Video Streaming: Concepts, Algorithms, and Systems

John G. Apostolopoulos, Wai-tian Tan, Susie J. Wee
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2002-260
September 18th , 2002*

E-mail: [japos, dtan, swee}@hpl.hp.com

video streaming, video delivery, streaming media content delivery networks, video coding, error-resilient; multiple description coding

Video has been an important media for communications and entertainment for many decades. Initially video was captured and transmitted in analog form. The advent of digital integrated circuits and computers led to the digitization of video, and digital video enabled a revolution in the compression and communication of video. Video compression became an important area of research in the late 1980's and 1990's and enabled a variety of applications including video storage on DVD's and Video-CD's, video broadcast over digital cable, satellite and terrestrial (over-the-air) digital television (DTV), and video conferencing and videophone over circuit-switched networks. The growth and popularity of the Internet in the mid-1990's motivated video communication over best-effort packet networks. Video over best-effort packet networks is complicated by a number of factors including unknown and time-varying bandwidth, delay, and losses, as well as many additional issues such as how to fairly share the network resources amongst many flows and how to efficiently perform one-to-many communication for popular content. This article examines the challenges that make simultaneous delivery and playback, or streaming, of video difficult, and explores *algorithms and systems* that enable streaming of pre-encoded or live video over packet networks such as the Internet.

We continue by providing a brief overview of the diverse range of video streaming and communication applications. Understanding the different classes of video applications is important, as they provide different sets of constraints and degrees of freedom in system design. Section 3 reviews video compression and video compression standards. Section 4 identifies the three fundamental challenges in video streaming: unknown and time-varying bandwidth, delay jitter, and loss. These fundamental problems and approaches for overcoming them are examined in depth in Sections 5, 6, and 7. Standardized media streaming protocols are described in Section 8, and additional issues in video streaming are highlighted in Section 9. We conclude by describing the design of emerging streaming media content delivery networks in Section 10.

* Internal Accession Date Only
Approved for External Publication
© Copyright Hewlett-Packard Company 2002

Google
Exhibit 1034

1

VIDEO STREAMING: CONCEPTS, ALGORITHMS, AND SYSTEMS

John G. Apostolopoulos, Wai-tian Tan, Susie J. Wee
Streaming Media Systems Group
Hewlett-Packard Laboratories
Palo Alto, CA, USA
{japos,dtan,swee}@hpl.hp.com

1. INTRODUCTION

Video has been an important media for communications and entertainment for many decades. Initially video was captured and transmitted in analog form. The advent of digital integrated circuits and computers led to the digitization of video, and digital video enabled a revolution in the compression and communication of video. Video compression became an important area of research in the late 1980's and 1990's and enabled a variety of applications including video storage on DVD's and Video-CD's, video broadcast over digital cable, satellite and terrestrial (over-the-air) digital television (DTV), and video conferencing and videophone over circuit-switched networks. The growth and popularity of the Internet in the mid-1990's motivated video communication over best-effort packet networks. Video over best-effort packet networks is complicated by a number of factors including unknown and time-varying bandwidth, delay, and losses, as well as many additional issues such as how to fairly share the network resources amongst many flows and how to efficiently perform one-to-many communication for popular content. This article examines the challenges that make simultaneous delivery and playback, or streaming, of video difficult, and explores *algorithms and systems* that enable streaming of pre-encoded or live video over packet networks such as the Internet.

We continue by providing a brief overview of the diverse range of video streaming and communication applications. Understanding the different classes of video applications is important, as they provide different sets of constraints and degrees of freedom in system design. Section 3 reviews video compression and video compression standards. Section 4 identifies the three fundamental challenges in video streaming: unknown and time-varying bandwidth, delay jitter, and loss. These fundamental problems and

approaches for overcoming them are examined in depth in Sections 5, 6, and 7. Standardized media streaming protocols are described in Section 8, and additional issues in video streaming are highlighted in Section 9. We conclude by describing the design of emerging streaming media content delivery networks in Section 10. Further overview articles include [1,2,3,4,5].

2. OVERVIEW OF VIDEO STREAMING AND COMMUNICATION APPLICATIONS

There exist a very diverse range of different video communication and streaming applications, which have very different operating conditions or properties. For example, video communication application may be for point-to-point communication or for multicast or broadcast communication, and video may be pre-encoded (stored) or may be encoded in real-time (e.g. interactive videophone or video conferencing). The video channels for communication may also be static or dynamic, packet-switched or circuit-switched, may support a constant or variable bit rate transmission, and may support some form of Quality of Service (QoS) or may only provide best effort support. The specific properties of a video communication application strongly influence the design of the system. Therefore, we continue by briefly discussing some of these properties and their effects on video communication system design.

Point-to-point, multicast, and broadcast communications
Probably the most popular form of video communication is one-to-many (basically one-to-all) communication or broadcast communication, where the most well known example is broadcast television. Broadcast is a very efficient form of communication for popular content, as it can often efficiently deliver popular content to all receivers at the same time. An important aspect of broadcast communications is that the system must be designed to provide every intended recipient with the required signal. This is an important issue, since different recipients may experience different channel characteristics, and as a result the system is often designed for the worst-case channel. An example of this is digital television broadcast where the source coding and channel coding were designed to provide adequate reception to receivers at the fringe of the required reception area, thereby sacrificing some quality to those receivers in areas with higher quality reception (e.g. in the center of the city). An important characteristic of broadcast communication is that, due to the large number of receivers involved, feedback from receiver to sender is generally infeasible – limiting the system's ability to adapt.

Another common form of communication is point-to-point or one-to-one communication, e.g. videophone and unicast video streaming over the Internet. In point-to-point communications, an important property is whether or not there is a back channel between the receiver and sender. If a back channel exists, the receiver can provide feedback to the sender which the sender can then use to adapt its processing. On the other hand, without a back channel the sender has limited knowledge about the channel.

Another form of communication with properties that lie between point-to-point and broadcast is multicast. Multicast is a one-to-many communication, but it is not one-to-all as in broadcast. An example of multicast is IP-Multicast over the Internet. However, as discussed later, IP

Multicast is currently not widely available in the Internet, and other approaches are being developed to provide multicast capability, e.g. application-layer multicast via overlay networks. To communicate to multiple receivers, multicast is more efficient than multiple unicast connections (i.e. one dedicated unicast connection to each client), and overall multicast provides many of the same advantages and disadvantages as broadcast.

Real-time encoding versus pre-encoded (stored) video

Video may be captured and encoded for real-time communication, or it may be pre-encoded and stored for later viewing. Interactive applications are one example of applications which require real-time encoding, e.g. videophone, video conferencing, or interactive games. However real-time encoding may also be required in applications that are not interactive, e.g. the live broadcast of a sporting event.

In many applications video content is pre-encoded and stored for later viewing. The video may be stored locally or remotely. Examples of local storage include DVD and Video CD, and examples of remote storage include video-on-demand (VOD), and video streaming over the Internet (e.g. as provided by RealNetworks and Microsoft). Pre-encoded video has the advantage that it does not require a real-time encoding constraint. This can enable more efficient encoding such as the multi-pass encoding that is typically performed for DVD content. On the other hand, it provides limited flexibility as, for example, the pre-encoded video can not be significantly adapted to channels that support different bit rates or to clients that support different display capabilities than that used in the original encoding.

