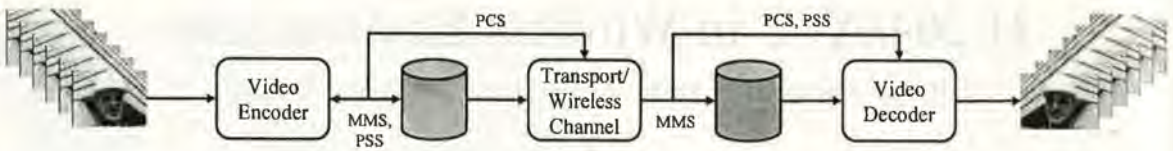


Fig. 1. Wireless video application MMS, PSS, and PCS: differentiation by real-time or offline processing for encoding, transmission, and decoding.

major service categories were identified in the H.264/AVC standardization process [7].

1) Circuit-switched [8] and packet-switched conversational services (PCS) [9] for video telephony and conferencing.
2) Live or pre-recorded video packet-switched streaming services (PSS) [10].
3) Video in multimedia messaging services (MMS) [11].

Although new services such as multimedia broadcast/multicast services (MBMS) [12] are planned for future releases of wireless networks, we restrict ourselves to single receiver applications. Mobile devices are hand-held and constrained in processing power and storage capacity. Therefore, a mobile video codec design must minimize terminal complexity while remaining consistent with the efficiency and robustness goals of the design. As complexity issues are discussed elsewhere in this issue [13], [14], we restrict ourselves to transmission constraints and properties.

The transmission requirements for the three identified applications can be distinguished with respect to requested data rate, the maximum allowed end-to-end delay, and the maximum delay jitter. This results in different system architectures for each of these applications. A simplified illustration is provided in Fig. 1. As MMS does not include any real-time constraints, encoding, transport, and decoding are completely separated. The recorded video signal is offline encoded and locally stored. The transmission is started using the stored signal at any time. The decoding process at the receiver is in general not started until the completion of the download. In PSS applications, the user typically requests pre-coded sequences, which are stored at a server. Whereas encoding and transmission are separated, decoding and display is started during transmission to minimize the initial delay and memory usage in mobile devices. Finally, in conversational services, the end-to-end delay has to be minimized to avoid any perceptual disturbances and to maintain synchronicity of audio and video. Therefore, encoding, transmission and decoding is performed simultaneously in real-time and, moreover, in both directions. These different ancillary conditions permit and require different strategies in encoding, transport, decoding, as well as in the underlying network and control architecture.

In general, the available bandwidth and therefore the bit-rate over the radio link are limited and the costs for a user are expected to be proportional to the reserved bit rate or the number of transmitted bits over the radio link. Thus, low bit rates are likely to be typical, and *compression efficiency* is the main requirement for a video coding standard to be successful in a mobile environment. This makes H.264/AVC a prime candidate for the use in wireless systems, because of its superior compression efficiency [3].

In addition, the mobile environment is characterized by harsh transmission conditions in terms of attenuation, shadowing, fading, and multi-user interference, which result in time- and location-varying channel conditions. The frequency of the channel variations highly depends on the environment, the user topology, the velocity of the mobile user, and the carrier frequency of the signal. For sufficiently long code words averaging over channel statistics is possible and transmission strategies can be used that are based on the long-term averages of fading states and the ergodic behavior of the channel. Many highly sophisticated radio link features such as broadband access, diversity techniques, space-time coding, multiple antenna systems, fast power control, interleaving, and forward error correction (FEC) by Turbo codes are used in 3G systems to reduce variations in channel conditions. However, only for fast-moving users and relatively large tolerated maximum delay can these advanced techniques provide a negligible bit error and radio block loss rate. Usually, some amount of residual errors has to be tolerated for low-delay applications due to the nonergodic behavior of the channel and the imperfectness of the applied signal processing. Therefore, in addition to high compression efficiency and reasonable complexity, a video coding standard to be applicable for conversational services in wireless environments has to be error resilient.

In addition, it is worth noting at this point that new directions in the design of wireless systems do not necessarily attempt to minimize the error rates in the system, but to maximize the throughput. This is especially appealing for services with relaxed delay constraints such as PSS and MMS. The nonergodic behavior of the channel is exploited such that in case of good channel states significantly higher data rate is supported than in bad channel states. In addition, reliable link layer protocols with persistent Automatic Repeat reQuest (ARQ) are usually used to guarantee error-free delivery. For example, in the high-speed downlink packet access (HSDPA) concept [15] ARQ, adaptive modulation schemes, and multiuser scheduling taking into account the channel states are combined to significantly enhance the throughput in wireless systems.

3G wireless transmission stacks usually consist of two different bearer types, dedicated and shared channels. Whereas in dedicated channels one user gets assigned a fixed data rate for the entire transmission interval, shared channels allow a dynamic bit-rate allocation similar to ATM or GSM GPRS. HSDPA will be an extension of the shared channel concept on the air interface. Except for MMS all streaming and conversational applications are assumed to use dedicated channels in the initial phase of 3G wireless systems due to their almost constant bit-rate behavior. In modern system designs, an application can request one of many different quality-of-service (QoS) classes. QoS classes contain parameters like a maximum error rate,

Google Exhibit No. 1063
Page 0977

TABLE I
QoS SERVICE CLASSES IN PACKET RADIO SYSTEMS [15]

Traffic Class	Fundamental Characteristics	Typical Examples
Conversational	Preserve time relation between information entities of the stream Conversational pattern (stringent and low delay)	Voice and video telephony, Video conferencing
Streaming	Preserve time relation (variation) between information entities of the stream	Streaming multimedia (video, audio, etc.)
Interactive	Request response pattern, Preserve data integrity	Web browsing, network games
Background	Destination is not expecting the data within a certain time Preserve data integrity	Background download of e-mails, files, etc.

maximum delay, and a guaranteed maximum data rate. Furthermore, according to [16], applications are usually divided into different service classes: conversational, streaming, interactive, and background traffic. Characteristics and typical examples are shown in Table I.

B. Transport of H.264/AVC Video in Wireless Systems

According to Fig. 2, H.264/AVC distinguishes between two different conceptual layers, the video coding layer (VCL) and the network abstraction layer (NAL). Both the VCL and the NAL are part of the H.264/AVC standard. The VCL specifies an efficient representation for the coded video signal. The NAL of H.264/AVC defines the interface between the video codec itself and the outside world. It operates on NAL units which give support for the packet-based approach of most existing networks. At the NAL decoder interface, it is assumed that the NAL units are delivered in decoding order and that packets are either received correctly, are lost, or an error flag in the NAL unit header can be raised if the payload contains bit errors. The latter feature is not part of the standard as the flag can be used for different purposes. However, it provides a way to signal an error indication through the entire network. Additionally, interface specifications are required for different transport protocols that will be specified by the responsible standardization bodies. The exact transport and encapsulation of NAL units for different transport systems, such as H.320 [17], MPEG-2 Systems [18], and RTP/IP [19], are also outside the scope of the H.264/AVC standardization. The NAL decoder interface is normatively defined in the standard, whereas the interface between the VCL and the NAL is conceptual and helps in describing and separating the tasks of the VCL and the NAL.

For real-time video services over 3G mobile networks, two protocol stacks are of major interest. 3GPP has specified a multimedia telephony service for circuit-switched channels [8] based on ITU-T Recommendation H.324M. For IP-based packet-switched communication, 3GPP has chosen to use SIP and SDP for call control [20] and RTP for media transport [9]. In other words, the IP-based protocol stack as presented in [21] will be used in packet-switched 3G mobile services. While the H.324 and the RTP/UDP/IP stacks have different roots and a completely different switching philosophy, the loss and delay effects on the media data when transmitting over wireless dedicated channels are very similar.

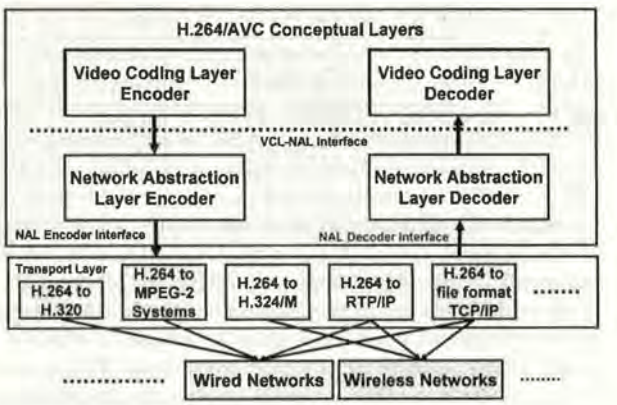


Fig. 2. H.264/AVC standard in transport environment.

H.324 [22] was primarily established by the ITU-T for low-bit-rate circuit-switched modem links and error prone extensions for mobile circuit switched low-bit-rate conversational services have been added. H.324M officially known as ITU-T Rec. H.324 Annex C, allows transmission over low, moderate, and high bit-error rate circuit switched links. 3GPP adopted H.324M including an error robust extension of the multiplexing protocol H.223 known as H.223 Annex B as the protocol used for circuit-switched video communication. This multiplexing protocol includes two layers: an error-resilient packet-based multiplex layer and an adaptation layer featuring common error detection capabilities, such as sequence numbering and cyclic redundancy checks (CRSs). Therefore, it is very similar to the RTP/UDP/IP stack (see [21]).

For packet-switched services, 3GPP/3GPP2 agreed on an IP-based stack. Fig. 3 shows a typical packetization of a NAL unit encapsulated in RTP/UDP/IP [19] through the 3GPP2 user plane protocol stack. After robust header compression (RoHC) [23] this IP/UDP/RTP packet is encapsulated into one packet data convergence protocol/point-to-point protocol (PDCP/PPP) packet that becomes a radio link control (RLC)-service data unit (SDU). The RLC protocol can operate in three modes: 1) transparent; 2) unacknowledged; and 3) acknowledged mode [26]. The RLC protocol provides segmentation and retransmission services for both users and control data. The transparent and unacknowledged mode RLC entities are defined to be

Google Exhibit No. 1063
Page 0978

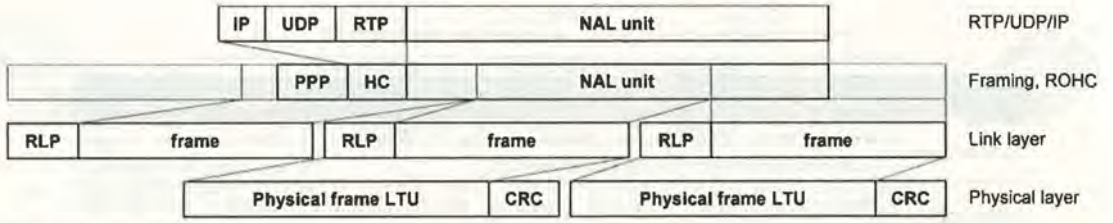


Fig. 3. Packetization through 3GPP2 user plane protocol stack.

TABLE II
BIT-ERROR PATTERNS

No.	Bit rate	Length	BER	Mobile Speed	Application
1	64 kbit/s	60 s	9.3e-3	3 km/h	Streaming
2	64 kbit/s	60 s	2.9e-3	3 km/h	Streaming
3	64 kbit/s	180 s	5.1e-4	3 km/h	Conversational
4	64 kbit/s	180 s	1.7e-4	50km/h	Conversational
5	128 kbit/s	180 s	5.0e-4	3 km/h	Conversational
6	128 kbit/s	180 s	2.0e-4	50km/h	Conversational

unidirectional and acknowledged mode entities are described as bi-directional. For all RLC modes, CRC error detection is performed on the physical layer and the result of the CRC check is delivered to the RLC together with the actual data. In the transparent mode no protocol overhead is added to higher layer data. Erroneous protocol data units (PDUs) can be discarded or marked erroneous. In the unacknowledged mode, no retransmission protocol is in use and data delivery is not guaranteed. Received erroneous data is either marked or discarded depending on the configuration. In the acknowledged mode, an automatic repeat request mechanism is used for backward error correction.

As video packets are of varying length by nature, the length of RLC-SDU's varies as well. If an RLC-SDU is larger than an RLC-PDU, the SDU is segmented into several PDUs. In the used, unacknowledged, and acknowledged modes, the flow of variable-size RLC-SDUs is continuous to avoid padding of bits as necessary for the transparent mode. In unacknowledged mode, if any of the RLC-PDUs containing data from a certain RLC-SDU have not been received correctly, the RLC-SDU is typically discarded. In acknowledged mode, the RLC/radio link protocol (RLP) layer can perform re-transmissions.

Additionally, both protocol stacks H.324 and RTP/IP/UDP use reliable setup and control protocols, H.245 and SIP, respectively. Hence, it can be assumed that a small amount of control information can be transported out-of-band in a reliable way. The resulting properties of real-time low-delay video transmission are therefore very similar in both cases. Packets are transmitted over underlying transports protocols and channels, which provide framing, encapsulation, error detection, and reliable setup. We focus hereafter on the RTP/IP-based transmission over wireless channels [21].

C. Common Test Conditions for Wireless Video

In the H.264/AVC standardization process, the importance of mobile video transmission has been recognized by adopting appropriate common test conditions for 3G mobile transmission for circuit switched conversational services based on H.324M [24] and for packet-switched conversational and streaming services [25]. These test conditions permit the selection of appropriate coding features, testing and evaluating error-resilience features, as well as meaningful anchor results. In this paper, we focus on the IP-based test conditions. The common test conditions define six test-case combinations for packet-switched conversational services as well as packet-switched streaming services over 3G mobile networks. Additionally, the test conditions include simplified offline 3GPP/3GPP2 simulation software, programming interfaces and evaluation criteria. Radio channel conditions are simulated with bit-error patterns, which were generated from simulated mobile radio channel conditions. The bit-error patterns are captured above the physical layer and below the RLC/RLP layer and, therefore, they are used as the physical layer simulation in practice. The properties bit rate, length, bit-error rate, and the mobile speed of the bit-error patterns are presented in Table II.

The bit errors in the files are statistically dependent, as channel coding and decoding included in 3G systems produces burst errors. This has been taken into account by evaluating the bit-error pattern files in the following. Patterns 1 and 2 are mostly suited to be used in video streaming applications, where RLP/RLC layer re-transmissions can correct many of the frame losses. The applied channel-coding scheme is a Turbo code scheme and power control targeting throughput maximization rather than error minimization. Patterns 1 and 2 are unrealistic for conversational service, as an acceptable quality cannot be achieved with such high error rates without retransmissions.

Patterns 3–6 are meant to simulate a more reliable, lower error-rate bearer that is required in conversational applications. Assuming a random byte starting position within the file the packet error probability $p_e(r)$ depending of the length of the packet r in bytes can be determined. These error probabilities $p_e(r)$ for all bit-error patterns are shown in Fig. 4. It is obvious

Google Exhibit No. 1063
Page 0979

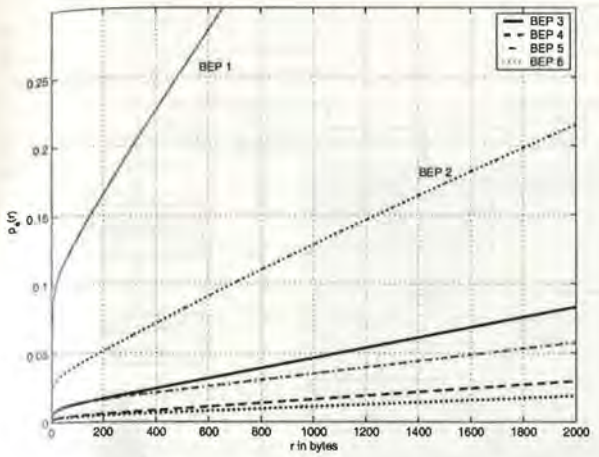


Fig. 4. Packet-loss probability $p_e(r)$ over packet length r in bytes for bit-error patterns according to [25].

that the error rate increases significantly with increasing packet length. For bit-error patterns (BEP1 and 2), the loss rates are not acceptable as already short packets, e.g., 500 bytes, have loss probabilities up to 25%. The high error rates require retransmission on the link layer.

The patterns useful for conversational services provide very decent error characteristics for reasonable packet lengths. The loss probability for reasonable packet sizes up to 1000 bytes is below 5%. This means that typical fixed Internet packet loss probabilities (compare [21]) are not exceeded. Note that with higher speed (50 km/h), the channel tends to be "more ergodic" than in case of the walking user (3 km/h). Therefore, the error rates are usually higher for slowly moving users than for fast-moving users.

During the standardization process, it was agreed in the very beginning that the standard should include error-resilience features for IP-based wired and wireless transmission. Usually two kinds of errors are present in today's transmission systems: bit inversion errors or packet losses. However, all relevant multiplexing protocols like H.223 and UDP/IP and almost all underlying mobile systems include packet loss and bit-error detection capabilities applying sequence numbering and block check sequences, respectively. Therefore, it can be assumed that a vast majority of erroneous transmission packets can be detected. Moreover, even if packets were detected to contain bit errors, decoding could be attempted. Some research has been conducted in this area and, in a few scenarios, gains have been reported for still image transmission [27]. However, in the standardization of H.264/AVC for the development of the reference software, bit-erroneous packets have been considered as being discarded by the receiver for the following reasons.

1) Processing of bit-erroneous packets is likely to be possible in a receiving mobile device only, as gateways and receivers having fixed network connection typically drop erroneous packets.
2) Joint source-channel decoding, such as trellis-based decoding of variable-length codes or iterative source and channel decoding might be applied. However, these techniques have not yet shown significant improvements for video decoding.
3) The decoding based on lost packets serves as a lower bound on the performance in bit-error-prone environments and, therefore, provides a valid benchmark of the performance of H.264/AVC in error-prone environment.
4) Finally, handling of bit errors generally complicates the implementation of decoder software significantly. As the test model software for H.264 was developed for multiple purposes, only simple but meaningful network interfaces have been integrated.

III. H.264/AVC—An Efficient and Flexible Video Coding Toolbox

A. Compression Efficiency and Encoder Flexibility

The features for compression efficiency are discussed elsewhere in this issue, we will only briefly present the key features of the standard, for more details we refer to [2]. Although the design of the H.264/AVC codec basically follows the hybrid design (motion compensation with lossy coding of residual signal) of prior video coding standards such as MPEG-2, H.263, and MPEG-4, it contains many new features that enable it to achieve a significant improvement in terms of compression efficiency. This is the main reason why H.264/AVC will be very attractive for use in wireless environments with the costly resource bit rate. The main features for significantly increased coding efficiency are multiframe motion-compensated prediction, adaptive block size for motion compensation, generalized B-pictures concepts, quartel-pel motion accuracy, intra coding utilizing prediction in the spatial domain, in-loop deblocking filters, and efficient entropy-coding methods.

The normative part of a video coding standard in general only consists of the appropriate definition of the order and semantics of the syntax elements and the decoding of error-free bit streams. This allows a significant flexibility at the encoder, which can, on the one hand, be exploited for pure compression efficiency, and on the other hand, several included features in the standard can be selected by the encoder for other purposes such as error resilience, random access, etc. A typical encoder with the main encoding options is shown in Fig. 5.

The encoding options relevant for wireless transmission are highlighted. The recorded video data is preprocessed by appropriate spatial and temporal preprocessing such that the data rates and displays in a wireless environment are well-matched. For the quantization of transform coefficients, H.264 coding uses scalar quantization. The quantizers are arranged in a way that there is an increase of approximately 12.5% from one quantization parameter (QP) to the next. The quantized transform coefficients are converted into coding symbols and all syntax elements of a macroblock (MB) including the coding symbols are conveyed by entropy coding methods. A MB can always be coded in one of several intra modes with and without prediction, as well as various efficient inter modes. Each motion-compensated mode corresponds to a specific partition of the MB into fixed-size blocks used for motion description, and up to 16 motion vectors may be transmitted for a MB. In addition, for each MB, a different reference frame can be selected. Finally, a NAL

Google Exhibit No. 1063
Page 0980

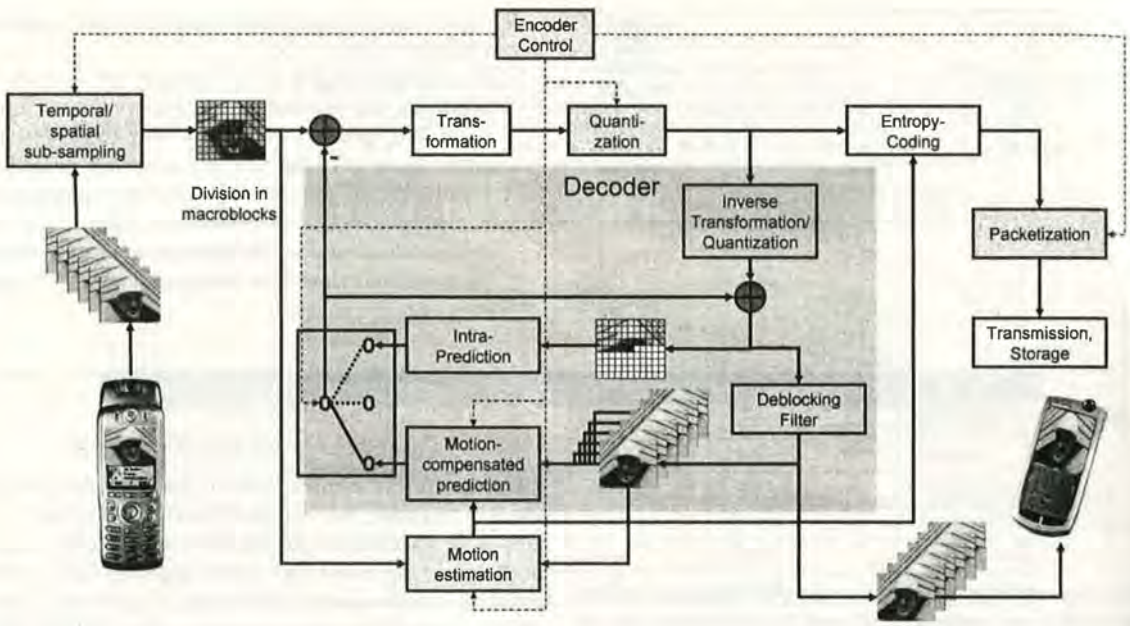


Fig. 5. H.264/AVC encoder realization with coding options.

unit in single-slice mode consists of the coded MBs of an entire frame or a subset of frame. More details on the packetization as well as on the appropriate selection of the presented different modes are discussed in Section IV.

B. Features for Multimedia Messaging and Wireless Packet-Based Streaming

In addition to pure compression efficiency features, additional tools for different purposes have been included in H.264/AVC. We will highlight those features with application to wireless video transmission. Because of the strict separation of encoding, transmission, and decoding, the main issue for MMS is compression efficiency. Other helpful features include the insertion of regular intra frames with instantaneous decoder refresh (IDR) for random access and fast forward. The rate control is typically applied such that video quality is almost constant over the sequence, regardless of the scene complexity except for constraints from the hypothetical reference decoder (HRD) [28]. If time, memory capabilities, and battery power permit, even several encoding passes for optimized rate-distortion (R-D) performance would be possible. On the link layer, reliable transmission strategies as known for data transmission such as file download are used.

Due to on-line transmission and decoding, streaming applications involve more technical challenges than MMS. Usually, pre-encoded data is requested by the user, which inherently does not allow an adaptation to the transmission conditions such as bit rate or error rate in the encoding process. However, the receiver usually buffers the received data and starts play-back after a few seconds. Once starting playback, a continuous presentation of the sequence should be guaranteed. As wireless channels usually show ergodic behavior within a window of a few seconds, reliable transmission schemes can be applied on the link layer, especially when the channel is known at the transmitter or retransmissions for erroneous link layer packets can be used as for example in the acknowledged mode. Slow variance due to distance, shadowing, or varying multiuser topology in the supported cell with renewed resource allocation transform the wireless channel in a slowly varying variable-bit-rate channel. With an appropriate setting of the initial delay and receiver buffer a certain quality of service can be guaranteed [28].

Furthermore, general channel-adaptive streaming technologies, which allow reacting to variable bit-rate channels, have gained significant interest recently. According to [30], these techniques can be grouped into three different categories. Firstly, *adaptive media playout* [31] is a new technique that allows a streaming media client, without the involvement of the server, to control the rate at which data is consumed by the playout process. Therefore, the probability of decoder buffer underflows and overflows can be reduced, but still noticeable artifacts in the displayed video occur. A second technology for a streaming media system is proposed, which makes decisions that govern how to allocate transmission resources among packets. Recent work [32] provides a flexible framework to allow *R-D optimized packet scheduling*. Finally, it is shown that this R-D-optimized transmission can be supported, if media streams are pre-encoded with appropriate packet dependencies, possibly adapted to the channel (*channel-adaptive packet dependency control*) [33].

The latter techniques are supported by H.264/AVC by various means. As the streaming server is in general aware of the current channel bit rate, the transmitter can decide to send one of several pre-encoded versions of the same content taking into account the expected channel behavior. If the channel rate fluctuates only in a small range, frame dropping of nonreference frames might be sufficient resulting in well-known temporal scalability.

Google Exhibit No. 1063
Page 0981

Switching of versions can be applied at I-frames that are also indicated as instantaneous decoder refresh (IDR) pictures to compensate large scale variations of the channel rate. In addition, H.264/AVC supports efficient version switching with the introduction of synchronization-predictive (SP) pictures. For more details on SP pictures, see [35]. Note that quality scalable video coding methods such as MPEG-4 fine-grain scalability (FGS) [34] are not supported by H.264/AVC and such extensions of H.264/AVC are currently not planned.

C. Features for Wireless Conversational Services—Error Resilience

A necessary requirement for conversational services is a low end-to-end delay being less that 250 ms. This delay constraint has two main impacts on the video transmitted over wireless bearer services with constant bit rate. Firstly, features have to be provided which allow adapting the bit-rate such that over a short window a constant bit-rate can be maintained. In addition, usually only temporally backward references in motion compensation are used in conversational applications, since prediction from future frames would introduce additional delay. Secondly, within the round-trip time of the communication, the channel usually shows nonergodic behavior and transmission errors cannot be avoided. Whereas the first issue can be solved by adapting the QP appropriately, the second issue requires error-resiliency tools in the video codec itself. More exactly, an error-resilient video coding standard suitable for conversational wireless services has to provide features to combat two problems: on the one hand, it is necessary to minimize the visual effect of errors within one frame. On the other hand, as errors cannot be avoided, the well-known problem of spatio-temporal error propagation in hybrid video coding has to be limited. We will present all error-resilience features included in the H.264/AVC standard and provide further details on the exact application in Section IV.

Packet loss probability and the visual degradation from packet losses can be reduced by introducing slice-structured coding. A slice is a sequence of MBs and provides spatially distinct resynchronization points within the video data for a single frame. No intra-frame prediction takes place across slice boundaries. With that, packet loss probability can be reduced if slices and, therefore, transmission packets are relatively small, since the probability of a bit-error hitting a short packet is generally lower than for large packets (see, e.g., Fig. 4). Moreover, short packets reduce the amount of lost information and, hence, the error is limited and error concealment methods can be applied successfully. However, the loss of intra-frame prediction and the increased overhead associated with decreasing slice sizes adversely affect coding performance and requires additional overhead per slice. Especially for mobile transmission, where the packet size clearly affects loss probability, a careful selection of the packet size is necessary. H.264/AVC specifies several enhanced concepts to reduce the artifacts caused by packet losses within one frame. Slices can be grouped by the use of aggregation packets into one packet and, therefore, concepts such as group-of-block (GOB) and *slice interleaving* [37], [38] are possible. This does not reduce the coding overhead in the VCL, but the costly RTP overhead of up to 40 bytes per packet can be avoided.

A more advanced and generalized concept is provided by a feature that has been called by the proponents *flexible MB ordering* (FMO) [39]. FMO permits the specification of different patterns for the mapping of MBs to slices including checkerboard-like patterns, sub-pictures within a picture (e.g., splitting a CIF picture into four QCIF pictures), or a dispersed mapping of MBs to slices. FMO is especially powerful in conjunction with appropriate error concealment when the samples of a missing slice are surrounded by many samples of correctly decoded slices. For more details on FMO, see [21].

Another error-resilience feature in H.264/AVC is *data partitioning*, which can also reduce visual artifacts resulting from packet losses, especially if prioritization or unequal error protection is provided by the network. For more details on the data-partitioning mode, we refer to [21]. In general, any kind of *forward error protection* (FEC) in combination with interleaving for packet lossy channels can be applied. A simple solution is provided by RFC2733 [40], more advanced schemes have been evaluated in many papers, e.g., [41], [42]. However, in the following, we do not consider FEC schemes in the transport layer as this requires a reasonable number of packets per codeword.

Despite all these techniques, packet losses and resulting reference frame mismatches between encoder and decoder are usually not avoidable. Then, the effects of spatio-temporal error propagation are, in general, severe. The impairment caused by transmission errors decays over time to some extent. However, the leakage in standardized video decoders, such as H.264/AVC, is not very strong, and quick recovery can only be achieved when image regions are encoded in intra mode, i.e., without reference to a previously coded frame. Completely intra-coded frames are usually not inserted in real-time and conversational video applications as the instantaneous bit rate and the resulting delay is increased significantly. Instead, H.264/AVC allows encoding of single MBs for regions that cannot be predicted efficiently as it is also known from other standards. In H.264/AVC, the efficient intra prediction can be constrained to intra MBs only to avoid error propagation from inter-coded MBs to refreshing intra-coded MBs. Another feature in H.264/AVC is the possibility to select the reference frame from the multiframe buffer. Both features have mainly been introduced for improved coding efficiency, but they can efficiently be used to limit the error propagation. Conservative approaches transmit a number of intra-coded MBs anticipating transmission errors. In this situation, the selection of intra-coded MBs can be done either randomly or preferably in a certain update pattern. For details and early work on this subject, see [43]–[45]. Multiple reference frames can also be used to limit the error propagation, for example in *video redundancy coding* schemes (see, e.g., [46]). In addition, a method known from H.263 under the acronym *redundant slices* will be supported in JVT coding. This will allow sending the same slice predicted from different reference frames which provides the decoder the possibility to predict this slice from error-free reference areas. Finally, multiple reference frames can be successfully combined with a feedback channel, which will be discussed in detail among others in Section IV.

Google Exhibit No. 1063
Page 0982

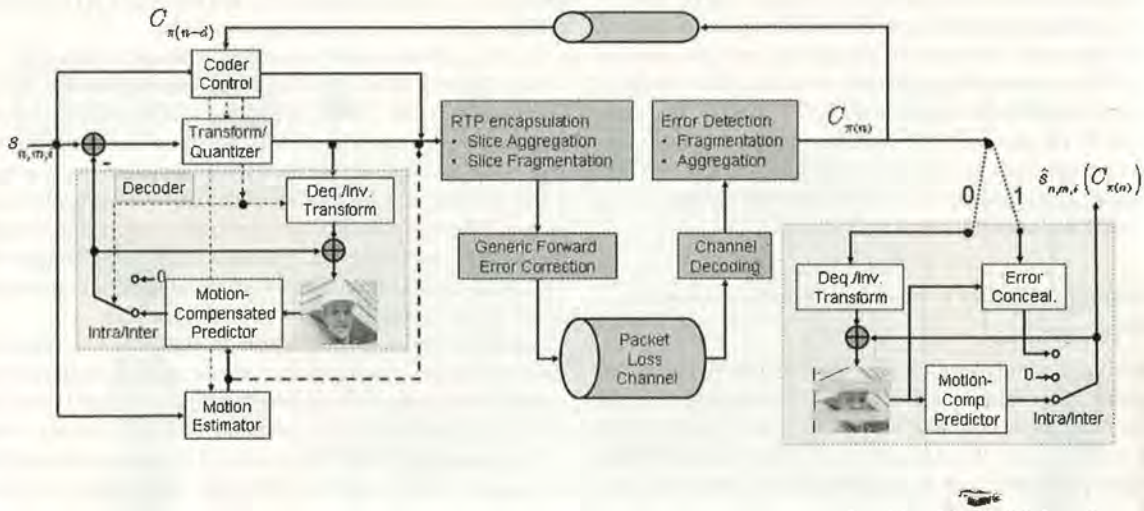


Fig. 6. H.264/AVC in IP-based packet-lossy environment with RTP encapsulation, generic forward error correction, delayed feedback information, and error concealment.

IV. Using H.264/AVC in Wireless Conversational Services

A. Problem Formulation and System Setup

The various error-resilience tools as described in the previous section provide a significant amount of freedom and flexibility for the implementation of H.264/AVC in wireless conversational services. A main criterion for the performance of the system is the appropriate selection of one or several of the mentioned tools together with the exact parameters, e.g., the position and number of intra MBs.

We will discuss in the following channel behavior and error concealment by formalizing the notation for sample representation. The investigated video transmission system is shown in Fig. 6. H.264/AVC video encoding is based on a sequential encoding of frames denoted with the index $n = 1, \ldots, N$ with N being the total number of frames to be encoded. In most existing video coding standards including H.264/AVC, within each frame video encoding is typically based on sequential encoding of MBs (except for FMO) denoted by index $m = 1, \ldots, M$, where M specifies the total number of MBs in one frame and depends on the spatial resolution of the video sequence. The encoding process creates slices by grouping a certain number of MBs. Picture number n and start MB address m_j are binary coded in the slice header. The coded representation of a slice is the payload of a NAL unit. The RTP payload specification specifies simple encapsulation of NAL units. In addition, several NAL units can be combined into one aggregation packet or one NAL unit can be fragmented into several transport packets [19], [21].

For notational convenience, let us define the number of transmission packets to transmit all frames up to n as $\pi(n)$. With that, we can define the packet loss or channel behavior c as a binary sequence $\{0,1\}^{\pi(n)}$ indicating whether a slice is lost (indicated by 1) or correctly received (indicated by 0). Obviously, if a NAL unit with the encapsulated slice is lost, all MBs contained by this slice are lost. It can be assumed that the decoder is aware of any lost packet as discussed previously. The channel-loss sequence is random and, therefore, we denote it as $C_{\pi(n)}$, where the statistics are in general unknown to the encoder. According to Fig. 6, in addition to the forward link it possible that a low-bit-rate reliable back-channel from the decoder to the encoder is available which allows reporting a d-frame delayed version $C_{\pi(n-d)}$ of the observed channel behavior at the decoder to the encoder. In RTP/IP environments, this is usually based on RTCP messages, and in wireless environments, internal protocols might be used.

The decoder processes the received sequence of packets. Whereas correctly received packets are decoded as usual for the lost packet, an error-concealment algorithm has to be invoked. The reconstructed sample $s_{n,m,i}$ at position i in MB m and frame n depends on the channel behavior and on the decoder error concealment. In inter-coding mode, i.e., when motion-compensated prediction (MCP) is utilized, the loss of information in one frame has a considerable impact on the quality of the following frames, if the concealed image content is referenced for MCP. Because errors remain visible for a longer period of time, the resulting artifacts are particularly annoying. Therefore, due to the motion-compensation process and the resulting error propagation, the reconstructed image depends not only on the lost packets for the current frame, but in general on the entire channel-loss sequence $C_{\pi(n)}$. We denote this dependency by $\hat{s}_{n,m,i}(o, C_{\pi(n)})$

In the following, we will discuss appropriate extensions of the encoder, the decoder, and the transmission environment, which are either necessary or at least beneficial to enhance the quality of the transmitted video.

B. Decoder Extensions—Loss Detection and Error Concealment

The H.264 standard defines how a decoder reacts to an error-free bit stream. In addition, a decoder implementation has also to deal with transmission errors. As we discussed earlier, it is assumed that bit errors are detected by the lower layer

Google Exhibit No. 1063
Page 0983

entities and any remaining transmission error results in a packet loss. Therefore, we address the reaction of the decoder to slice losses. Two major issues are important for an error-resilient decoder: a robust video decoder has to detect transmission errors, and appropriate concealment on detected errors has to be applied. In this section, we present how the H.264 test model decoder meets the goal of error resiliency.

The error detection capabilities of the test model decoder are based on two assumptions about the error detection operation of the underlying system. First, bit-erroneous slices are discarded prior to passing the slices to the test model decoder. Second, received data is buffered in a way that the correct decoding order is recovered. In other words, the test model decoder expects noncorrupted slices in a correct decoding order. Temporal and spatial localization of lost packets is left to the decoder. In particular, the decoder has to detect if an entire picture or one or more slices of a picture were lost. Losses of entire pictures are detected using frame numbers associated with each frame and carried in slice headers. A frame number n is incremented by one for each coded and transmitted frame that is further used for motion compensation. These frames are herein referred to as reference frames. For disposable nonreference pictures, such as conventional B-pictures, the frame number is incremented relative to the value in the most recent reference frame that precedes the disposable picture in the bit-stream order.

The decoder generates a slice structure for each received single slice packet and forwards it to the VCL decoder which maintains a state machine that keeps track of the expected frame number n_e. Moreover, the decoder maintains a loss indication for each MB within the current picture. At the beginning of a new picture, the binary map is reset to indicate that all MBs were lost. If the frame number n of the next slice to be decoded equals n_e, the decoder decodes the slice and updates the binary map. If n is greater than n_e, it is deduced that all the received slices of the previous picture have been decoded. Then, the binary map is investigated, and if the picture is not fully covered by correctly decoded MBs, a slice loss is inferred and losses are concealed as presented in the following. Moreover, if n is greater than $n_e + 1$, the decoder infers a loss of pictures and inserts concealed pictures to the reference picture buffer as if the lost pictures were decoded. The concealment is accomplished by copying the previous decoded picture. Finally, the decoder resets the binary map and decodes the next slice of the next picture.

Error concealment is a nonnormative feature in the H.264 test model. The target for the selected error concealment is to provide a basic level of error resiliency for the decoder. Any error-robust coding scheme proposed for H.264 should be compared against the H.264 test model equipped with the selected error concealment algorithms. Two well-known concealment algorithms, weighted pixel value averaging for intra pictures [47] and boundary-matching-based motion vector recovery for inter pictures [48], were tailored for H.264 as summarized below and described in details in [49] and [50].

Weighted pixel value averaging operates as follows. If a MB has not been received, it is concealed from the pixel values of spatially adjacent MBs. If a lost MB has at least two correctly decoded neighboring MBs, only these neighbors are used in the concealment process. Otherwise, previously concealed neighboring MBs take part in the process, too. Each pixel value in a MB to be concealed is formed as a weighted sum of the closest boundary pixels of the selected adjacent MBs. The weight associated with each boundary pixel is relative to the inverse distance between the pixel to be concealed and the boundary pixel. The performance of the intra concealment method is shown in Fig. 7



Fig. 7. Intra frame error concealment.

In the motion vector recovery algorithm, the motion activity of the correctly received slices of the current picture is investigated first. If the average length of a motion vector component is smaller than a pre-defined threshold (currently 1/4 pixels), all the lost slices are copied from co-located positions in the reference frame. Otherwise, motion-compensated error concealment is used, and the motion vectors of the lost MBs are predicted as described in the following paragraphs. The image is scanned MB-column-wise from left and right edges to the center of the image. Consecutive lost MBs in a column are concealed starting from top and bottom of the lost area toward to center of the area. This processing order is used to ensure that lost MBs at the center of an image are concealed using as many neighboring concealed MBs as possible.

Each 8×8 luminance block of a MB to be concealed is handled separately. If a block has spatially adjacent blocks whose motion vectors are correctly received, these motion vectors and their reference pictures are used to obtain a candidate prediction block. If all the motion vectors in the adjacent blocks are lost, neighboring concealed motion vectors are used similarly. In addition, the spatially co-located block from the previous frame is always one of the candidates. The candidate prediction block whose boundary matching error is the smallest is chosen. The boundary matching error is defined as the sum of the pixel-wise absolute differences of the adjacent luminance pixels in the concealed block and its decoded or concealed neighbor blocks. The lost prediction error block is not concealed. The performance of the inter-frame concealment is shown in Fig. 8.

C. Encoder Extensions

Let us now consider algorithms and rules for the appropriate selection of different encoding parameters according to the presentation in Fig. 5 for wireless conversational services. First, the rate control has to guarantee that the delay jitter is as small as possible. A good choice is to adapt for each frame to obtain almost constant encoding, but keep the QP within one frame constant. In the case that the highest QP cannot achieve the required bit rate, frame dropping is introduced; otherwise, the frame rate

Google Exhibit No. 1063
Page 0984



Fig. 8. Inter-frame error concealment.

is kept constant. Obviously, this results in varying quality depending of the complexity of the scene. Low-motion scenes with little complexity have higher quality than for example frames at the beginning of a sequence or at a scene cut.

As discussed in Section III, H.264/AVC provides a large flexibility on the selection of appropriate coding modes. The concept of selecting appropriate coding options in optimized encoder designs for many video coding standards is based on R-D optimization algorithms [51],[52]. The two cost terms "rate" and "distortion" are linearly combined and the mode is selected such that the total cost is minimized. This can be formalized by defining the set of selectable coding options for one MB as $\mathcal{O}$. Assuming the introduced distortion when encoding with a certain mode o as $D(o)$ and the corresponding rate as $R(o)$, the rate-constrained mode decision selects the coding option o^* such that the Lagrangian cost functional is minimized, i.e.,

$$o^* = \arg\min_{o \in \mathcal{O}} (D(o) + \lambda R(o)) \tag{1}$$

with λ being the Lagrange parameter for appropriate weighting of rate and distortion. In the H.264/AVC test model, the Lagrangian mode selection is used for motion vector search as well as MB mode and reference frame selection. Then, for the distortion $D(o)$, the sum of squared sample differences (SSD) or the sum of absolute sample difference (SAD) are used, and for the rate $R(o)$, the number of bits for the encoding is used. The selection of the Lagrange parameter depends on the selected QPs (for details, see [53]). In addition, the mode selection might be extended by selecting the appropriate QP for each MB, as the QP can be changed at least in small ranges for each MB. This is not considered in the following.

As discussed in Section III-C, the tools for increased error resilience, in particular those to limit error propagation, do not significantly differ from those used for compression efficiency. Features like multiframe prediction or intra MB coding are not exclusively error-resilience tools. They are also used to increase coding efficiency in error-free environments providing a trade-off that is left to the encoder. This also means that bad decisions at the encoder can lead to poor results in coding efficiency or error resiliency or both. Therefore, the selection of the coding mode according to (1) is modified taking into account the influence of the random lossy channel. In the case of error-prone transmission, the distortion in (1) is replaced with the expected decoder distortion when encoding with mode o. Assuming that the encoder can access information about the channel statistics $C_{\pi(n)}$, the encoder can get an estimate of the expected decoder distortion $D_{n,m}(o, C_{\pi(n)})$ when encoding MB m in frame n at the decoder by the expected distortion as

$$D_{n,m}\left(o, C_{\pi(n)}\right) = \sum_{i=1}^{I} E_{C_{\pi(n)}} \left| s_{n,m,i} - \hat{s}_{n,m,i}\left(o, C_{\pi(n)}\right) \right|^2 \tag{2}$$

where the expectation is over the random process characterizing the channel $C_{\pi(n)}$. With the expected distortion measure, the mode selection for lossy channels is identical to that in (1) except for modified distortion term according to (2). However, the same parameter λ [54] and the same set of possible coding options $\mathcal{O}$ are used.

This leaves the problem of computing the expected decoder distortion at the encoder which depends on the coding mode o, the channel statistics $C_{\pi(n)}$, and the applied error concealment in the decoder. The estimate of the expected sample distortion in packet loss environment has been addressed in several publications. For example, in [58], [55], or [56], methods to estimate the distortion introduced due the transmission errors and the resulting drift are presented. A similar approach has recently been proposed within the H.264/AVC standardization process which attempts to measure the drift noise between encoder and decoder [57]. In suboptimal approaches [55]–[57], the quantization noise and the distortion introduced by the transmission errors are linearly combined. The encoder keeps track of an estimated sample distortion and, therefore, requires additional complexity in encoder, which is dependent on the actual method chosen.

The most recognized out of these methods, called recursive optimal per-sample estimate (ROPE) algorithm [58], provides an estimation by keeping track of the first- and second-order moment of $\hat{s}_{n,m,i}, E\left\{\hat{s}_{n,m,i}(o, C_{\pi(n)})\right\}$, and $E\left\{\hat{s}^2_{n,m,i}(o, C_{\pi(n)})\right\}$, respectively. For H.263-like encoding ROPE can provide an exact estimate of the expected decoder distortion. As two moments for each sample have to be tracked in the encoder, the added complexity of ROPE is approximately twice the complexity of the decoder. However, the extension of the ROPE algorithm to H.264/AVC is not straightforward. The in-loop deblocking filter, the fractional sample motion accuracy, the complex intra prediction and the advanced error concealment require taking into account the expectation of products of samples at different positions to obtain an accurate estimation which makes the ROPE either infeasible or inaccurate in this case.

Therefore, a powerful yet complex method has been introduced into the H.264/AVC test model to estimate the expected decoder distortion [54]. Let us assume that we have K copies of the random variable channel behavior at the encoder, denoted as $C_{\pi(n)}(k)$. Additionally, assume that the set of random variables $C_{\pi(n)}(k), k = 1, \cdots, K$ are *identically* and *independently* distributed (*i.i.d.*). Then, as $K \to \infty$, it follows by the strong law of large numbers that

$$\frac{1}{K}\sum_{k=1}^{K} \left| s_{n,m,i} - \hat{s}_{n,m,i}\left(C_{\pi(n)}(k)\right) \right|^2 = E_{C_{\pi(n)}} \left| s_{n,m,i} - \hat{s}_{n,m,i}\left(C_{\pi(n)}\right) \right|^2 \tag{3}$$

Google Exhibit No. 1063
Page 0985

holds with probability 1. An interpretation of the left-hand side leads to a simple solution of the previously stated problem to estimate the expected sample distortion. In the encoder, K copies of the random variable channel behavior and the decoder are operated. The reconstruction of the sample value depends on the channel behavior $C_{\pi(n)}(k)$ and the decoder including error concealment. The K copies of channel and decoder pairs in the encoder operate independently. Therefore, the expected distortion at the decoder can be estimated accurately in the encoder if K is chosen large enough. However, the added complexity in the encoder is obviously at least K times the decoder complexity. For details on the implementation as well as comparison of suboptimal modes based on ROPE, we refer to [54].

It was later demonstrated in [59] and [60] that a technique known as isolated regions, which combines periodical intra MB coding and limitation of inter prediction, can provide competitive simulation results compared to the error-robust MB mode selection of H.264 test model, and the best simulation results were obtained when the two MB mode-selection algorithms were combined.

The packet size is usually upper bounded by a MTU size of the underlying network. However, if bit errors cause entire packet losses, it might be an advantage to reduce the packet size. Conservative approaches limit the packet or (in our simulation environment) slice size to a fixed number of MBs, e.g., a line of MBs is transmitted in one packet. However, this makes especially important image parts very susceptible to packet losses as for example suddenly appearing objects need intra coding or high motion areas require an extensive motion vector rate. A more suitable, but still conservative, approach is the introduction of slice boundaries such that slices have almost the same length in number of bytes. This makes all packets similarly susceptible to bit errors. This method has been proven to work quite well already for MPEG-4 and can also be applied with the flexible encoding framework of H.264/AVC.

D. Exploiting Feedback in H.264/AVC

Multiple reference frames used in error-prone environment can most successfully be combined with a feedback channel. Due to the bidirectional nature of conversational applications, it is common that the encoder has knowledge of the experienced channel at the decoder, usually with a small delay. In our framework this can be expressed by the knowledge of a d-frame delayed version of the random channel $C_{\pi(n-d)}$ at the encoder. This characteristic can be conveyed from the decoder to the encoder by acknowledging correctly received slices (ACK), sending a not-acknowledge message (NAK) for missing slices or both types of messages. In general, it can be assumed that the reverse channel is error-free and the overhead is negligible. In common data transmission applications, the lost packets would obviously be re-transmitted. However, in a low-delay environment, this is not feasible, but the observed channel characteristic are still useful at the encoder even if the erroneous frame has already been decoded and concealed. The support of these techniques is out of the focus of the standardization of H.264, and a full support of the presented concepts might not be possible with current transport and control protocols. However, it seems worth to mention and discuss these concepts in here in detail as it shows the flexibility of H.264 as well as it provides motivation for inclusion of feedback in system designs.

In previous standards and transport environments similar approaches have already been discussed, usually limited by the reduced syntax capabilities of the video standard. A simple yet powerful approach suitable for video codecs using just one reference frame such as MPEG-2, H.261, or H.263 version 1 has been introduced in [61] and [62] under the name *error tracking*. When receiving a NAK on parts of frame $n-d$ or the entire frame $n-d$, the encoder attempts to track the error to obtain an estimate of the quality of frame $n-1$, which serves as reference for frame n. Appropriate actions after having tracked the error are for example presented in [56], and [61]–[64]. Note that with this concept, error propagation in frame n is only removed if frames $n-d+1,\ldots,n-1$ have been received at the decoder without any error.

A technique addressing the problem of continuing error propagation has been introduced, among others, in [65]–[67] under the acronym *NEWPRED*. Based on these early nonstandard compliant solutions in H.263 Annex N [68], a reference picture selection (RPS) for each GOB is specified such that the NEWPRED technique can be applied. RPS can be operated in two different modes. In the negative acknowledgment mode (NAM), the encoder only alters its operation in the case of reception of a NAK. Then, the encoder attempts to use an intact reference frame for the erroneous GOBs. To completely eliminate error propagation, this mode has to be combined with independent segment decoding (ISD) according to Annex R of H.263 [68]. In the positive acknowledgment mode (PAM), the encoder is only allowed to reference confirmed GOBs. If no GOBs are available to be referenced, intra coding has to be applied. NEWPRED allows to completely eliminate error propagation in frame n, even if additional errors have occurred in frames $n-d+1,\ldots,n-1$.

The flexibility provided in H.263 Annex U [68] and especially H.264/AVC to select the MB mode and reference frames on MB or subMB basis allows incorporating NEWPRED in a straightforward manner [56]. Therefore, let us define three different states of transmitted packets at the encoder: ACK, NAK, and outstanding acknowledgment (OAK). Then, based on the exploitation of these messages in the encoder, we discuss three modes which can elegantly be integrated into the R-D optimized mode selection according to (1).

1) Feedback Mode 1: Restricting Reference Areas to ACK: In this case, the reference area is restricted to frames or slices which have been acknowledged. This can be formalized in the context of (1) by applying the encoding distortion $D(o)$ in (1), but altering the accessible coding options $\mathcal{O}$ such that only acknowledged areas can be used for reference. In addition, if no reference area is available, or if no satisfying match is found in the allowed area, intra coding is applied. Although in the presentation of a single frame an error might be visible, error propagation and reference frame mismatch between encoder can be completely avoided independent of the error concealment applied at the decoder if correctly decoded samples are not altered. However, in the used test model JM1.7, the deblocking filter operation in the motion-compensation

Google Exhibit No. 1063
Page 0986

loop is applied over slice boundaries, which restricts the area to be referenced significantly, or a complete removal of encoder and decoder mismatch is not possible. Although the influence of this mismatch is, in general, negligible, in the final design of H.264/AVC the deblocking filter can adaptively be switched on and off at slice boundaries to allow mismatch-free operation.

2) Feedback Mode 2: Referencing ACK and Error Concealed NAK: In this case, the reference area is again restricted; however, in addition to acknowledged areas, also the areas which the decoder signaled as lost can be referenced. Therefore, the reference frames in the encoders multi-frame frame buffer are updated with reception of each ACK and NAK by applying the identical error concealment as the decoder applies. This is obviously very critical, as the error concealment is—for good reasons—nonnormative in common video coding standards including H.264/AVC. Only if the encoder is aware of the decoder's error concealment by any external means, this mode can provide benefits compared to feedback mode 1. In the context of the mode decision in (1), we have again a restricted set of coding modes $\mathcal{O}$ and, in addition, the encoding distortion is replaced by the deterministic decoder distortion.

3) Feedback Mode 3: Unrestricted Reference Areas With Expected Distortion Updating: In [56] and [58], techniques have been proposed which allow combining the error-resilient MB mode selection with feedback information. In this case, the expected decoder distortion computation is altered such that, for all packets $1, \ldots, \pi(n-d)$, the channel is deterministic at the encoder, and the expected distortion is computed for the packets with status OAK. In our case, packets containing MBs in frames $\pi(n-d)+1, \ldots, \pi(n)$ are random. The set of selectable coding options $\mathcal{O}$ is not altered compared to pure coding efficiency mode selection. This method is especially beneficial compared to mode 1 and 2, if the feedback is significantly delayed. In the case of the multiple decoder approach, this can be integrated by applying feedback mode 2 not only to the reference frames, but also to all decoders in the encoder. In combination with ROPE, however, the complexity of this method increases since the moments of the previous d frames have to be re-tracked [58].

V. Selected Simulation Results for Different System Concepts

A. Simulation Conditions and Evaluation Criteria

In the following, we will present simulation results based on the test conditions that show the influence of the selection of different error-resilience features for the quality of the decoded images. For all following tests the H.264/AVC test model software version JM1.7 is used. Note that in the final standard [1], the zig-zag scanning and run-length coding is replaced by context-adaptive variable length codes (CVLC). All UVLCs are replaced by CVLC, which are adapted to the statistics of different syntax elements. In addition, quantizer values have been shifted. However, all changes from JM1.7 to the final standard are of little relevance to the presented results and conclusions in this paper.

The reported PSNR is the arithmetic mean over the decoded luminance PSNR over all frames of the encoded sequence and over 100 transmission and decoding runs. The 100 starting positions for the error patterns have been selected such that they are almost equally distributed over the error pattern. For all comparable results, the same starting positions and, therefore, the same channel statistics have been applied. In addition, we present results for the cumulative distribution of the decoded luminance PSNR for each frame, i.e., the likelihood that the PSNR of the frames of the sequence is smaller than the value on the x axis. This shows the variance in the decoded quality. It is assumed that the high-level syntax parameters have been transmitted in advance and out-of-band applying a reliable setup protocol. The NAL overhead, the RTP/UDP/IP overhead after RoHC, and the link layer overhead is taken into account in the bit-rate constraints according to [25].

For the following simulations, we concentrate on test case 5 from [25], which includes the QCIF test sequence "Foreman" (30 Hz, 300 frames) at a constant frame rate of 7.5 Hz at bit-error pattern 3 according to Table II, i.e., a mobile user at 3 km/h and maximum bit-rate 64 kbit/s. This is the most critical case in terms of error probability, additional test results will be made available online[2]. For the following tests, entropy coding based on the UVLC and only one reference frame has been applied, if not stated otherwise. The encoded sequences are I-P-P-P... sequences; B-pictures are excluded due to the unacceptable delay involved in the encoding process. Note that due to the repeated decoding of an encoded file, every 75th frame is an I-frame, i.e., an I-frame occurs every 10 s. Constrained intra has been used to avoid error propagation from inter MBs to intra MBs. For all encoding runs, R-D optimized mode selection and motion vector selection according to (1) have been used. The distortion $D(o)$ and the set of coding modes $\mathcal{O}$ is appropriately selected according to the applied features. In the case of using the expected decoder distortion, the number of decoders operated in the encoder has been fixed to $K = 100$. Unless stated otherwise, the error concealment in the decoder is based on the advanced error concealment as presented in Section IV-B, whereas the multiple decoders in the encoder always apply previous frame concealment. This reduces encoding complexity significantly and results only in negligible performance losses for the investigated cases.

B. Low-Delay Rate Control

Version JM1.7 of the H.264/AVC test model encoder which is used in the experiments does not include a rate control to achieve a constant bit rate for wireless conversational services. Moreover, the rate control introduced in later versions of H.264/AVC test model encoder is not suitable for constant rate encoding. For this reason we have added a rate control which provides an almost constant bit-rate encoding for each frame by adapting the QP for each frame appropriately. Therefore, before we investigate the error-resilience features in H.264/AVC, we will first focus on the effect of bit-rate control that is necessary for constant bit-rate conversational applications. Fig. 9 shows the cumulative distribution of the encoding PSNR for the applied rate control and a fixed QP such that both encoded files result in the

[2]For additional simulation results, we refer to http://www.lnt.ei.tum.de/~stockhammer.

Google Exhibit No. 1063
Page 0987

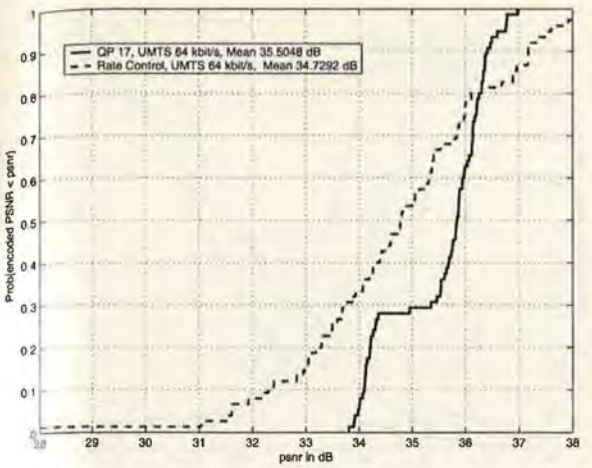


Fig. 9. Cumulative distribution of encoding luminance PSNR for each frame for constant QP 17 and constant bit rate such that rate constraint of 64 kbit/s for UMTS is fulfilled.

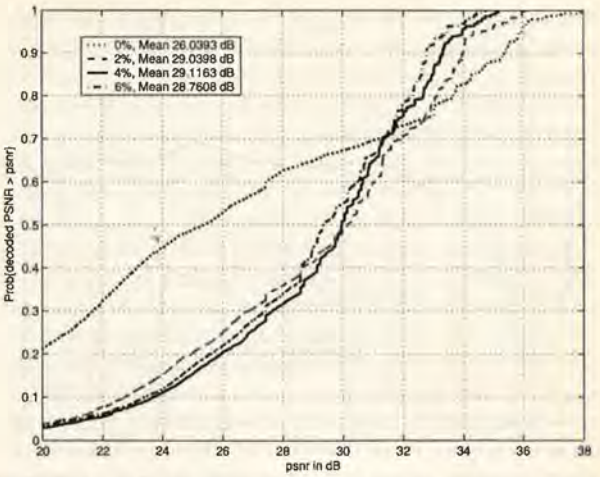


Fig. 10. Cumulative distribution of decoded PSNR for different NAL unit erasure rates for the estimation of the expected distortion in the encoder.

same total bit-rate, and, additionally the file can be transmitted over the 64 kbit/s link including NAL unit, packet, and link layer overhead. However, the fixed QP results in an extremely varying data rate and, therefore, the introduced delay jitter is not acceptable. In the encoding process, no error-resilience tools have been used, i.e., one frame is encapsulated into one NAL unit, for $D(o)$ the encoding distortion is applied, and the set of coding modes $\mathcal{O}$ is unrestricted.

The average PSNR for the rate control is about 0.8 dB below the PSNR for the fixed QP. In addition, the probability for low encoded PSNR is significantly higher as for complex scenes the QP has to be adapted appropriately. Therefore, it can be seen that the rate control necessary for conversational applications involves an inherent decrease in quality if the average PSNR is the measure of interest.

C. $R - E\{D\}$-Optimized MB Mode Selection

In this section we investigate the performance of the system in case that the channel statistics are taken into account into the selection of the coding options in the encoder. For this purpose we replace the encoding distortion $D(o)$ in (1) by the expected decoder distortion assuming a channel producing independent packet losses with probability p. As we use the mapping of one frame to one transport packet and apply the strict rate control producing almost a constant number of bytes for each encoded frame, the size of each frame results in roughly 1000 bytes. The corresponding loss rate for packets of this size for bit-error pattern 3 is approximately 4%–5% according to Fig. 4. Fig. 10 shows the cumulative distribution of decoded PSNR for different NAL unit erasure rates $p = \{0, 0.02, 0.04, 0.06\}$ for the estimation of the expected distortion in the encoder. Looking at the results for the average PSNR, it can be seen that the introduction of loss-aware R-D optimization ($p > 0$) increases the decoded quality significantly compared to the results with pure encoding distortion ($p = 0$). The average PSNR increases by at least 3 dB when compared to the to pure R-D optimization. The advantage of $R-E\{D\}$ optimization is even more evident when looking at the cumulative distribution of the different strategies. Whereas in the case of no error resilience, the probability of bad frames (frames below 22 dB) is at an unacceptable ratio of about 30%, for loss-aware coding, this is reduced significantly to less than 8%. It is also obvious that if the expected error rate matches the experienced error rate on the channel, the performance is optimal (see $p = 4\%$). However, it can also be seen that a mismatch in the expected error rate in the encoder does not have significant influence. The performance of $p = 2\%$ and $p = 6\%$ is only slightly inferior to the matching expected error rate. Therefore, a rough estimation of the expected decoder distortion at the encoder seems to be good enough for a good mode selection. Note the significant loss in average PSNR for wireless transmission compared to the error-free transmission according to Fig. 9 of more than 5 dB.

D. Slices and Error Concealment

The introduction of slices in the encoding has two beneficial aspects when transmitting over wireless channels, but adversely affects the coding efficiency due to increased packet overhead and reduced prediction within one frame, as e.g., motion vector prediction and spatial intra prediction is not allowed over slice boundaries. The two positive effects with the introduction of slices are the reduced error probability of shorter packets (see Fig. 4) and, the re-synchronization possibility within one frame. The latter technique allows restarting the decoding process at each slice and, in addition, it allows applying advanced error concealment as for example presented in Section IV-B. However, for packet-lossy transmission over the Internet, the introduction of slices does not, in general, provide gains in the decoded quality [69] as long as the slice size is below the MTU size, as the loss of a packet is then independent of its length. This is different for wireless transmission: Fig. 11 shows the average decoded PSNR for different number of packets per frame N_p and MB mode decision with encoding distortion. The experimental conditions are similar to the above section otherwise. For a given number of packets per frame, the size of each packet is selected such that the packets roughly have the same number of bytes. This makes their susceptibility to bit errors almost identical.

Google Exhibit No. 1063
Page 0988

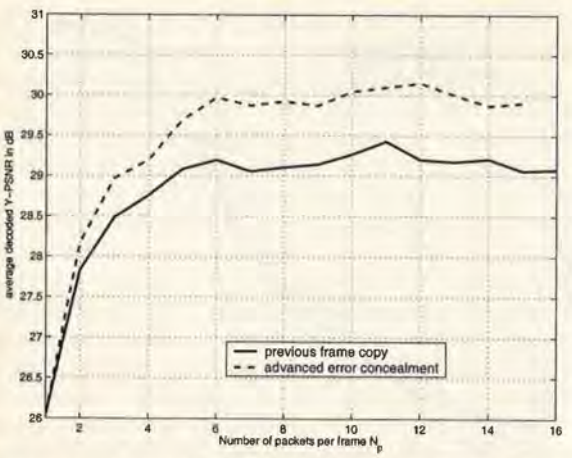


Fig. 11. Average decoded PSNR for different number of packets per frame and different error-concealment schemes; mode selection with encoding distortion.

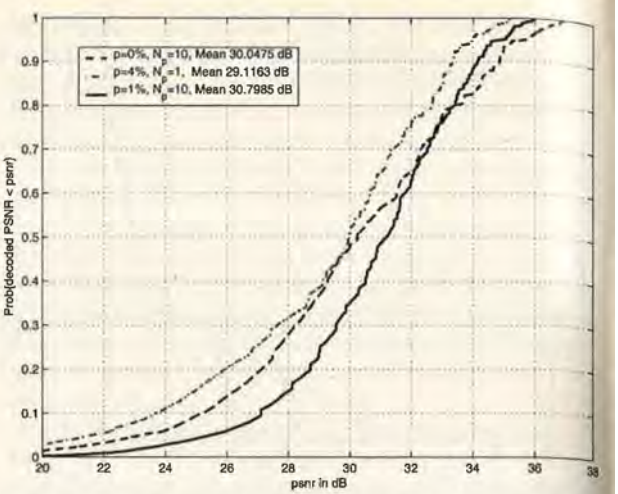


Fig. 12. Cumulative distribution of decoded PSNR for different error-resilience strategies: $R - E\{D\}$-optimized intra updates with and without slice structuring and delay $d = 2$ for different assumed loss probabilities p.

The average decoded PSNR is shown for previous frame copy error concealment and advanced error concealment according to Section IV-B. The benefit of introducing slice structuring in the encoding process is obvious from the results. Compared to the "one frame—one packet" mode indicated by $N_p = 1$ the introduction of shorter packets increases the decoded quality significantly for both error concealment methods. For about $N_p = 6$, the curve flattens out and decreases again for higher $N_p > 12$ due to increasing packet overhead and the reduced compression efficiency. Although a clear maximum cannot be determined, for the wireless environment according to our test conditions, a reasonable number of packets per frame is about 10. The resulting packet size in this case is in the range of 100 bytes. However, note that for the simulations, RoHC was applied, which reduces the typical IP/UDP/RTP overhead from 40 bytes to about 3 bytes and, therefore, the packetization overhead is less significant. The benefits of the advanced error concealment are also obvious from Fig. 11. As can be seen, the gains for advanced error concealment increase with increasing number of packets, as better concealment is possible due to increased number of neighboring MBs in case of losing a single slice.

For the experiments in Fig. 11, no explicit means to reduce the error propagation have been used. However, we can obviously combine the MB mode selection based on the expected decoder distortion with the slice structured coding. As indicated, the loss rate for decreased packet size decreases compared to the single packet for one frame. The loss rate can be estimated by dividing the approximated number of bytes for each frame, roughly 1000 bytes, by the number of packets. The loss probability can then again be estimated with Fig. 4 using the resulting average packet length. The combination of slice-structured coding and adaptive intra MB updates has been investigated and a comparison with the best cases of the previous two system designs is provided in Fig. 12 based on the cumulative distribution of the decoded PSNR. For the slice-structured coding with encoding distortion ($p = 0\%$) the number of packets is selected as $N_p = 10$. For the expected decoder distortion without slice-structuring ($N_p = 1$), the adapted loss-rate $p = 4\%$ is chosen. Finally, for the combination of slice-structured coding and channel-adaptive intra updates, the number of packets per frame is selected as $N_p = 10$ and, therefore, the appropriate loss probability to compute the expected decoder distortion according to Fig. 4 is about $p = 1\%$.

The average decoded PSNR indicates that an optimized combination of both error-resilience schemes outperforms each of the presented error-resilience schemes significantly. The result that slice-structured coding is superior to intra updates cannot be generalized for all sequences. The repeated I-frame insertion and the camera pan in the test sequence "Foreman" results in a significant amount of intra information, even if only the encoding distortion is chosen in the mode selection. This might change for different or longer sequences with less intra information. From the cumulative distribution, it can be observed that the probability for bad frames below 22 dB in PSNR is almost vanishing for the combined mode. The presented results indicate that slices in combination with advanced error concealment significantly outperform the single packet for one frame approach. However, the loss compared to error-free transmission is still about 4 dB in average PSNR.

From the results, it can be conjectured that for wireless transmission as investigated in this case, other approaches, which reduce the artifacts within one frame, might provide additional benefits. This includes concepts such as FMO, slice interleaving, or even generic forward-error correction in combination with a NAL unit fragmentation scheme as recently introduced in the draft RTP payload specification for H.264/AVC [19]. Also, data partitioning with appropriate unequal error protection might enhance the quality of the decoded video. In addition, it is conjectured from the results that a better adaptation of the link layer error protection scheme with appropriate interleaving could increase the overall system performance. These features are currently investigated and are subject of future work.

E. Exploiting Feedback in Video Encoding

Finally, we investigate a system which exploits multiple reference frames and network feedback. We restrict our simula-

Google Exhibit No. 1063
Page 0989

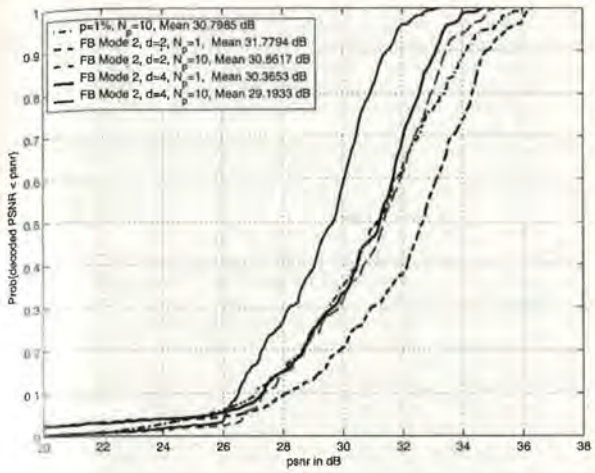


Fig. 13. Cumulative distribution of decoded PSNR for different error-resilience strategies: R-D-optimized intra updates with slice structuring, and feedback mode 2 with and without slice structuring for delay $d = 2$ and $d = 4$.

tion results to feedback mode 2 according to the presentation in Section IV-D with advanced error concealment. There are two reasons for this. On the one hand, the deblocking and mismatch problem present for JM1.7 can be avoided. On the other hand, it has been shown previously that feedback mode 1 and feedback mode 2 result in almost identical performance [70]. Therefore, we chose feedback mode 2 due to the simpler implementation, at least in our simulation environment. Feedback mode 3 is excluded as, on the one hand, the complexity of this mode is rather high and, on the other hand, as we operate with low feedback delays, the expected benefits are only marginal. In contrast to the previous simulations, we use five reference frames for the feedback mode. Fig. 13 shows the cumulative distribution of decoded PSNR for different error-resilience strategies: R-D-optimized intra updates with slice structuring, as well as feedback mode 2 with and without slice structuring for delay $d = 2$ and $d = 4$ frames, which corresponds to a round-trip time of about 250 and 500 ms, respectively.

Let us focus on the delay $d = 2$ case first. The results indicate that the optimized intra mode with slice structuring and MB mode selection with expected decoder distortion and feedback mode 2 perform very similar based on the cumulative distribution and the average decoded PSNR. The feedback mode might still be beneficial in this case as the complex estimation of the decoder distortion is not necessary for the feedback case. However, much more interesting is the case with feedback and no slice structuring. In contrast to the case without feedback (see Fig. 12), the renouncement of slice-structured coding provides a significantly higher average decoded PSNR. Initially, this is obviously surprising, as packet loss rate is still much lower when several slices are used and also the visual effects for a decoded frame when losing a single slice should be lower than in case of losing an entire frame. The first effect can indeed be observed from the cumulative distribution. The probability of bad frames (PSNR below 22 dB) is higher for $N_p = 1$ than for $N_p = 10$. However, in the case of no errors, the increased coding efficiency when not using slices provides many frames with significantly higher PSNR than for slice structuring. As we avoid error propagation, the correctly received frames are really error-free, which is not the case if we use intra updates. Therefore, if feedback information is available and several completely lost frames are tolerable, it is better to use no slice structured coding than harming the compression efficiency. For increased feedback delay $d = 4$, the curves are shifted to the left compared to feedback delay 2. However, the $d = 4$ and $N_p = 1$ performs almost as well as the best case without feedback. Therefore, in the case of available feedback, this very simple system without considering expected decoder distortion and slice structuring and just relying on multiple reference frames outperforms many highly sophisticated error-resilience schemes as long as the delay of the feedback is reasonable. The combination of these methods according to feedback mode 3 is currently investigated and should allow adaptively selecting the best methods, however, with significantly increased encoding complexity.

VI. Conclusions

H.264/AVC promises some significant advances of the state-of-the-art of standardized video coding in mobile applications. In addition to excellent coding efficiency, the design of H.264/AVC also takes into account network adaptation providing large flexibility for its use in wireless applications. The tools provided in H.264/AVC for error resilience do not necessarily differ from the compression efficiency features such as intra MBs or multiple reference frames. However, in the case of error-prone transmission, the selection methods have to be changed by using the expected decoder distortion or by restricting the set of accessible coding options. In experimental results based on common test conditions, it has been shown that in case without any feedback, several slices in combination with channel-adaptive R-D optimized mode selection is a promising approach. In this case, further investigation with advanced error-resilience tools such as flexible MB ordering, data partitioning, and generic forward error correction, might provide benefits. However, in the case of available feedback, the application of multiple reference frames to exclude error propagation without slice structuring provides excellent results.

Acknowledgment

The authors would like to thank T. Oelbaum, D. Kontopodis, Y.-K. Wang, V. Varsa, and A. Hourunranta for implementing and testing parts of the algorithms, V. Varsa and G. Liebl for providing the test conditions and software simulator, S. Wenger, N. Färber, K. Stuhlmüller, E. Steinbach, and B. Girod for useful discussions, and JVT for the collaborative work and the technically outstanding discussions and contributions.

References

[1] "Final committee draft: Editor's proposed revisions," in *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG*, T. Wiegand, Ed., Feb. 2003, JVT-F100.

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, July 2003.