Interactive versus Non-interactive Applications

Interactive applications such as videophone or interactive games have a real-time constraint. Specifically the information has a time-bounded usefulness, and if the information arrives, but is late, it is useless. This is equivalent to a maximum acceptable end-to-end latency on the transmitted information, where by end-to-end we mean: capture, encode, transmission, receive, decode, display. The maximum acceptable latency depends on the application, but often is on the order of 150 ms. Non-interactive applications have looser latency constraints, for example many seconds or potentially even minutes. Examples of non-interactive applications include multicast of popular events or multicast of a lecture; these applications require timely delivery, but have a much looser latency constraint. Note that interactive applications require real-time encoding, and non-interactive applications may also require real-time encoding, however the end-to-end latency for non-interactive applications is much looser, and this has a dramatic effect on the design of video communication systems.

Static versus Dynamic Channels

Video communication system design varies significantly if the characteristics of the communication channel, such as bandwidth, delay, and loss, are static or dynamic (time-varying). Examples of static channels include ISDN (which provides a fixed bit rate and delay, and a very low loss rate) and video storage on a DVD. Examples of dynamic channels include communication over wireless channels or over the Internet. Video communication over a dynamic channel is much more difficult than over a static channel.

Furthermore, many of the challenges of video streaming, as are discussed later in this article, relate to the dynamic attributes of the channels.

Constant-bit-rate (CBR) or Variable-bit-rate (VBR) Channel
Some channels support CBR, for example ISDN or DTV, and some channels support VBR, for example DVD storage and communication over shared packet networks. On the other hand, a video sequence typically has time-varying complexity. Therefore coding a video to achieve a constant visual quality requires a variable bit rate, and coding for a constant bit rate would produce time-varying quality. Clearly, it is very important to match the video bit rate to what the channel can support. To achieve this a buffer is typically used to couple the video encoder to the channel, and a buffer control mechanism provides feedback based on the buffer fullness to regulate the coarseness/fineness of the quantization and thereby the video bit rate.

Packet-Switched or Circuit-Switched Network
A key network attribute that affects the design of media streaming systems is whether they are packet-switched or circuit-switched. Packet-switched networks, such as Ethernet LANs and the Internet, are shared networks where the individual packets of data may exhibit variable delay, may arrive out of order, or may be completely lost. Alternatively, circuit-switched networks, such as the public switched telephone network (PSTN) or ISDN, reserve resources and the data has a fixed delay, arrives in order, however the data may still be corrupted by bit errors or burst errors.

Quality of Service (QoS) Support
An important area of network research over the past two decades has been QoS support. QoS is a vague, and all-encompassing term, which is used to convey that the network provides some type of preferential delivery service or performance guarantees, e.g. guarantees on throughput, maximum loss rates or delay. Network QoS support can greatly facilitate video communication, as it can enable a number of capabilities including provisioning for video data, prioritizing delay-sensitive video data relative to other forms of data traffic, and also prioritize among the different forms of video data that must be communicated. Unfortunately, QoS is currently not widely supported in packet-switched networks such as the Internet. However, circuit-switched networks such as the PSTN or ISDN do provide various guarantees on delay, bandwidth, and loss rate. The current Internet does not provide any QoS support, and it is often referred to as Best Effort (BE), since the basic function is to provide simple network connectivity by best effort (without any guarantees) packet delivery . Different forms of network QoS that are under consideration for the Internet include Differentiated Services (DiffServ) and Integrated Services (IntServ), and these will be discussed further later in this writeup.

3. REVIEW OF VIDEO COMPRESSION

This section provides a very brief overview of video compression and video compression standards. The limited space precludes a detailed discussion, however we highlight some of the important principles and practices of current and emerging video compression algorithms and standards that are especially relevant for video communication and video streaming. An important motivation for this discussion is that both the standards (H.261/3/4, MPEG-1/2/4) and the most popular proprietary solutions (e.g.

RealNetworks [6] and Microsoft Windows Media [7]) are based on the same basic principles and practices, and therefore by understanding them one can gain a basic understanding for both standard and proprietary video streaming systems. Another goal of this section is to describe what are the different video compression standards, what do these standards actual specify, and which standards are most relevant for video streaming.

3.1 BRIEF OVERVIEW OF VIDEO COMPRESSION

Video compression is achieved by exploiting the similarities or redundancies that exists in a typical video signal. For example, consecutive frames in a video sequence exhibit temporal redundancy since they typically contain the same objects, perhaps undergoing some movement between frames. Within a single frame there is spatial redundancy as the amplitudes of nearby pixels are often correlated. Similarly, the Red, Green, and Blue color components of a given pixel are often correlated. Another goal of video compression is to reduce the irrelevancy in the video signal, that is to only code video features that are perceptually important and not to waste valuable bits on information that is not perceptually important or irrelevant. Identifying and reducing the redundancy in a video signal is relatively straightforward, however identifying what is perceptually relevant and what is not is very difficult and therefore irrelevancy is difficult to exploit.

To begin, we consider image compression, such as the JPEG standard, which is designed to exploit the spatial and color redundancy that exists in a single still image. Neighboring pixels in an image are often highly similar, and natural images often have most of their energies concentrated in the low frequencies. JPEG exploits these features by partitioning an image into 8x8 pixel blocks and computing the 2-D Discrete Cosine Transform (DCT) for each block. The motivation for splitting an image into small blocks is that the pixels within a small block are generally more similar to each other than the pixels within a larger block. The DCT compacts most of the signal energy in the block into only a small fraction of the DCT coefficients, where this small fraction of the coefficients are sufficient to reconstruct an accurate version of the image. Each 8x8 block of DCT coefficients is then quantized and processed using a number of techniques known as zigzag scanning, run-length coding, and Huffman coding to produce a compressed bitstream [8]. In the case of a color image, a color space conversion is first applied to convert the RGB image into a luminance/chrominance color space where the different human visual perception for the luminance (intensity) and chrominance characteristics of the image can be better exploited.

A video sequence consists of a sequence of video frames or images. Each frame may be coded as a separate image, for example by independently applying JPEG-like coding to each frame. However, since neighboring video frames are typically very similar much higher compression can be achieved by exploiting the similarity between frames. Currently, the most effective approach to exploit the similarity between frames is by coding a given frame by (1) first predicting it based on a previously coded frame, and then (2) coding the error in this prediction. Consecutive video frames typically contain the same imagery, however possibly at different spatial locations because of motion. Therefore, to improve the predictability it is important to estimate the motion between the frames and then to form an appropriate prediction that compensates for the motion. The process of estimating the

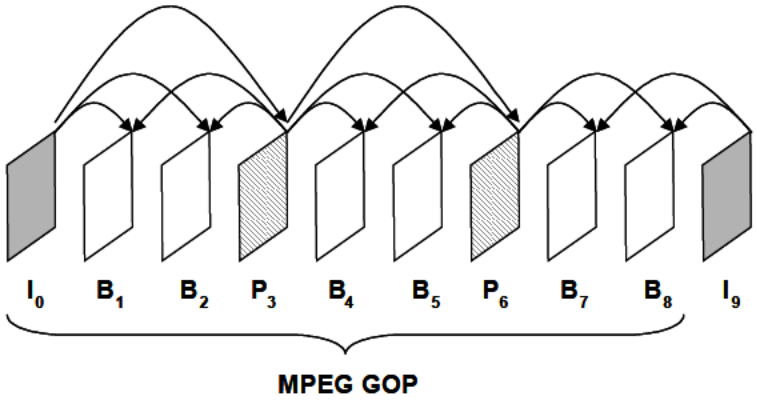

Figure 1: Example of the prediction dependencies between frames.

motion between frames is known as *motion estimation (ME)*, and the process of forming a prediction while compensating for the relative motion between two frames is referred to as *motion-compensated prediction (MC-P)*. Block-based ME and MC-prediction is currently the most popular form of ME and MC-prediction: the current frame to be coded is partitioned into 16x16-pixel blocks, and for each block a prediction is formed by finding the best-matching block in the previously coded reference frame. The relative motion for the best-matching block is referred to as the *motion vector*.