Google Exhibit No. 1063
Page 0990

[3] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 688–703, July 2003.
[4] "Video Coding for Low Bitrate Communication, Version 1," ITU-T, ITU-T Recommendation H.263, 1995.
[5] "Coding of Audio-Visual Objects—Part 2: Visual," ISO/IEC JTC1, ISO/IEC 14 496-2 (MPEG-4 visual version 1), 1999.
[6] "Multimedia Messaging Service (MMS); Media Formats and Codecs," 3GPP Technical Specification 3GPP TR 26.140.
[7] S. Wenger, M. Hannuksela, and T. Stockhammer, "Identified H.26L Applications," ITU-T SG 16, Doc. VCEG-L34, Eibsee, Germany, 2001.
[8] "Codec for Circuit Switched Multimedia Telephony Service; General Description," 3GPP Technical Specification 3GPP TR 26.110.
[9] "Packet Switched Conversational Multimedia Applications; Default Codecs," 3GPP Technical Specification 3GPP TR 26.235.
[10] "Transparent End-to-End Packet Switched Streaming Service (PSS); RTP Usage Model," 3GPP Technical Specification 3GPP TR 26.937.
[11] "Multimedia Messaging Service (MMS); Media Formats and Codecs," 3GPP Technical Specification 3GPP TR 26.140.
[12] "Multimedia Broadcast/Multicast Services," 3GPP Technical Specification 3GPP TR 29.846.
[13] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 704–716, July 2003.
[14] V. Lappalainen, A. Hallapuro, and T. D. Hämäläinen, "Complexity of optimized H.264/AVC video decoder implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 717–725, July 2003.
[15] "High Speed Downlink Packet Access (HSDPA); Overall UTRAN Description," 3GPP Technical Specification 3GPP TR 25.855.
[16] H. Holma and A. Toskala, Eds., *WCDMA for UMTS: Radio Access For Third Generation Mobile Communications.* New York: Wiley, 2000.
[17] *Narrow-Band Visual Telephone Systems and Terminal Equipment, Rev. 4*, ITU-T Recommendation H.320, 1999.
[18] "Generic Coding of Moving Pictures and Associated Audio Information," ISO/IEC International Standard 13 818, 1994.
[19] S. Wenger, T. Stockhammer, and M. M. Hannuksela, "RTP payload format for H.264 video," in *Internet Draft, Work in Progress*, Mar. 2003, Draft-wenger-avt-rtp-h264-01.txt.
[20] "3rd GPP; Technical Specification Group Core Network; IP Multimedia Call Control Protocol Based on SIP and SDP," 3GPP Technical Specification 3GPP TS 24.229.
[21] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 645–656, July 2003.
[22] D. Lindberg, "The H.324 multimedia communication standard," *IEEE Commun. Mag.*, vol. 34, pp. 46–51, Dec. 1996.
[23] H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "RObust header compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," in RFC 3095, July 2001.
[24] "Common conditions for video performance evaluation in H.324/M error-prone systems, VCEG (SG16/Q15)," in Ninth Meeting, Redbank, NJ, Oct. 1999, ITU-T Q15-I-60.
[25] G. Roth, R. Sjöberg, G. Liebl, T. Stockhammer, V. Varsa, and M. Karczewicz, "Common Test Conditions for RTP/IP Over 3GPP/3GPP2,", Austin, TX, ITU-T SG16 Doc. VCEG-M77, 2001.
[26] "Radio Link Control (RLC) Protocol Specification," 3GPP Technical Specification 3GPP TS 25.322.
[27] S. S. Hemami, "Robust image transmission using resynchronizing variable-length codes and error concealment," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 927–939, June 2000.
[28] J. Ribas-Corbera, P. A. Chou, and S. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 674–687, July 2003.
[29] H. Jenkac, T. Stockhammer, and G. Kuhn, "Streaming media in variable bit-rate environments," presented at the Packet Video Workshop, Nantes, France, Apr..
[30] B. Girod, M. Kalman, Y. J. Liang, and R. Zhang, "Advances in video channel-adaptive streaming," in ICIP 2002, Rochester, NY, Sept. 2002.
[31] E. G. Steinbach, N. Färber, and B. Girod, "Adaptive play-out for law latency video streaming," presented at the ICIP 2001, Thessaloniki, Greece, Oct. 2001.
[32] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, submitted for publication.
[33] Y. J. Liang and B. Girod, "Rate-distortion optimized low—Latency video streaming using channel-adaptive bitstream assembly," presented at the ICME2002, Lausanne, Switzerland, Aug. 2002.
[34] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, pp. 53–68, Mar. 2001.
[35] M. Karczewicz and R. Kurçeren, "The SP and SI frames design for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 637–644, July 2003.
[36] V. Varsa, M. M. Hannuksela, and Y. Wang, "Non-Normative Error Concealment Algorithms," ITU-T VCEG-N62, 2001.
[37] S. Wenger and G. Cotê, "Using RFC 2429 and H.263+ at low to medium bit-rates for low-latency applications," presented at the Proc. Packet Video Workshop, New York, NY, Apr. 1999.
[38] V. Varsa and M. Karczewicz, "Slice interleaving in compressed video packetization," presented at the Packet Video Workshop 2000, Forte Village, Italy, May 2000.
[39] S. Wenger and M. Horowitz, "Flexible MB ordering—A new error resilience tool for IP-based video," presented at the IWDC 2002, Capri, Italy, Sept. 2002.
[40] J. Rosenberg and H. Schulzrine, "An RTP payload format for generic forward error correction," in RFC 2733, Dec. 1999.
[41] G. Carle and E. W. Biersack, "Survey of error recovery techniques for IP-based audio-visual multicast applications," *IEEE Network Mag.*, vol. 11, pp. 2–14, Nov. 1997.
[42] U. Horn, K. Stuhlmüller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *IEEE Trans. Image Processing*, vol. 15, pp. 77–94, Sept. 1999.
[43] Q. F. Zhu and L. Kerofsky, "Joint source coding, transport processing, and error concealment for H.323-based packet video," *Proc. SPIE VCIP*, vol. 3653, pp. 52–62, Jan. 1999.
[44] P. Haskell and D. Messerschmitt, "Resynchronization of motion-compensated video affected by ATM cell loss," *Proc. IEEE ICASSP*, vol. 3, pp. 545–548, 1992.
[45] J. Liao and J. Villasenor, "Adaptive intra update for video coding over noisy channels," *Proc. ICIP*, vol. 3, pp. 763–766, Oct. 1996.
[46] S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error resilience support in H.263+," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 867–877, Nov. 1998.
[47] P. Salama, N. B. Shroff, and E. J. Delp, "Error concealment in encoded video," in *Image Recovery Techniques for Image Compression Applications.* Norwell, MA: Kluwer, 1998.
[48] W. M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," in *Proc. ICASSP*, vol. 5, Mar. 1993, pp. 417–420.
[49] V. Varsa, M. M. Hannuksela, and Y.-K. Wang, "Non-Normative Error Concealment Algorithms," ITU-T VCEG-N62, 2001.
[50] Y.-K. Wang, M. M. Hannuksela, V. Varsa, A. Hourunranta, and M. Gabbouj, "The error concealment feature in the H.26L test model," in *Proc. ICIP*, vol. 2, Sept. 2002, pp. 729–732.
[51] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.
[52] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Mag.*, vol. 15, pp. 74–90, Nov. 1998.
[53] T. Wiegand and B. Girod, "Lagrangian multiplier selection in hybrid video coder control," presented at the Proc. ICIP 2001, Oct. 2001.
[54] T. Stockhammer, D. Kontopodis, and T. Wiegand, "Rate-distortion optimization for H.26L video coding in packet loss environment," presented at the Packet Video Workshop 2002, Pittsburgh, PA, Apr. 2002.
[55] G. Cote, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error-prone networks," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 952–965, Dec. 2000.
[56] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod, "Error-resilient video transmission using long-term memory motion-compensated prediction," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1050—1050-, Dec. 2000.
[57] C. W. Kim, D. W. Kang, and I. S. Kwang, "High-complexity mode decision for error prone environment," in JVT-C101, May 2002.
[58] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 966–976, 2000.
[59] Y.-K. Wang and M. M. Hannuksela, "Error-robust video coding using isolated regions," in JVT-C073, May 2002.
[60] Y.-K. Wang, M. M. Hannuksela, and M. Gabbouj, "Error-robust inter/intra mode selection using isolated regions," in Proc. Int. Packet Video Workshop 2003, Nantes, France, Apr. 2003.

Google Exhibit No. 1063
Page 0991

[61] E. Steinbach, N. Färber, and B. Girod, "Standard compatible extension of H.263 for robust video transmission in mobile environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 872–881, Dec. 1997.
[62] B. Girod and N. Färber, "Feedback-based error control for mobile video transmission," *Proc. IEEE*, vol. 97, pp. 1707–1723, Oct. 1999.
[63] W. Wada, "Selective recovery of video packet losses using error concealment," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 807–814, June 1989.
[64] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," *Proc. IEEE*, vol. 86, pp. 974–997, May 1998.
[65] T. Nakai and Y. Tomita, "Core Experiments on Feedback Channel Operation for H.263+," ITU-T SG15 LBC96-308, 1996.
[66] S. Fukunaga, T. Nakai, and H. Inoue, "Error resilient video coding by dynamic replacing of reference pictures," in *Proc. IEEE GLOBECOM*, vol. 3, Nov. 1996, pp. 1503–1508.
[67] Y. Tomita, T. Kimura, and T. Ichikawa, "Error resilient modified inter-frame coding system for limited reference picture memories," presented at the Picture Coding Symposium, Berlin, Germany, Sept. 1997.
[68] "Video Coding for Low Bit-Rate Communication, Version 1," ITU-T Recommendation H.263, 1995.
[69] T. Stockhammer, T. Wiegand, T. Oelbaum, and F. Obermeier, "Video coding and transport layer techniques for H.264-based transmission over packet-lossy networks," presented at the ICIP, Barcelona, Spain, Sept. 2003.
[70] T. Stockhammer and S. Wenger, "Standard-compliant enhancement of JVT coded video for transmission over fixed and wireless IP," presented at the IWDC 2002, Capri, Italy, Sept. 2002.



Thomas Stockhammer received the Diplom.-Ing. degree in electrical engineering in 1996 from the Munich University of Technology (TUM), Munich, Germany, where he is currently working toward the Dr.-Ing. degree in the area of source and video transmission over mobile and packet-lossy channels.

In 1996, he visited Rensselear Polytechnic Institute (RPI), Troy, NY to perform his diploma thesis in the area of combined source channel coding for video and coding theory. There he began research in video transmission and combined source and channel coding. In 2000, he was Visiting Researcher in the Information Coding Laboratory, University of San Diego at California (UCSD). Since then, he has published several conference and journal papers and holds several patents. He regularly participates and contributes to different standardization activities, e.g., ITU-T H.324, H.264, ISO/IEC MPEG, JVT, and IETF. He acts as a member of several technical program committees, as Reviewer for different journals, and as an Evaluator for the European Commission. His research interests include joint source and channel coding, video transmission, system design, rate-distortion optimization, information theory, and mobile communications.



Miska M. Hannuksela received the M.S. degree in engineering from Tampere University of Technology, Tampere, Finland, in 1997.

He is currently a Research Manager in the Visual Communications Laboratory, Nokia Research Center, Tampere, Finland. From 1996 to 1999, he was a Research Engineer with Nokia Research Center in the area of mobile video communications. From 2000 to 2003, he was a Project Team Leader and a specialist in various mobile multimedia research and product projects in Nokia Mobile Phones. He has been an active participant in the ITU-T Video Coding Experts Group since 1999 and in the Joint Video Team of ITU-T and ISO/IEC since its foundation in 2001. He has co-authored more than 80 technical contributions to these standardization groups. His research interests include video error resilience, scalable video coding, and video communication systems.



Thomas Wiegand received the Dr.-Ing. degree from the University of Erlangen-Nuremberg, Erlangen-Nuremberg, Germany, in 2000 and the Dipl.-Ing. degree in electrical engineering from the Technical University of Hamburg-Harburg, Hamburg-Harburg, Germany, in 1995.

He is the Head of the Image Communication Group in the Image Processing Department, Fraunhofer–Institute for Telecommunications – Heinrich Hertz Institute (HHI), Berlin, Germany. During 1997 to 1998, he was a Visiting Researcher at Stanford University, Stanford, CA, and served as a Consultant to 8x8, Inc., Santa Clara, CA. From 1993 to 1994, he was a Visiting Researcher at Kobe University, Kobe, Japan. In 1995, he was a Visiting Scholar at the University of California at Santa Barbara, where he began his research on video compression and transmission. Since then, he has published several conference and journal papers on the subject and has contributed successfully to the ITU-T Video Coding Experts Group (ITU-T SG16 Q.6—VCEG)/ISO/IEC Moving Pictures Experts Group (ISO/IEC JTC1/SC29/WG11—MPEG)/Joint Video Team (JVT) standardization efforts and holds various international patents in this field. He has been appointed as the Associated Rapporteur of the ITU-T VCEG (October 2000), the Associated Rapporteur/Co-Chair of the JVT that has been created by ITU-T VCEG and ISO/IEC MPEG for finalization of the H.264/AVC video coding standard (December 2001), and the Editor of the H.264/AVC video coding standard (February 2002).

Google Exhibit No. 1063
Page 0992

Ex. 1027B

LIBRARY CATALOG

1 of 1

PERIODICAL OR NEWSPAPER

IEEE transactions on circuits and systems for video technology : a publication ...

Full Record MARC Tags

Main title

IEEE transactions on circuits and systems for video technology : a publication of the Circuits and Systems Society.

Published/Created

New York, NY : Institute of Electrical and Electronics Engineers, 1991-

Request this Item LC Find It

More Information

LCCN Permalink	https://lccn.loc.gov/91642188
Publication history	Vol. 1, no. 1 (Mar. 1991)-
Description	volumes : illustrations ; 28 cm
Current frequency	Monthly, 2001-
Former frequency	Quarterly, 1991-1992 Six no. per year, 1993-1997 Eight no. per year, 1998-2000

Google Exhibit No. 1063
Page 0994

ISSN	1051-8215
Linking ISSN	1051-8215
LC classification	TK6630.A1 I37
Variant title	Institute of Electrical and Electronics Engineers transactions on circuits and systems for video technology
Portion of title	Circuits and systems for video technology
Serial key title	IEEE transactions on circuits and systems for video technology (Print)
Abbreviated title	IEEE trans. circuits syst. video technol. (Print)
Related names	IEEE Circuits and Systems Society.
LC Subjects	Television--Equipment and supplies--Periodicals. Television circuits--Periodicals. Image processing--Periodicals. Signal processing--Periodicals.
Other Subjects	Bildverarbeitung. Videotechnik. Zeitschrift. Image processing. Signal processing. Television circuits. Television--Equipment and supplies. Bildverarbeitung. Videotechnik. Zeitschrift.
Form/Genre	Periodicals. Periodicals.
Browse by shelf order	TK6630.A1
Notes	Title from cover.
Reproduction no./Source	IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331
Additional formats	Online version: IEEE transactions on circuits and systems for video

Google Exhibit No. 1063
Page 0995

technology (Online) 1558-2205 (DLC) 2005215235 (OCoLC)44466031

LCCN	91642188
Invalid LCCN	sn 90001281
CODEN	ITCTEM
Dewey class no.	621
National bib no.	01605380X
National bib agency no.	01605380X 1074294-3
Other system no.	(OCoLC)ocm22134460
Additional Links	http://ieeexplore.ieee.org/servlet/opac?punumber=76
Type of material	Periodical or Newspaper
Content type	text
Media type	unmediated
Carrier type	volume
Source of description	Latest issue consulted: Vol. 1, no. 2 (June 1991).

Item Availability

CALL NUMBER	TK6630.A1 I37 Set 1
Request in	Jefferson or Adams Building Reading Rooms
Status	Not Charged
Older receipts	v.5-v.6 (1995-1996:Apr.), v.6-v.16:no.1 (1996:Aug.-2006:Jan.), v.16:no.3-v.16:no.10 (2006:Mar.-2006:Oct.), v.16:no.12-v.18:no.3 (2006:Dec.-2008:Mar.), v.18:no.6-v.33:no.12 (2008:June-2023:Dec.)

Google Exhibit No. 1063
Page 0996

CALL NUMBER	TK6630.A1 I37 Copy 1 Unbound issues
Request in	Newspaper & Current Periodical Reading Room (Madison LM133)
Latest receipts	v. 34, no. 12 (2024 Dec.) 2 Pts. v. 34, no. 11 (2024 Nov.) 2 Pts. v. 34, no. 10 (2024 Oct.) 2 Pts. v. 34, no. 9 (2024 Sept.) v. 34, no. 8 (2024 Aug.) v. 34, no. 7 (2024 July) v. 34, no. 6 (2024 June) v. 34, no. 5 (2024 May) v. 34, no. 4 (2024 Apr.) v. 34, no. 3 (2024 Mar.) v. 34, no. 2 (2024 Feb.) v. 34, no. 1 (2024 Jan.)

Google Exhibit No. 1063
Page 0997

LIBRARY CATALOG

1 of 1

PERIODICAL OR NEWSPAPER

IEEE transactions on circuits and systems for video technology : a publication ...

Full Record MARC Tags

000		03727cas a2200793 a 4500
001		11385851
005		20220506065655.0
008		900724c19919999nyumr p 0 a0eng c
010	__	|a 91642188 |z sn 90001281
015	__	|a 01605380X |2 dnb
016	7_	|a 01605380X |2 DE-101
016	7_	|a 1074294-3 |2 DE-600
022	0_	|a 1051-8215 |l 1051-8215 |2 1
030	__	|a TCTEM
032	__	|a 006329 |b JSPS
035	__	|a (OCoLC)ocm22134460
037	__	|b IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331
040	__	|a NSD |b eng |c NSD |d NYG |d MYG |d HUL |d NSD |d DLC |d NSD |d DLC |d CAS | d WAU |d OCL |d NSD |d ELM |d NST |d NSD |d DLC |d MYG |d CUS |d OCL |d MYG | d SYS |d EEEX |d OCLCQ |d DLC |d NSD |d OCLCQ |d HEBIS |d DEBBG |d GBVCP | d OCLCO |d OCLCQ |d OCLCF |d OCLCO |d OCLCA |d IUL |d GILDS |d UMC
042	__	|a nsdp |a pcc
050	00	|a TK6630.A1 |b I37
082	10	|a 621 |2 12
210	0_	|a EEE trans. circuits syst. video technol. |b (Print)
222	_0	|a EEE transactions on circuits and systems for video technology |b (Print)

Google Exhibit No. 1063
Page 0998

245	00	|a IEEE transactions on circuits and systems for video technology : |b a publication of the Circuits and Systems Society.
246	3_	|a Institute of Electrical and Electronics Engineers transactions on circuits and systems for video technology
246	30	|a Circuits and systems for video technology
260	__	|a New York, NY : |b Institute of Electrical and Electronics Engineers, |c 1991-
300	__	|a volumes : |b illustrations ; |c 28 cm
310	__	|a Monthly, |b 2001-
321	__	|a Quarterly, |b 1991-1992
321	__	|a Six no. per year, |b 1993-1997
321	__	|a Eight no. per year, |b 1998-2000
336	__	|a text |b txt |2 rdacontent
337	__	|a unmediated |b n |2 rdamedia
338	__	|a volume |b nc |2 rdacarrier
362	0_	|a Vol. 1, no. 1 (Mar. 1991)-
500	__	|a Title from cover.
588	__	|a Latest issue consulted: Vol. 1, no. 2 (June 1991).
590	__	|a SERBIB/SERLOC merged record
650	_0	|a Television |x Equipment and supplies |v Periodicals.
650	_0	|a Television circuits |v Periodicals.
650	_0	|a Image processing |v Periodicals.
650	_0	|a Signal processing |v Periodicals.
650	07	|a Bildverarbeitung. |2 swd
650	07	|a Videotechnik. |2 swd
650	07	|a Zeitschrift. |2 swd
650	_7	|a Image processing. |2 fast |0 (OCoLC)fst00967501
650	_7	|a Signal processing. |2 fast |0 (OCoLC)fst01118281
650	_7	|a Television circuits. |2 fast |0 (OCoLC)fst01146830
650	_7	|a Television |x Equipment and supplies. |2 fast |0 (OCoLC)fst01146552
650	07	|a Bildverarbeitung. |0 (DE-588)4006684-8 |2 gnd
650	07	|a Videotechnik. |0 (DE-588)4063466-8 |2 gnd
650	07	|a Zeitschrift. |0 (DE-588)4067488-5 |2 gnd
655	_7	|a Periodicals. |2 fast |0 (OCoLC)fst01411641
655	_7	|a Periodicals. |2 lcgft
710	2_	|a IEEE Circuits and Systems Society.
776	08	|i Online version: |t IEEE transactions on circuits and systems for video technology (Online) |x 1558-2205 |w (DLC) 2005215235 |w (OCoLC)44466031

Google Exhibit No. 1063
Page 0999

850	__	|a AAP |a AU |a ArU |a AzTeS |a CCC |a CPT |a CSt |a CU-A |a CU-RivP |a CU-SB | a CoFS |a CoU |a DGW |a DLC |a GAT |a ICarbS |a IaU |a InLP |a KyU |a MBU |a MCM | a MH-GM |a MdBJ |a MMet |a MiEM |a MiU |a MoKL |a MoU |a MsSM |a N |a NN |a NNC | a NRU |a NSyU |a NcD |a NcRS |a NjP |a NjR |a NmU |a OCU |a OU |a OrCS |a PPD | a PPiC |a PU |a RPB-S |a TU |a TxCM |a TxHR |a ViBlbV |a WM |a WaU
859	41	|u http://ieeexplore.ieee.org/servlet/opac?punumber=76
853	33	|8 1 |a v. |b no. |i (year) |j (month)
863	40	|8 1 |a <1>- |i <1991>- |x provisional
890	__	|a IEEE transactions on circuits and systems for video technology. |i 91-642188
906	__	|a 7 |b cbc |c serials |d 2 |e ncip |f 19 |g n-oclcserc
920	__	|a Keep 1
923	__	|d 20020725 |s Group Registration
984	__	|a srvf
985	__	|e eserial 200406
991	__	|b c-GenColl |h TK6630.A1 |i I37 |w SERIALS
992	__	|b SER |w SERLOC

Request this Item | LC Find It

Item Availability

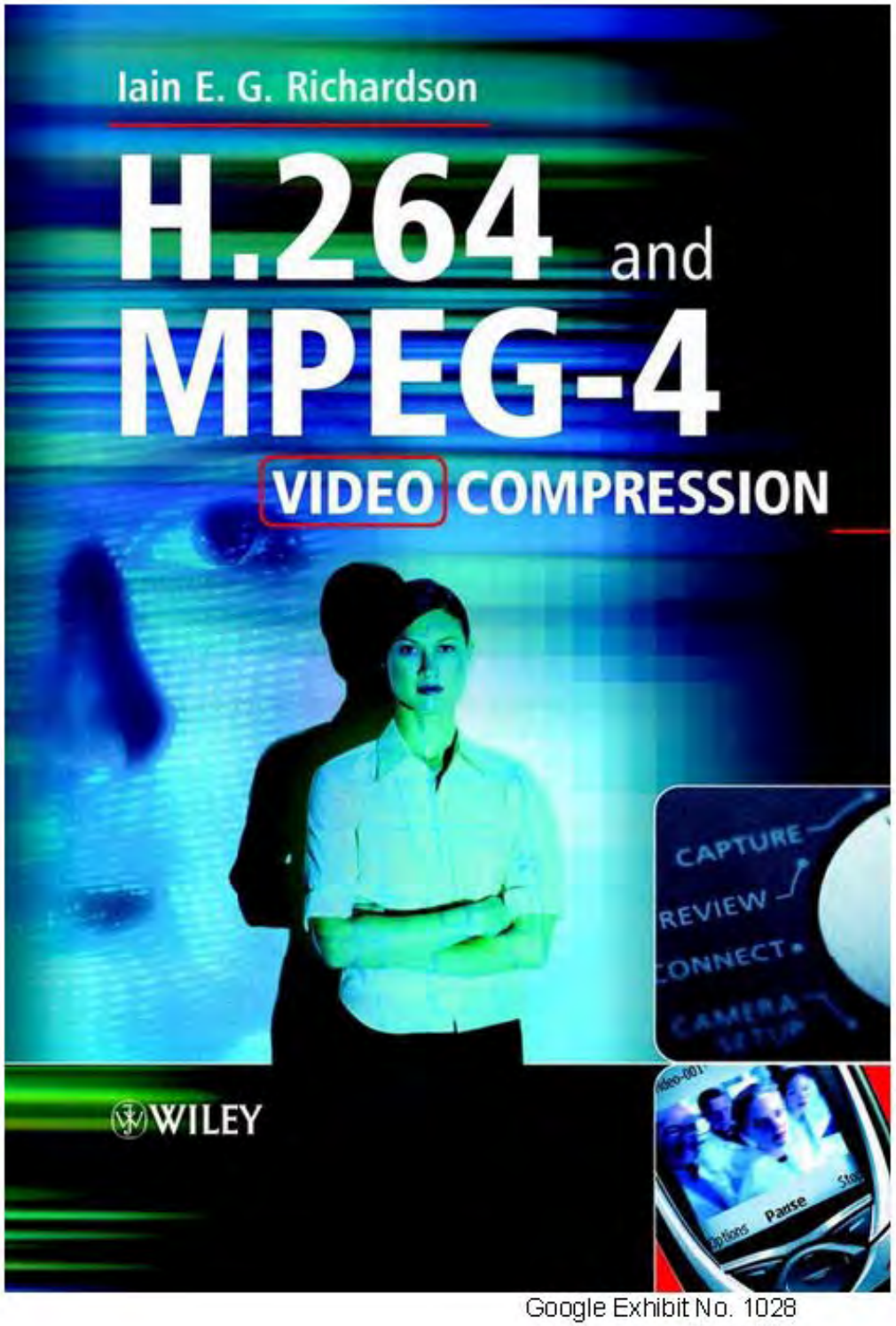
CALL NUMBER	TK6630.A1 I37 Set 1
Request in	Jefferson or Adams Building Reading Rooms
Status	Not Charged
Older receipts	v.5-v.6 (1995-1996:Apr.), v.6-v.16:no.1 (1996:Aug.-2006:Jan.), v.16:no.3-v.16:no.10 (2006:Mar.-2006:Oct.), v.16:no.12-v.18:no.3 (2006:Dec.-2008:Mar.), v.18:no.6-v.33:no.12 (2008:June-2023:Dec.)

CALL NUMBER	TK6630.A1 I37 Copy 1 Unbound issues

Google Exhibit No. 1063
Page 1000

Request in	Newspaper & Current Periodical Reading Room (Madison LM133)
Latest receipts	v. 34, no. 12 (2024 Dec.) 2 Pts. v. 34, no. 11 (2024 Nov.) 2 Pts. v. 34, no. 10 (2024 Oct.) 2 Pts. v. 34, no. 9 (2024 Sept.) v. 34, no. 8 (2024 Aug.) v. 34, no. 7 (2024 July) v. 34, no. 6 (2024 June) v. 34, no. 5 (2024 May) v. 34, no. 4 (2024 Apr.) v. 34, no. 3 (2024 Mar.) v. 34, no. 2 (2024 Feb.) v. 34, no. 1 (2024 Jan.)

Google Exhibit No. 1063
Page 1001



Google Exhibit No. 1028
Page 001

H.264 and MPEG-4 Video Compression

Google Exhibit No. 1028
Page 002

H.264 and MPEG-4 Video Compression

Video Coding for Next-generation Multimedia

Iain E. G. Richardson
The Robert Gordon University, Aberdeen, UK



Google Exhibit No. 1028
Page 003

Copyright © 2003 John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wileyeurope.com or www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1T 4LP, UK, without the permission in writing of the Publisher. Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or emailed to permreq@wiley.co.uk, or faxed to (+44) 1243 770620.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 33 Park Road, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 22 Worcester Road, Etobicoke, Ontario, Canada M9W 1L1

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-470-84837-5

Typeset in 10/12pt Times roman by TechBooks, New Delhi, India
Printed and bound in Great Britain by Antony Rowe, Chippenham, Wiltshire
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Google Exhibit No. 1028
Page 004

Google Exhibit No. 1063
Page 1005

To Phyllis

Google Exhibit No. 1028
Page 005

Contents

About the Author xiii

Foreword xv

Preface xix

Glossary xxi

1 Introduction 1
1.1 The Scene 1
1.2 Video Compression 3
1.3 MPEG-4 and H.264 5
1.4 This Book 6
1.5 References 7

2 Video Formats and Quality 9
2.1 Introduction 9
2.2 Natural Video Scenes 9
2.3 Capture 10
2.3.1 Spatial Sampling 11
2.3.2 Temporal Sampling 11
2.3.3 Frames and Fields 13
2.4 Colour Spaces 13
2.4.1 RGB 14
2.4.2 YCbCr 15
2.4.3 YCbCr Sampling Formats 17
2.5 Video Formats 19
2.6 Quality 20
2.6.1 Subjective Quality Measurement 21
2.6.2 Objective Quality Measurement 22
2.7 Conclusions 24
2.8 References 24

Google Exhibit No. 1028
Page 006

Google Exhibit No. 1063
Page 1007

3 Video Coding Concepts **27**
3.1 Introduction 27
3.2 Video CODEC 28
3.3 Temporal Model 30
3.3.1 Prediction from the Previous Video Frame 30
3.3.2 Changes due to Motion 30
3.3.3 Block-based Motion Estimation and Compensation 32
3.3.4 Motion Compensated Prediction of a Macroblock 33
3.3.5 Motion Compensation Block Size 34
3.3.6 Sub-pixel Motion Compensation 37
3.3.7 Region-based Motion Compensation 41
3.4 Image model 42
3.4.1 Predictive Image Coding 44
3.4.2 Transform Coding 45
3.4.3 Quantisation 51
3.4.4 Reordering and Zero Encoding 56
3.5 Entropy Coder 61
3.5.1 Predictive Coding 61
3.5.2 Variable-length Coding 62
3.5.3 Arithmetic Coding 69
3.6 The Hybrid DPCM/DCT Video CODEC Model 72
3.7 Conclusions 82
3.8 References 83

4 The MPEG-4 and H.264 Standards **85**
4.1 Introduction 85
4.2 Developing the Standards 85
4.2.1 ISO MPEG 86
4.2.2 ITU-T VCEG 87
4.2.3 JVT 87
4.2.4 Development History 88
4.2.5 Deciding the Content of the Standards 88
4.3 Using the Standards 89
4.3.1 What the Standards Cover 90
4.3.2 Decoding the Standards 90
4.3.3 Conforming to the Standards 91
4.4 Overview of MPEG-4 Visual/Part 2 92
4.5 Overview of H.264 / MPEG-4 Part 10 93
4.6 Comparison of MPEG-4 Visual and H.264 94
4.7 Related Standards 95
4.7.1 JPEG and JPEG2000 95
4.7.2 MPEG-1 and MPEG-2 95
4.7.3 H.261 and H.263 96
4.7.4 Other Parts of MPEG-4 97
4.8 Conclusions 97
4.9 References 98

Google Exhibit No. 1028
Page 007

Google Exhibit No. 1063
Page 1008

5 MPEG-4 Visual **99**
5.1 Introduction 99
5.2 Overview of MPEG-4 Visual (Natural Video Coding) 100
5.2.1 Features 100
5.2.2 Tools, Objects, Profiles and Levels 100
5.2.3 Video Objects 103
5.3 Coding Rectangular Frames 104
5.3.1 Input and Output Video Format 106
5.3.2 The Simple Profile 106
5.3.3 The Advanced Simple Profile 115
5.3.4 The Advanced Real Time Simple Profile 121
5.4 Coding Arbitrary-shaped Regions 122
5.4.1 The Core Profile 124
5.4.2 The Main Profile 133
5.4.3 The Advanced Coding Efficiency Profile 138
5.4.4 The N-bit Profile 141
5.5 Scalable Video Coding 142
5.5.1 Spatial Scalability 142
5.5.2 Temporal Scalability 144
5.5.3 Fine Granular Scalability 145
5.5.4 The Simple Scalable Profile 148
5.5.5 The Core Scalable Profile 148
5.5.6 The Fine Granular Scalability Profile 149
5.6 Texture Coding 149
5.6.1 The Scalable Texture Profile 152
5.6.2 The Advanced Scalable Texture Profile 152
5.7 Coding Studio-quality Video 153
5.7.1 The Simple Studio Profile 153
5.7.2 The Core Studio Profile 155
5.8 Coding Synthetic Visual Scenes 155
5.8.1 Animated 2D and 3D Mesh Coding 155
5.8.2 Face and Body Animation 156
5.9 Conclusions 156
5.10 References 156

6 H.264/MPEG-4 Part 10 **159**
6.1 Introduction 159
6.1.1 Terminology 159
6.2 The H.264 CODEC 160
6.3 H.264 structure 162
6.3.1 Profiles and Levels 162
6.3.2 Video Format 162
6.3.3 Coded Data Format 163
6.3.4 Reference Pictures 163
6.3.5 Slices 164
6.3.6 Macroblocks 164

Google Exhibit No. 1028
Page 008

Google Exhibit No. 1063
Page 1009

6.4 The Baseline Profile 165
6.4.1 Overview 165
6.4.2 Reference Picture Management 166
6.4.3 Slices 167
6.4.4 Macroblock Prediction 169
6.4.5 Inter Prediction 170
6.4.6 Intra Prediction 177
6.4.7 Deblocking Filter 184
6.4.8 Transform and Quantisation 187
6.4.9 4 × 4 Luma DC Coefficient Transform and Quantisation (16 × 16 Intra-mode Only) 194
6.4.10 2 × 2 Chroma DC Coefficient Transform and Quantisation 195
6.4.11 The Complete Transform, Quantisation, Rescaling and Inverse Transform Process 196
6.4.12 Reordering 198
6.4.13 Entropy Coding 198
6.5 The Main Profile 207
6.5.1 B Slices 207
6.5.2 Weighted Prediction 211
6.5.3 Interlaced Video 212
6.5.4 Context-based Adaptive Binary Arithmetic Coding (CABAC) 212
6.6 The Extended Profile 216
6.6.1 SP and SI slices 216
6.6.2 Data Partitioned Slices 220
6.7 Transport of H.264 220
6.8 Conclusions 222
6.9 References 222

7 Design and Performance 225
7.1 Introduction 225
7.2 Functional Design 225
7.2.1 Segmentation 226
7.2.2 Motion Estimation 226
7.2.3 DCT/IDCT 234
7.2.4 Wavelet Transform 238
7.2.5 Quantise/Rescale 238
7.2.6 Entropy Coding 238
7.3 Input and Output 241
7.3.1 Interfacing 241
7.3.2 Pre-processing 242
7.3.3 Post-processing 243
7.4 Performance 246
7.4.1 Criteria 246
7.4.2 Subjective Performance 247
7.4.3 Rate–distortion Performance 251

Google Exhibit No. 1028
Page 009

Google Exhibit No. 1063
Page 1010

7.4.4 Computational Performance 254
7.4.5 Performance Optimisation 255
7.5 Rate control 256
7.6 Transport and Storage 262
7.6.1 Transport Mechanisms 262
7.6.2 File Formats 263
7.6.3 Coding and Transport Issues 264
7.7 Conclusions 265
7.8 References 265

8 Applications and Directions 269
8.1 Introduction 269
8.2 Applications 269
8.3 Platforms 270
8.4 Choosing a CODEC 270
8.5 Commercial issues 272
8.5.1 Open Standards? 273
8.5.2 Licensing MPEG-4 Visual and H.264 274
8.5.3 Capturing the Market 274
8.6 Future Directions 275
8.7 Conclusions 276
8.8 References 276

Bibliography 277

Index 279

Google Exhibit No. 1028
Page 010

Google Exhibit No. 1063
Page 1011

About the Author

Iain Richardson is a lecturer and researcher at The Robert Gordon University, Aberdeen, Scotland. He was awarded the degrees of MEng (Heriot-Watt University) and PhD (The Robert Gordon University) in 1990 and 1999 respectively. He has been actively involved in research and development of video compression systems since 1993 and is the author of over 40 journal and conference papers and two previous books. He leads the Image Communication Technology Research Group at The Robert Gordon University and advises a number of companies on video compression technology issues.

Google Exhibit No. 1028
Page 011

Google Exhibit No. 1063
Page 1012

Foreword

Work on the emerging "Advanced Video Coding" standard now known as ITU-T Recommendation H.264 and as ISO/IEC 14496 (MPEG-4) Part 10 has dominated the video coding standardization community for roughly the past three years. The work has been stimulating, intense, dynamic, and all consuming for those of us most deeply involved in its design. The time has arrived to see what has been accomplished.

Although not a direct participant, Dr Richardson was able to develop a high-quality, up-to-date, introductory description and analysis of the new standard. The timeliness of this book is remarkable, as the standard itself has only just been completed.