There are three basic common types of coded frames: (1) intra-coded frames, or I-frames, where the frames are coded independently of all other frames, (2) predictively coded, or P-frames, where the frame is coded based on a previously coded frame, and (3) bi-directionally predicted frames, or B-frames, where the frame is coded using both previous and future coded frames. Figure 1 illustrates the different coded frames and prediction dependencies for an example MPEG Group of Pictures (GOP). The selection of prediction dependencies between frames can have a significant effect on video streaming performance, e.g. in terms of compression efficiency and error resilience.

Current video compression standards achieve compression by applying the same basic principles [9, 10]. The temporal redundancy is exploited by applying MC-prediction, the spatial redundancy is exploited by applying the DCT, and the color space redundancy is exploited by a color space conversion. The resulting DCT coefficients are quantized, and the nonzero quantized DCT coefficients are runlength and Huffman coded to produce the compressed bitstream.

3.2 VIDEO COMPRESSION STANDARDS

Video compression standards provide a number of benefits, foremost of which is ensuring interoperability, or communication between encoders and decoders made by different people or different companies. In this way standards lower the risk for both consumer and manufacturer, and this can lead to quicker acceptance and widespread use. In addition, these standards are designed for a large variety of applications, and the resulting economies of scale lead to reduced cost and further widespread use.

Currently there are two families of video compression standards, performed under the auspices of the International Telecommunications Union-Telecommunications (ITU-T, formerly the International Telegraph and Telephone Consultative Committee, CCITT) and the International Organization for Standardization (ISO). The first video compression standard to gain widespread acceptance was the ITU H.261 [11], which was designed for videoconferencing over the integrated services digital network (ISDN). H.261 was adopted as a standard in 1990. It was designed to operate at p = 1,2, …, 30 multiples of the baseline ISDN data rate, or p x 64 kb/s. In 1993, the ITU-T initiated a standardization effort with the primary goal of videotelephony over the public switched telephone network (PSTN) (conventional analog telephone lines), where the total available data rate is only about 33.6 kb/s. The video compression portion of the standard is H.263 and its first phase was adopted in 1996 [12]. An enhanced H.263, H.263 Version 2 (V2), was finalized in 1997, and a completely new algorithm, originally referred to as H.26L, is currently being finalized as H.264/AVC.

The Moving Pictures Expert Group (MPEG) was established by the ISO in 1988 to develop a standard for compressing moving pictures (video) and associated audio on digital storage media (CD-ROM). The resulting standard, commonly known as MPEG-1, was finalized in 1991 and achieves approximately VHS quality video and audio at about 1.5 Mb/s [13]. A second phase of their work, commonly known as MPEG-2, was an extension of MPEG-1 developed for application toward digital television and for higher bit rates [14]. A third standard, to be called MPEG-3, was originally envisioned for higher bit rate applications such as HDTV, but it was recognized that those applications could also be addressed within the context of MPEG-2; hence those goals were wrapped into MPEG-2 (consequently, there is no MPEG-3 standard). Currently, the video portion of digital television (DTV) and high definition television (HDTV) standards for large portions of North America, Europe, and Asia is based on MPEG-2. A third phase of work, known as MPEG-4, was designed to provide improved compression efficiency and error resilience features, as well as increased functionality, including object-based processing, integration of both natural and synthetic (computer generated) content, content-based interactivity [15].

Table 1. Current and emerging video compression standards.

Video Coding Standard	Primary Intended Applications	Bit Rate
H.261	Video telephony and teleconferencing over ISDN	p x 64 kb/s
MPEG-1	Video on digital storage media (CD-ROM)	1.5 Mb/s
MPEG-2	Digital Television	2-20 Mb/s
H.263	Video telephony over PSTN	33.6 kb/s and up
MPEG-4	Object-based coding, synthetic content, interactivity, video streaming	Variable
H.264/MPEG-4 Part 10 (AVC)	Improved video compression	10's to 100's of kb/s

The H.26L standard is being finalized by the Joint Video Team, from both ITU and ISO MPEG. It achieves a significant improvement in compression over all prior video coding standards, and it will be adopted by both ITU and ISO and called H.264 and MPEG-4 Part 10, Advanced Video Coding (AVC).

Currently, the video compression standards that are primarily used for video communication and video streaming are H.263 V2, MPEG-4, and the emerging H.264/MPEG-4 Part 10 AVC will probably gain wide acceptance.

What Do The Standards Specify?

An important question is what is the scope of the video compression standards, or what do the standards actually specify. A video compression system is composed of an encoder and a decoder with a common interpretation for compressed bit-streams. The encoder takes original video and compresses it to a bitstream, which is passed to the decoder to produce the reconstructed video. One possibility is that the standard would specify both the encoder and decoder. However this approach turns out to be overly restrictive. Instead, the standards have a limited scope to ensure interoperability while enabling as much differentiation as possible.

The standards do not specify the encoder nor the decoder. Instead they specify the bitstream syntax and the decoding process. The bitstream syntax is the format for representing the compressed data. The decoding process is the set of rules for interpreting the bitstream. Note that specifying the decoding process is different from specifying a specific decoder implementation. For example, the standard may specify that the decoder use an IDCT, but not how to implement the IDCT. The IDCT may be implemented in a direct form, or using a fast algorithm similar to the FFT, or using MMX instructions. The specific implementation is not standardized and this allows different designers and manufacturers to provide standard-compatible enhancements and thereby differentiate their work.

The encoder process is deliberately not standardized. For example, more sophisticated encoders can be designed that provide improved performance over baseline low-complexity encoders. In addition, improvements can be incorporated even after a standard is finalized, e.g. improved algorithms for motion estimation or bit allocation may be incorporated in a standard-compatible manner. The only constraint is that the encoder produces a syntactically correct bitstream that can be properly decoded by a standard-compatible decoder.

Limiting the scope of standardization to the bitstream syntax and decoding process enables improved encoding and decoding strategies to be employed in a standard-compatible manner - thereby ensuring interoperability while enabling manufacturers to differentiate themselves. As a result, it is important to remember that “not all encoders are created equal”, even if they correspond to the same standard.

4. CHALLENGES IN VIDEO STREAMING

This section discusses some of the basic approaches and key challenges in video streaming. The three fundamental problems in video streaming are briefly highlighted and are examined in depth in the following three sections.

Video Delivery via File Download

Probably the most straightforward approach for video delivery of the Internet is by something similar to a file download, but we refer to it as video download to keep in mind that it is a video and not a generic file. Specifically, video download is similar to a file download, but it is a LARGE file. This approach allows the use of established delivery mechanisms, for example TCP as the transport layer or FTP or HTTP at the higher layers. However, it has a number of disadvantages. Since videos generally correspond to very large files, the download approach usually requires long download times and large storage spaces. These are important practical constraints. In addition, the entire video must be downloaded before viewing can begin. This requires patience on the viewers part and also reduces flexibility in certain circumstances, e.g. if the viewer is unsure of whether he/she wants to view the video, he/she must still download the entire video before viewing it and making a decision.

Video Delivery via Streaming

Video delivery by video streaming attempts to overcome the problems associated with file download, and also provides a significant amount of additional capabilities. The basic idea of video streaming is to split the video into parts, transmit these parts in succession, and enable the receiver to decode and playback the video as these parts are received, without having to wait for the entire video to be delivered. Video streaming can conceptually be thought to consist of the follow steps:

1) Partition the compressed video into packets
2) Start delivery of these packets
3) Begin decoding and playback at the receiver while the video is still being delivered

Video streaming enables *simultaneous delivery and playback* of the video. This is in contrast to file download where the entire video must be delivered before playback can begin. In video streaming there usually is a short delay (usually on the order of 5-15 seconds) between the start of delivery and the beginning of playback at the client. This delay, referred to as the pre-roll delay, provides a number of benefits which are discussed in Section 6.

Video streaming provides a number of benefits including low delay before viewing starts and low storage requirements since only a small portion of the video is stored at the client at any point in time. The length of the delay is given by the time duration of the pre-roll buffer, and the required storage is approximately given by the amount of data in the pre-roll buffer.

Expressing Video Streaming as a Sequence of Constraints

A significant amount of insight can be obtained by expressing the problem of video streaming as a *sequence of constraints*. Consider the time interval between displayed frames to be denoted by Δ, e.g. Δ is 33 ms for 30 frames/s video and 100 ms for 10 frames/s video. Each frame must be delivered and decoded by its playback time, therefore the sequence of frames has an associated sequence of deliver/decode/display deadlines:

- Frame N must be delivered and decoded by time T_N
- Frame N+1 must be delivered and decoded by time $T_N + \Delta$
- Frame N+2 must be delivered and decoded by time $T_N + 2\Delta$
- Etc.

Any data that is lost in transmission cannot be used at the receiver. Furthermore, any data that arrives late is also useless. Specifically, any data that arrives after its decoding and display deadline is too late to be displayed. (Note that certain data may still be useful even if it arrives after its display time, for example if subsequent data depends on this "late" data.) Therefore, an important goal of video streaming is to perform the streaming in a manner so that this sequence of constraints is met.

4.1 Basic Problems in Video Streaming

There are a number of basic problems that afflict video streaming. In the following discussion, we focus on the case of video streaming over the Internet since it is an important, concrete example that helps to illustrate these problems. Video streaming over the Internet is difficult because the Internet only offers best effort service. That is, it provides no guarantees on bandwidth, delay jitter, or loss rate. Specifically, these characteristics are unknown and dynamic. Therefore, a key goal of video streaming is to design a system to reliably deliver high-quality video over the Internet when dealing with unknown and dynamic:

- Bandwidth
- Delay jitter
- Loss rate

The bandwidth available between two points in the Internet is generally unknown and time-varying. If the sender transmits faster than the available bandwidth then congestion occurs, packets are lost, and there is a severe drop in video quality. If the sender transmits slower than the available bandwidth then the receiver produces sub-optimal video quality. The goal to overcome the bandwidth problem is to estimate the available bandwidth and than match the transmitted video bit rate to the available bandwidth. Additional considerations that make the bandwidth problem very challenging include accurately estimating the available bandwidth, matching the pre-encoded video to the estimated channel bandwidth, transmitting at a rate that is fair to other concurrent flows in the Internet, and solving this problem in a multicast situation where a single sender streams data to multiple receivers where each may have a different available bandwidth.

The end-to-end delay that a packet experiences may fluctuate from packet to packet. This variation in end-to-end delay is referred to as the delay jitter. Delay jitter is a problem because the receiver must receive/decode/display frames at a constant rate, and any late frames resulting from the delay jitter can produce problems in the reconstructed video, e.g. jerks in the video. This problem is typically addressed by including a playout buffer at the receiver. While the playout buffer can compensate for the delay jitter, it also introduces additional delay.

The third fundamental problem is losses. A number of different types of losses may occur, depending on the particular network under consideration. For example, wired packet networks such as the Internet are afflicted by packet loss, where an entire packet is erased (lost). On the other hand, wireless channels are typically afflicted by bit errors or burst errors. Losses can have a very destructive effect on the reconstructed video quality. To combat the effect of losses, a video streaming system is designed with error control. Approaches for error control can be roughly grouped into four

classes: (1) forward error correction (FEC), (2) retransmissions, (3) error concealment, and (4) error-resilient video coding.

The three fundamental problems of unknown and dynamic *bandwidth, delay jitter, and loss,* are considered in more depth in the following three sections. Each section focuses on one of these problems and discusses various approaches for overcoming it.

5. TRANSPORT AND RATE CONTROL FOR OVERCOMING TIME-VARYING BANDWIDTHS

This section begins by discussing the need for streaming media systems to adaptively control its transmission rate according to prevalent network condition. We then discuss some ways in which appropriate transmission rates can be estimated dynamically at the time of streaming, and survey how media coding has evolved to support such dynamic changes in transmission rates.

5.1 The Need for Rate Control

Congestion is a common phenomenon in communication networks that occurs when the offered load exceeds the designed limit, causing degradation in network performance such as throughput. Useful throughput can decrease for a number of reasons. For example, it can be caused by collisions in multiple access networks, or by increased number of retransmissions in systems employing such technology. Besides a decrease in useful throughput, other symptoms of congestion in packet networks may include packet losses, higher delay and delay jitter. As we have discussed in Section 4, such symptoms represent significant challenges to streaming media systems. In particular, packet losses are notoriously difficult to handle, and is the subject of the Section 7.

To avoid the undesirable symptoms of congestion, control procedures are often employed to limit the amount of network load. Such control procedures are called rate control, sometimes also known as congestion control. It should be noted that different network technologies may implement rate control in different levels, such as hop-to-hop level or network level [16]. Nevertheless, for inter-networks involving multiple networking technologies, it is common to rely on rate control performed by the end-hosts. The rest of this section examines rate control mechanisms performed by the sources or sinks of streaming media sessions.

5.2 Rate Control for Streaming Media

For environments like the Internet where little can be assumed about the network topology and load, determining an appropriate transmission rate can be difficult. Nevertheless, the rate control mechanism implemented in the Transmission Control Protocol (TCP) has been empirically proven to be sufficient in most cases. Being the dominant traffic type in the Internet, TCP is the workhorse in the delivery of web-pages, emails, and some streaming media. Rate control in TCP is based on a simple "Additive Increase Multiplicative Decrease" (AIMD) rule [17]. Specifically, end-to-end observations are used to infer packet losses or congestion. When no congestion is inferred, packet transmission is increased at a constant rate

(additive increase). Conversely, when congestion is inferred, packet transmission rate is halved (multiplicative decrease).