The new H.264/AVC standard is designed to provide a technical solution appropriate for a broad range of applications, including:

- Broadcast over cable, satellite, cable modem, DSL, terrestrial.
- Interactive or serial storage on optical and magnetic storage devices, DVD, etc.
- Conversational services over ISDN, Ethernet, LAN, DSL, wireless and mobile networks, modems.
- Video-on-demand or multimedia streaming services over cable modem, DSL, ISDN, LAN, wireless networks.
- Multimedia messaging services over DSL, ISDN.

The range of bit rates and picture sizes supported by H.264/AVC is correspondingly broad, addressing video coding capabilities ranging from very low bit rate, low frame rate, "postage stamp" resolution video for mobile and dial-up devices, through to entertainment-quality standard-definition television services, HDTV, and beyond. A flexible system interface for the coded video is specified to enable the adaptation of video content for use over this full variety of network and channel-type environments. However, at the same time, the technical design is highly focused on providing the two limited goals of high coding efficiency and robustness to network environments for conventional rectangular-picture camera-view video content. Some potentially-interesting (but currently non-mainstream) features were deliberately left out (at least from the first version of the standard) because of that focus (such as support of arbitrarily-shaped video objects, some forms of bit rate scalability, 4:2:2 and 4:4:4 chroma formats, and color sampling accuracies exceeding eight bits per color component).

Google Exhibit No. 1028
Page 012

Google Exhibit No. 1063
Page 1013

In the work on the new H.264/AVC standard, a number of relatively new technical developments have been adopted. For increased coding efficiency, these include improved prediction design aspects as follows:

- Variable block-size motion compensation with small block sizes,
- Quarter-sample accuracy for motion compensation,
- Motion vectors over picture boundaries,
- Multiple reference picture motion compensation,
- Decoupling of referencing order from display order,
- Decoupling of picture representation methods from the ability to use a picture for reference,
- Weighted prediction,
- Improved "skipped" and "direct" motion inference,
- Directional spatial prediction for intra coding, and
- In-the-loop deblocking filtering.

In addition to improved prediction methods, other aspects of the design were also enhanced for improved coding efficiency, including:

- Small block-size transform,
- Hierarchical block transform,
- Short word-length transform,
- Exact-match transform,
- Arithmetic entropy coding, and
- Context-adaptive entropy coding.

And for robustness to data errors/losses and flexibility for operation over a variety of network environments, some key design aspects include:

- Parameter set structure,
- NAL unit syntax structure,
- Flexible slice size,
- Flexible macroblock ordering,
- Arbitrary slice ordering,
- Redundant pictures,
- Data partitioning, and
- SP/SI synchronization switching pictures.

Prior to the H.264/AVC project, the big recent video coding activity was the MPEG-4 Part 2 (Visual) coding standard. That specification introduced a new degree of creativity and flexibility to the capabilities of the representation of digital visual content, especially with its coding of video "objects", its scalability features, extended N-bit sample precision and 4:4:4 color format capabilities, and its handling of synthetic visual scenes. It introduced a number of design variations (called "profiles" and currently numbering 19 in all) for a wide variety of applications. The H.264/AVC project (with only 3 profiles) returns to the narrower and more traditional focus on efficient compression of generic camera-shot rectangular video pictures with robustness to network losses – making no attempt to cover the ambitious breadth of MPEG-4 Visual. MPEG-4 Visual, while not quite as "hot off the press", establishes a landmark in recent technology development, and its capabilities are yet to be fully explored.

Google Exhibit No. 1028
Page 013

Google Exhibit No. 1063
Page 1014

Most people first learn about a standard in publications other than the standard itself. My personal belief is that if you want to know about a standard, you should also obtain a copy of it, read it, and refer to that document alone as the ultimate authority on its content, its boundaries, and its capabilities. No tutorial or overview presentation will provide all of the insights that can be obtained from careful analysis of the standard itself.

At the same time, no standardized specification document (at least for video coding), can be a complete substitute for a good technical book on the subject. Standards specifications are written primarily to be precise, consistent, complete, and correct and not to be particularly readable. Standards tend to leave out information that is not *absolutely* necessary to comply with them. Many people find it surprising, for example, that video coding standards say almost nothing about how an encoder works or how one should be designed. In fact an encoder is essentially allowed to do anything that produces bits that can be correctly decoded, regardless of what picture quality comes out of that decoding process. People, however, can usually only understand the principles of video coding if they think from the perspective of the encoder, and nearly all textbooks (including this one) approach the subject from the encoding perspective. A good book, such as this one, will tell you why a design is the way it is and how to make use of that design, while a good standard may only tell you exactly what it is and abruptly (deliberately) stop right there.

In the case of H.264/AVC or MPEG-4 Visual, it is highly advisable for those new to the subject to read some introductory overviews such as this one, and even to get a copy of an older and simpler standard such as H.261 or MPEG-1 and try to understand that first. The principles of digital video codec design are not too complicated, and haven't really changed much over the years – but those basic principles have been wrapped in layer-upon-layer of technical enhancements to the point that the simple and straightforward concepts that lie at their core can become obscured. The entire H.261 specification was only 25 pages long, and only 17 of those pages were actually required to fully specify the technology that now lies at the heart of all subsequent video coding standards. In contrast, the H.264/AVC and MPEG-4 Visual and specifications are more than 250 and 500 pages long, respectively, with a high density of technical detail (despite completely leaving out key information such as how to encode video using their formats). They each contain areas that are difficult even for experts to fully comprehend and appreciate.

Dr Richardson's book is not a completely exhaustive treatment of the subject. However, his approach is highly informative and provides a good initial understanding of the key concepts, and his approach is conceptually superior (and in some aspects more objective) to other treatments of video coding publications. This and the remarkable timeliness of the subject matter make this book a strong contribution to the technical literature of our community.

Gary J. Sullivan

Biography of Gary J. Sullivan, PhD

Gary J. Sullivan is the chairman of the Joint Video Team (JVT) for the development of the latest international video coding standard known as H.264/AVC, which was recently completed as a joint project between the ITU-T video coding experts group (VCEG) and the ISO/IEC moving picture experts group (MPEG).

Google Exhibit No. 1028
Page 014

Google Exhibit No. 1063
Page 1015

He is also the Rapporteur of Advanced Video Coding in the ITU-T, where he has led VCEG (ITU-T Q.6/SG16) for about seven years. He is also the ITU-T video liaison representative to MPEG and served as MPEG's (ISO/IEC JTC1/SC29/WG11) video chairman from March of 2001 to May of 2002.

He is currently a program manager of video standards and technologies in the eHome A/V platforms group of Microsoft Corporation. At Microsoft he designed and remains active in the extension of DirectX® Video Acceleration API/DDI feature of the Microsoft Windows® operating system platform.

Google Exhibit No. 1028
Page 015

Google Exhibit No. 1063
Page 1016

Preface

With the widespread adoption of technologies such as digital television, Internet streaming video and DVD-Video, video compression has become an essential component of broadcast and entertainment media. The success of digital TV and DVD-Video is based upon the 10-year-old MPEG-2 standard, a technology that has proved its effectiveness but is now looking distinctly old-fashioned. It is clear that the time is right to replace MPEG-2 video compression with a more effective and efficient technology that can take advantage of recent progress in processing power. For some time there has been a running debate about which technology should take up MPEG-2's mantle. The leading contenders are the International Standards known as MPEG-4 Visual and H.264.

This book aims to provide a clear, practical and unbiased guide to these two standards to enable developers, engineers, researchers and students to understand and apply them effectively. Video and image compression is a complex and extensive subject and this book keeps an unapologetically limited focus, concentrating on the standards themselves (and in the case of MPEG-4 Visual, on the elements of the standard that support coding of 'real world' video material) and on video coding concepts that directly underpin the standards. The book takes an application-based approach and places particular emphasis on tools and features that are helpful in practical applications, in order to provide practical and useful assistance to developers and adopters of these standards.

I am grateful to a number of people who helped to shape the content of this book. I received many helpful comments and requests from readers of my book *Video Codec Design*. Particular thanks are due to Gary Sullivan for taking the time to provide helpful and detailed comments, corrections and advice and for kindly agreeing to write a Foreword; to Harvey Hanna (Impact Labs Inc), Yafan Zhao (The Robert Gordon University) and Aitor Garay for reading and commenting on sections of this book during its development; to members of the Joint Video Team for clarifying many of the details of H.264; to the editorial team at John Wiley & Sons (and especially to the ever-helpful, patient and supportive Kathryn Sharples); to Phyllis for her constant support; and finally to Freya and Hugh for patiently waiting for the long-promised trip to Storybook Glen!

I very much hope that you will find this book enjoyable, readable and above all useful. Further resources and links are available at my website, http://www.vcodex.com/. I always appreciate feedback, comments and suggestions from readers and you will find contact details at this website.

Iain Richardson

Google Exhibit No. 1028
Page 016

Google Exhibit No. 1063
Page 1017

Glossary

4:2:0 (sampling)	Sampling method: chrominance components have half the horizontal and vertical resolution of luminance component
4:2:2 (sampling)	Sampling method: chrominance components have half the horizontal resolution of luminance component
4:4:4 (sampling)	Sampling method: chrominance components have same resolution as luminance component
arithmetic coding	Coding method to reduce redundancy
artefact	Visual distortion in an image
ASO	Arbitrary Slice Order, in which slices may be coded out of raster sequence
BAB	Binary Alpha Block, indicates the boundaries of a region (MPEG-4 Visual)
BAP	Body Animation Parameters
Block	Region of macroblock (8×8 or 4×4) for transform purposes
block matching	Motion estimation carried out on rectangular picture areas
blocking	Square or rectangular distortion areas in an image
B-picture (slice)	Coded picture (slice) predicted using bidirectional motion compensation
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAE	Context-based Arithmetic Encoding
CAVLC	Context Adaptive Variable Length Coding
chrominance	Colour difference component
CIF	Common Intermediate Format, a colour image format
CODEC	*CO*der / *DEC*oder pair
colour space	Method of representing colour images
DCT	Discrete Cosine Transform
Direct prediction	A coding mode in which no motion vector is transmitted
DPCM	Differential Pulse Code Modulation
DSCQS	Double Stimulus Continuous Quality Scale, a scale and method for subjective quality measurement
DWT	Discrete Wavelet Transform

Google Exhibit No. 1028
Page 017

Google Exhibit No. 1063
Page 1018

entropy coding	Coding method to reduce redundancy
error concealment	Post-processing of a decoded image to remove or reduce visible error effects
Exp-Golomb	Exponential Golomb variable length codes
FAP	Facial Animation Parameters
FBA	Face and Body Animation
FGS	Fine Granular Scalability
field	Odd- or even-numbered lines from an interlaced video sequence
flowgraph	Pictorial representation of a transform algorithm (or the algorithm itself)
FMO	Flexible Macroblock Order, in which macroblocks may be coded out of raster sequence
Full Search	A motion estimation algorithm
GMC	Global Motion Compensation, motion compensation applied to a complete coded object (MPEG-4 Visual)
GOP	Group Of Pictures, a set of coded video images
H.261	A video coding standard
H.263	A video coding standard
H.264	A video coding standard
HDTV	High Definition Television
Huffman coding	Coding method to reduce redundancy
HVS	Human Visual System, the system by which humans perceive and interpret visual images
hybrid (CODEC)	CODEC model featuring motion compensation and transform
IEC	International Electrotechnical Commission, a standards body
Inter (coding)	Coding of video frames using temporal prediction or compensation
interlaced (video)	Video data represented as a series of fields
intra (coding)	Coding of video frames without temporal prediction
I-picture (slice)	Picture (or slice) coded without reference to any other frame
ISO	International Standards Organisation, a standards body
ITU	International Telecommunication Union, a standards body
JPEG	Joint Photographic Experts Group, a committee of ISO (also an image coding standard)
JPEG2000	An image coding standard
latency	Delay through a communication system
Level	A set of conformance parameters (applied to a Profile)
loop filter	Spatial filter placed within encoding or decoding feedback loop
Macroblock	Region of frame coded as a unit (usually 16×16 pixels in the original frame)
Macroblock partition	Region of macroblock with its own motion vector (H.264)
Macroblock sub-partition	Region of macroblock with its own motion vector (H.264)
media processor	Processor with features specific to multimedia coding and processing
motion compensation	Prediction of a video frame with modelling of motion
motion estimation	Estimation of relative motion between two or more video frames

Google Exhibit No. 1028
Page 018

Google Exhibit No. 1063
Page 1019

motion vector	Vector indicating a displaced block or region to be used for motion compensation
MPEG	Motion Picture Experts Group, a committee of ISO/IEC
MPEG-1	A multimedia coding standard
MPEG-2	A multimedia coding standard
MPEG-4	A multimedia coding standard
NAL	Network Abstraction Layer
objective quality	Visual quality measured by algorithm(s)
OBMC	Overlapped Block Motion Compensation
Picture (coded)	Coded (compressed) video frame
P-picture (slice)	Coded picture (or slice) using motion-compensated prediction from one reference frame
profile	A set of functional capabilities (of a video CODEC)
progressive (video)	Video data represented as a series of complete frames
PSNR	Peak Signal to Noise Ratio, an objective quality measure
QCIF	Quarter Common Intermediate Format
quantise	Reduce the precision of a scalar or vector quantity
rate control	Control of bit rate of encoded video signal
rate–distortion	Measure of CODEC performance (distortion at a range of coded bit rates)
RBSP	Raw Byte Sequence Payload
RGB	Red/Green/Blue colour space
ringing (artefacts)	'Ripple'-like artefacts around sharp edges in a decoded image
RTP	Real Time Protocol, a transport protocol for real-time data
RVLC	Reversible Variable Length Code
scalable coding	Coding a signal into a number of layers
SI slice	Intra-coded slice used for switching between coded bitstreams (H.264)
slice	A region of a coded picture
SNHC	Synthetic Natural Hybrid Coding
SP slice	Inter-coded slice used for switching between coded bitstreams (H.264)
sprite	Texture region that may be incorporated in a series of decoded frames (MPEG-4 Visual)
statistical redundancy	Redundancy due to the statistical distribution of data
studio quality	Lossless or near-lossless video quality
subjective quality	Visual quality as perceived by human observer(s)
subjective redundancy	Redundancy due to components of the data that are subjectively insignificant
sub-pixel (motion compensation)	Motion-compensated prediction from a reference area that may be formed by interpolating between integer-valued pixel positions
test model	A software model and document that describe a reference implementation of a video coding standard
Texture	Image or residual data
Tree-structured motion compensation	Motion compensation featuring a flexible hierarchy of partition sizes (H.264)

Google Exhibit No. 1028
Page 019

Google Exhibit No. 1063
Page 1020

TSS	Three Step Search, a motion estimation algorithm
VCEG	Video Coding Experts Group, a committee of ITU
VCL	Video Coding Layer
video packet	Coded unit suitable for packetisation
VLC	Variable Length Code
VLD	Variable Length Decoder
VLE	Variable Length Encoder
VLSI	Very Large Scale Integrated circuit
VO	Video Object
VOP	Video Object Plane
VQEG	Video Quality Experts Group
VQEG	Video Quality Experts Group
Weighted prediction	Motion compensation in which the prediction samples from two references are scaled
YCbCr	Luminance, Blue chrominance, Red chrominance colour space
YUV	A colour space (see YCbCr)

Google Exhibit No. 1028
Page 020

Google Exhibit No. 1063
Page 1021

1

Introduction

1.1 THE SCENE

Scene 1: Your avatar (a realistic 3D model with your appearance and voice) walks through a sophisticated virtual world populated by other avatars, product advertisements and video walls. On one virtual video screen is a news broadcast from your favourite channel; you want to see more about the current financial situation and so you interact with the broadcast and pull up the latest stock market figures. On another screen you call up a videoconference link with three friends. The video images of the other participants, neatly segmented from their backgrounds, are presented against yet another virtual backdrop.

Scene 2: Your new 3G vidphone rings; you flip the lid open and answer the call. The face of your friend appears on the screen and you greet each other. Each sees a small, clear image of the other on the phone's screen, without any of the obvious 'blockiness' of older-model video phones. After the call has ended, you call up a live video feed from a football match. The quality of the basic-rate stream isn't too great and you switch seamlessly to the higher-quality (but more expensive) 'premium' stream. For a brief moment the radio signal starts to break up but all you notice is a slight, temporary distortion in the video picture.

These two scenarios illustrate different visions of the next generation of multimedia applications. The first is a vision of MPEG-4 Visual: a rich, interactive on-line world bringing together synthetic, natural, video, image, 2D and 3D 'objects'. The second is a vision of H.264/AVC: highly efficient and reliable video communications, supporting two-way, 'streaming' and broadcast applications and robust to channel transmission problems. The two standards, each with their advantages and disadvantages and each with their supporters and critics, are contenders in the race to provide video compression for next-generation communication applications.

Turn on the television and surf through tens or hundreds of digital channels. Play your favourite movies on the DVD player and breathe a sigh of relief that you can throw out your antiquated VHS tapes. Tune in to a foreign TV news broadcast on the web (still just a postage-stamp video window but the choice and reliability of video streams is growing all the time). Chat to your friends and family by PC videophone. These activities are now commonplace and unremarkable, demonstrating that digital video is well on the way to becoming a ubiquitous

H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia
Iain E. G. Richardson. © 2003 John Wiley & Sons, Ltd. ISBN: 0-470-84837-5

Google Exhibit No. 1028
Page 021

Google Exhibit No. 1063
Page 1022

and essential component of the entertainment, computing, broadcasting and communications industries.

Pervasive, seamless, high-quality digital video has been the goal of companies, researchers and standards bodies over the last two decades. In some areas (for example broadcast television and consumer video storage), digital video has clearly captured the market, whilst in others (videoconferencing, video email, mobile video), market success is perhaps still too early to judge. However, there is no doubt that digital video is a globally important industry which will continue to pervade businesses, networks and homes. The continuous evolution of the digital video industry is being driven by commercial and technical forces. The commercial drive comes from the huge revenue potential of persuading consumers and businesses (a) to replace analogue technology and older digital technology with new, efficient, high-quality digital video products and (b) to adopt new communication and entertainment products that have been made possible by the move to digital video. The technical drive comes from continuing improvements in processing performance, the availability of higher-capacity storage and transmission mechanisms and research and development of video and image processing technology.

Getting digital video from its source (a camera or a stored clip) to its destination (a display) involves a chain of components or processes. Key to this chain are the processes of compression (encoding) and decompression (decoding), in which bandwidth-intensive 'raw' digital video is reduced to a manageable size for transmission or storage, then reconstructed for display. Getting the compression and decompression processes 'right' can give a significant technical and commercial edge to a product, by providing better image quality, greater reliability and/or more flexibility than competing solutions. There is therefore a keen interest in the continuing development and improvement of video compression and decompression methods and systems. The interested parties include entertainment, communication and broadcasting companies, software and hardware developers, researchers and holders of potentially lucrative patents on new compression algorithms.

The early successes in the digital video industry (notably broadcast digital television and DVD-Video) were underpinned by international standard ISO/IEC 13818 [1], popularly known as 'MPEG-2' (after the working group that developed the standard, the Moving Picture Experts Group). Anticipation of a need for better compression tools has led to the development of two further standards for video compression, known as ISO/IEC 14496 Part 2 ('MPEG-4 Visual') [2] and ITU-T Recommendation H.264/ISO/IEC 14496 Part 10 ('H.264') [3]. MPEG-4 Visual and H.264 share the same ancestry and some common features (they both draw on well-proven techniques from earlier standards) but have notably different visions, seeking to improve upon the older standards in different ways. The vision of MPEG-4 Visual is to move away from a restrictive reliance on rectangular video images and to provide an open, flexible framework for visual communications that uses the best features of efficient video compression and object-oriented processing. In contrast, H.264 has a more pragmatic vision, aiming to do what previous standards did (provide a mechanism for the compression of rectangular video images) but to do it in a more efficient, robust and practical way, supporting the types of applications that are becoming widespread in the marketplace (such as broadcast, storage and streaming).

At the present time there is a lively debate about which (if either) of these standards will come to dominate the market. MPEG-4 Visual is the more mature of the two new standards (its first Edition was published in 1999, whereas H.264 became an International

Google Exhibit No. 1028
Page 022

Google Exhibit No. 1063
Page 1023

Standard/Recommendation in 2003). There is no doubt that H.264 can out-perform MPEG-4 Visual in compression efficiency but it does not have the older standard's bewildering flexibility. The licensing situation with regard to MPEG-4 Visual is clear (and not popular with some parts of the industry) but the cost of licensing H.264 remains to be agreed. This book is about these two important new standards and examines the background to the standards, the core concepts and technical details of each standard and the factors that will determine the answer to the question 'MPEG-4 Visual or H.264?'.

1.2 VIDEO COMPRESSION

Network bitrates continue to increase (dramatically in the local area and somewhat less so in the wider area), high bitrate connections to the home are commonplace and the storage capacity of hard disks, flash memories and optical media is greater than ever before. With the price per transmitted or stored bit continually falling, it is perhaps not immediately obvious why video compression is necessary (and why there is such a significant effort to make it better). Video compression has two important benefits. First, it makes it possible to use digital video in transmission and storage environments that would not support uncompressed ('raw') video. For example, current Internet throughput rates are insufficient to handle uncompressed video in real time (even at low frame rates and/or small frame size). A Digital Versatile Disk (DVD) can only store a few seconds of raw video at television-quality resolution and frame rate and so DVD-Video storage would not be practical without video and audio compression. Second, video compression enables more efficient use of transmission and storage resources. If a high bitrate transmission channel is available, then it is a more attractive proposition to send high-resolution compressed video or multiple compressed video channels than to send a single, low-resolution, uncompressed stream. Even with constant advances in storage and transmission capacity, compression is likely to be an essential component of multimedia services for many years to come.

An information-carrying signal may be compressed by removing redundancy from the signal. In a lossless compression system statistical redundancy is removed so that the original signal can be perfectly reconstructed at the receiver. Unfortunately, at the present time lossless methods can only achieve a modest amount of compression of image and video signals. Most practical video compression techniques are based on lossy compression, in which greater compression is achieved with the penalty that the decoded signal is not identical to the original. The goal of a video compression algorithm is to achieve efficient compression whilst minimising the distortion introduced by the compression process.

Video compression algorithms operate by removing redundancy in the temporal, spatial and/or frequency domains. Figure 1.1 shows an example of a single video frame. Within the highlighted regions, there is little variation in the content of the image and hence there is significant spatial redundancy. Figure 1.2 shows the same frame after the background region has been low-pass filtered (smoothed), removing some of the higher-frequency content. The human eye and brain (Human Visual System) are more sensitive to lower frequencies and so the image is still recognisable despite the fact that much of the 'information' has been removed. Figure 1.3 shows the next frame in the video sequence. The sequence was captured from a camera at 25 frames per second and so there is little change between the two frames in the short interval of 1/25 of a second. There is clearly significant temporal redundancy, i.e. most

Google Exhibit No. 1028
Page 023

Google Exhibit No. 1063
Page 1024



Figure 1.1 Video frame (showing examples of homogeneous regions)



Figure 1.2 Video frame (low-pass filtered background)

Google Exhibit No. 1028
Page 024

Google Exhibit No. 1063
Page 1025



Figure 1.3 Video frame 2

of the image remains unchanged between successive frames. By removing different types of redundancy (spatial, frequency and/or temporal) it is possible to compress the data significantly at the expense of a certain amount of information loss (distortion). Further compression can be achieved by encoding the processed data using an entropy coding scheme such as Huffman coding or Arithmetic coding.

Image and video compression has been a very active field of research and development for over 20 years and many different systems and algorithms for compression and decompression have been proposed and developed. In order to encourage interworking, competition and increased choice, it has been necessary to define standard methods of compression encoding and decoding to allow products from different manufacturers to communicate effectively. This has led to the development of a number of key International Standards for image and video compression, including the JPEG, MPEG and H.26× series of standards.

1.3 MPEG-4 AND H.264

MPEG-4 Visual and H.264 (also known as Advanced Video Coding) are standards for the coded representation of visual information. Each standard is a document that primarily defines two things, a coded representation (or syntax) that describes visual data in a compressed form and a method of decoding the syntax to reconstruct visual information. Each standard aims to ensure that compliant encoders and decoders can successfully interwork with each other, whilst allowing manufacturers the freedom to develop competitive and innovative products. The standards specifically do not define an encoder; rather, they define the output that an encoder should

Google Exhibit No. 1028
Page 025

Google Exhibit No. 1063
Page 1026

produce. A decoding method is defined in each standard but manufacturers are free to develop alternative decoders as long as they achieve the same result as the method in the standard.

MPEG-4 Visual (Part 2 of the MPEG-4 group of standards) was developed by the Moving Picture Experts Group (MPEG), a working group of the International Organisation for Standardisation (ISO). This group of several hundred technical experts (drawn from industry and research organisations) meet at 2–3 month intervals to develop the MPEG series of standards. MPEG-4 (a multi-part standard covering audio coding, systems issues and related aspects of audio/visual communication) was first conceived in 1993 and Part 2 was standardised in 1999. The H.264 standardisation effort was initiated by the Video Coding Experts Group (VCEG), a working group of the International Telecommunication Union (ITU-T) that operates in a similar way to MPEG and has been responsible for a series of visual telecommunication standards. The final stages of developing the H.264 standard have been carried out by the Joint Video Team, a collaborative effort of both VCEG and MPEG, making it possible to publish the final standard under the joint auspices of ISO/IEC (as MPEG-4 Part 10) and ITU-T (as Recommendation H.264) in 2003.

MPEG-4 Visual and H.264 have related but significantly different visions. Both are concerned with compression of visual data but MPEG-4 Visual emphasises flexibility whilst H.264's emphasis is on efficiency and reliability. MPEG-4 Visual provides a highly flexible toolkit of coding techniques and resources, making it possible to deal with a wide range of types of visual data including rectangular frames ('traditional' video material), video objects (arbitrary-shaped regions of a visual scene), still images and hybrids of natural (real-world) and synthetic (computer-generated) visual information. MPEG-4 Visual provides its functionality through a set of coding tools, organised into 'profiles', recommended groupings of tools suitable for certain applications. Classes of profiles include 'simple' profiles (coding of rectangular video frames), object-based profiles (coding of arbitrary-shaped visual objects), still texture profiles (coding of still images or 'texture'), scalable profiles (coding at multiple resolutions or quality levels) and studio profiles (coding for high-quality studio applications).

In contrast with the highly flexible approach of MPEG-4 Visual, H.264 concentrates specifically on efficient compression of video frames. Key features of the standard include compression efficiency (providing significantly better compression than any previous standard), transmission efficiency (with a number of built-in features to support reliable, robust transmission over a range of channels and networks) and a focus on popular applications of video compression. Only three profiles are currently supported (in contrast to nearly 20 in MPEG-4 Visual), each targeted at a class of popular video communication applications. The Baseline profile may be particularly useful for "conversational" applications such as videoconferencing, the Extended profile adds extra tools that are likely to be useful for video streaming across networks and the Main profile includes tools that may be suitable for consumer applications such as video broadcast and storage.

1.4 THIS BOOK

The aim of this book is to provide a technically-oriented guide to the MPEG-4 Visual and H.264/AVC standards, with an emphasis on practical issues. Other works cover the details of the other parts of the MPEG-4 standard [4–6] and this book concentrates on the application of MPEG-4 Visual and H.264 to the coding of natural video. Most practical applications of

Google Exhibit No. 1028
Page 026

Google Exhibit No. 1063
Page 1027

MPEG-4 (and emerging applications of H.264) make use of a subset of the tools provided by each standard (a 'profile') and so the treatment of each standard in this book is organised according to profile, starting with the most basic profiles and then introducing the extra tools supported by more advanced profiles.

Chapters 2 and 3 cover essential background material that is required for an understanding of both MPEG-4 Visual and H.264. Chapter 2 introduces the basic concepts of digital video including capture and representation of video in digital form, colour-spaces, formats and quality measurement. Chapter 3 covers the fundamentals of video compression, concentrating on aspects of the compression process that are common to both standards and introducing the transform-based CODEC 'model' that is at the heart of all of the major video coding standards.

Chapter 4 looks at the standards themselves and examines the way that the standards have been shaped and developed, discussing the composition and procedures of the VCEG and MPEG standardisation groups. The chapter summarises the content of the standards and gives practical advice on how to approach and interpret the standards and ensure conformance. Related image and video coding standards are briefly discussed.

Chapters 5 and 6 focus on the technical features of MPEG-4 Visual and H.264. The approach is based on the structure of the Profiles of each standard (important conformance points for CODEC developers). The Simple Profile (and related Profiles) have shown themselves to be by far the most popular features of MPEG-4 Visual to date and so Chapter 5 concentrates first on the compression tools supported by these Profiles, followed by the remaining (less commercially popular) Profiles supporting coding of video objects, still texture, scalable objects and so on. Because this book is primarily about compression of natural (real-world) video information, MPEG-4 Visual's synthetic visual tools are covered only briefly. H.264's Baseline Profile is covered first in Chapter 6, followed by the extra tools included in the Main and Extended Profiles. Chapters 5 and 6 make extensive reference back to Chapter 3 (Video Coding Concepts). H.264 is dealt with in greater technical detail than MPEG-4 Visual because of the limited availability of reference material on the newer standard.

Practical issues related to the design and performance of video CODECs are discussed in Chapter 7. The design requirements of each of the main functional modules required in a practical encoder or decoder are addressed, from motion estimation through to entropy coding. The chapter examines interface requirements and practical approaches to pre- and post-processing of video to improve compression efficiency and/or visual quality. The compression and computational performance of the two standards is compared and rate control (matching the encoder output to practical transmission or storage mechanisms) and issues faced in transporting and storing of compressed video are discussed.

Chapter 8 examines the requirements of some current and emerging applications, lists some currently-available CODECs and implementation platforms and discusses the important implications of commercial factors such as patent licenses. Finally, some predictions are made about the next steps in the standardisation process and emerging research issues that may influence the development of future video coding standards.

1.5 REFERENCES

1. ISO/IEC 13818, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information, 2000.

Google Exhibit No. 1028
Page 027

Google Exhibit No. 1063
Page 1028

2. ISO/IEC 14496-2, Coding of Audio-Visual Objects – Part 2:Visual, 2001.
3. ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.
4. F. Pereira and T. Ebrahimi (eds), *The MPEG-4 Book*, IMSC Press, 2002.
5. A. Walsh and M. Bourges-Sévenier (eds), *MPEG-4 Jump Start*, Prentice-Hall, 2002.
6. ISO/IEC JTC1/SC29/WG11 N4668, MPEG-4 Overview, http://www.m4if.org/resources/Overview.pdf, March 2002.

Google Exhibit No. 1028
Page 028

Google Exhibit No. 1063
Page 1029

2

Video Formats and Quality

2.1 INTRODUCTION

Video coding is the process of compressing and decompressing a digital video signal. This chapter examines the structure and characteristics of digital images and video signals and introduces concepts such as sampling formats and quality metrics that are helpful to an understanding of video coding. Digital video is a representation of a natural (real-world) visual scene, sampled spatially and temporally. A scene is sampled at a point in time to produce a frame (a representation of the complete visual scene at that point in time) or a field (consisting of odd- or even-numbered lines of spatial samples). Sampling is repeated at intervals (e.g. 1/25 or 1/30 second intervals) to produce a moving video signal. Three sets of samples (components) are typically required to represent a scene in colour. Popular formats for representing video in digital form include the ITU-R 601 standard and the set of 'intermediate formats'. The accuracy of a reproduction of a visual scene must be measured to determine the performance of a visual communication system, a notoriously difficult and inexact process. Subjective measurements are time consuming and prone to variations in the response of human viewers. Objective (automatic) measurements are easier to implement but as yet do not accurately match the opinion of a 'real' human.

2.2 NATURAL VIDEO SCENES

A typical 'real world' or 'natural' video scene is composed of multiple objects each with their own characteristic shape, depth, texture and illumination. The colour and brightness of a natural video scene changes with varying degrees of smoothness throughout the scene ('continuous tone'). Characteristics of a typical natural video scene (Figure 2.1) that are relevant for video processing and compression include spatial characteristics (texture variation within scene, number and shape of objects, colour, etc.) and temporal characteristics (object motion, changes in illumination, movement of the camera or viewpoint and so on).

H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia.
Iain E. G. Richardson. © 2003 John Wiley & Sons, Ltd. ISBN: 0-470-84837-5

Google Exhibit No. 1028
Page 029

Google Exhibit No. 1063
Page 1030



Figure 2.1 Still image from natural video scene

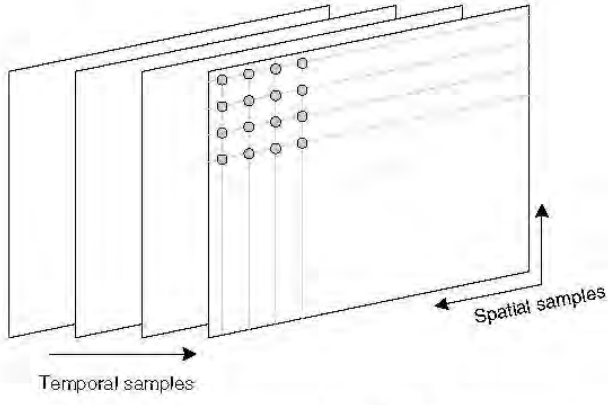


Figure 2.2 Spatial and temporal sampling of a video sequence

2.3 CAPTURE

A natural visual scene is spatially and temporally continuous. Representing a visual scene in digital form involves sampling the real scene spatially (usually on a rectangular grid in the video image plane) and temporally (as a series of still frames or components of frames sampled at regular intervals in time) (Figure 2.2). Digital video is the representation of a sampled video scene in digital form. Each spatio-temporal sample (picture element or pixel) is represented as a number or set of numbers that describes the brightness (luminance) and colour of the sample.

Google Exhibit No. 1028
Page 030

Google Exhibit No. 1063
Page 1031



Figure 2.3 Image with 2 sampling grids

To obtain a 2D sampled image, a camera focuses a 2D projection of the video scene onto a sensor, such as an array of Charge Coupled Devices (CCD array). In the case of colour image capture, each colour component is separately filtered and projected onto a CCD array (see Section 2.4).

2.3.1 Spatial Sampling

The output of a CCD array is an analogue video signal, a varying electrical signal that represents a video image. Sampling the signal at a point in time produces a sampled image or frame that has defined values at a set of sampling points. The most common format for a sampled image is a rectangle with the sampling points positioned on a square or rectangular grid. Figure 2.3 shows a continuous-tone frame with two different sampling grids superimposed upon it. Sampling occurs at each of the intersection points on the grid and the sampled image may be reconstructed by representing each sample as a square picture element (pixel). The visual quality of the image is influenced by the number of sampling points. Choosing a 'coarse' sampling grid (the black grid in Figure 2.3) produces a low-resolution sampled image (Figure 2.4) whilst increasing the number of sampling points slightly (the grey grid in Figure 2.3) increases the resolution of the sampled image (Figure 2.5).

2.3.2 Temporal Sampling

A moving video image is captured by taking a rectangular 'snapshot' of the signal at periodic time intervals. Playing back the series of frames produces the appearance of motion. A higher temporal sampling rate (frame rate) gives apparently smoother motion in the video scene but requires more samples to be captured and stored. Frame rates below 10 frames per second are sometimes used for very low bit-rate video communications (because the amount of data

Google Exhibit No. 1028
Page 031

Google Exhibit No. 1063
Page 1032



Figure 2.4 Image sampled at coarse resolution (black sampling grid)



Figure 2.5 Image sampled at slightly finer resolution (grey sampling grid)

Google Exhibit No. 1028
Page 032

Google Exhibit No. 1063
Page 1033

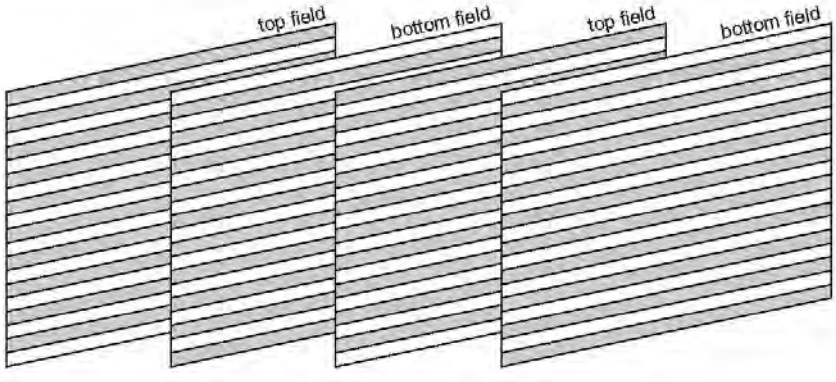


Figure 2.6 Interlaced video sequence

is relatively small) but motion is clearly jerky and unnatural at this rate. Between 10 and 20 frames per second is more typical for low bit-rate video communications; the image is smoother but jerky motion may be visible in fast-moving parts of the sequence. Sampling at 25 or 30 complete frames per second is standard for television pictures (with interlacing to improve the appearance of motion, see below); 50 or 60 frames per second produces smooth apparent motion (at the expense of a very high data rate).