Streaming Media over TCP

Given the success and ubiquity of TCP, it may seem natural to employ TCP for streaming media. There are indeed a number of important advantages of using TCP. First, TCP rate control has empirically proven stability and scalability. Second, TCP provides guaranteed delivery, effectively eliminating the much dreaded packet losses. Therefore, it may come as a surprise to realize that streaming media today are often carried using TCP only as a last resort, e.g., to get around firewalls. Practical difficulties with using TCP for streaming media include the following. First, delivery guarantee of TCP is accomplished through persistent retransmission with potentially increasing wait time between consecutive retransmissions, giving rise to potentially very long delivery time. Second, the "Additive Increase Multiplicative Decrease" rule gives rise to a widely varying instantaneous throughput profile in the form of a saw-tooth pattern not suitable for streaming media transport.

Streaming Media over Rate-controlled UDP

We have seen that both the retransmission and the rate control mechanisms of TCP possess characteristics that are not suitable for streaming media. Current streaming systems for the Internet rely instead on the best-effort delivery service in the form of User Datagram Protocol (UDP). This allows more flexibility both in terms of error control and rate control. For instance, instead of relying on retransmissions alone, other error control techniques can be incorporated or substituted. For rate control, the departure from the AIMD algorithm of TCP is a mixed blessing: it promises the end of wildly varying instantaneous throughput, but also the proven TCP stability and scalability.

Recently, it has been observed that the *average* throughput of TCP can be inferred from end-to-end measurements of observed quantities such as round-trip-time and packet losses [18,19]. Such observation gives rise to TCP-friendly rate control that attempts to mimic TCP throughput on a macroscopic scale and without the instantaneous fluctuations of TCP's AIMD algorithm [20,21]. One often cited importance of TCP-friendly rate control is its ability to coexist with other TCP-based applications. Another benefit though, is more predictable stability and scalability properties compared to an arbitrary rate control algorithm. Nevertheless, by attempting to mimic average TCP throughput under the same network conditions, TCP friendly rate control also inherits characteristics that may not be natural for streaming media. One example is the dependence of transmission rate on packet round-trip time.

Other Special Cases

Some media streaming systems do not perform rate control. Instead, media content is transmitted without regard to the prevalent network condition. This can happen in scenarios where an appropriate transmission rate is itself difficult to define, e.g., one-to-many communication where an identical stream is transmitted to all recipients via channels of different levels of congestion. Another possible reason is the lack of a feedback channel.

Until now we have only considered rate control mechanisms that are implemented at the sender, now we consider an example where rate control is performed at the receiver. In the last decade, a scheme known as *layered multicast* has been proposed as a possible way to achieve rate control in Internet multicast of streaming media. Specifically, a scalable or layered compression scheme is assumed that produces multiple layers of compressed media, with a base layer that offers low but usable quality, and each additional layer provides further refinement to the quality. Each *receiver* can then individually decide how many layers to receive [22]. In other words, rate control is performed at the receiving end instead of the transmitting end. Multicast rate control is still an area of active research.

5.3 Meeting Transmission Bandwidth Constraints

The incorporation of rate control introduces additional complexity in streaming media system. Since transmission rate is dictated by channel conditions, problems may arise if the determined transmission rate is lower than the media bit rate. Client buffering helps to a certain degree to overcome occasional short-term drops in transmission rate. Nevertheless, it is not possible to stream a long 200 *kbps* stream through a 100 *kbps* channel, and the media bit rate needs to be modified to conform with the transmission constraints.

Transcoding

A direct method to modify the media bit rate is recompression, whereby the media is decoded and then re-encoded to the desired bit rate. There are two drawbacks with this approach. First, the media resulted from recompression is generally of lower quality than if the media was coded directly from the original source to the same bit rate. Second, media encoding generally requires extensive computation, making the approach prohibitively expensive. The complexity problem is solved by a technique known as compressed-domain transcoding. The basic idea is to selectively re-use compression decisions already made in the compressed media to reduce computation. Important transcoding operations include bit rate reduction, spatial downsampling, frame rate reduction, and changing compression formats [23].

Multiple File Switching

Another commonly used technique is multi-rate switching whereby multiple copies of the same content at different bit-rates are made available. Early implementations of streaming media systems coded the same content at a few strategic media rates targeted for common connection speeds (e.g. one for dialup modem and one for DSL/cable) and allowed the client to choose the appropriate media rate at the beginning of the session. However, these early systems only allowed the media rate to be chosen once at the beginning of each session. In contrast, multi-rate switching enables dynamic switching between different media rates within a single streaming media session. This mid-session switching between different media rates enables better adaptation to longer-term fluctuations in available bandwidth than can be achieved by the use of the client buffer alone. Examples include Intelligent Streaming from Microsoft and SureStream from Real Networks.

This approach overcomes both limitations of transcoding, as very little computation is needed for switching between the different copies of the

stream, and no recompression penalty is incurred. However, there are a number of disadvantages. First, the need to store multiple copies of the same media incurs higher storage cost. Second, for practical implementation, only a small number of copies are used, limiting its ability to adapt to varying transmission rates.

Scalable Compression

A more elegant approach to adapt to longer-term bandwidth fluctuations is to use layered or scalable compression. This is similar in spirit to multi-rate switching, but instead of producing multiple copies of the same content at different bit rates, layered compression produces a set of (ordered) bitstreams (sometimes referred to as layers) and different subsets of these bitstreams can be selected to represent the media at different target bit rates [20]. Many commonly used compression standards, such as MPEG-2, MPEG-4 and H.263 have extensions for layered coding. Nevertheless, layered or scalable approaches are not widely used because they incur a significant compression penalty as compared to non-layered/non-scalable approaches.

5.4 Evolving Approaches

Rate control at end-hosts avoids congestion by dynamically adapting the transmission rate. Alternatively, congestion can also be avoided by providing unchanging amount of resources to each flow, but instead limiting the addition of new flows. This is similar to the telephone system that provides performance guarantees although with a possibility for call blocking.

With all the difficulties facing streaming media systems in the Internet, there has been work towards providing some Quality of Service (QoS) support in the Internet. The Integrated Services (IntServ) model of the Internet [24], for instance, is an attempt to provide end-to-end QoS guarantees in terms of bandwidth, packet loss rate, and delay, on a per-flow basis. QoS guarantees are established using explicit resource allocation based on the Resource Reservation Protocol (RSVP). The guarantees in terms of bandwidth and packet loss rate would have greatly simplified streaming media systems. Nevertheless, this is only at the expense of additional complexity in the network. The high complexity and cost of deployment of the RSVP-based service architecture eventually led the IETF to consider other QoS mechanisms. The Differentiated Services (DiffServ) model, in particular, is specifically designed to achieve low complexity and easy deployment at the cost of less stringent QoS guarantees than IntServ. Under DiffServ, service differentiation is no longer provided on a per-flow basis. Instead, it is based on the *code-point* or tag in each packet. Thus, packets having the same tags are given the same treatment under DiffServ regardless of where they originate. The cost of easy deployment for DiffServ compared to IntServ is the reduced level of QoS support. Specific ways in which streaming media systems can take advantage of a DiffServ Internet is currently an area of active research.

6. PLAYOUT BUFFER FOR OVERCOMING DELAY JITTER

It is common for streaming media clients to have a 5 to 15 second buffering before playback starts. As we have seen in Section 4, streaming can be viewed as a sequence of constraints for individual media samples. The use of

buffering essentially relaxes all the constraints by an identical amount. Critical to the performance of streaming systems over best-effort networks such as the Internet, buffering provides a number of important advantages:

1. Jitter reduction: Variations in network conditions cause the time it takes for packets to travel between identical end-hosts to vary. Such variations can be due to a number of possible causes, including queuing delays and link-level retransmissions. Jitter can cause jerkiness in playback due to the failure of same samples to meet their presentation deadlines, and have to be therefore skipped or delayed. The use of buffering effectively extends the presentation deadlines for all media samples, and in most cases, practically eliminates playback jerkiness due to delay jitter. The benefits of a playback buffer are illustrated in Figure 2, where packets are transmitted and played at a constant rate, and the playback buffer reduces the number of packets that arrive after their playback deadline.
2. Error recovery through retransmissions: The extended presentation deadlines for the media samples allow retransmission to take place when packets are lost, e.g., when UDP is used in place of TCP for transport. Since compressed media streams are often sensitive to errors, the ability to recover losses greatly improves streaming media quality.
3. Error resilience through Interleaving: Losses in some media streams, especially audio, can often be better *concealed* if the losses are isolated instead of concentrated. The extended presentation deadlines with the use of buffering allow interleaving to transform possible burst loss in the channel into isolated losses, thereby enhancing the concealment of the subsequent losses. As we shall discuss in the next section, the extended deadlines also allow other forms of error control schemes such as the use of error control codes, which are particularly effective when used with interleaving.
4. Smoothing throughput fluctuation: Since time varying channel gives rise to time varying throughput, the buffer can provide needed data to sustain streaming when throughput is low. This is especially important when streaming is performed using TCP (or HTTP), since the server typically does not react to a drop in channel throughput by reducing media rate.

The benefits of buffering do come at a price though. Besides additional storage requirements at the streaming client, buffering also introduces additional delay before playback can begin or resume (after a pause due to buffer depletion). Adaptive Media Playout (AMP) is a new technique that enables an valuable tradeoff between delay and reliability [25,4].

7. ERROR CONTROL FOR OVERCOMING CHANNEL LOSSES

The third fundamental problem that afflicts video communication is losses. Losses can have a very destructive effect on the reconstructed video quality, and if the system is not designed to handle losses, even a single bit error can have a catastrophic effect. A number of different types of losses may occur, depending on the particular network under consideration. For example, wired packet networks such as the Internet are afflicted by packet loss, where congestion may cause an entire packet to be discarded (lost). In this

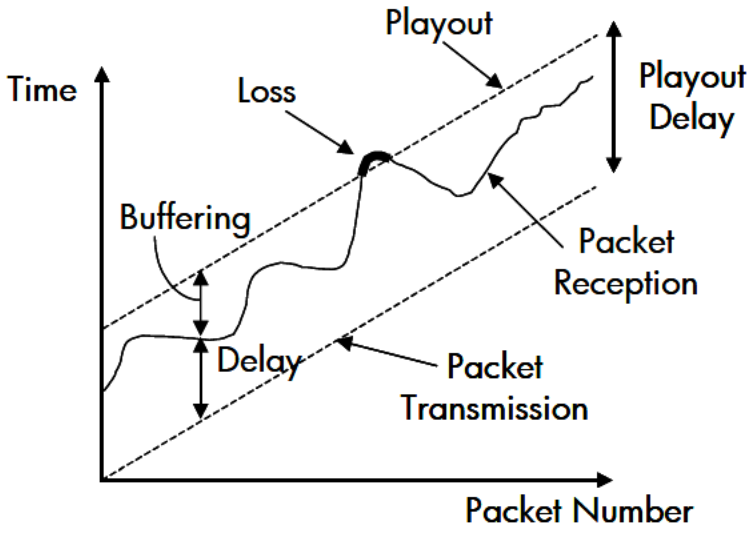

Figure 2: Effect of playout buffer on reducing the number of late packets.

case the receiver will either completely receive a packet in its entirety or completely lose a packet. On the other hand, wireless channels are typically afflicted by bit errors or burst errors at the physical layer. These errors may be passed up from the physical layer to the application as bit or burst errors, or alternatively, entire packets may be discarded when any errors are detected in these packets. Therefore, depending on the interlayer communication, a video decoder may expect to always receive "clean" packets (without any errors) or it may receive "dirty" packets (with errors). The loss rate can vary widely depending on the particular network, and also for a given network depending on the amount of cross traffic. For example, for video streaming over the Internet one may see a packet loss rate of less than 1 %, or sometimes greater than 5-10 %.

A video streaming system is designed with error control to combat the effect of losses. There are four rough classes of approaches for error control: (1) retransmissions, (2) forward error correction (FEC), (3) error concealment, and (4) error-resilient video coding. The first two classes of approaches can be thought of as channel coding approaches for error control, while the last two are source coding approaches for error control. These four classes of approaches are discussed in the following four subsections. A video streaming system is typically designed using a number of these different approaches. In addition, joint design of the source coding and channel coding is very important and this is discussed in Section 7.5. Additional information, and specifics for H.263 and MPEG-4, are available in [26,27,28].

7.1 RETRANSMISSIONS

In retransmission-based approaches the receiver uses a back-channel to notify the sender which packets were correctly received and which were not, and this enables the sender to resend the lost packets. This approach efficiently uses the available bandwidth, in the sense that only lost packets are resent, and it also easily adapts to changing channel conditions. However, it also has some disadvantages. Retransmission leads to additional delay corresponding roughly to the round-trip-time (RTT) between receiver-sender-receiver. In addition, retransmission requires a back-channel, and

this may not be possible or practical in various applications such as broadcast, multicast, or point-to-point without a back-channel.

In many applications the additional delay incurred from using retransmission is acceptable, e.g. Web browsing, FTP, telnet. In these cases, when guaranteed delivery is required (and a backchannel is available) then feedback-based retransmits provide a powerful solution to channel losses. On the other hand, when a back channel is not available or the additional delay is not acceptable, then retransmission is not an appropriate solution.

There exist a number of important variations on retransmission-based schemes. For example, for video streaming of time-sensitive data one may use delay-constrained retransmission where packets are only retransmitted if they can arrive by their time deadline, or priority-based retransmission, where more important packets are retransmitted before less important packets. These ideas lead to interesting scheduling problems, such as which packet should be transmitted next (e.g. [29,4]).

7.2 FORWARD ERROR CORRECTION

The goal of FEC is to add specialized redundancy that can be used to recover from errors. For example, to overcome packet losses in a packet network one typically uses block codes (e.g. Reed Solomon or Tornado codes) that take K data packets and output N packets, where N-K of the packets are redundant packets. For certain codes, as long as *any* K of the N packets are correctly received the original data can be recovered. On the other hand, the added redundancy increases the required bandwidth by a factor of N/K. FEC provides a number of advantages and disadvantages. Compared to retransmissions, FEC does not require a back-channel and may provide lower delay since it does not depend on the round-trip-time of retransmits. Disadvantages of FEC include the overhead for FEC even when there are no losses, and possible latency associated with reconstruction of lost packets. Most importantly, FEC-based approaches are designed to overcome a predetermined amount of loss and they are quite effective *if* they are appropriately matched to the channel. If the losses are less than a threshold, then the transmitted data can be perfectly recovered from the received, lossy data. However, if the losses are greater than the threshold, then only a portion of the data can be recovered, and depending on the type of FEC used, the data may be completely lost. Unfortunately the loss characteristics for packet networks are often unknown and time varying. Therefore the FEC may be poorly matched to the channel -- making it ineffective (too little FEC) or inefficient (too much FEC).