2.3.3 Frames and Fields

A video signal may be sampled as a series of complete frames (*progressive* sampling) or as a sequence of interlaced fields (*interlaced* sampling). In an interlaced video sequence, half of the data in a frame (one field) is sampled at each temporal sampling interval. A field consists of either the odd-numbered or even-numbered lines within a complete video frame and an interlaced video sequence (Figure 2.6) contains a series of fields, each representing half of the information in a complete video frame (e.g. Figure 2.7 and Figure 2.8). The advantage of this sampling method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence with the same data rate, giving the appearance of smoother motion. For example, a PAL video sequence consists of 50 fields per second and, when played back, motion can appears smoother than in an equivalent progressive video sequence containing 25 frames per second.

2.4 COLOUR SPACES

Most digital video applications rely on the display of colour video and so need a mechanism to capture and represent colour information. A monochrome image (e.g. Figure 2.1) requires just one number to indicate the brightness or luminance of each spatial sample. Colour images, on the other hand, require at least three numbers per pixel position to represent colour accurately. The method chosen to represent brightness (luminance or luma) and colour is described as a colour space.

Google Exhibit No. 1028
Page 033

Google Exhibit No. 1063
Page 1034



Figure 2.7 Top field



Figure 2.8 Bottom field

2.4.1 RGB

In the RGB colour space, a colour image sample is represented with three numbers that indicate the relative proportions of Red, Green and Blue (the three additive primary colours of light). Any colour can be created by combining red, green and blue in varying proportions. Figure 2.9 shows the red, green and blue components of a colour image: the red component consists of all the red samples, the green component contains all the green samples and the blue component contains the blue samples. The person on the right is wearing a blue sweater and so this appears 'brighter' in the blue component, whereas the red waistcoat of the figure on the left

Google Exhibit No. 1028
Page 034

Google Exhibit No. 1063
Page 1035



Figure 2.9 Red, Green and Blue components of colour image

appears brighter in the red component. The RGB colour space is well-suited to capture and display of colour images. Capturing an RGB image involves filtering out the red, green and blue components of the scene and capturing each with a separate sensor array. Colour Cathode Ray Tubes (CRTs) and Liquid Crystal Displays (LCDs) display an RGB image by separately illuminating the red, green and blue components of each pixel according to the intensity of each component. From a normal viewing distance, the separate components merge to give the appearance of 'true' colour.

2.4.2 YCbCr

The human visual system (HVS) is less sensitive to colour than to luminance (brightness). In the RGB colour space the three colours are equally important and so are usually all stored at the same resolution but it is possible to represent a colour image more efficiently by separating the luminance from the colour information and representing luma with a higher resolution than colour.

The YCbCr colour space and its variations (sometimes referred to as YUV) is a popular way of efficiently representing colour images. Y is the luminance (luma) component and can be calculated as a weighted average of R, G and B:

$$Y = k_r R + k_g G + k_b B \tag{2.1}$$

where k are weighting factors.

The colour information can be represented as *colour difference* (chrominance or chroma) components, where each chrominance component is the difference between R, G or B and the luminance Y:

$$\begin{aligned} Cb &= B - Y \\ Cr &= R - Y \\ Cg &= G - Y \end{aligned} \tag{2.2}$$

The complete description of a colour image is given by Y (the luminance component) and three colour differences Cb, Cr and Cg that represent the difference between the colour intensity and the mean luminance of each image sample. Figure 2.10 shows the chroma components (red, green and blue) corresponding to the RGB components of Figure 2.9. Here, mid-grey is zero difference, light grey is a positive difference and dark grey is a negative difference. The chroma components only have significant values where there is a large

Google Exhibit No. 1028
Page 035

Google Exhibit No. 1063
Page 1036



Figure 2.10 Cr, Cg and Cb components

difference between the colour component and the luma image (Figure 2.1). Note the strong blue and red difference components.

So far, this representation has little obvious merit since we now have four components instead of the three in RGB. However, $Cb + Cr + Cg$ is a constant and so only two of the three chroma components need to be stored or transmitted since the third component can always be calculated from the other two. In the YCbCr colour space, only the luma (Y) and blue and red chroma (Cb, Cr) are transmitted. YCbCr has an important advantage over RGB, that is the Cr and Cb components may be represented with a *lower resolution* than Y because the HVS is less sensitive to colour than luminance. This reduces the amount of data required to represent the chrominance components without having an obvious effect on visual quality. To the casual observer, there is no obvious difference between an RGB image and a YCbCr image with reduced chrominance resolution. Representing chroma with a lower resolution than luma in this way is a simple but effective form of image compression.

An RGB image may be converted to YCbCr after capture in order to reduce storage and/or transmission requirements. Before displaying the image, it is usually necessary to convert back to RGB. The equations for converting an RGB image to and from YCbCr colour space and vice versa are given in Equation 2.3 and Equation 2.4[1]. Note that there is no need to specify a separate factor k_g (because $k_b + k_r + k_g = 1$) and that G can be extracted from the YCbCr representation by subtracting Cr and Cb from Y, demonstrating that it is not necessary to store or transmit a Cg component.

$$\begin{aligned} Y &= k_r R + (1 - k_b - k_r)G + k_b B \\ Cb &= \frac{0.5}{1 - k_b}(B - Y) \\ Cr &= \frac{0.5}{1 - k_r}(R - Y) \end{aligned} \tag{2.3}$$

$$\begin{aligned} R &= Y + \frac{1 - k_r}{0.5}Cr \\ G &= Y - \frac{2k_b(1 - k_b)}{1 - k_b - k_r}Cb - \frac{2k_r(1 - k_r)}{1 - k_b - k_r}Cr \\ B &= Y + \frac{1 - k_b}{0.5}Cb \end{aligned} \tag{2.4}$$

[1] Thanks to Gary Sullivan for suggesting the form of Equations 2.3 and 2.4

Google Exhibit No. 1028
Page 036

Google Exhibit No. 1063
Page 1037

ITU-R recommendation BT.601 [1] defines $k_b = 0.114$ and $k_r = 0.299$. Substituting into the above equations gives the following widely-used conversion equations:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) \\ Cr &= 0.713(R - Y) \end{aligned} \tag{2.5}$$

$$\begin{aligned} R &= Y + 1.402Cr \\ G &= Y - 0.344Cb - 0.714Cr \\ B &= Y + 1.772Cb \end{aligned} \tag{2.6}$$

2.4.3 YCbCr Sampling Formats

Figure 2.11 shows three sampling patterns for Y, Cb and Cr that are supported by MPEG-4 Visual and H.264. 4:4:4 sampling means that the three components (Y, Cb and Cr) have the same resolution and hence a sample of each component exists at every pixel position. The numbers indicate the relative sampling rate of each component in the *horizontal* direction, i.e. for every four luminance samples there are four Cb and four Cr samples. 4:4:4 sampling preserves the full fidelity of the chrominance components. In 4:2:2 sampling (sometimes referred to as YUY2), the chrominance components have the same vertical resolution as the luma but half the horizontal resolution (the numbers 4:2:2 mean that for every four luminance samples in the horizontal direction there are two Cb and two Cr samples). 4:2:2 video is used for high-quality colour reproduction.

In the popular 4:2:0 sampling format ('YV12'), Cb and Cr each have half the horizontal and vertical resolution of Y. The term '4:2:0' is rather confusing because the numbers do not actually have a logical interpretation and appear to have been chosen historically as a 'code' to identify this particular sampling pattern and to differentiate it from 4:4:4 and 4:2:2. 4:2:0 sampling is widely used for consumer applications such as video conferencing, digital television and digital versatile disk (DVD) storage. Because each colour difference component contains one quarter of the number of samples in the Y component, 4:2:0 YCbCr video requires exactly half as many samples as 4:4:4 (or R:G:B) video.

Example

Image resolution: 720 × 576 pixels
Y resolution: 720 × 576 samples, each represented with eight bits

4:4:4 Cb, Cr resolution: 720 × 576 samples, each eight bits
Total number of bits: 720 × 576 × 8 × 3 = 9 953 280 bits

4:2:0 Cb, Cr resolution: 360 × 288 samples, each eight bits
Total number of bits: (720 × 576 × 8) + (360 × 288 × 8 × 2) = 4 976 640 bits

The 4:2:0 version requires half as many bits as the 4:4:4 version.

Google Exhibit No. 1028
Page 037

Google Exhibit No. 1063
Page 1038

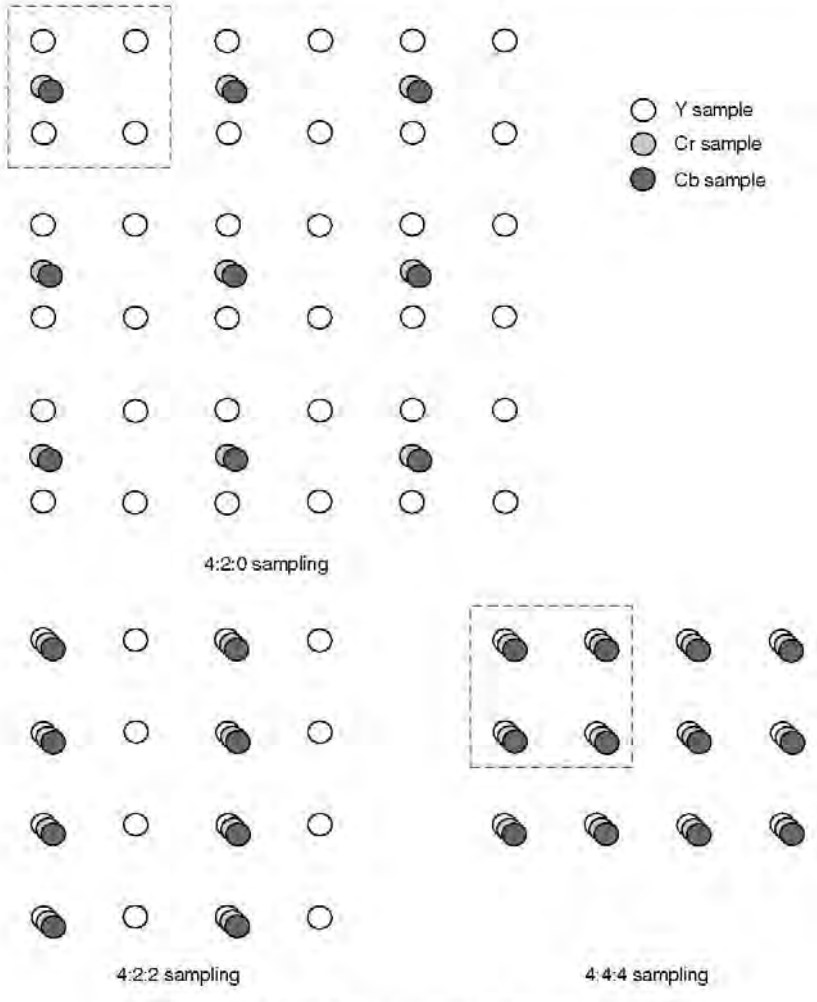


Figure 2.11 4:2:0, 4:2:2 and 4:4:4 sampling patterns (progressive)

4:2:0 sampling is sometimes described as '12 bits per pixel'. The reason for this can be seen by examining a group of four pixels (see the groups enclosed in dotted lines in Figure 2.11). Using 4:4:4 sampling, a total of 12 samples are required, four each of Y, Cb and Cr, requiring a total of $12 \times 8 = 96$ bits, an average of $96/4 = 24$ bits per pixel. Using 4:2:0 sampling, only six samples are required, four Y and one each of Cb, Cr, requiring a total of $6 \times 8 = 48$ bits, an average of $48/4 = 12$ bits per pixel.

In a 4:2:0 interlaced video sequence, the Y, Cb and Cr samples corresponding to a complete video frame are allocated to two fields. Figure 2.12 shows the method of allocating

Google Exhibit No. 1028
Page 038

Google Exhibit No. 1063
Page 1039

Table 2.1 Video frame formats

Format	Luminance resolution (horiz. × vert.)	Bits per frame (4:2:0, eight bits per sample)
Sub-QCIF	128 × 96	147456
Quarter CIF (QCIF)	176 × 144	304128
CIF	352 × 288	1216512
4CIF	704 × 576	4866048

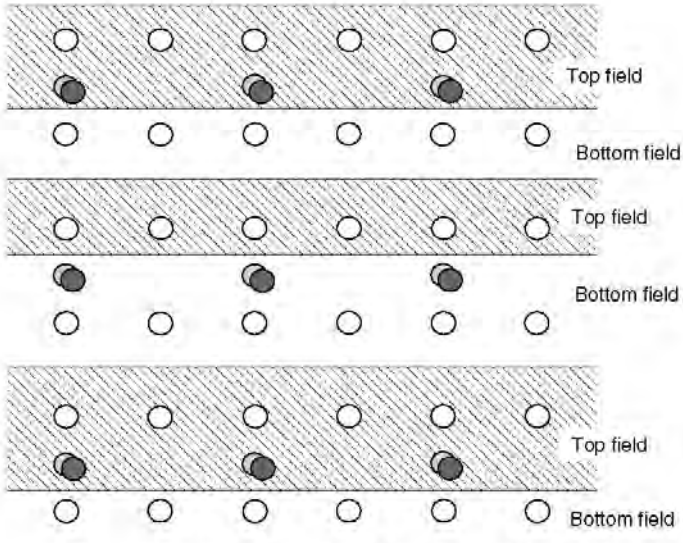


Figure 2.12 Allocaton of 4:2:0 samples to top and bottom fields

Y, Cb and Cr samples to a pair of interlaced fields adopted in MPEG-4 Visual and H.264. It is clear from this figure that the total number of samples in a pair of fields is the same as the number of samples in an equivalent progressive frame.

2.5 VIDEO FORMATS

The video compression standards described in this book can compress a wide variety of video frame formats. In practice, it is common to capture or convert to one of a set of 'intermediate formats' prior to compression and transmission. The Common Intermediate Format (CIF) is the basis for a popular set of formats listed in Table 2.1. Figure 2.13 shows the luma component of a video frame sampled at a range of resolutions, from 4CIF down to Sub-QCIF. The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF

Google Exhibit No. 1028
Page 039

Google Exhibit No. 1063
Page 1040

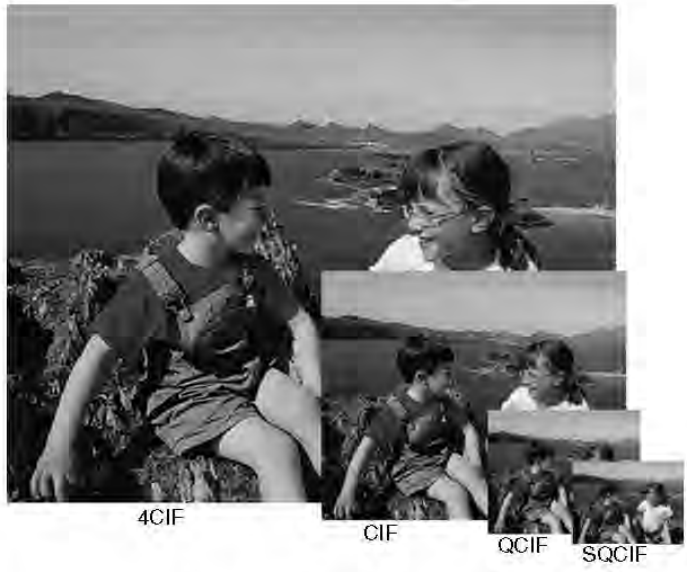


Figure 2.13 Video frame sampled at range of resolutions

are popular for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bitrate are limited. Table 2.1 lists the number of bits required to represent one uncompressed frame in each format (assuming 4:2:0 sampling and 8 bits per luma and chroma sample).

A widely-used format for digitally coding video signals for television production is ITU-R Recommendation BT.601-5 [1] (the term 'coding' in the Recommendation title means conversion to digital format and does not imply compression). The luminance component of the video signal is sampled at 13.5 MHz and the chrominance at 6.75 MHz to produce a 4:2:2 Y:Cb:Cr component signal. The parameters of the sampled digital signal depend on the video frame rate (30 Hz for an NTSC signal and 25 Hz for a PAL/SECAM signal) and are shown in Table 2.2. The higher 30 Hz frame rate of NTSC is compensated for by a lower spatial resolution so that the total bit rate is the same in each case (216 Mbps). The actual area shown on the display, the *active area*, is smaller than the total because it excludes horizontal and vertical blanking intervals that exist 'outside' the edges of the frame.

Each sample has a possible range of 0 to 255. Levels of 0 and 255 are reserved for synchronisation and the active luminance signal is restricted to a range of 16 (black) to 235 (white).

2.6 QUALITY

In order to specify, evaluate and compare video communication systems it is necessary to determine the quality of the video images displayed to the viewer. Measuring visual quality is

Google Exhibit No. 1028
Page 040

Google Exhibit No. 1063
Page 1041

Table 2.2 ITU-R BT.601-5 Parameters

	30 Hz frame rate	25 Hz frame rate
Fields per second	60	50
Lines per complete frame	525	625
Luminance samples per line	858	864
Chrominance samples per line	429	432
Bits per sample	8	8
Total bit rate	216 Mbps	216 Mbps
Active lines per frame	480	576
Active samples per line (Y)	720	720
Active samples per line (Cr,Cb)	360	360

a difficult and often imprecise art because there are so many factors that can affect the results. Visual quality is inherently *subjective* and is influenced by many factors that make it difficult to obtain a completely accurate measure of quality. For example, a viewer's opinion of visual quality can depend very much on the task at hand, such as passively watching a DVD movie, actively participating in a videoconference, communicating using sign language or trying to identify a person in a surveillance video scene. Measuring visual quality using *objective* criteria gives accurate, repeatable results but as yet there are no objective measurement systems that completely reproduce the subjective experience of a human observer watching a video display.

2.6.1 Subjective Quality Measurement

2.6.1.1 Factors Influencing Subjective Quality

Our perception of a visual scene is formed by a complex interaction between the components of the Human Visual System (HVS), the eye and the brain. The perception of visual quality is influenced by spatial fidelity (how clearly parts of the scene can be seen, whether there is any obvious distortion) and temporal fidelity (whether motion appears natural and 'smooth'). However, a viewer's opinion of 'quality' is also affected by other factors such as the viewing environment, the observer's state of mind and the extent to which the observer interacts with the visual scene. A user carrying out a specific task that requires concentration on part of a visual scene will have a quite different requirement for 'good' quality than a user who is passively watching a movie. For example, it has been shown that a viewer's opinion of visual quality is measurably higher if the viewing environment is comfortable and non-distracting (regardless of the 'quality' of the visual image itself).

Other important influences on perceived quality include visual attention (an observer perceives a scene by fixating on a sequence of points in the image rather than by taking in everything simultaneously) and the so-called 'recency effect' (our opinion of a visual sequence is more heavily influenced by recently-viewed material than older video material) [2, 3]. All of these factors make it very difficult to measure visual quality accurately and quantitavely.

Google Exhibit No. 1028
Page 041

Google Exhibit No. 1063
Page 1042

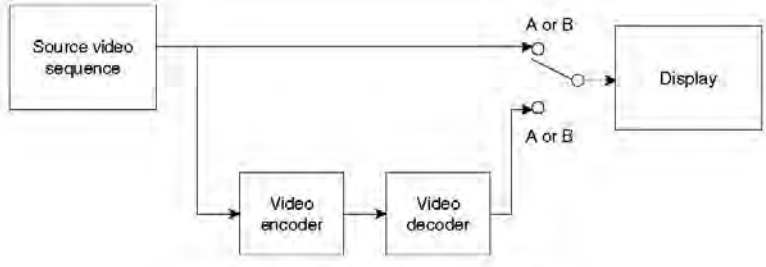


Figure 2.14 DSCQS testing system

2.6.1.2 ITU-R 500

Several test procedures for subjective quality evaluation are defined in ITU-R Recommendation BT.500-11 [4]. A commonly-used procedure from the standard is the Double Stimulus Continuous Quality Scale (DSCQS) method in which an assessor is presented with a pair of images or short video sequences A and B, one after the other, and is asked to give A and B a 'quality score' by marking on a continuous line with five intervals ranging from 'Excellent' to 'Bad'. In a typical test session, the assessor is shown a series of pairs of sequences and is asked to grade each pair. Within each pair of sequences, one is an unimpaired "reference" sequence and the other is the same sequence, modified by a system or process under test. Figure 2.14 shows an experimental set-up appropriate for the testing of a video CODEC in which the original sequence is compared with the same sequence after encoding and decoding. The selection of which sequence is 'A' and which is 'B' is randomised.

The order of the two sequences, original and "impaired", is randomised during the test session so that the assessor does not know which is the original and which is the impaired sequence. This helps prevent the assessor from pre-judging the impaired sequence compared with the reference sequence. At the end of the session, the scores are converted to a normalised range and the end result is a score (sometimes described as a 'mean opinion score') that indicates the *relative* quality of the impaired and reference sequences.

Tests such as DSCQS are accepted to be realistic measures of subjective visual quality. However, this type of test suffers from practical problems. The results can vary significantly depending on the assessor and the video sequence under test. This variation is compensated for by repeating the test with several sequences and several assessors. An 'expert' assessor (one who is familiar with the nature of video compression distortions or 'artefacts') may give a biased score and it is preferable to use 'nonexpert' assessors. This means that a large pool of assessors is required because a nonexpert assessor will quickly learn to recognise characteristic artefacts in the video sequences (and so become 'expert'). These factors make it expensive and time consuming to carry out the DSCQS tests thoroughly.

2.6.2 Objective Quality Measurement

The complexity and cost of subjective quality measurement make it attractive to be able to measure quality automatically using an algorithm. Developers of video compression and video

Google Exhibit No. 1028
Page 042

Google Exhibit No. 1063
Page 1043



Figure 2.15 PSNR examples: (a) original; (b) 30.6 dB; (c) 28.3 dB



Figure 2.16 Image with blurred background (PSNR = 27.7 dB)

processing systems rely heavily on so-called objective (algorithmic) quality measures. The most widely used measure is Peak Signal to Noise Ratio (PSNR) but the limitations of this metric have led to many efforts to develop more sophisticated measures that approximate the response of 'real' human observers.

2.6.2.1 PSNR

Peak Signal to Noise Ratio (PSNR) (Equation 2.7) is measured on a logarithmic scale and depends on the mean squared error (MSE) of between an original and an impaired image or video frame, relative to $(2^n - 1)^2$ (the square of the highest-possible signal value in the image, where n is the number of bits per image sample).

$$PSNR_{dB} = 10\log_{10}\frac{(2^n - 1)^2}{MSE} \tag{2.7}$$

PSNR can be calculated easily and quickly and is therefore a very popular quality measure, widely used to compare the 'quality' of compressed and decompressed video images. Figure 2.15 shows a close-up of 3 images: the first image (a) is the original and (b) and (c) are degraded (blurred) versions of the original image. Image (b) has a measured PSNR of 30.6 dB whilst image (c) has a PSNR of 28.3 dB (reflecting the poorer image quality).

Google Exhibit No. 1028
Page 043

Google Exhibit No. 1063
Page 1044

The PSNR measure suffers from a number of limitations. PSNR requires an unimpaired original image for comparison but this may not be available in every case and it may not be easy to verify that an 'original' image has perfect fidelity. PSNR does not correlate well with subjective video quality measures such as those defined in ITU-R 500. For a given image or image sequence, high PSNR usually indicates high quality and low PSNR usually indicates low quality. However, a particular value of PSNR does not necessarily equate to an 'absolute' subjective quality. For example, Figure 2.16 shows a distorted version of the original image from Figure 2.15 in which only the background of the image has been blurred. This image has a PSNR of 27.7 dB relative to the original. Most viewers would rate this image as significantly better than image (c) in Figure 2.15 because the face is clearer, contradicting the PSNR rating. This example shows that PSNR ratings do not necessarily correlate with 'true' subjective quality. In this case, a human observer gives a higher importance to the face region and so is particularly sensitive to distortion in this area.

2.6.2.2 Other Objective Quality Metrics

Because of the limitations of crude metrics such as PSNR, there has been a lot of work in recent years to try to develop a more sophisticated objective test that more closely approaches subjective test results. Many different approaches have been proposed [5, 6, 7] but none of these has emerged as a clear alternative to subjective tests. As yet there is no standardised, accurate system for objective ('automatic') quality measurement that is suitable for digitally coded video. In recognition of this, the ITU-T Video Quality Experts Group (VQEG) aim to develop standards for objective video quality evaluation [8]. The first step in this process was to test and compare potential models for objective evaluation. In March 2000, VQEG reported on the first round of tests in which ten competing systems were tested under identical conditions. Unfortunately, none of the ten proposals was considered suitable for standardisation and VQEG are completing a second round of evaluations in 2003. Unless there is a significant breakthrough in automatic quality assessment, the problem of accurate objective quality measurement is likely to remain for some time to come.

2.7 CONCLUSIONS

Sampling analogue video produces a digital video signal, which has the advantages of accuracy, quality and compatibility with digital media and transmission but which typically occupies a prohibitively large bitrate. Issues inherent in digital video systems include spatial and temporal resolution, colour representation and the measurement of visual quality. The next chapter introduces the basic concepts of video compression, necessary to accommodate digital video signals on practical storage and transmission media.

2.8 REFERENCES

1. Recommendation ITU-R BT.601-5, Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, ITU-T, 1995.

Google Exhibit No. 1028
Page 044

Google Exhibit No. 1063
Page 1045

2. N. Wade and M. Swanston, *Visual Perception: An Introduction*, 2nd edition, Psychology Press, London, 2001.
3. R. Aldridge, J. Davidoff, D. Hands, M. Ghanbari and D. E. Pearson, Recency effect in the subjective assessment of digitally coded television pictures, *Proc. Fifth International Conference on Image Processing and its Applications*, Heriot-Watt University, Edinburgh, UK, July 1995.
4. Recommendation ITU-T BT.500-11, Methodology for the subjective assessment of the quality of television pictures, ITU-T, 2002.
5. C. J. van den Branden Lambrecht and O. Verscheure, Perceptual quality measure using a spatio-temporal model of the Human Visual System, Digital Video Compression Algorithms and Technologies, *Proc. SPIE*, **2668**, San Jose, 1996.
6. H. Wu, Z. Yu, S. Winkler and T. Chen, Impairment metrics for MC/DPCM/DCT encoded digital video, *Proc. PCS01*, Seoul, April 2001.
7. K. T. Tan and M. Ghanbari, A multi-metric objective picture quality measurement model for MPEG video, *IEEE Trans. Circuits and Systems for Video Technology*, **10** (7), October 2000.
8. http://www.vqeg.org/ (Video Quality Experts Group).

Google Exhibit No. 1028
Page 045

Google Exhibit No. 1063
Page 1046

5

MPEG-4 Visual

5.1 INTRODUCTION

ISO/IEC Standard 14496 Part 2 [1] (MPEG-4 Visual) improves on the popular MPEG-2 standard both in terms of compression efficiency (better compression for the same visual quality) and flexibility (enabling a much wider range of applications). It achieves this in two main ways, by making use of more advanced compression algorithms and by providing an extensive set of 'tools' for coding and manipulating digital media. MPEG-4 Visual consists of a 'core' video encoder/decoder model together with a number of additional coding tools. The core model is based on the well-known hybrid DPCM/DCT coding model (see Chapter 3) and the basic function of the core is extended by tools supporting (among other things) enhanced compression efficiency, reliable transmission, coding of separate shapes or 'objects' in a visual scene, mesh-based compression and animation of face or body models.

It is unlikely that any single application would require all of the tools available in the MPEG-4 Visual framework and so the standard describes a series of *profiles*, recommended sets or groupings of tools for particular types of application. Examples of profiles include Simple (a minimal set of tools for low-complexity applications), Core and Main (with tools for coding multiple arbitrarily-shaped video objects), Advanced Real Time Simple (with tools for error-resilient transmission with low delay) and Advanced Simple (providing improved compression at the expense of increased complexity).

MPEG-4 Visual is embodied in ISO/IEC 14496-2, a highly detailed document running to over 500 pages. Version 1 was released in 1998 and further tools and profiles were added in two Amendments to the standard culminating in Version 2 in late 2001. More tools and profiles are planned for future Amendments or Versions but the 'toolkit' structure of MPEG-4 means that any later versions of 14496-2 should remain backwards compatible with Version 1.

This chapter is a guide to the tools and features of MPEG-4 Visual. Practical implementations of MPEG-4 Visual are based on one or more of the profiles defined in the standard and so this chapter is organised according to profiles. After an overview of the standard and its approach and features, the profiles for coding rectangular video frames are discussed (Simple,

H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia
Iain E. G. Richardson. © 2003 John Wiley & Sons, Ltd. ISBN: 0-470-84837-5

Google Exhibit No. 1028
Page 046

Google Exhibit No. 1063
Page 1047

Advanced Simple and Advanced Real-Time Simple profiles). These are by far the most popular profiles in use at the present time and so they are covered in some detail. Tools and profiles for coding of arbitrary-shaped objects are discussed next (the Core, Main and related profiles), followed by profiles for scalable coding, still texture coding and high-quality ('studio') coding of video.

In addition to tools for coding of 'natural' (real-world) video material, MPEG-4 Visual defines a set of profiles for coding of 'synthetic' (computer-generated) visual objects such as 2D and 3D meshes and animated face and body models. The focus of this book is very much on coding of natural video and so these profiles are introduced only briefly. Coding tools in the MPEG-4 Visual standard that are not included in any Profile (such as Overlapped Block Motion Compensation, OBMC) are (perhaps contentiously!) not covered in this chapter.

5.2 OVERVIEW OF MPEG-4 VISUAL (NATURAL VIDEO CODING)

5.2.1 Features

MPEG-4 Visual attempts to satisfy the requirements of a wide range of visual communication applications through a toolkit-based approach to coding of visual information. Some of the key features that distinguish MPEG-4 Visual from previous visual coding standards include:

- Efficient compression of progressive and interlaced 'natural' video sequences (compression of sequences of rectangular video frames). The core compression tools are based on the ITU-T H.263 standard and can out-perform MPEG-1 and MPEG-2 video compression. Optional additional tools further improve compression efficiency.
- Coding of video objects (irregular-shaped regions of a video scene). This is a new concept for standard-based video coding and enables (for example) independent coding of foreground and background objects in a video scene.
- Support for effective transmission over practical networks. Error resilience tools help a decoder to recover from transmission errors and maintain a successful video connection in an error-prone network environment and scalable coding tools can help to support flexible transmission at a range of coded bitrates.
- Coding of still 'texture' (image data). This means, for example, that still images can be coded and transmitted within the same framework as moving video sequences. Texture coding tools may also be useful in conjunction with animation-based rendering.
- Coding of animated visual objects such as 2D and 3D polygonal meshes, animated faces and animated human bodies.
- Coding for specialist applications such as 'studio' quality video. In this type of application, visual quality is perhaps more important than high compression.

5.2.2 Tools, Objects, Profiles and Levels

MPEG-4 Visual provides its coding functions through a combination of *tools*, *objects* and *profiles*. A *tool* is a subset of coding functions to support a specific feature (for example, basic

Google Exhibit No. 1028
Page 047

Google Exhibit No. 1063
Page 1048

Table 5.1 MPEG-4 Visual profiles for coding natural video

MPEG-4 Visual profile	Main features
Simple	Low-complexity coding of rectangular video frames
Advanced Simple	Coding rectangular frames with improved efficiency and support for interlaced video
Advanced Real-Time Simple	Coding rectangular frames for real-time streaming
Core	Basic coding of arbitrary-shaped video objects
Main	Feature-rich coding of video objects
Advanced Coding Efficiency	Highly efficient coding of video objects
N-Bit	Coding of video objects with sample resolutions other than 8 bits
Simple Scalable	Scalable coding of rectangular video frames
Fine Granular Scalability	Advanced scalable coding of rectangular frames
Core Scalable	Scalable coding of video objects
Scalable Texture	Scalable coding of still texture
Advanced Scalable Texture	Scalable still texture with improved efficiency and object-based features
Advanced Core	Combines features of Simple, Core and Advanced Scalable Texture Profiles
Simple Studio	Object-based coding of high quality video sequences
Core Studio	Object-based coding of high quality video with improved compression efficiency.

Table 5.2 MPEG-4 Visual profiles for coding synthetic or hybrid video

MPEG-4 Visual profile	Main features
Basic Animated Texture	2D mesh coding with still texture
Simple Face Animation	Animated human face models
Simple Face and Body Animation	Animated face and body models
Hybrid	Combines features of Simple, Core, Basic Animated Texture and Simple Face Animation profiles

video coding, interlaced video, coding object shapes, etc.). An *object* is a video element (e.g. a sequence of rectangular frames, a sequence of arbitrary-shaped regions, a still image) that is coded using one or more tools. For example, a simple video object is coded using a limited subset of tools for rectangular video frame sequences, a core video object is coded using tools for arbitrarily-shaped objects and so on. A *profile* is a set of object types that a CODEC is expected to be capable of handling.

The MPEG-4 Visual profiles for coding 'natural' video scenes are listed in Table 5.1 and these range from Simple Profile (coding of rectangular video frames) through profiles for arbitrary-shaped and scalable object coding to profiles for coding of studio-quality video. Table 5.2 lists the profiles for coding 'synthetic' video (animated meshes or face/body models) and the hybrid profile (incorporates features from synthetic and natural video coding). These profiles are not (at present) used for natural video compression and so are not covered in detail in this book.

Google Exhibit No. 1028
Page 048

Google Exhibit No. 1063
Page 1049

Profile \ Object types	Simple	Advanced Simple	Advanced Real-Time Simple	Core	Main	Advanced Coding Efficiency	N-bit	Simple Scalable	Fine Granular Scalability	Core Scalable	Scalable Texture	Advanced Scalable Texture	Simple Studio	Core Studio	Simple Face Animation	Simple Face and Body Animation	Basic Animated Texture	Animated 2D Mesh
Simple	✓																	
Advanced Simple	✓	✓																
Advanced Real-Time Simple	✓		✓															
Core	✓			✓														
Advanced Core	✓			✓								✓						
Main	✓			✓	✓						✓							
Advanced Coding Efficiency	✓			✓		✓												
N-bit	✓			✓			✓											
Simple Scalable	✓							✓										
Fine Granular Scalability	✓	✓							✓									
Core Scalable	✓			✓				✓		✓								
Scalable Texture											✓							
Advanced Scalable Texture												✓						
Simple Studio													✓					
Core Studio													✓	✓				
Basic Animated Texture											✓				✓		✓	
Simple Face Animation															✓			
Simple FBA																✓		
Hybrid	✓			✓							✓				✓		✓	✓

Figure 5.1 MPEG-4 Visual profiles and objects

Figure 5.1 lists each of the MPEG-4 Visual profiles (left-hand column) and visual object types (top row). The table entries indicate which object types are contained within each profile. For example, a CODEC compatible with Simple Profile must be capable of coding and decoding Simple objects and a Core Profile CODEC must be capable of coding and decoding Simple *and* Core objects.

Profiles are an important mechanism for encouraging interoperability between CODECs from different manufacturers. The MPEG-4 Visual standard describes a diverse range of coding tools and it is unlikely that any commercial CODEC would require the implementation of all the tools. Instead, a CODEC designer chooses a profile that contains adequate tools for the target application. For example, a basic CODEC implemented on a low-power processor may use Simple profile, a CODEC for streaming video applications may choose Advanced Real Time Simple and so on. To date, some profiles have had more of an impact on the marketplace than others. The Simple and Advanced Simple profiles are particularly popular with manufacturers and users whereas the profiles for the coding of arbitrary-shaped objects have had very limited commercial impact (see Chapter 8 for further discussion of the commercial impact of MPEG-4 Profiles).