7.3 ERROR CONCEALMENT

Transmission errors may result in lost information. The basic goal of error concealment is to estimate the lost information or missing pixels in order to conceal the fact that an error has occurred. The key observation is that video exhibits a significant amount of correlation along the spatial and temporal dimensions. This correlation was used to achieve video compression, and unexploited correlaton can also be used to estimate the lost information. Therefore, the basic approach in error concealment is to exploit the correlation by performing some form of spatial and/or temporal interpolation (or extrapolation) to estimate the lost information from the correctly received data.

To illustrate the basic idea of error concealment, consider the case where a single 16x16 block of pixels (a macroblock in MPEG terminology) is lost. This example is not representative of the information typically lost in video streaming, however it is a very useful example for conveying the basic concepts. The missing block of pixels may be assumed to have zero amplitude, however this produces a black/green square in the middle of the video frame which would be highly distracting. Three general approaches for error concealment are: (1) spatial interpolation, (2) temporal interpolation (freeze frame), and (3) motion-compensated temporal interpolation. The goal of spatial interpolation is to estimate the missing pixels by smoothly extrapolating the surrounding correctly received pixels. Correctly recovering the missing pixels is extremely difficult, however even correctly estimating the DC (average) value is very helpful, and provides significantly better concealment than assuming the missing pixels have an amplitude of zero. The goal of temporal extrapolation is to estimate the missing pixels by copying the pixels at the same spatial location in the previous correctly decoded frame (freeze frame). This approach is very effective when there is little motion, but problems arise when there is significant motion. The goal of motion-compensated temporal extrapolation is to estimate the missing block of pixels as a motion compensated block from the previous correctly decoded frame, and thereby hopefully overcome the problems that arise from motion. The key problem in this approach is how to accurately estimate the motion for the missing pixels. Possible approach include using the coded motion vector for that block (if it is available), use a neighboring motion vector as an estimate of the missing motion vector, or compute a new motion vector by leveraging the correctly received pixels surrounding the missing pixels.

Various error concealment algorithms have been developed that apply different combinations of spatial and/or temporal interpolation. Generally, a motion-compensated algorithm usually provides the best concealment (assuming an accurate motion vector estimate). This problem can also be formulated as a signal recovery or inverse problem, leading to the design of sophisticated algorithms (typically iterative algorithms) that provide improved error concealment in many cases.

The above example where a 16x16 block of pixels is lost illustrates many of the basic ideas of error concealment. However, it is important to note that errors typically lead to the loss of much more than a single 16x16 block. For example, a packet loss may lead to the loss of a significant fraction of an entire frame, or for low-resolution video (e.g. 176x144 pixels/frame) an entire coded frame may fit into a single packet, in which case the loss of the packet leads to the loss of the entire frame. When an entire frame is lost, it is not possible to perform any form of spatial interpolation as there is no spatial information available (all of the pixels in the frame are lost), and therefore only temporal information can be used for estimating the lost frame. Generally, the lost frame is estimated as the last correctly received frame (freeze frame) since this approach typically leads to the fewest artifacts.

A key point about error concealment is that it is performed at the decoder. As a result, error concealment is outside the scope of video compression standards. Specifically, as improved error concealment algorithms are

developed they can be incorporated as standard-compatible enhancements to conventional decoders.

7.4 ERROR RESILIENT VIDEO CODING

Compressed video is highly vulnerable to errors. The goal of error-resilient video coding is to design the video compression algorithm and the compressed bitstream so that it is resilient to specific types of errors. This section provides an overview of error-resilient video compression. It begins by identifying the basic problems introduced by errors and then discusses the approaches developed to overcome these problems. In addition, we focus on which problems are most relevant for video streaming and also which approaches to overcome these problems are most successful and when. In addition, scalable video coding and multiple description video coding are examined as possible approaches for providing error resilient video coding.

7.4.1 Basic problems introduced by errors

Most video compression systems possess a similar architecture based on motion-compensated (MC) prediction between frames, Block-DCT (or other spatial transform) of the prediction error, followed by entropy coding (e.g. runlength and Huffman coding) of the parameters. The two basic error-induced problems that afflict a system based on this architecture are:

1) Loss of bitstream synchronization
2) Incorrect state and error propagation

The first class of problems, loss of bitstream synchronization, refers to the case when an error can cause the decoder to become confused and lose synchronization with the bitstream, i.e. the decoder may loss track of what bits correspond to what parameters. The second class of problems, incorrect state and error propagation, refers to what happens when a loss afflicts a system that uses predictive coding.

7.4.2 Overcoming Loss of Bitstream Synchronization

Loss of bitstream synchonization corresponds to the case when an error causes the decoder to loss track of what bits correspond to what parameters. For example, consider what happens when a bit error afflicts a Huffman codeword or other variable length codeword (VLC). Not only would the codeword be incorrectly decoded by the decoder, but because of the variable length nature of the codewords it is highly probably that the codeword would be incorrectly decoded to a codeword of a different length, and thereby all the subsequent bits in the bitstream (until the next resync) will be misinterpreted. Even a *single* bit error can lead to significant subsequent loss of information.

It is interesting to note that fixed length codes (FLC) do not have this problem, since the beginning and ending locations of each codeword are known, and therefore losses are limited to a single codeword. However, FLC's do not provide good compression. VLC's provide significantly better compression and therefore are widely used.

The key to overcoming the problem of loss of bitstream synchronization is to provide mechanisms that enable the decoder to quickly isolate the problem

and resynchronize to the bitstream after an error has occurred. We now consider a number of mechanisms that enable bitstream resynchonization.

Resync Markers

Possibly the simplest approach to enable bitstream resynchonization is by the use of resynchronization markers, commonly referred to as resync markers. The basic idea is to place unique and easy to find entry points in the bitstream, so that if the decoder losses sync, it can look for the next entry point and then begin decoding again after the entry point. Resync markers correspond to a bitstream pattern that the decoder can unmistakably find. These markers are designed to be distinct from all codewords, concatenations of codewords, and minor perturbations of concatenated codewords. An example of a resync marker is the three-byte sequence consisting of 23 zeros followed by a one. Sufficient information is typically included after each resync marker to enable the restart of bitstream decoding.

An important question is where to place the resync markers. One approach is to place the resync markers at strategic locations in the compressed video hierarchy, e.g. picture or slice headers. This approach is used in MPEG-1/2 and H.261/3. This approach results in resyncs being placed every fixed number of blocks, which corresponds to a variable number of bits. An undesired consequence of this is that active areas, which require more bits, would in many cases have a higher probability of being corrupted. To overcome this problem, MPEG-4 provides the capability to place the resync markers periodically after every fixed number of bits (and variable number of coding blocks). This approach provides a number of benefits including reduces the probability that active areas be corrupted, simplifies the search for resync markers, and supports application-aware packetization.

Reversible Variable Length Codes (RVLCs)

Conventional VLC's, such as Huffman codes, are uniquely decodeable in the forward direction. RVLCs in addition have the property that they are also uniquely decodable in the backward direction. This property can be quite beneficial in recovering data that would otherwise be lost. For example, if an error is detected in the bitstream, the decoder would typically jump to the next resync marker. Now, if RVLCs are used, instead of discarding all the data between the error and the resync, the decoder can start decoding backwards from the resync until it identifies another error, and thereby enables partial recovery of data (which would otherwise be discarded). Nevertheless, RVLC are typically less efficient than VLC.