Profiles define a subset of coding tools and *Levels* define constraints on the parameters of the bitstream. Table 5.3 lists the Levels for the popular Simple-based profiles (Simple,

Google Exhibit No. 1028
Page 049

Google Exhibit No. 1063
Page 1050

Table 5.3 Levels for Simple-based profiles

Profile	Level	Typical resolution	Max. bitrate	Max. objects
Simple	L0	176 × 144	64 kbps	1 simple
	L1	176 × 144	64 kbps	4 simple
	L2	352 × 288	128 kbps	4 simple
	L3	352 × 288	384 kbps	4 simple
Advanced Simple (AS)	L0	176 × 144	128 kbps	1 AS or simple
	L1	176 × 144	128 kbps	4 AS or simple
	L2	352 × 288	384 kbps	4 AS or simple
	L3	352 × 288	768 kbps	4 AS or simple
	L4	352 × 576	3 Mbps	4 AS or simple
	L5	720 × 576	8 Mbps	4 AS or simple
Advanced Real-Time Simple (ARTS)	L1	176 × 144	64 kbps	4 ARTS or simple
	L2	352 × 288	128 kbps	4 ARTS or simple
	L3	352 × 288	384 kbps	4 ARTS or simple
	L4	352 × 288	2 Mbps	16 ARTS or simple

Advanced Simple and Advanced Real Time Simple). Each Level places constraints on the maximum performance required to decode an MPEG-4 coded sequence. For example, a multimedia terminal with limited processing capabilities and a small amount of memory may only support Simple Profile @ Level 0 bitstream decoding. The Level definitions place restrictions on the amount of buffer memory, the decoded frame size and processing rate (in macroblocks per second) and the number of video objects (one in this case, a single rectangular frame). A terminal that can cope with these parameters is guaranteed to be capable of successfully decoding any conforming Simple Profile @ Level 0 bitstream. Higher Levels of Simple Profile require a decoder to handle up to four Simple Profile video objects (for example, up to four rectangular objects covering the QCIF or CIF display resolution).

5.2.3 Video Objects

One of the key contributions of MPEG-4 Visual is a move away from the 'traditional' view of a video sequence as being merely a collection of rectangular frames of video. Instead, MPEG-4 Visual treats a video sequence as a collection of one or more *video objects*. MPEG-4 Visual defines a video object as a flexible 'entity that a user is allowed to access (seek, browse) and manipulate (cut and paste)' [1]. A video object (VO) is an area of the video scene that may occupy an arbitrarily-shaped region and may exist for an arbitrary length of time. An instance of a VO at a particular point in time is a *video object plane* (VOP).

This definition encompasses the traditional approach of coding complete frames, in which each VOP is a single frame of video and a sequence of frames forms a VO (for example, Figure 5.2 shows a VO consisting of three rectangular VOPs). However, the introduction of the VO concept allows more flexible options for coding video. Figure 5.3 shows a VO that consists of three irregular-shaped VOPs, each one existing within a frame and each coded separately (*object-based coding*).

Google Exhibit No. 1028
Page 050

Google Exhibit No. 1063
Page 1051

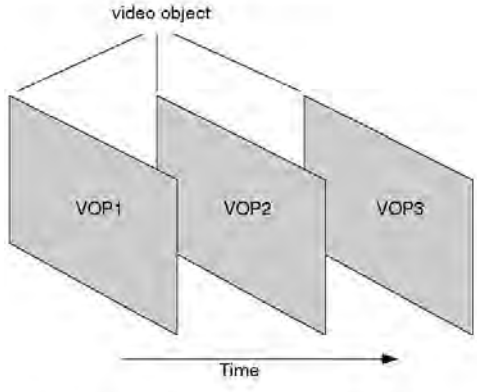


Figure 5.2 VOPs and VO (rectangular)

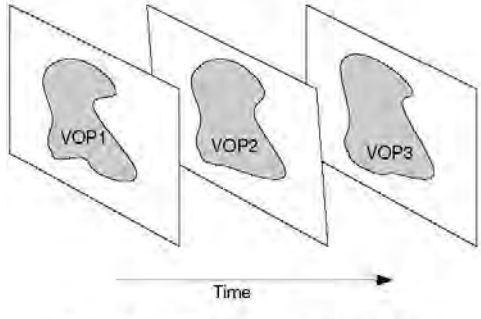


Figure 5.3 VOPs and VO (arbitrary shape)

A video scene (e.g. Figure 5.4) may be made up of a background object (VO3 in this example) and a number of separate foreground objects (VO1, VO2). This approach is potentially much more flexible than the fixed, rectangular frame structure of earlier standards. The separate objects may be coded with different visual qualities and temporal resolutions to reflect their 'importance' to the final scene, objects from multiple sources (including synthetic and 'natural' objects) may be combined in a single scene and the composition and behaviour of the scene may be manipulated by an end-user in highly interactive applications. Figure 5.5 shows a new video scene formed by adding VO1 from Figure 5.4, a new VO2 and a new background VO. Each object is coded separately using MPEG-4 Visual (the compositing of visual and audio objects is assumed to be handled separately, for example by MPEG-4 Systems [2]).

5.3 CODING RECTANGULAR FRAMES

Notwithstanding the potential flexibility offered by object-based coding, the most popular application of MPEG-4 Visual is to encode complete frames of video. The tools required

Google Exhibit No. 1028
Page 051

Google Exhibit No. 1063
Page 1052

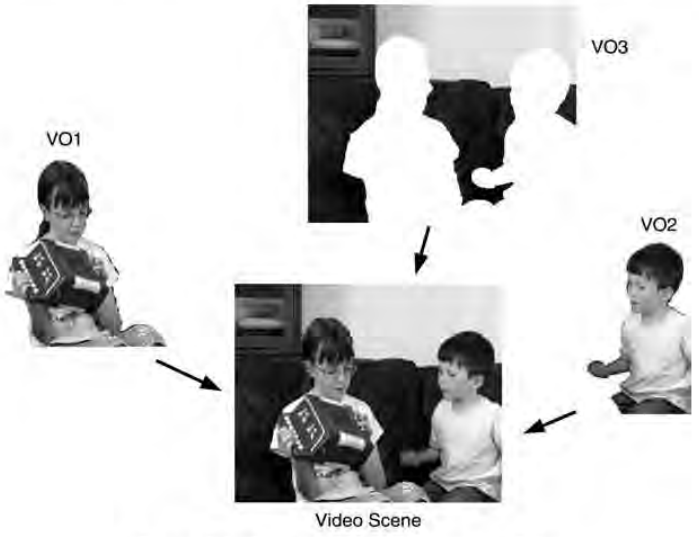


Figure 5.4 Video scene consisting of three VOs



Figure 5.5 Video scene composed of VOs from separate sources

to handle rectangular VOPs (typically complete video frames) are grouped together in the so-called *simple* profiles. The tools and objects for coding rectangular frames are shown in Figure 5.6. The basic tools are similar to those adopted by previous video coding standards, DCT-based coding of macroblocks with motion compensated prediction. The Simple profile is based around the well-known hybrid DPCM/DCT model (see Chapter 3, Section 3.6) with

Google Exhibit No. 1028
Page 052

Google Exhibit No. 1063
Page 1053

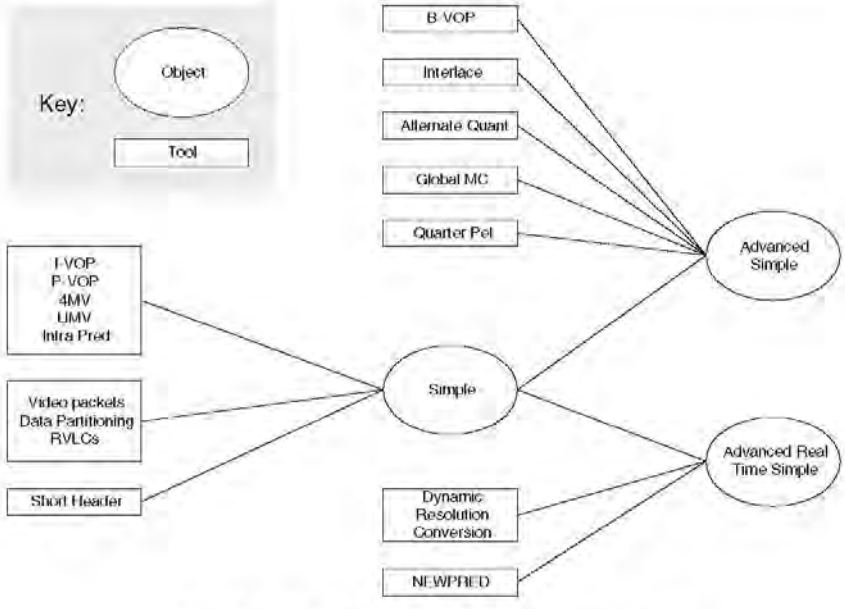


Figure 5.6 Tools and objects for coding rectangular frames

additional tools to improve coding efficiency and transmission efficiency. Because of the widespread popularity of Simple profile, enhanced profiles for rectangular VOPs have been developed. The Advanced Simple profile improves further coding efficiency and adds support for interlaced video and the Advanced Real-Time Simple profile adds tools that are useful for real-time video streaming applications.

5.3.1 Input and output video format

The input to an MPEG-4 Visual encoder and the output of a decoder is a video sequence in 4:2:0, 4:2:2 or 4:4:4 progressive or interlaced format (see Chapter 2). MPEG-4 Visual uses the sampling arrangement shown in Figure 2.11 for progressive sampled frames and the method shown in Figure 2.12 for allocating luma and chroma samples to each pair of fields in an interlaced sequence.

5.3.2 The Simple Profile

A CODEC that is compatible with Simple Profile should be capable of encoding and decoding Simple Video Objects using the following tools:

- I-VOP (Intra-coded rectangular VOP, progressive video format);
- P-VOP (Inter-coded rectangular VOP, progressive video format);

Google Exhibit No. 1028
Page 053

Google Exhibit No. 1063
Page 1054

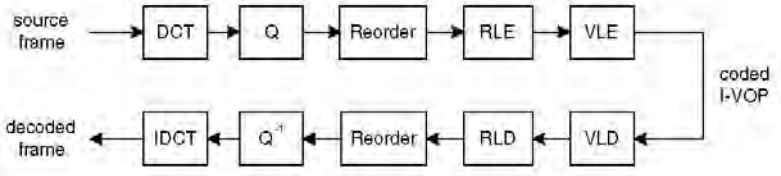


Figure 5.7 I-VOP encoding and decoding stages

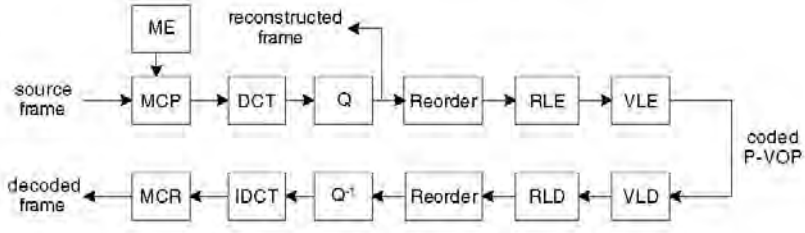


Figure 5.8 P-VOP encoding and decoding stages

- short header (mode for compatibility with H.263 CODECs);
- compression efficiency tools (four motion vectors per macroblock, unrestricted motion vectors, Intra prediction);
- transmission efficiency tools (video packets, Data Partitioning, Reversible Variable Length Codes).

5.3.2.1 The Very Low Bit Rate Video Core

The Simple Profile of MPEG-4 Visual uses a CODEC model known as the Very Low Bit Rate Video (VLBV) Core (the hybrid DPCM/DCT model described in Chapter 3). In common with other standards, the architecture of the encoder and decoder are not specified in MPEG-4 Visual but a practical implementation will require to carry out the functions shown in Figure 5.7 (coding of Intra VOPs) and Figure 5.8 (coding of Inter VOPs). The basic tools required to encode and decode rectangular I-VOPs and P-VOPs are described in the next section (Section 3.6 of Chapter 3 provides a more detailed 'walk-through' of the encoding and decoding process). The tools in the VLBV Core are based on the H.263 standard and the 'short header' mode enables direct compatibility (at the frame level) between an MPEG-4 Simple Profile CODEC and an H.263 Baseline CODEC.

5.3.2.2 Basic coding tools

I-VOP

A rectangular I-VOP is a frame of video encoded in Intra mode (without prediction from any other coded VOP). The encoding and decoding stages are shown in Figure 5.7.

Google Exhibit No. 1028
Page 054

Google Exhibit No. 1063
Page 1055

Table 5.4 Values of *dc_scaler* parameter depending on QP range

Block type	$QP \leq 4$	$5 \leq QP \leq 8$	$9 \leq QP \leq 24$	$25 \leq QP$
Luma	8	$2 \times QP$	$QP + 8$	$(2 \times QP) - 16$
Chroma	8	$(QP + 13)/2$	$(QP + 13)/2$	$QP - 6$

DCT and IDCT: Blocks of luma and chroma samples are transformed using an 8×8 Forward DCT during encoding and an 8×8 Inverse DCT during decoding (see Section 3.4).

Quantisation: The MPEG-4 Visual standard specifies the method of rescaling ('inverse quantising') quantised transform coefficients in a decoder. Rescaling is controlled by a quantiser scale parameter, *QP*, which can take values from 1 to 31 (larger values of *QP* produce a larger quantiser step size and therefore higher compression and distortion). Two methods of rescaling are described in the standard: 'method 2' (basic method) and 'method 1' (more flexible but also more complex). Method 2 inverse quantisation operates as follows. The DC coefficient in an Intra-coded macroblock is rescaled by:

$$DC = DC_Q . dc\_scaler \tag{5.1}$$

DC_Q is the quantised coefficient, DC is the rescaled coefficient and *dc_scaler* is a parameter defined in the standard. In short header mode (see below), *dc_scaler* is 8 (i.e. all Intra DC coefficients are rescaled by a factor of 8), otherwise *dc_scaler* is calculated according to the value of *QP* (Table 5.4). All other transform coefficients (including AC and Inter DC) are rescaled as follows:

$$\begin{aligned} |F| &= QP \cdot (2 \cdot |F_Q| + 1) && \text{(if } QP \text{ is odd and } F_Q \neq 0) \\ |F| &= QP \cdot (2 \cdot |F_Q| + 1) - 1 && \text{(if } QP \text{ is even and } F_Q \neq 0) \\ F &= 0 && \text{(if } F_Q = 0) \end{aligned} \tag{5.2}$$

F_Q is the quantised coefficient and F is the rescaled coefficient. The sign of F is made the same as the sign of F_Q. Forward quantisation is not defined by the standard.

Zig-zag scan: Quantised DCT coefficients are reordered in a zig-zag scan prior to encoding (see Section 3.4).

Last-Run-Level coding: The array of reordered coefficients corresponding to each block is encoded to represent the zero coefficients efficiently. Each nonzero coefficient is encoded as a triplet of (last, run, level), where 'last' indicates whether this is the final nonzero coefficient in the block, 'run' signals the number of preceding zero coefficients and 'level' indicates the coefficient sign and magnitude.

Entropy coding: Header information and (last, run, level) triplets (see Section 3.5) are represented by variable-length codes (VLCs). These codes are similar to Huffman codes and are defined in the standard, based on pre-calculated coefficient probabilities

A coded I-VOP consists of a VOP header, optional video packet headers and coded macroblocks. Each macroblock is coded with a header (defining the macroblock type, identifying which blocks in the macroblock contain coded coefficients, signalling changes in quantisation parameter, etc.) followed by coded coefficients for each 8×8 block.

Google Exhibit No. 1028
Page 055

Google Exhibit No. 1063
Page 1056

In the decoder, the sequence of VLCs are decoded to extract the quantised transform coefficients which are re-scaled and transformed by an 8 × 8 IDCT to reconstruct the decoded I-VOP (Figure 5.7).

P-VOP

A P-VOP is coded with Inter prediction from a previously encoded I- or P-VOP (a reference VOP). The encoding and decoding stages are shown in Figure 5.8.

Motion estimation and compensation: The basic motion compensation scheme is block-based compensation of 16 × 16 pixel macroblocks (see Chapter 3). The offset between the current macroblock and the compensation region in the reference picture (the motion vector) may have half-pixel resolution. Predicted samples at sub-pixel positions are calculated using bilinear interpolation between samples at integer-pixel positions. The method of motion estimation (choosing the 'best' motion vector) is left to the designer's discretion. The matching region (or prediction) is subtracted from the current macroblock to produce a residual macroblock (Motion-Compensated Prediction, MCP in Figure 5.8).

After motion compensation, the residual data is transformed with the DCT, quantised, reordered, run-level coded and entropy coded. The quantised residual is rescaled and inverse transformed in the encoder in order to reconstruct a local copy of the decoded MB (for further motion compensated prediction). A coded P-VOP consists of VOP header, optional video packet headers and coded macroblocks each containing a header (this time including differentially-encoded motion vectors) and coded residual coefficients for every 8 × 8 block.

The decoder forms the same motion-compensated prediction based on the received motion vector and its own local copy of the reference VOP. The decoded residual data is added to the prediction to reconstruct a decoded macroblock (Motion-Compensated Reconstruction, MCR in Figure 5.8).

Macroblocks within a P-VOP may be coded in Inter mode (with motion compensated prediction from the reference VOP) or Intra mode (no motion compensated prediction). Inter mode will normally give the best coding efficiency but Intra mode may be useful in regions where there is not a good match in a previous VOP, such as a newly-uncovered region.

Short Header

The 'short header' tool provides compatibility between MPEG-4 Visual and the ITU-T H.263 video coding standard. An I- or P-VOP encoded in 'short header' mode has identical syntax to an I-picture or P-picture coded in the baseline mode of H.263. This means that an MPEG-4 I-VOP or P-VOP should be decodeable by an H.263 decoder and vice versa.

In short header mode, the macroblocks within a VOP are organised in Groups of Blocks (GOBs), each consisting of one or more complete rows of macroblocks. Each GOB may (optionally) start with a resynchronisation marker (a fixed-length binary code that enables a decoder to resynchronise when an error is encountered, see Section 5.3.2.4).

5.3.2.3 Coding Efficiency Tools

The following tools, part of the Simple profile, can improve compression efficiency. They are only used when short header mode is not enabled.

Google Exhibit No. 1028
Page 056

Google Exhibit No. 1063
Page 1057

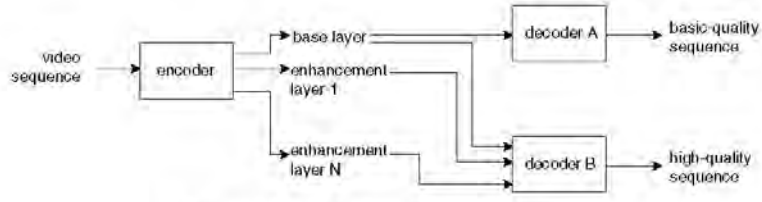


Figure 5.57 Scalable coding: general concept

5.5 SCALABLE VIDEO CODING

Scalable encoding of video data enables a decoder to decode selectively only part of the coded bitstream. The coded stream is arranged in a number of *layers*, including a 'base' layer and one or more 'enhancement' layers (Figure 5.57). In this figure, decoder A receives only the base layer and can decode a 'basic' quality version of the video scene, whereas decoder B receives all layers and decodes a high quality version of the scene. This has a number of applications, for example, a low-complexity decoder may only be capable of decoding the base layer; a low-rate bitstream may be extracted for transmission over a network segment with limited capacity; and an error-sensitive base layer may be transmitted with higher priority than enhancement layers.

MPEG-4 Visual supports a number of scalable coding modes. *Spatial scalability* enables a (rectangular) VOP to be coded at a hierarchy of spatial resolutions. Decoding the base layer produces a low-resolution version of the VOP and decoding successive enhancement layers produces a progressively higher-resolution image. *Temporal scalability* provides a low frame-rate base layer and enhancement layer(s) that build up to a higher frame rate. The standard also supports *quality scalability*, in which the enhancement layers improve the visual quality of the VOP and *complexity scalability*, in which the successive layers are progressively more complex to decode. *Fine Grain Scalability* (FGS) enables the quality of the sequence to be increased in small steps. An application for FGS is streaming video across a network connection, in which it may be useful to scale the coded video stream to match the available bit rate as closely as possible.

5.5.1 Spatial Scalability

The base layer contains a reduced-resolution version of each coded frame. Decoding the base layer alone produces a low-resolution output sequence and decoding the base layer with enhancement layer(s) produces a higher-resolution output. The following steps are required to *encode* a video sequence into two spatial layers:

1. Subsample each input video frame (Figure 5.58) (or video object) horizontally and vertically (Figure 5.59).
2. Encode the reduced-resolution frame to form the base layer.
3. Decode the base layer and up-sample to the original resolution to form a prediction frame (Figure 5.60).
4. Subtract the full-resolution frame from this prediction frame (Figure 5.61).
5. Encode the difference (residual) to form the enhancement layer.

Google Exhibit No. 1028
Page 057

Google Exhibit No. 1063
Page 1058



Figure 5.58 Original video frame



Figure 5.59 Sub-sampled frame to be encoded as base layer



Figure 5.60 Base layer frame (decoded and upsampled)

Google Exhibit No. 1028
Page 058

Google Exhibit No. 1063
Page 1059



Figure 5.61 Residual to be encoded as enhancement layer

A single-layer decoder decodes only the base layer to produce a reduced-resolution output sequence. A two-layer decoder can reconstruct a full-resolution sequence as follows:

1. Decode the base layer and up-sample to the original resolution.
2. Decode the enhancement layer.
3. Add the decoded residual from the enhancement layer to the decoded base layer to form the output frame.

An **I-VOP** in an enhancement layer is encoded without any spatial prediction, i.e. as a complete frame or object at the enhancement resolution. In an enhancement layer *P-VOP*, the decoded, up-sampled base layer VOP (at the same position in time) is used as a prediction *without* any motion compensation. The difference between this prediction and the input frame is encoded using the texture coding tools, i.e. no motion vectors are transmitted for an enhancement P-VOP. An enhancement layer **B-VOP** is predicted from two directions. The backward prediction is formed by the decoded, up-sampled base layer VOP (at the same position in time), without any motion compensation (and hence without any MVs). The forward prediction is formed by the previous VOP in the enhancement layer (even if this is itself a B-VOP), with motion-compensated prediction (and hence MVs).

If the VOP has arbitrary (binary) shape, a base layer and enhancement layer BAB is required for each MB. The base layer BAB is encoded as usual, based on the shape and size of the base layer object. A BAB in a P-VOP enhancement layer is coded using prediction from an up-sampled version of the base layer BAB. A BAB in a B-VOP enhancement layer may be coded in the same way, or using forward prediction from the previous enhancement VOP (as described in Section 5.4.1.1).

5.5.2 Temporal Scalability

The base layer of a temporal scalable sequence is encoded at a low video frame rate and a temporal enhancement layer consists of I-, P- and/or B-VOPs that can be decoded together with the base layer to provide an increased video frame rate. Enhancement layer VOPs are predicted using motion-compensated prediction according to the following rules.

Google Exhibit No. 1028
Page 059

Google Exhibit No. 1063
Page 1060

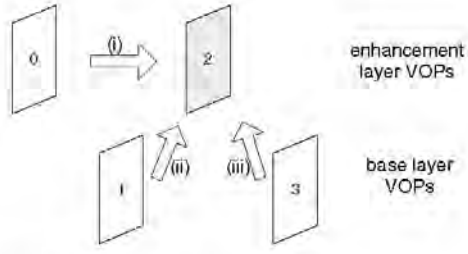


Figure 5.62 Temporal enhancement P-VOP prediction options

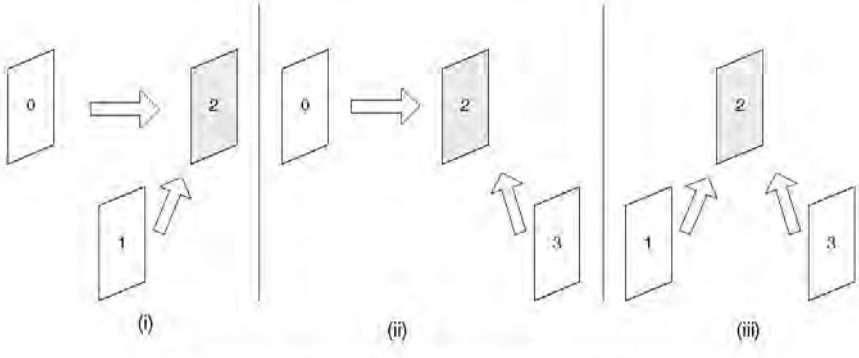


Figure 5.63 Temporal enhancement B-VOP prediction options

An enhancement *I-VOP* is encoded without any prediction. An enhancement *P-VOP* is predicted from (i) the previous enhancement VOP, (ii) the previous base layer VOP or (iii) the next base layer VOP (Figure 5.62). An enhancement *B-VOP* is predicted from (i) the previous enhancement and previous base layer VOPs, (ii) the previous enhancement and next base layer VOPs or (iii) the previous and next base layer VOPs (Figure 5.63).

5.5.3 Fine Granular Scalability

Fine Granular Scalability (FGS) [5] is a method of encoding a sequence as a base layer and enhancement layer. The enhancement layer can be truncated during or after encoding (reducing the bitrate and the decoded quality) to give highly flexible control over the transmitted bitrate. FGS may be useful for video streaming applications, in which the available transmission bandwidth may not be known in advance. In a typical scenario, a sequence is coded as a base layer and a high-quality enhancement layer. Upon receiving a request to send the sequence at a particular bitrate, the streaming server transmits the base layer and a truncated version of the enhancement layer. The amount of truncation is chosen to match the available transmission bitrate, hence maximising the quality of the decoded sequence without the need to re-encode the video clip.

Google Exhibit No. 1028
Page 060

Google Exhibit No. 1063
Page 1061

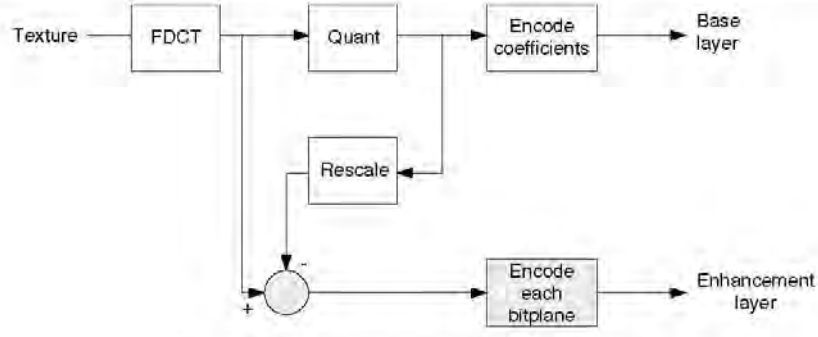


Figure 5.64 FGS encoder block diagram (simplified)

13	-11	0	0	...
0	17	0	...	
0	-3	...		
0	...			
...				

Figure 5.65 Block of residual coefficients (top-left corner)

Encoding

Figure 5.64 shows a simplified block diagram of an FGS encoder (motion compensation is not shown). In the Base Layer, the texture (after motion compensation) is transformed with the forward DCT, quantised and encoded. The quantised coefficients are re-scaled ('inverse quantised') and these re-scaled coefficients are subtracted from the unquantised DCT coefficients to give a set of difference coefficients. The difference coefficients for each block are encoded as a series of *bitplanes*. First, the residual coefficients are reordered using a zigzag scan. The highest-order bits of each coefficient (zeros or ones) are encoded first (the MS bitplane) followed by the next highest-order bits and so on until the LS bits have been encoded.

Example

A block of residual coefficients is shown in Figure 5.65 (coefficients not shown are zero). The coefficients are reordered in a zigzag scan to produce the following list:

$$+13, -11, 0, 0, +17, 0, 0, 0, -3, 0, 0 \ldots.$$

The bitplanes corresponding to the magnitude of each residual coefficient are shown in Table 5.6. In this case, the highest plane containing nonzero bits is plane 4 (because the highest magnitude is 17).

Google Exhibit No. 1028
Page 061

Google Exhibit No. 1063
Page 1062

Table 5.6 Residual coefficient bitplanes (magnitude)

Value	+13	−11	0	0	+17	0	0	0	−3	0...
Plane 4 (MSB)	0	0	0	0	1	0	0	0	0	0...
Plane 3	1	1	0	0	0	0	0	0	0	0...
Plane 2	1	1	0	0	0	0	0	0	0	0...
Plane 1	0	0	0	0	0	0	0	0	1	0...
Plane 0 (LSB)	1	1	0	0	1	0	0	0	1	0...

Table 5.7 Encoded values

Plane	Encoded values
4	(4, EOP) (+)
3	(0) (+) (0, EOP) (−)
2	(0, EOP)
1	(1) (6, EOP) (−)
0	(0) (0) (2) (3, EOP)

Each bitplane contains a series of zeros and ones. The ones are encoded as (run, EOP) where 'EOP' indicates 'end of bitplane' and each (run, EOP) pair is transmitted as a variable-length code. Whenever the MS bit of a coefficient is encoded, it is immediately followed in the bitstream by a sign bit. Table 5.7 lists the encoded values for each bitplane. Bitplane 4 contains four zeros, followed by a 1. This is the last nonzero bit and so is encoded as (4, EOP). This also the MS bit of the coefficient '+17' and so the sign of this coefficient is encoded.

This example illustrates the processing of one block. The encoding procedure for a complete frame is as follows:

1. Find the highest bit position of any difference coefficient in the frame (the MSB).
2. Encode each bitplane as described above, starting with the plane containing the MSB.

Each complete encoded bitplane is preceded by a start code, making it straightforward to truncate the bitstream by sending only a limited number of encoded bitplanes.

Decoding

The decoder decodes the base layer and enhancement layer (which may be truncated). The difference coefficients are reconstructed from the decoded bitplanes, added to the base layer coefficients and inverse transformed to produce the decoded enhancement sequence (Figure 5.66).

If the enhancement layer has been truncated, then the accuracy of the difference coefficients is reduced. For example, assume that the enhancement layer described in the above example is truncated after bitplane 3. The MS bits (and the sign) of the first three nonzero coefficients are decoded (Table 5.8); if the remaining (undecoded) bitplanes are filled with

Google Exhibit No. 1028
Page 062

Google Exhibit No. 1063
Page 1063

Table 5.8 Decoded values (truncated after plane 3)

Plane 4 (MSB)	0	0	0	0	1	0	0	0	0	0...
Plane 3	1	1	0	0	0	0	0	0	0	0...
Plane 2	0	0	0	0	0	0	0	0	0	0...
Plane 1	0	0	0	0	0	0	0	0	0	0...
Plane 0 (LSB)	0	0	0	0	0	0	0	0	0	0...
Decoded value	+8	−8	0	0	+16	0	0	0	0	0...

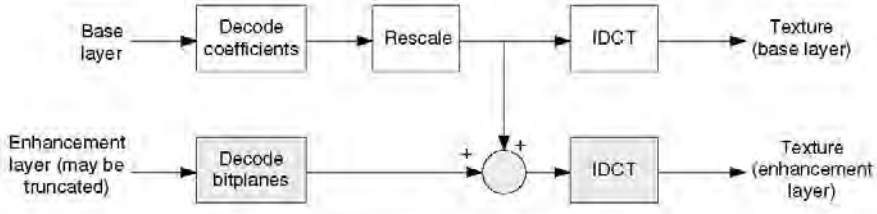


Figure 5.66 FGS decoder block diagram (simplified)

zeros then the list of output values becomes:

$$+8, -8, 0, 0, +16, 0 \ldots.$$

Optional enhancements to FGS coding include selective enhancement (in which bit planes of selected MBs are bit-shifted up prior to encoding, in order to give them a higher priority and a higher probability of being included in a truncated bitstream) and frequency weighting (in which visually-significant low frequency DCT coefficients are shifted up prior to encoding, again in order to give them higher priority in a truncated bitstream).

5.5.4 The Simple Scalable Profile

The Simple Scalable profile supports Simple and Simple Scalable objects. The Simple Scalable object contains the following tools:

- I-VOP, P-VOP, 4MV, unrestricted MV and Intra Prediction;
- Video packets, Data Partitioning and Reversible VLCs;
- B-VOP;
- Rectangular Temporal Scalability (1 enhancement layer) (Section 5.5.2);
- Rectangular Spatial Scalability (1 enhancement layer) (Section 5.5.1).

The last two tools support scalable coding of rectangular VOs.

5.5.5 The Core Scalable Profile

The Core Scalable profile includes Simple, Simple Scalable and Core objects, plus the Core Scalable object which features the following tools, in each case with up to two enhancement layers per object:

Google Exhibit No. 1028
Page 063

Google Exhibit No. 1063
Page 1064

- Rectangular Temporal Scalability (Section 5.5.2);
- Rectangular Spatial Scalability (Section 5.5.1);
- Object-based Spatial Scalability (Section 5.5.1).

5.5.6 The Fine Granular Scalability Profile

The FGS profile includes Simple and Advanced Simple objects plus the FGS object which includes these tools:

- B-VOP, Interlace and Alternate Quantiser tools;
- FGS Spatial Scalability;
- FGS Temporal Scalability.

FGS 'Spatial Scalability' uses the encoding and decoding techniques described in Section 5.5.3 to encode each frame as a base layer and an FGS enhancement layer. FGS 'Temporal Scalability' combines FGS (Section 5.5.3) with temporal scalability (Section 5.5.2). An enhancement-layer frame is encoded using forward or bidirectional prediction from *base layer* frame(s) only. The DCT coefficients of the enhancement-layer frame are encoded in bitplanes using the FGS technique.

5.6 TEXTURE CODING

The applications targeted by the developers of MPEG4 include scenarios where it is necessary to transmit still texture (i.e. still images). Whilst block transforms such as the DCT are widely considered to be the best practical solution for motion-compensated video coding, the Discrete Wavelet Transform (DWT) is particularly effective for coding still images (see Chapter 3) and MPEG-4 Visual uses the DWT as the basis for tools to compress still texture. Applications include the coding of rectangular texture objects (such as complete image frames), coding of arbitrary-shaped texture regions and coding of texture to be mapped onto animated 2D or 3D meshes (see Section 5.8).

The basic structure of a still texture encoder is shown in Figure 5.68. A 2D DWT is applied to the texture object, producing a DC component (low-frequency subband) and a number of AC (high-frequency) subbands (see Chapter 3). The DC subband is quantised, predictively encoded (using a form of DPCM) and entropy encoded using an arithmetic encoder. The AC subbands are quantised and reordered ('scanned'), zero-tree encoded and entropy encoded.

Discrete Wavelet Transform

The DWT adopted for MPEG-4 Still Texture coding is the Daubechies (9,3)-tap biorthogonal filter [6]. This is essentially a matched pair of filters, one low pass (with three filter coefficients or 'taps') and one high pass (with nine filter taps).

Quantisation

The DC subband is quantised using a scalar quantiser (see Chapter 3). The AC subbands may be quantised in one of three ways:

Google Exhibit No. 1028
Page 064

Google Exhibit No. 1063
Page 1065

6

H.264/MPEG4 Part 10

6.1 INTRODUCTION

The Moving Picture Experts Group and the Video Coding Experts Group (MPEG and VCEG) have developed a new standard that promises to outperform the earlier MPEG-4 and H.263 standards, providing better compression of video images. The new standard is entitled 'Advanced Video Coding' (AVC) and is published jointly as Part 10 of MPEG-4 and ITU-T Recommendation H.264 [1, 3].

6.1.1 Terminology

Some of the important terminology adopted in the H.264 standard is as follows (the details of these concepts are explained in later sections):

A field (of interlaced video) or a frame (of progressive or interlaced video) is encoded to produce a *coded picture*. A coded frame has a *frame number* (signalled in the bitstream), which is not necessarily related to decoding order, and each coded field of a progressive or interlaced frame has an associated *picture order count*, which defines the decoding order of fields. Previously coded pictures (*reference pictures*) may be used for inter prediction of further coded pictures. Reference pictures are organised into one or two lists (sets of numbers corresponding to reference pictures), described as *list 0* and *list 1*.

A coded picture consists of a number of *macroblocks*, each containing 16 × 16 luma samples and associated chroma samples (8 × 8 Cb and 8 × 8 Cr samples in the current standard). Within each picture, macroblocks are arranged in *slices*, where a slice is a set of macroblocks in raster scan order (but not necessarily contiguous – see section 6.4.3). An *I slice* may contain only I macroblock types (see below), a *P slice* may contain P and I macroblock types and a *B slice* may contain B and I macroblock types. (There are two further slice types, SI and SP, discussed in section 6.6.1).

I macroblocks are predicted using intra prediction from decoded samples in the current slice. A prediction is formed either (a) for the complete macroblock or (b) for each 4 × 4 block

H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia
Iain E. G. Richardson. © 2003 John Wiley & Sons, Ltd. ISBN: 0-470-84837-5

Google Exhibit No. 1028
Page 065

Google Exhibit No. 1063
Page 1066

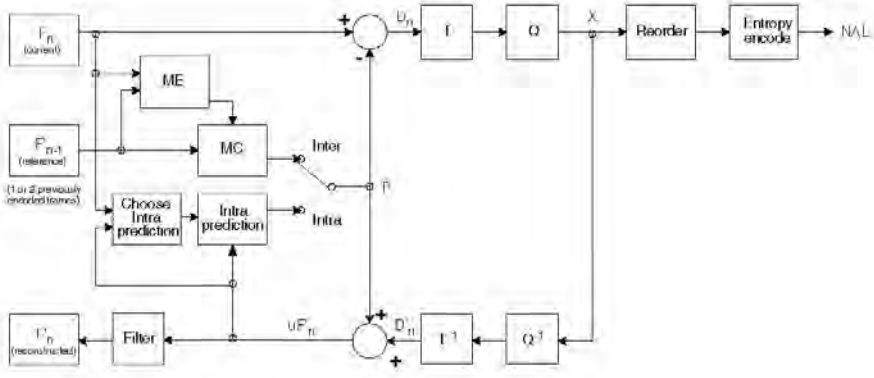


Figure 6.1 H.264 Encoder

of luma samples (and associated chroma samples) in the macroblock. (An alternative to intra prediction, I_PCM mode, is described in section 6.4.6).

P macroblocks are predicted using inter prediction from reference picture(s). An inter coded macroblock may be divided into *macroblock partitions*, i.e. blocks of size 16 × 16, 16 × 8, 8 × 16 or 8 × 8 luma samples (and associated chroma samples). If the 8 × 8 partition size is chosen, each 8 × 8 *sub-macroblock* may be further divided into *sub-macroblock partitions* of size 8 × 8, 8 × 4, 4 × 8 or 4 × 4 luma samples (and associated chroma samples). Each macroblock partition may be predicted from one picture in list 0. If present, every sub-macroblock partition in a sub-macroblock is predicted from the same picture in list 0.

B macroblocks are predicted using inter prediction from reference picture(s). Each macroblock partition may be predicted from one or two reference pictures, one picture in list 0 and/or one picture in list 1. If present, every sub-macroblock partition in a sub-macroblock is predicted from (the same) one or two reference pictures, one picture in list 0 and/or one picture in list 1.

6.2 THE H.264 CODEC

In common with earlier coding standards, H.264 does not explicitly define a CODEC (enCOder / DECoder pair) but rather defines the syntax of an encoded video bitstream together with the method of decoding this bitstream. In practice, a compliant encoder and decoder are likely to include the functional elements shown in Figure 6.1 and Figure 6.2. With the exception of the deblocking filter, most of the basic functional elements (prediction, transform, quantisation, entropy encoding) are present in previous standards (MPEG-1, MPEG-2, MPEG-4, H.261, H.263) but the important changes in H.264 occur in the details of each functional block.

The Encoder (Figure 6.1) includes two dataflow paths, a 'forward' path (left to right) and a 'reconstruction' path (right to left). The dataflow path in the Decoder (Figure 6.2) is shown from right to left to illustrate the similarities between Encoder and Decoder. Before examining the detail of H.264, we will describe the main steps in encoding and decoding a frame (or field)

Google Exhibit No. 1028
Page 066

Google Exhibit No. 1063
Page 1067

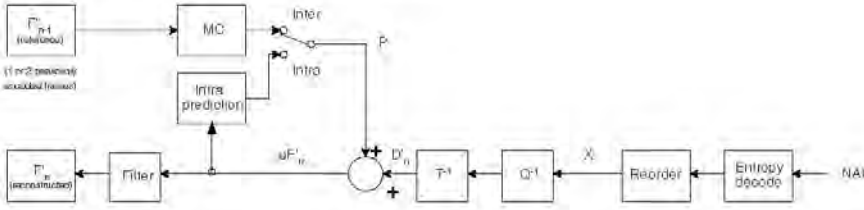


Figure 6.2 H.264 Decoder

of video. The following description is simplified in order to provide an overview of encoding and decoding. The term "block" is used to denote a macroblock partition or sub-macroblock partition (inter coding) or a 16 × 16 or 4 × 4 block of luma samples and associated chroma samples (intra coding).

Encoder (forward Path)

An input frame or field F_n is processed in units of a macroblock. Each macroblock is encoded in intra or inter mode and, for each block in the macroblock, a prediction PRED (marked 'P' in Figure 6.1) is formed based on reconstructed picture samples. In Intra mode, PRED is formed from samples in the current slice that have previously encoded, decoded and reconstructed (uF'_n in the figures; note that *unfiltered* samples are used to form PRED). In Inter mode, PRED is formed by motion-compensated prediction from one or two reference picture(s) selected from the set of list 0 and/or list 1 reference pictures. In the figures, the reference picture is shown as the previous encoded picture F'_{n-1} but the prediction reference for each macroblock partition (in inter mode) may be chosen from a selection of past or future pictures (in display order) that have already been encoded, reconstructed and filtered.

The prediction PRED is subtracted from the current block to produce a residual (difference) block D_n that is transformed (using a block transform) and quantised to give X, a set of quantised transform coefficients which are reordered and entropy encoded. The entropy-encoded coefficients, together with side information required to decode each block within the macroblock (prediction modes, quantiser parameter, motion vector information, etc.) form the compressed bitstream which is passed to a Network Abstraction Layer (NAL) for transmission or storage.

Encoder (Reconstruction Path)

As well as encoding and transmitting each block in a macroblock, the encoder decodes (reconstructs) it to provide a reference for further predictions. The coefficients X are scaled (Q^{-1}) and inverse transformed (T^{-1}) to produce a difference block D'_n. The prediction block PRED is added to D'_n to create a reconstructed block uF'_n (a decoded version of the original block; u indicates that it is unfiltered). A filter is applied to reduce the effects of blocking distortion and the reconstructed reference picture is created from a series of blocks F'_n.

Decoder

The decoder receives a compressed bitstream from the NAL and entropy decodes the data elements to produce a set of quantised coefficients X. These are scaled and inverse transformed

Google Exhibit No. 1028
Page 067

Google Exhibit No. 1063
Page 1068

to give D'_n (identical to the D'_n shown in the Encoder). Using the header information decoded from the bitstream, the decoder creates a prediction block PRED, identical to the original prediction PRED formed in the encoder. PRED is added to D'_n to produce uF'_n which is filtered to create each decoded block F'_n.

6.3 H.264 STRUCTURE

6.3.1 Profiles and Levels

H.264 defines a set of three *Profiles*, each supporting a particular set of coding functions and each specifying what is required of an encoder or decoder that complies with the Profile. The *Baseline Profile* supports intra and inter-coding (using I-slices and P-slices) and entropy coding with context-adaptive variable-length codes (CAVLC). The *Main Profile* includes support for interlaced video, inter-coding using B-slices, inter coding using weighted prediction and entropy coding using context-based arithmetic coding (CABAC). The *Extended Profile* does not support interlaced video or CABAC but adds modes to enable efficient switching between coded bitstreams (SP- and SI-slices) and improved error resilience (Data Partitioning). Potential applications of the Baseline Profile include videotelephony, videoconferencing and wireless communications; potential applications of the Main Profile include television broadcasting and video storage; and the Extended Profile may be particularly useful for streaming media applications. However, each Profile has sufficient flexibility to support a wide range of applications and so these examples of applications should not be considered definitive.

Figure 6.3 shows the relationship between the three Profiles and the coding tools supported by the standard. It is clear from this figure that the Baseline Profile is a subset of the Extended Profile, but not of the Main Profile. The details of each coding tool are described in Sections 6.4, 6.5 and 6.6 (starting with the Baseline Profile tools).

Performance limits for CODECs are defined by a set of Levels, each placing limits on parameters such as sample processing rate, picture size, coded bitrate and memory requirements.

6.3.2 Video Format

H.264 supports coding and decoding of 4:2:0 progressive or interlaced video[1] and the default sampling format for 4:2:0 progressive frames is shown in Figure 2.11 (other sampling formats may be signalled as Video Usability Information parameters). In the default sampling format, chroma (Cb and Cr) samples are aligned horizontally with every 2nd luma sample and are located vertically between two luma samples. An interlaced frame consists of two fields (a top field and a bottom field) separated in time and with the default sampling format shown in Figure 2.12.

[1] An extension to H.264 to support alternative colour sampling structures and higher sample accuracy is currently under development.

Google Exhibit No. 1028
Page 068

Google Exhibit No. 1063
Page 1069

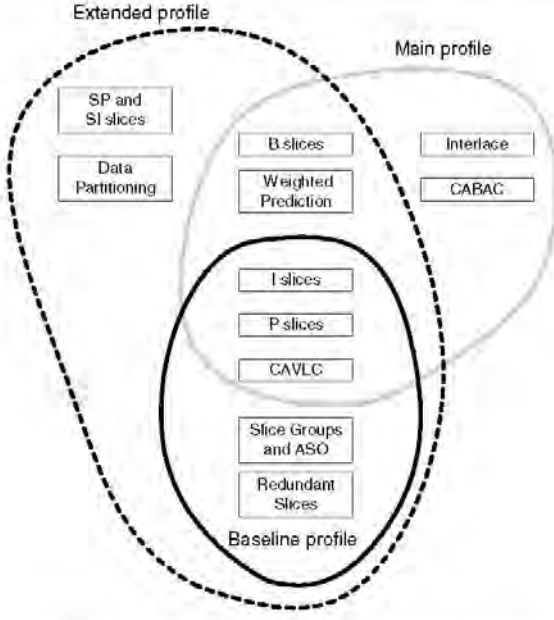


Figure 6.3 H.264 Baseline, Main and Extended profiles

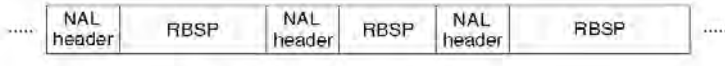


Figure 6.4 Sequence of NAL units

6.3.3 Coded Data Format

H.264 makes a distinction between a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). The output of the encoding process is VCL data (a sequence of bits representing the coded video data) which are mapped to NAL units prior to transmission or storage. Each NAL unit contains a Raw Byte Sequence Payload (RBSP), a set of data corresponding to coded video data or header information. A coded video sequence is represented by a sequence of NAL units (Figure 6.4) that can be transmitted over a packet-based network or a bitstream transmission link or stored in a file. The purpose of separately specifying the VCL and NAL is to distinguish between coding-specific features (at the VCL) and transport-specific features (at the NAL). Section 6.7 describes the NAL and transport mechanisms in more detail.

6.3.4 Reference Pictures

An H.264 encoder may use one or two of a number of previously encoded pictures as a reference for motion-compensated prediction of each inter coded macroblock or macroblock

Google Exhibit No. 1028
Page 069

Google Exhibit No. 1063
Page 1070

Table 6.1 H.264 slice modes

Slice type	Description	Profile(s)
I (Intra)	Contains only I macroblocks (each block or macroblock is predicted from previously coded data within the same slice).	All
P (Predicted)	Contains P macroblocks (each macroblock or macroblock partition is predicted from one list 0 reference picture) and/or I macroblocks.	All
B (Bi-predictive)	Contains B macroblocks (each macroblock or macroblock partition is predicted from a list 0 and/or a list 1 reference picture) and/or I macroblocks.	Extended and Main
SP (Switching P)	Facilitates switching between coded streams; contains P and/or I macroblocks.	Extended
SI (Switching I)	Facilitates switching between coded streams; contains SI macroblocks (a special type of intra coded macroblock).	Extended

partition. This enables the encoder to search for the best 'match' for the current macroblock partition from a wider set of pictures than just (say) the previously encoded picture.

The encoder and decoder each maintain one or two lists of reference pictures, containing pictures that have previously been encoded and decoded (occurring before and/or after the current picture in display order). Inter coded macroblocks and macroblock partitions in P slices (see below) are predicted from pictures in a single list, **list 0**. Inter coded macroblocks and macroblock partitions in a B slice (see below) may be predicted from two lists, **list 0** and **list 1**.

6.3.5 Slices

A video picture is coded as one or more slices, each containing an integral number of macroblocks from 1 (1 MB per slice) to the total number of macroblocks in a picture (1 slice per picture) The number of macroblocks per slice need not be constant within a picture. There is minimal inter-dependency between coded slices which can help to limit the propagation of errors. There are five types of coded slice (Table 6.1) and a coded picture may be composed of different types of slices. For example, a Baseline Profile coded picture may contain a mixture of I and P slices and a Main or Extended Profile picture may contain a mixture of I, P and B slices.

Figure 6.5 shows a simplified illustration of the syntax of a coded slice. The slice header defines (among other things) the slice type and the coded picture that the slice 'belongs' to and may contain instructions related to reference picture management (see Section 6.4.2). The slice data consists of a series of coded macroblocks and/or an indication of skipped (not coded) macroblocks. Each MB contains a series of header elements (see Table 6.2) and coded residual data.

6.3.6 Macroblocks

A macroblock contains coded data corresponding to a 16 × 16 sample region of the video frame (16 × 16 luma samples, 8 × 8 Cb and 8 × 8 Cr samples) and contains the syntax elements described in Table 6.2. Macroblocks are numbered (addressed) in raster scan order within a frame.

Google Exhibit No. 1028
Page 070

Google Exhibit No. 1063
Page 1071

Table 6.2 Macroblock syntax elements

mb_type	Determines whether the macroblock is coded in intra or inter (P or B) mode; determines macroblock partition size (see Section 6.4.2).
mb_pred	Determines intra prediction modes (intra macroblocks); determines list 0 and/or list 1 references and differentially coded motion vectors for each macroblock partition (inter macroblocks, except for inter MBs with 8 × 8 macroblock partition size).
sub_mb_pred	(Inter MBs with 8 × 8 macroblock partition size only) Determines sub-macroblock partition size for each sub-macroblock; list 0 and/or list 1 references for each macroblock partition; differentially coded motion vectors for each macroblock sub-partition.
coded_block_pattern	Identifies which 8 × 8 blocks (luma and chroma) contain coded transform coefficients.
mb_qp_delta	Changes the quantiser parameter (see Section 6.4.8).
residual	Coded transform coefficients corresponding to the residual image samples after prediction (see Section 6.4.8).

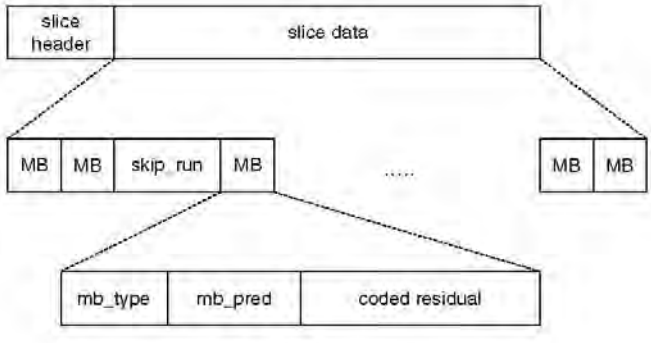


Figure 6.5 Slice syntax

6.4 THE BASELINE PROFILE

6.4.1 Overview

The Baseline Profile supports coded sequences containing I- and P-slices. I-slices contain intra-coded macroblocks in which each 16 × 16 or 4 × 4 luma region and each 8 × 8 chroma region is predicted from previously-coded samples in the same slice. P-slices may contain intra-coded, inter-coded or skipped MBs. Inter-coded MBs in a P slice are predicted from a number of previously coded pictures, using motion compensation with quarter-sample (luma) motion vector accuracy.

After prediction, the residual data for each MB is transformed using a 4 × 4 integer transform (based on the DCT) and quantised. Quantised transform coefficients are reordered and the syntax elements are entropy coded. In the Baseline Profile, transform coefficients are entropy coded using a context-adaptive variable length coding scheme (CAVLC) and all other

Google Exhibit No. 1028
Page 071

Google Exhibit No. 1063
Page 1072

syntax elements are coded using fixed-length or Exponential-Golomb Variable Length Codes. Quantised coefficients are scaled, inverse transformed, reconstructed (added to the prediction formed during encoding) and filtered with a de-blocking filter before (optionally) being stored for possible use in reference pictures for further intra- and inter-coded macroblocks.

6.4.2 Reference Picture Management

Pictures that have previously been encoded are stored in a reference buffer (the decoded picture buffer, DPB) in both the encoder and the decoder. The encoder and decoder maintain a list of previously coded pictures, reference picture list 0, for use in motion-compensated prediction of inter macroblocks in P slices. For P slice prediction, list 0 can contain pictures before and after the current picture in display order and may contain both *short term* and *long term* reference pictures. By default, an encoded picture is reconstructed by the encoder and marked as a short term picture, a recently-coded picture that is available for prediction. Short term pictures are identified by their frame number. Long term pictures are (typically) older pictures that may also be used for prediction and are identified by a variable LongTermPicNum. Long term pictures remain in the DPB until explicitly removed or replaced.

When a picture is encoded and reconstructed (in the encoder) or decoded (in the decoder), it is placed in the decoded picture buffer and is either (a) marked as 'unused for reference' (and hence not used for any further prediction), (b) marked as a short term picture, (c) marked as a long term picture or (d) simply output to the display. By default, short term pictures in list 0 are ordered from the highest to the lowest PicNum (a variable derived from the frame number) and long term pictures are ordered from the lowest to the highest LongTermPicNum. The encoder may signal a change to the default reference picture list order. As each new picture is added to the short term list at position 0, the indices of the remaining short-term pictures are incremented. If the number of short term and long term pictures is equal to the maximum number of reference frames, the oldest short-term picture (with the highest index) is removed from the buffer (known as *sliding window* memory control). The effect that this process is that the encoder and decoder each maintain a 'window' of N short-term reference pictures, including the current picture and $(N - 1)$ previously encoded pictures.

Adaptive memory control commands, sent by the encoder, manage the short and long term picture indexes. Using these commands, a short term picture may be assigned a long term frame index, or any short term or long term picture may be marked as 'unused for reference'.

The encoder chooses a reference picture from list 0 for encoding each macroblock partition in an inter-coded macroblock. The choice of reference picture is signalled by an index number, where index 0 corresponds to the first frame in the short term section and the indices of the long term frames start after the last short term frame (as shown in the following example).

Example: Reference buffer management (P-slice)

Current frame number = 250

Number of reference frames = 5

Google Exhibit No. 1028
Page 072

Google Exhibit No. 1063
Page 1073

Operation	Reference picture list				
	0	1	2	3	4
Initial state	–	–	–	–	–
Encode frame 250	250	–	–	–	–
Encode 251	251	250	–	–	–
Encode 252	252	251	250	–	–
Encode 253	253	252	251	250	–
Assign 251 to LongTermPicNum 0	253	252	250	0	–
Encode 254	254	253	252	250	0
Assign 253 to LongTermPicNum 4	254	252	250	0	4
Encode 255	255	254	252	0	4
Assign 255 to LongTermPicNum 3	254	252	0	3	4
Encode 256	256	254	0	3	4

(Note that in the above example, 0, 3 and 4 correspond to the decoded frames 251, 255 and 253 respectively).

Instantaneous Decoder Refresh Picture

An encoder sends an IDR (Instantaneous Decoder Refresh) coded picture (made up of I- or SI-slices) to clear the contents of the reference picture buffer. On receiving an IDR coded picture, the decoder marks all pictures in the reference buffer as 'unused for reference'. All subsequent transmitted slices can be decoded without reference to any frame decoded prior to the IDR picture. The first picture in a coded video sequence is always an IDR picture.

6.4.3 Slices

A bitstream conforming to the the Baseline Profile contains coded I and/or P slices. An I slice contains only intra-coded macroblocks (predicted from previously coded samples in the same slice, see Section 6.4.6) and a P slice can contain inter coded macroblocks (predicted from samples in previously coded pictures, see Section 6.4.5), intra coded macroblocks or Skipped macroblocks. When a Skipped macroblock is signalled in the bitstream, no further data is sent for that macroblock. The decoder calculates a vector for the skipped macroblock (see Section 6.4.5.3) and reconstructs the macroblock using motion-compensated prediction from the first reference picture in list 0.

An H.264 encoder may optionally insert a picture delimiter RBSP unit at the boundary between coded pictures. This indicates the start of a new coded picture and indicates which slice types are allowed in the following coded picture. If the picture delimiter is not used, the decoder is expected to detect the occurrence of a new picture based on the header of the first slice in the new picture.

Redundant coded picture

A picture marked as 'redundant' contains a redundant representation of part or all of a coded picture. In normal operation, the decoder reconstructs the frame from 'primary'

Google Exhibit No. 1028
Page 073

Google Exhibit No. 1063
Page 1074

Table 6.3 Macroblock to slice group map types

Type	Name	Description
0	Interleaved	run_length MBs are assigned to each slice group in turn (Figure 6.6).
1	Dispersed	MBs in each slice group are dispersed throughout the picture (Figure 6.7).
2	Foreground and background	All but the last slice group are defined as rectangular regions within the picture. The last slice group contains all MBs not contained in any other slice group (the 'background'). In the example in Figure 6.8, group 1 overlaps group 0 and so MBs not already allocated to group 0 are allocated to group 1.
3	Box-out	A 'box' is created starting from the centre of the frame (with the size controlled by encoder parameters) and containing group 0; all other MBs are in group 1 (Figure 6.9).
4	Raster scan	Group 0 contains MBs in raster scan order from the top-left and all other MBs are in group 1 (Figure 6.9).
5	Wipe	Group 0 contains MBs in vertical scan order from the top-left and all other MBs are in group 1 (Figure 6.9).
6	Explicit	A parameter, slice_group_id, is sent for each MB to indicate its slice group (i.e. the macroblock map is entirely user-defined).

(nonredundant)' pictures and discards any redundant pictures. However, if a primary coded picture is damaged (e.g. due to a transmission error), the decoder may replace the damaged area with decoded data from a redundant picture if available.

Arbitrary Slice Order (ASO)

The Baseline Profile supports Arbitrary Slice Order which means that slices in a coded frame may follow any decoding order. ASO is defined to be in use if the first macroblock in any slice in a decoded frame has a smaller macroblock address than the first macroblock in a *previously* decoded slice in the same picture.

Slice Groups

A *slice group* is a subset of the macroblocks in a coded picture and may contain one or more slices. Within each slice in a slice group, MBs are coded in raster order. If only one slice group is used per picture, then all macroblocks in the picture are coded in raster order (unless ASO is in use, see above). Multiple slice groups (described in previous versions of the draft standard as Flexible Macroblock Ordering or FMO) make it possible to map the sequence of coded MBs to the decoded picture in a number of flexible ways. The allocation of macroblocks is determined by a *macroblock to slice group map* that indicates which slice group each MB belongs to. Table 6.3 lists the different types of macroblock to slice group maps.

Example: 3 slice groups are used and the map type is 'interleaved' (Figure 6.6). The coded picture consists of first, all of the macroblocks in slice group 0 (filling every 3rd row of macroblocks); second, all of the macroblocks in slice group 1; and third, all of the macroblocks in slice group 0. Applications of multiple slice groups include error resilience, for example if one of the slice groups in the dispersed map shown in Figure 6.7 is 'lost' due to errors, the missing data may be concealed by interpolation from the remaining slice groups.

Google Exhibit No. 1028
Page 074

Google Exhibit No. 1063
Page 1075

6.6 THE EXTENDED PROFILE

The Extended Profile (known as the X Profile in earlier versions of the draft H.264 standard) may be particularly useful for applications such as video streaming. It includes all of the features of the Baseline Profile (i.e. it is a superset of the Baseline Profile, unlike Main Profile), together with B-slices (Section 6.5.1), Weighted Prediction (Section 6.5.2) and additional features to support efficient streaming over networks such as the Internet. SP and SI slices facilitate switching between different coded streams and 'VCR-like' functionality and Data Partitioned slices can provide improved performance in error-prone transmission environments.

6.6.1 SP and SI slices

SP and SI slices are specially-coded slices that enable (among other things) efficient switching between video streams and efficient random access for video decoders [10]. A common requirement in a streaming application is for a video decoder to switch between one of several encoded streams. For example, the same video material is coded at multiple bitrates for transmission across the Internet and a decoder attempts to decode the highest-bitrate stream it can receive but may require switching automatically to a lower-bitrate stream if the data throughput drops.

Example

A decoder is decoding Stream A and wants to switch to decoding Stream B (Figure 6.48). For simplicity, assume that each frame is encoded as a single slice and predicted from one reference (the previous decoded frame). After decoding P-slices A_0 and A_1, the decoder wants to switch to Stream B and decode B_2, B_3 and so on. If all the slices in Stream B are coded as P-slices, then the decoder will not have the correct decoded reference frame(s) required to reconstruct B_2 (since B_2 is predicted from the decoded picture B_1 which does not exist in stream A). One solution is to code frame B_2 as an I-slice. Because it is coded without prediction from any other frame, it can be decoded independently of preceding frames in stream B and the decoder can therefore switch between stream A and stream B as shown in Figure 6.49. Switching can be accommodated by inserting an I-slice at regular intervals in the coded sequence to create 'switching points'. However, an I-slice is likely to contain much more coded data than a P-slice and the result is an undesirable peak in the coded bitrate at each switching point.

SP-slices are designed to support switching between similar coded sequences (for example, the same source sequence encoded at various bitrates) without the increased bitrate penalty of I-slices (Figure 6.49). At the switching point (frame 2 in each sequence), there are three SP-slices, each coded using motion compensated prediction (making them more efficient than I-slices). SP-slice A_2 can be decoded using reference picture A_1 and SP-slice B_2 can be decoded using reference picture B_1. The key to the switching process is SP-slice AB_2 (known as a *switching SP-slice*), created in such a way that it can be decoded using motion-compensated reference picture A_1, to produce decoded frame B_2 (i.e. the decoder output frame B_2 is identical whether decoding B_1 followed by B_2 or A_1 followed by AB_2). An extra SP-slice is required at each switching point (and in fact another SP-slice, BA_2, would be required to switch in the other direction) but this is likely to be more efficient than encoding frames A_2

Google Exhibit No. 1028
Page 075

Google Exhibit No. 1063
Page 1076

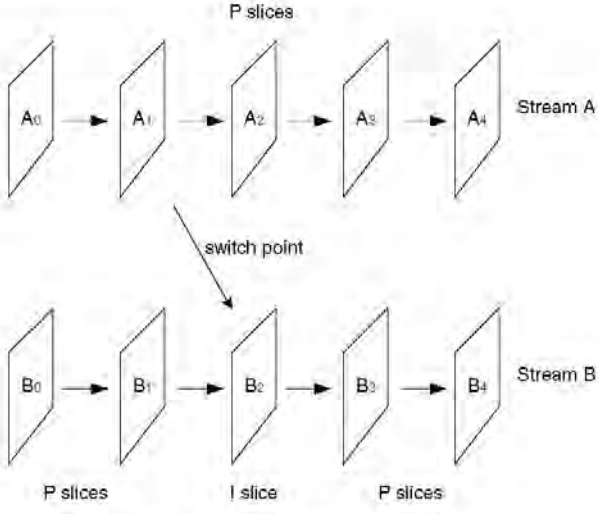


Figure 6.48 Switching streams using I-slices

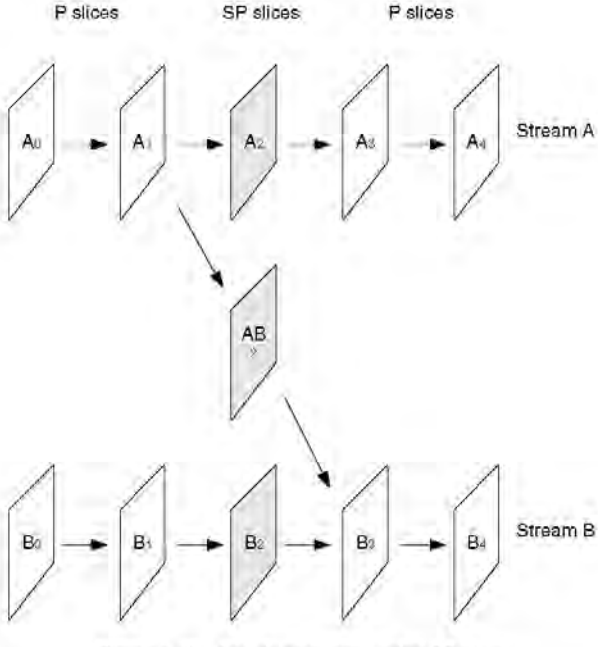


Figure 6.49 Switching streams using SP-slices

Google Exhibit No. 1028
Page 076

Google Exhibit No. 1063
Page 1077

Table 6.18 Switching from stream A to stream B using SP-slices

Input to decoder	MC reference	Output of decoder
P-slice A_0	[earlier frame]	Decoded frame A_0
P-slice A_1	Decoded frame A_0	Decoded frame A_1
SP-slice AB_2	Decoded frame A_1	Decoded frame B_2
P-slice B_3	Decoded frame B_2	Decoded frame B_3
....	....	....

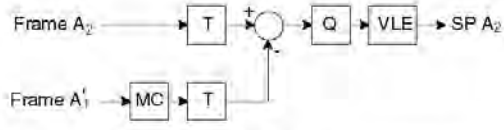


Figure 6.50 Encoding SP-slice A_2 (simplified)

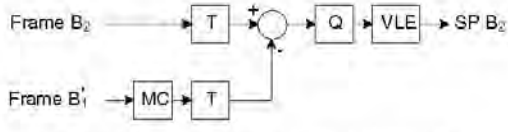


Figure 6.51 Encoding SP-slice B_2 (simplified)

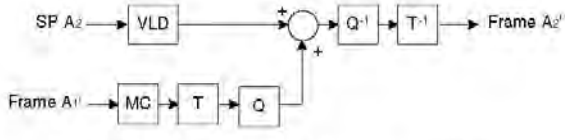


Figure 6.52 Decoding SP-slice A_2 (simplified)

and B_2 as I-slices. Table 6.18 lists the steps involved when a decoder switches from stream A to stream B.

Figure 6.50 shows a simplified diagram of the encoding process for SP-slice A_2, produced by subtracting a motion-compensated version of A'_1 (decoded frame A_1) from frame A_2 and then coding the residual. Unlike a 'normal' P-slice, the subtraction occurs in the transform domain (after the block transform). SP-slice B_2 is encoded in the same way (Figure 6.51). A decoder that has previously decoded frame A_1 can decode SP-slice A_2 as shown in Figure 6.52. Note that these diagrams are simplified; in practice further quantisation and rescaling steps are required to avoid mismatch between encoder and decoder and a more detailed treatment of the process can be found in [11].

Google Exhibit No. 1028
Page 077

Google Exhibit No. 1063
Page 1078

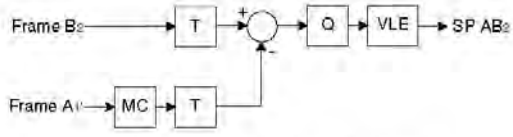


Figure 6.53 Encoding SP-slice AB_2 (simplified)

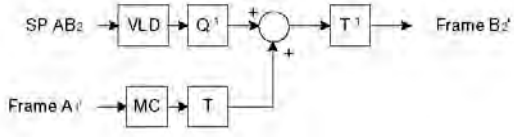


Figure 6.54 Decoding SP-slice AB_2 (simplified)

SP-slice AB_2 is encoded as shown in Figure 6.53 (simplified). Frame B_2 (the frame we are switching to) is transformed and a motion-compensated prediction is formed from A_1' (the decoded frame from which we are switching). The 'MC' block in this diagram attempts to find the best match for each MB of frame B_2 *using decoded picture* A_1 *as a reference*. The motion-compensated prediction is transformed, then subtracted from the transformed B_2 (i.e. in the case of a switching SP slice, subtraction takes place in the transform domain). The residual (after subtraction) is quantized, encoded and transmitted.

A decoder that has previously decoded A_1' can decode SP-slice AB_2 to produce B_2' (Figure 6.54). A_1' is motion compensated (using the motion vector data encoded as part of AB_2), transformed and added to the decoded and scaled (inverse quantized) residual, then the result is inverse transformed to produce B_2'.

If streams A and B are versions of the same original sequence coded at different bitrates, the motion-compensated prediction of B_2 from A_1' (SP-slice AB_2) should be quite efficient. Results show that using SP-slices to switch between different versions of the same sequence is significantly more efficient than inserting I-slices at switching points. Another application of SP-slices is to provide random access and 'VCR-like' functionalities. For example, an SP-slice and a switching SP-slice are placed at the position of frame 10 (Figure 6.55). A decoder can fast-forward from A_0 directly to A_{10} by (a) decoding A_0, then (b) decoding switching SP-slice A_{0-10} to produce A_{10} by prediction from A_0.

A further type of switching slice, the SI-slice, is supported by the Extended Profile. This is used in a similar way to a switching SP-slice, except that the prediction is formed using the 4×4 Intra Prediction modes (see Section 6.4.6.1) from previously-decoded samples of the reconstructed frame. This slice mode may be used (for example) to switch from one sequence to a completely different sequence (in which case it will not be efficient to use motion compensated prediction because there is no correlation between the two sequences).

Google Exhibit No. 1028
Page 078

Google Exhibit No. 1063
Page 1079

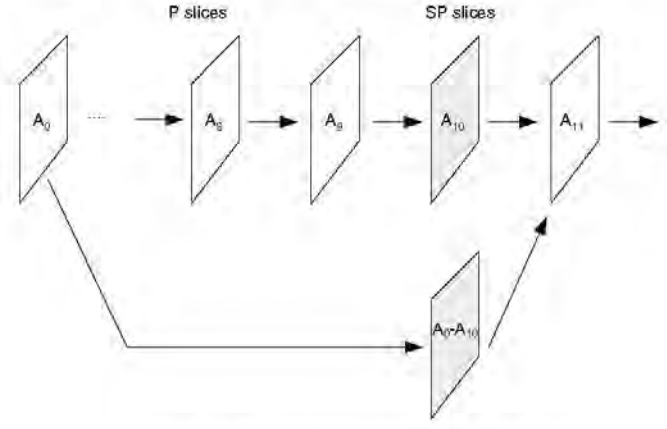


Figure 6.55 Fast-forward using SP-slices

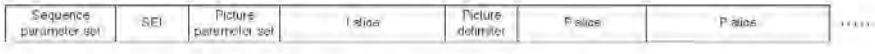


Figure 6.56 Example sequence of RBSP elements

6.6.2 Data Partitioned Slices

The coded data that makes up a slice is placed in three separate Data Partitions (A, B and C), each containing a subset of the coded slice. Partition A contains the slice header and header data for each macroblock in the slice, Partition B contains coded residual data for Intra and SI slice macroblocks and Partition C contains coded residual data for inter coded macroblocks (forward and bi-directional). Each Partition can be placed in a separate NAL unit and may therefore be transported separately.

If Partition A data is lost, it is likely to be difficult or impossible to reconstruct the slice, hence Partition A is highly sensitive to transmission errors. Partitions B and C can (with careful choice of coding parameters) be made to be independently decodeable and so a decoder may (for example) decode A and B only, or A and C only, lending flexibility in an error-prone environment.

6.7 TRANSPORT OF H.264

A coded H.264 video sequence consists of a series of NAL units, each containing an RBSP (Table 6.19). Coded slices (including Data Partitioned slices and IDR slices) and the End of Sequence RBSP are defined as VCL NAL units whilst all other elements are just NAL units.

An example of a typical sequence of RBSP units is shown in Figure 6.56. Each of these units is transmitted in a separate NAL unit. The header of the NAL unit (one byte) signals the type of RBSP unit and the RBSP data makes up the rest of the NAL unit.

Google Exhibit No. 1028
Page 079

Google Exhibit No. 1063
Page 1080

Table 6.19

RBSP type	Description
Parameter Set	'Global' parameters for a sequence such as picture dimensions, video format, macroblock allocation map (see Section 6.4.3).
Supplemental Enhancement Information	Side messages that are not essential for correct decoding of the video sequence.
Picture Delimiter	Boundary between video pictures (optional). If not present, the decoder infers the boundary based on the frame number contained within each slice header.
Coded slice	Header and data for a slice; this RBSP unit contains actual coded video data.
Data Partition A, B or C	Three units containing Data Partitioned slice layer data (useful for error resilient decoding). Partition A contains header data for all MBs in the slice, Partition B contains intra coded data and partition C contains inter coded data.
End of sequence	Indicates that the next picture (in decoding order) is an IDR picture (see Section 6.4.2). (Not essential for correct decoding of the sequence).
End of stream	Indicates that there are no further pictures in the bitstream. (Not essential for correct decoding of the sequence).
Filler data	Contains 'dummy' data (may be used to increase the number of bytes in the sequence). (Not essential for correct decoding of the sequence).

Parameter sets

H.264 introduces the concept of *parameter sets*, each containing information that can be applied to a large number of coded pictures. A *sequence parameter set* contains parameters to be applied to a complete video sequence (a set of consecutive coded pictures). Parameters in the sequence parameter set include an identifier (seq_parameter_set_id), limits on frame numbers and picture order count, the number of reference frames that may be used in decoding (including short and long term reference frames), the decoded picture width and height and the choice of progressive or interlaced (frame or frame / field) coding. A *picture parameter set* contains parameters which are applied to one or more decoded pictures within a sequence. Each picture parameter set includes (among other parameters) an identifier (pic_parameter_set_id), a selected seq_parameter_set_id, a flag to select VLC or CABAC entropy coding, the number of slice groups in use (and a definition of the type of slice group map), the number of reference pictures in list 0 and list 1 that may be used for prediction, initial quantizer parameters and a flag indicating whether the default deblocking filter parameters are to be modified.

Typically, one or more sequence parameter set(s) and picture parameter set(s) are sent to the decoder prior to decoding of slice headers and slice data. A coded slice header refers to a pic_parameter_set_id and this 'activates' that particular picture parameter set. The 'activated' picture parameter set then remains active until a different picture parameter set is activated by being referred to in another slice header. In a similar way, a picture parameter set refers to a seq_parameter_set_id which 'activates' that sequence parameter set. The activated set remains in force (i.e. its parameters are applied to all consecutive coded pictures) until a different sequence parameter set is activated.

The parameter set mechanism enables an encoder to signal important, infrequently-changing sequence and picture parameters separately from the coded slices themselves. The parameter sets may be sent well ahead of the slices that refer to them, or by another transport

Google Exhibit No. 1028
Page 080

Google Exhibit No. 1063
Page 1081

mechanism (e.g. over a reliable transmission channel or even 'hard wired' in a decoder implementation). Each coded slice may 'call up' the relevant picture and sequence parameters using a single VLC (pic_parameter_set_id) in the slice header.

Transmission and Storage of NAL units

The method of transmitting NAL units is not specified in the standard but some distinction is made between transmission over packet-based transport mechanisms (e.g. packet networks) and transmission in a continuous data stream (e.g. circuit-switched channels). In a packet-based network, each NAL unit may be carried in a separate packet and should be organised into the correct sequence prior to decoding. In a circuit-switched transport environment, a start code prefix (a uniquely-identifiable delimiter code) is placed before each NAL unit to make a *byte stream* prior to transmission. This enables a decoder to search the stream to find a start code prefix identifying the start of a NAL unit.

In a typical application, coded video is required to be transmitted or stored together with associated audio track(s) and side information. It is possible to use a range of transport mechanisms to achieve this, such as the Real Time Protocol and User Datagram Protocol (RTP/UDP). An Amendment to MPEG-2 Systems specifies a mechanism for transporting H.264 video (see Chapter 7) and ITU-T Recommendation H.241 defines procedures for using H.264 in conjunction with H.32× multimedia terminals. Many applications require storage of multiplexed video, audio and side information (e.g. streaming media playback, DVD playback). A forthcoming Amendment to MPEG-4 Systems (Part 1) specifies how H.264 coded data and associated media streams can be stored in the ISO Media File Format (see Chapter 7).

6.8 CONCLUSIONS

H.264 provides mechanisms for coding video that are optimised for compression efficiency and aim to meet the needs of practical multimedia communication applications. The range of available coding tools is more restricted than MPEG-4 Visual (due to the narrower focus of H.264) but there are still many possible choices of coding parameters and strategies. The success of a practical implementation of H.264 (or MPEG-4 Visual) depends on careful design of the CODEC and effective choices of coding parameters. The next chapter examines design issues for each of the main functional blocks of a video CODEC and compares the performance of MPEG-4 Visual and H.264.

6.9 REFERENCES

1. ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.
2. T. Wiegand, G. Sullivan, G. Bjontegaard and A. Luthra, Overview of the H.264 / AVC Video Coding Standard, IEEE Transactions on Circuits and Systems for Video Technology, to be published in 2003.
3. A. Hallapuro, M. Karczewicz and H. Malvar, Low Complexity Transform and Quantization – Part I: Basic Implementation, JVT document JVT-B038, Geneva, February 2002.
4. H.264 Reference Software Version JM6.1d, http://bs.hhi.de/~suehring/tml/, March 2003.

Google Exhibit No. 1028
Page 081

Google Exhibit No. 1063
Page 1082

5. S. W. Golomb, Run-length encoding, *IEEE* Trans. on Inf. Theory, **IT-12**, pp. 399–401, 1966.
6. G. Bjøntegaard and K. Lillevold, Context-adaptive VLC coding of coefficients, JVT document JVT-C028, Fairfax, May 2002.
7. D. Marpe, G. Blättermann and T. Wiegand, Adaptive codes for H.26L, ITU-T SG16/6 document VCEG-L13, Eibsee, Germany, January 2001.
8. H. Schwarz, D. Marpe and T. Wiegand, CABAC and slices, JVT document JVT-D020, Klagenfurt, Austria, July 2002
9. D. Marpe, H. Schwarz and T. Wiegand, Context-Based Adaptive Binary Arithmetic Coding in the H.264 / AVC Video Compression Standard, IEEE Transactions on Circuits and Systems for Video Technology, to be published in 2003.
10. M. Karczewicz and R. Kurceren, A proposal for SP-frames, ITU-T SG16/6 document VCEG-L27, Eibsee, Germany, January 2001.
11. M. Karczewicz and R. Kurceren, The SP and SI Frames Design for H.264/AVC, IEEE Transactions on Circuits and Systems for Video Technology, to be published in 2003.

Google Exhibit No. 1028
Page 082

Google Exhibit No. 1063
Page 1083

Ex. 1028A



BOOK

H.264 and MPEG-4 video compression : video coding for next generation multimedia

Richardson, Iain E. G.

c2003

Available at Colchester, Albert Sloman Library Floor 2 north (TK 6680.5.R5) >

TOP
SEND TO
GET IT
LINKS
DETAILS
VIRTUAL BROWSE

Send to

REPORT BROKEN LINK | QR | EXPORT RIS | ENDNOTE | PRINT

PERMALINK | CITATION | EMAIL

https://essex.primo.exlibrisgroup.com/permalink/44UOES_INST/ngp9lg/alma991004277709707346

COPY THE PERMALINK TO CLIPBOARD

Get It

Please sign in to check if there are any request options. Sign in

BACK TO LOCATIONS

LOCATION ITEMS

Colchester, Albert Sloman Library
Available , Floor 2 north ; TK 6680.5.R5
(1 copy, 1 available, 0 requests)

Item in place
Log in for more information

Talk to us live

Links

Display Source Record

Details

Title	H.264 and MPEG-4 video compression : video coding for next generation multimedia
Creator	Richardson, Iain E. G.
Publisher	Chichester : Wiley
Creation Date	c2003
Format	xxiv, 281 p. : ill ; 26 cm.
Subject	Video compression MPEG (Video coding standard) Multimedia systems
Source	Library Catalog
Identifier	OCLC : (OCoLC)53361968
MMS ID	991004277709707346

Virtual Browse



mation nology-- G audio

MPEG video compression standard ...

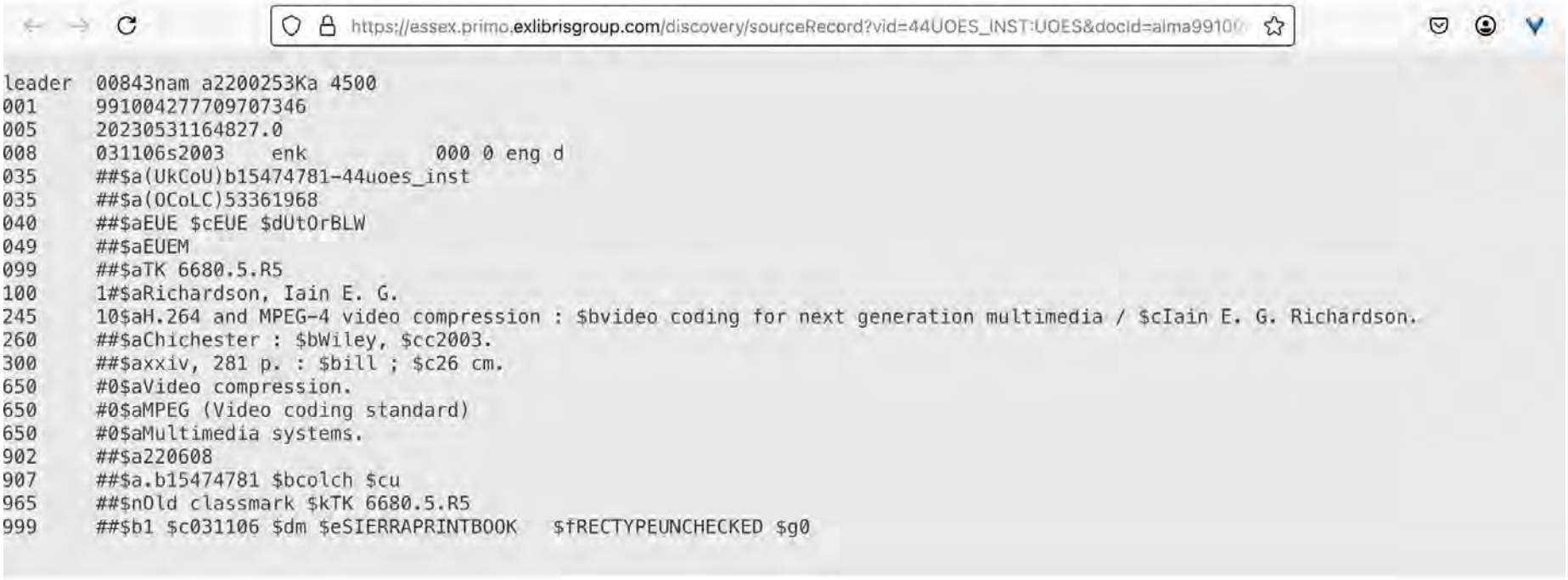


A technical introduction to digital

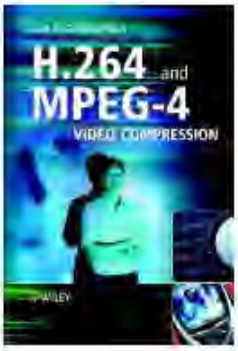


Video codec design : developing


nologies.
1. MPEG ...
17
c1996
video ...
c1996
image and
video ...
c2002


https://essex.primo.exlibrisgroup.com/discovery/sourceRecord?vid=44UOES_INST:UOES&docid=alma99100
leader 00843nam a2200253Ka 4500
001 991004277709707346
005 20230531164827.0
008 031106s2003 enk 000 0 eng d
035 ##$a(UkCoU)b15474781-44uoes_inst
035 ##$a(OCoLC)53361968
040 ##$aEUE $cEUE $dUtOrBLW
049 ##$aEUEM
099 ##$aTK 6680.5.R5
100 1#$aRichardson, Iain E. G.
245 10$aH.264 and MPEG-4 video compression : $bvideo coding for next generation multimedia / $cIain E. G. Richardson.
260 ##$aChichester : $bWiley, $cc2003.
300 ##$axxiv, 281 p. : $bill ; $c26 cm.
650 #0$aVideo compression.
650 #0$aMPEG (Video coding standard)
650 #0$aMultimedia systems.
902 ##$a220608
907 ##$a.b15474781 $bcolch $cu
965 ##$nOld classmark $kTK 6680.5.R5
999 ##$b1 $c031106 $dm $eSIERRAPRINTBOOK $fRECTYPEUNCHECKED $g0

Ex. 1028B



H.264 and MPEG-4 video compression : video coding for next-generation multimedia /

Show other versions (1)

Main Author:	Richardson, Iain E. G.
Format:	Book
Language:	English
Published:	Chichester : Wiley, [2003]
Subjects:	Digital video > Standards. Video compression. MPEG (Video coding standard) Coding theory.

Holdings Description Other Versions (1) Similar Items Staff View

MARC

LEADER	00000pam a2200000 a 4500		
001	in00001815737		
005	20151025203832.0		
008	031124s2003 enka b 001 0 eng		
015			**|a** GBA3-Y8304
020			**|a** 0470848375
035			**|a** (OCoLC)ocm53361968
040			**|a** UKM **|c** UKM **|d** TXA **|d** UtOrBLW
049			**|a** TXAM
082	0	4	**|a** 621.388 **|2** 21

Google Exhibit No. 1063
Page 1090

090			**|a** TK6680.5 **|b** .R524 2003
100	1		**|a** Richardson, Iain E. G.
245	1	0	**|a** H.264 and MPEG-4 video compression : **|b** video coding for next-generation multimedia / **|c** Iain E.G. Richardson.
264		1	**|a** Chichester : **|b** Wiley, **|c** [2003]
264		4	**|c** ©2003
300			**|a** xxiv, 281 pages : **|b** illustrations ; **|c** 26 cm.
336			**|a** text **|b** txt **|2** rdacontent
337			**|a** unmediated **|b** n **|2** rdamedia
338			**|a** volume **|b** nc **|2** rdacarrier
504			**|a** Includes bibliographical references (page [277]) and index.
650		0	**|a** Digital video **|x** Standards.
650		0	**|a** Video compression.
650		0	**|a** MPEG (Video coding standard)
650		0	**|a** Coding theory.

Google Exhibit No. 1063
Page 1091

948			**|a** cataloged **|b** h **|c** 2004/1/28 **|d** c **|e** mphillip **|f** 2:15:44 pm
994			**|a** E0 **|b** TXA
944			**|a** stk **|h** REPL<3200468>REP-SAME<>REF<jstephen> **|m** Replacement on order
999			**|a** MARS
999	f	f	**|s** f8903c2e-dbb9-31e1-88f7-0e01fb4cf32a **|i** d895a291-d9dd-3beb-8652-9d4133ae50e9 **|t** 0
952	f	f	**|p** normal **|a** Texas A&M University **|b** College Station **|c** Sterling C. Evans Library **|d** Evans: Library Stacks **|t** 0 **|e** TK6680.5 .R524 2003 **|h** Library of Congress classification **|i** unmediated -- volume **|m** A14829957247
998	f	f	**|a** TK6680.5 .R524 2003 **|t** 0 **|l** Evans: Library Stacks

Google Exhibit No. 1063
Page 1092

Low-power H.264 video decoder with graceful degradation

Arnaud Bourge*, Joël Jung*
Philips Digital Systems Laboratories - Paris
51, rue Carnot, BP301, 92156 Suresnes Cedex, France

ABSTRACT

In this paper, we present a novel H.264 video decoder where memory transfers and energy consumption are significantly reduced. A power-friendly solution is indeed required in mobile applications, where autonomy is a key feature. Whereas low-power design is usually regarded as a pure architectural and implementation problem, our solution is based on an algorithmic approach. We first observe that in modern hardware architectures, computational complexity plays a second role in terms of energy dissipation, which is dominated by memory transfers. We thus identify the motion compensation module as the most consuming part of the standard H.264 decoder, due to its numerous memory accesses to the reference frame(s). Through an algorithmic modification that uses concurrent embedded compression techniques and relies on the data structure of the memory, we reduce memory transfers, and hence power dissipation, while maintaining the visual quality. Experimental results prove that the new method is close to the reference H.264 baseline decoder both in terms of objective and subjective measurements. In the meantime, memory transfers have been reduced by 55% in average, implying power savings which lengthen the battery life of the mobile device, increase the reliability of the chip and lower production costs.

Keywords: Low Power, Battery life, H.264, Video Decoding, Memory Transfers, Complexity, Motion Compensation, Embedded Compression, Subjective Quality.

1. INTRODUCTION

The evolution of mobile devices shows a fast integration of multimedia functionalities. Today the capture and display of color images at reasonable resolutions are common features in cellular phones. Video is already present in some products (high-end mobile phones, PDAs) and is expected to spread with the emergence of the MMS (Multimedia Messaging System). However, this integration comes at the expense of increasing power consumption. Indeed, mobile devices providing video-decoding capabilities suffer from a shortened battery life, whereas autonomy remains a key-feature in the mobile world. Advances in battery capacities only partially answer this problem, because the complexity of codecs and the requirements in picture size and frame-rate keep increasing too. Directly reducing the energy dissipation of a video decoding circuit is thus crucial. It does not only extend battery life, but also increases the reliability of the chip. Furthermore, power saving allows a significant attenuation of the cooling effort, which simplifies the hardware management and lowers production costs.

Low-power design is usually regarded as a pure architectural problem. For instance, hardware/software partitioning allows an improved management of the power consumption[1] by allocating the most consuming parts to dedicated components. Embedded memories also help because they are located close to the CPU and co-processor, so an access to their data goes through less cable and dissipates less energy[2]. Finally, optimizing the implementation of a given algorithm for a specific platform is primordial[3]. In this classic approach, however, power matters are considered at the latest stage and result in a non-optimal solution.

On the contrary, we take the energy dissipation into account at the algorithmic stage. This approach, if combined with architectural considerations, leads to the most promising gains[4] because some algorithms intrinsically have the property to enable power-efficient implementations[5]. The counterpart is that evaluating the actual power consumption at the algorithmic stage is a tricky task that can only result in qualitative estimations. In practice, we translate consumption properties of a generic platform into guidelines and constraints for the design of algorithms[5]. First of all, low power should not be regarded as a mere complexity problem. Today, power consumption is led by memory transfers, and especially by accesses to an external memory. Computations only represent a minor (though not negligible) portion of

* {arnaud.bourge ; joel.jung }@philips.com

Visual Communications and Image Processing 2004, edited by Sethuraman Panchanathan, Bhaskaran Vasudev, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 5308
© 2004 SPIE and IS&T · 0277-786X/04/$15

Google Exhibit No. 1029
Page 001

Google Exhibit No. 1063
Page 1093

the overall dissipation[7]. As a consequence, a low-power video-decoding algorithm should mainly minimize the number of accesses to the memory, even at the expense of a moderate complexity increase[8]. Hence, we do not measure performance through a classical rate-distortion trade-off, but through a power-rate-distortion criterion. It means that a loss in visual quality can be compensated by a gain in power consumption.

We have chosen to start from the H.264 decoding scheme - that is the state-of-the-art standard in terms of coding efficiency - at the Baseline Profile, which is intended to be used in the mobile world. In this paper, we present a novel H.264 video decoder where memory transfers and power consumption are significantly reduced. Section 2 is a preliminary study. We present the state-of-the-art in low-power video decoding, show that the motion compensation is the most costly module, explain why we choose to apply embedded compression, and highlight some difficulties inherent to this kind of scheme. In Section 3 we define our strategy, which consists in imposing some specific constraints to the embedded compression, and we describe our algorithmic solution. Section 4 is dedicated to the experimental results, where we show that the memory size can be reduced by a factor of 3, and the associated memory transfers by a factor of 2.3 while maintaining the visual quality. Results also include complexity evaluations, rate-distortion curves and subjective quality assessments. Section 5 concludes this paper.

2. PRELIMINARY STUDY

2.1. State of the art

We have seen in the introduction some examples of low-power decoder designs based on architectural optimizations[1,2,3]. Although this is the most common approach, our method is more focused on the algorithmic side. Following the same principle, authors have built their decompression algorithm based on underlying hardware properties to minimize power dissipation. Most of them propose computational simplifications, but this method is not sufficient anymore to reduce energy consumption. Others put the strain on minimizing data transfers, both spatially and temporally. In the spatial domain, isometries were thus proposed as a way to significantly reduced a vector quantization (VQ) codebook[9]. Moreover in [10], "power scalability", depicted as the ability for the codec to change its power consumption depending on the available bandwidth and on the required video quality, is introduced through the use of a hierarchical VQ. Yet we will see that the spatial domain is not the most critical path regarding memory transfers in a conventional decoder. In the temporal domain, the Motion Estimation / Motion Compensation can be replaced by a so-called Conditional Replenishment[11] but at the cost of compression performance. Alternatively, an efficient trade-off between data transfers and coding efficiency is introduced in [6] with the notion of "Collocated Motion Estimation".
All the above solutions imply the use of a non-conventional encoder to generate the incoming bit-stream. They are not applicable in a context where interoperability and exchanges impose the use of an accepted standard, which is generally the case in mobile communications. Strictly speaking, one cannot come up with a completely new scheme only for the decoder: the corresponding encoding scheme must be accepted too. However, it is possible to decode a standard bit-stream in a manner that differs from the standard algorithm. The resulting sequence is necessarily degraded, but a sound control of the modifications can keep the loss small while some costly operations are saved. In fact this notion of "graceful degradation" introduces a trade-off between the quality of the reconstruction and the cost of the decoding[12]. Each modification of the standard scheme introduces a deviation that impacts the visual result. The goal is then to find modifications that maximize the gain in complexity while minimizing the visual losses. Finally the algorithm can adapt itself to the available resources: depending on the CPU utilization, some operations are done, simplified or even skipped. For instance the Inverse DCT can be truncated, or the sub-pixel interpolation filters of the motion compensation can be approximated.

2.2. Identification of the power-costly module: Motion Compensation

We have seen that in the literature "graceful degradation" is commonly applied to computational complexity. We extend this notion to power consumption by adapting the decoding scheme to the available energy resources, thus introducing power-related graceful degradation. We start with the identification of the power-consuming part of the H.264 decoder, which we will then modify.
Memory transfers, and especially accesses to an external memory, have a major influence on power consumption[8]. Computations (arithmetic and logical operations) still represent a significant portion of the overall dissipation, but there

Google Exhibit No. 1029
Page 002

Google Exhibit No. 1063
Page 1094

is a trend towards a reduction of its relative weight[7]. The design of low-power algorithms is thus primarily a matter of minimizing memory transfers, which is closely related to bandwidth reduction between the memory and the processing modules. This is especially true in hardware-only and in mixed hardware/software solutions. In software-only environments, however, the power repartition between memory transfers and computations is more balanced and difficult to measure.

In the scope of an H.264 video decoder, the Motion Compensation consumes most of the memory transfers. This is because the other parts of the decoder can generally be performed locally, on a block-by-block basis. In this paper, the term "Motion Compensation" (MC) defines all the operations (logical, arithmetic and memory) that are part of the feedback loop of the predictive scheme, as shown in Figure 1. It includes storing the decoded frame to be used as a reference for the next frame(s), calculating the Motion Vectors (MV), reading the memory to get the reference pixels, and finally performing the interpolation filters that generate the sub-pixel grid.

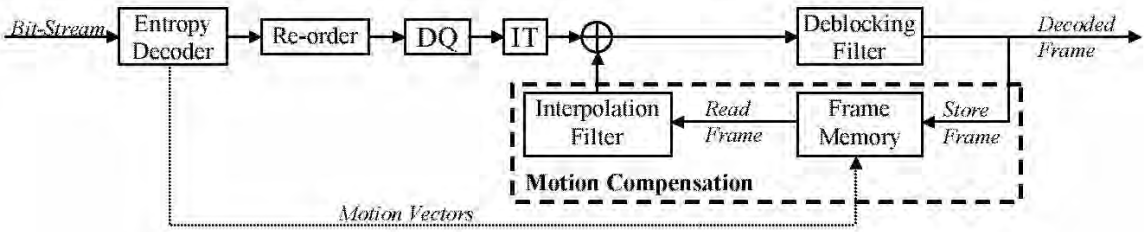


Figure 1: General scheme of a standard H.264 decoder.

The storing of the latest decoded frame is unavoidable in the standard scheme because all its pixels may serve as references for the next frame(s). In CIF, with a 4:2:0 format, this implies $352x288x1.5=152,064$ bytes to be stored. Moreover, even in the Baseline Profile, up to 5 preceding frames are used as references, so with the currently decoded frame there are 6 frames in the memory. Consequently, a 1MB external memory is required. It consumes much more power than a small internal memory, partly due to longer connections. During the decoding of one frame, around 300kB of data have to be transferred to and from this memory (read/write operations). No other module needs to store and access so much data: most of them can be performed on the fly, except the deblocking filter in case FMO (Flexible Macroblock Ordering) is enabled. We do not consider this possibility in the remaining of this paper. The Motion Compensation is thus responsible for a major part of the bandwidth between the external memory and the processing modules.

2.3. An existing technique for reducing memory size and transfers in the MC: Embedded Compression

Recent work has put the strain on reducing this bandwidth by presenting an appropriate memory management[13]. The pixels of the stored reference frame(s) are mapped to the structure of a DDR-SDRAM block-by-block instead of line-by-line, yielding excellent results. This method optimized the implementation of the MC without changing the decoding algorithm itself. It was then proposed to go further by using embedded compression techniques[14]. Embedded Compression (EC) consists in compressing the reference frame(s) in the feedback loop of a conventional predictive video coding scheme[15] as depicted in Figure 2. The additional encoder and decoder that perform the EC have a computational cost (complexity). This cost must be moderate, because in an extreme case the associated increase in CPU dissipation could exceed the expected power savings on memory accesses.

On quality aspects, if the selected compression technique is lossless it has no impact on the reconstructed sequence, but of course the compression ratio (and thus the expected gain) is limited (two at most). On the contrary if the compression is lossy the memory savings are much higher. It comes at the expense of a quality degradation that worsens along a Group of Pictures (GoP). The reference frames are compressed with loss, so the decoding of the next frame is erroneous. The predictive structure of video coding schemes make this reconstruction error propagate and accumulate: visual quality gets inexorably poorer and poorer. This phenomenon is known as the drift effect. Designing a low-power decoder with embedded compression, we have to minimize this unwanted artifact that is temporally very annoying.

Google Exhibit No. 1029
Page 003

Google Exhibit No. 1063
Page 1095

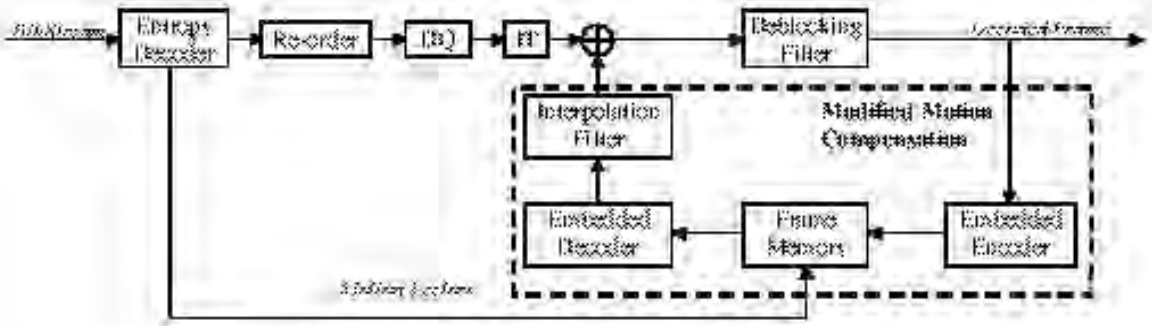


Figure 2. Internal structure of an H.264 decoder with Embedded Compression

3.4. Drawback of Embedded Compression that is removed

When the goal of Embedded Compression (EC) is purely to reduce the required memory size for cost reasons, virtually any efficient still image compression technique can be performed. In practice, block-based schemes are convenient because they match the block-oriented structure of the processing bit-stream, allowing on-the-fly processing. However, we also aim at reducing the bandwidth, and the relation between the Compression Ratio (CR) of the EC and the gain in memory transfers is not direct, due to overhead.

We define overhead as the ratio between the number of pixels that are actually accessed during the MC of a block and the number of pixels that are really useful to the reference block. This ratio is superior to 1 because of the block-based nature of EC techniques. For instance let us consider that the reference frames are encoded on an 8x8 block grid. The reference frame no longer has a pixel-oriented structure, but a block-oriented one (Figure 3): to access the value of one pixel we must fetch and decode the whole block it belongs to.

This is not a problem as long as the reference pixels for the MC form blocks that are aligned with the grid of the EC, but it becomes annoying when the motion vector points to a reference that overlaps several encoded blocks. It was already true with previous standards,[3] but it becomes even more serious with H.264. Indeed, a 16x16 macroblock can be divided in 16x8, 8x16 or 8x8 blocks during the MC. Any 8x8 block can then be sub-divided into 8x4, 4x8 or 4x4 sub-blocks. In the case depicted in Figure 4, this has a dramatic impact on the overhead. Indeed the compressed block is a 4x4 block (allowed in H.264) that is not at all aligned with the 8x8 grid. Four encoded blocks, corresponding to 256 pixels, have thus to be read and decoded just to get the useful 16 pixels. The overhead in that worst-case scenario is 16. The long 6-tap filters that perform sub-pixel interpolation also contribute to this phenomenon.

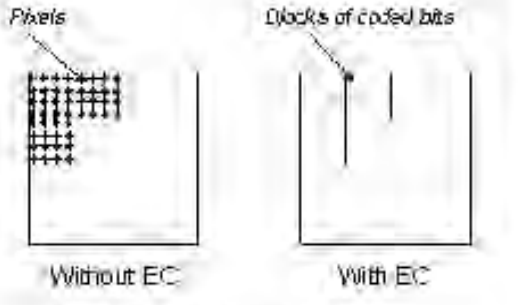


Figure 3. Block-oriented bandwidth

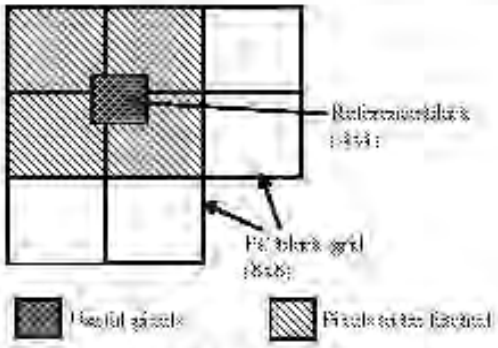


Figure 4. Worst-case example

Google Exhibit No. 1029
Page 004

Google Exhibit No. 1063
Page 1096

3. PROPOSED LOW POWER H.264 DECODER

We propose a low-power H.264 decoder based on embedded compression. Its high-level scheme is similar to the one depicted in Figure 2, and we impose some extra constraints to obtain an actual reduction of the memory transfers. Two techniques that fulfill these constraints are then detailed. Our solution takes benefits from both techniques via an efficient competitive mode.

3.1. Requirements for the embedded codec and consequences

To overcome overhead, we impose three main constraints on our EC algorithm:

1- the compression is block-based, and each block is independently encoded;
2- the compression ratio is fixed for all blocks and maps the memory structure;
3- luminance and chrominance planes are jointly stored in the memory.

First, having a block-based EC fits well with the structure of the standard decoder since it allows on-the-fly processing and avoids extra buffering. Furthermore, independent encoding of the blocks ensures that a limited amount of data must be fetched to recover some pixels pointed by the current motion vector. This excludes any kind of prediction between blocks, in particular spatial prediction, because one would have to load and decode also the previous blocks in the reference frame, which would annihilate the desired effect of memory transfer reduction.
The second constraint enables fast and easy access to any block in the reference frame. The start and end addresses of each encoded block are known, so that they do not have to be retrieved in an extra table in memory. The full benefit in terms of transfers and power comes when this constraint is coupled with a good data mapping strategy. In a typical SDRAM, memory addresses point to words of N bits (e.g. N=32), where N corresponds to the bus width. Moreover, data can be accessed by bursts of M consecutive words. Long bursts dissipate less energy than small ones relatively to the amount of transferred data. Generally, the maximum burst length is M=8 words (256 bits in the case of a 32-bit data bus). Hence, we set a fixed compression ratio CR so that the pixels of the source block fit in the length of one (or multiple) burst(s).
The third constraint increases data locality since motion vectors are identical for luminance and chrominance. Thus, pixels of the three planes can be accessed in one transfer, saving much energy in data-bus communications. We work with 4:2:0 progressive materials, so an 8x8 block in the pixel domain contains 512 bits of luminance and 256 bits of chrominance data. It exactly fits in one 8-word burst of 256 bits with the above memory structure if we apply a compression factor of 3.

While they clearly act in favor of a limited overhead effect, our first two constraints make the efficiency of our scheme difficult in terms of compression ratio and visual quality. Indeed, we cannot benefit from correlations between adjacent blocks. Furthermore, the fixed CR implies that too few bits might be used to correctly encode complex blocks while some easy blocks can certainly be losslessly encoded with less than 256 bits. On the contrary, the third constraint gives a double advantage since it decreases power through a reduction of the memory requests and also enables a joint bit-allocation between the Y, U and V components, which allows more flexibility for coding efficiency.

3.2. A competitive embedded compression scheme

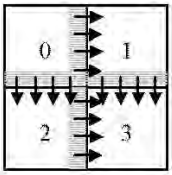
The above constraints limit the power dissipation due to memory accesses by imposing a block-based processing that fits well with the data structure of conventional memories and with the communication properties of the data bus. Once these constraints have been set, the choice of an embedded compression algorithm purely depends on coding efficiency. We have built a scheme that uses a competition between two complementary solutions through an *a priori* selection: a modified version of the Intra mode of H.264, and a zonal bit-plane coding scheme[15]. The former is known for its efficiency and is especially convenient to implement in an H.264 codec platform since the main functions are already available. The latter is bit-rate scalable and thus does not need any rate regulation module, resulting in an optimal use of the allocated bit budget.

Google Exhibit No. 1029
Page 005

Google Exhibit No. 1063
Page 1097

Modified H.264 Intra mode

In the H.264 standard, if a macroblock is encoded in Intra mode, a spatial prediction is formed based on the previously encoded blocks. 4x4 integer transform, quantization and entropy coding are then applied to the prediction error. In our case we cannot enable the complete prediction scheme, but a restrained version is possible. As a matter of fact we encode blocks of size 8x8 pixels where there are four 4x4 subblocks, so spatial predictions between these subblocks are performed. For instance the up-right subblock can be predicted from the pixels in the right column of the first subblock, either from their average value ("DC prediction") or by horizontal padding (Horizontal prediction"). Only the first subblock cannot at all be predicted from the other ones. Instead, we extract its DC coefficient truncated on 8 bits and use it as a predictor. The DC coefficient of the first subblock is thus encoded with a fair and constant precision, independently of the chosen quantization step. It helps limiting the drift effect. Figure 5 summarizes the available prediction modes for each subblock. One can refer to the H.264 standard for further information about each mode.



0: predicted by its own average value on 8 bits
1: DC prediction or horizontal prediction from subblock 0
2: DC prediction or vertical prediction from subblock 0
3: DC, horizontal, vertical, vertical-right, diagonal-right, horizontal-down, or horizontal-up prediction from subblocks 0, 1 and 2

Figure 5: Spatial prediction modes available for each 4x4 subblock in the modified H.264 Intra mode.

One bit is enough to transmit the prediction mode of subblock 1, and another one for subblock 2, while 3 bits are needed for subblock 3, leading to a total of 5 bits. Chrominance pixels directly form 4x4 blocks, so we do not perform any spatial prediction there, except that their average value truncated on 8 bits is also losslessly encoded for each UV component. Then the 4x4 transform and the quantization process are similar to the standard, but with a limited number of 32 possible quantization steps. The Level and Run tables are encoded using a variable-length coding (VLC). In fact, a complex entropy coding (e.g. the Context-based Adaptive Binary Arithmetic Coding - CABAC) is not justified in this application because the context would have to be reset after each block, leading to a sub-optimal solution. VLC is just as efficient here. QP is adjusted so that the total length of the bit-stream does not exceed the chosen value (in our case 256 bits or a burst of eight 32-bit words). Finally, the bit-stream corresponding to one block is composed as depicted in Figure 6.

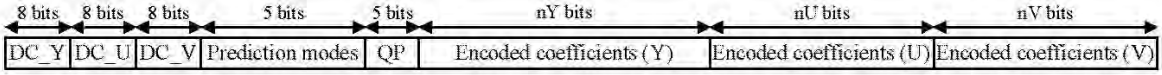


Figure 6: Bit-stream composition in the modified H.264 Intra mode.

Scalable Rmax/Cmax mode

The other scheme[16] that we have studied encodes DCT coefficients bit plane by bit plane, starting with the Most Significant Bits. As in the above solution, we encode the DC coefficients losslessly to counter the drift effect. For the AC coefficients, compression is achieved through a zonal coding: for each bit plane the significance map of the previously insignificant coefficients is represented by the maximum row Rmax and the maximum column Cmax where a bit is equal to 1 (see Figure 7). The data beyond this zone does not need to be transmitted. Since the energy usually lies in the low frequency coefficients, this technique implies a good bit-rate reduction. Its great advantage is to provide a fine-grain scalability: the process stops when the bit budget is exhausted. This property, although it generally comes at the expense of coding efficiency, is important in our context since we have imposed a severe rate limitation on a block basis. With the conventional quantization/VLC solution above, some bits are always wasted, while here all the available bits can efficiently be used.

Google Exhibit No. 1029
Page 006

Google Exhibit No. 1063
Page 1098