Data Partitioning

An important observation is that bits which closely follow a resync marker are more likely to be accurately decoded than those further away. This motivates the idea of placing the most important information immediately after each resync (e.g. motion vectors, DC DCT coefficients, and shape information for MPEG-4) and placing the less important information later (AC DCT coefficients). This approach is referred to as data partitioning in MPEG-4. Note that this approach is in contrast with the conventional approach used in MPEG-1/2 and H.261/3, where the data is ordered in the bitstream in a consecutive macroblock by macroblock manner, and without accounting for the importance of the different types of data.

To summarize, the basic idea to overcome the problem of loss of bitstream synchronization is to first isolate (localize) the corrupted information, and second enable a fast resynchronization.

Application-Aware Packetization: Application Level Framing (ALF)
Many applications involve communication over a packet network, such as the Internet, and in these cases the losses have an important structure that can be exploited. Specifically, either a packet is accurately received in its entirety or it is completely lost. This means that the boundaries for lost information are exactly determined by the packet boundaries. This motivates the idea that to combat packet loss, one should *design (frame) the packet payload to minimize the effect of loss*. This idea was crystallized in the Application Level Framing (ALF) principle presented in [30], who basically said that the "application knows best" how to handle packet loss, out-of-order delivery, and delay, and therefore the application should design the packet payloads and related processing. For example, if the video encoder knows the packet size for the network, it can design the packet payloads so that each packet is independently decodable, i.e. bitstream resynchronization is supported at the packet level so that each correctly received packet can be straightforwardly parsed and decoded. MPEG-4, H.263V2, and H.264/MPEG-4 AVC support the creation of different forms of independently decodable *video packets*. As a result, the careful usage of the application level framing principle can often overcome the bitstream synchronization problem. Therefore, the major obstacle for reliable video streaming over lossy packet networks such as the Internet, is the error propagation problem, which is discussed next.

7.4.3 Overcoming Incorrect State and Error Propagation

If a loss has occurred, and even if the bitstream has been resynchronized, another crucial problem is that the state of the representation at the decoder may be different from the state at the encoder. In particular, when using MC-prediction an error causes the reconstructed frame (state) at the decoder to be incorrect. The decoder's state is then different from the encoder's, leading to incorrect (mismatched) predictions and often significant error propagation that can afflict many subsequent frames, as illustrated in Figure 3. We refer to this problem as having incorrect (or mismatched) state at the decoder, because the state of the representation at the decoder (the previous coded frame) is not the same as the state at the encoder. This problem also arises in other contexts (e.g. random access for DVD's or channel acquisition for Digital TV) where a decoder attempts to decode beginning at an arbitrary position in the bitstream.

A number of approaches have been developed over the years to overcome this problem, where these approaches have the common goal of trying to limit the effect of error propagation. The simplest approach to overcome this problem is by using I-frame only. Clearly by not using any temporal prediction, this approach avoids the error propagation problem, however it also provides very poor compression and therefore it is generally not an appropriate streaming solution. Another approach is to use periodic I-frames, e.g. the MPEG GOP. For example, with a 15-frame GOP there is an I-frame every 15 frames and this periodic reinitialization of the prediction loop limits error propagation to a maximum of one GOP (15 frames in this example). This approach is used in DVD's to provide random access and Digital TV to provide rapid channel

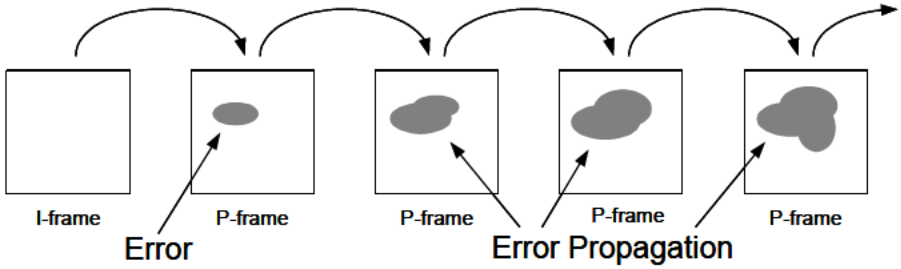

Figure 3: Example of error propagation that can result from a single error.

acquisition. However, the use of periodic I-frames limits the compression, and therefore this approach is often inappropriate for very low bit rate video, e.g. video over wireless channels or over the Internet.

More sophisticated methods of intra coding often apply partial intra-coding of each frame, where individual macroblocks (MBs) are intra-coded as opposed to entire frames. The simplest approach of this form is periodic intra-coding of all MBs: $1/N$ of the MB's in each frame are intra-coded in a predefined order, and after N frames all the MBs have been intra-coded. A more effective method is pre-emptive intra-coding, where one optimizes the intra-inter mode decision for each macroblock based on the macroblocks's content, channel loss model, and the macroblock's estimated vulnerability to losses.

The use of intra-coding to reduce the error propagation problem has a number of advantages and disadvantages. The advantages include: (1) intra coding does successfully limit error propagation by reinitializing the prediction loop, (2) the sophistication is at the encoder, while the decoder is quite simple, (3) the intra-inter mode decisions are outside the scope of the standards, and more sophisticated algorithms may be incorporated in a standard-compatible manner. However, intra-coding also has disadvantages including: (1) it requires a significantly higher bit rate than inter coding, leading to a sizable compression penalty, (2) optimal intra usage depends on accurate knowledge of channel characteristics. While intra coding limits error propagation, the high bit rate it requires limits its use in many applications.

Point-to-Point Communication with Back-Channel

The special case of point-to-point transmission with a back-channel and with real-time encoding facilitates additional approaches for overcoming the error propagation problem [31]. For example, when a loss occurs the decoder can notify the encoder of the loss and tell the encoder to reinitialize the prediction loop by coding the next frame as an I-frame. While this approach uses I-frames to overcome error propagation (similar to the previous approaches described above), the key is that I-frames are *only used* when necessary. Furthermore, this approach can be extended to provide improved compression efficiency by using P-frames as opposed to I-frames to overcome the error propagation. The basic idea is that both the encoder and decoder store multiple previously coded frames. When a loss occurs the decoder notifies the encoder which frames where correctly/erroneously received and therefore which frame should be used as the reference for the next prediction. These capabilities are provided by the Reference Picture Selection (RPS) in H.263 V2 and NewPred in MPEG-4 V2. To summarize, for point-to-

point communications with real-time encoding and a reliable back-channel with a sufficiently short round-trip-time (RTT), feedback-based approaches provide a very powerful approach for overcoming channel losses. However, the effectiveness of this approach decreases as the RTT increases (measured in terms of frame intervals), and the visual degradation can be quite significant for large RTTs [32].

Partial Summary: Need for Other Error-Resilient Coding Approaches
This section discussed the two major classes of problems that afflict compressed video communication in error-prone environments: (1) bitstream synchronization and (2) incorrect state and error propagation. The bitstream synchronization problem can often be overcome through appropriate algorithm and system design based on the application level framing principle. However, the error propagation problem remains a major obstacle for reliable video communication over lossy packet networks such as the Internet. While this problem can be overcome in certain special cases (e.g. point-to-point communication with a backchannel and with sufficiently short and reliable RTT), many important applications do not have a backchannel, or the backchannel may have a long RTT, thereby severely limiting effectiveness. Therefore, it is important to be able to overcome the error propagation problem in the *feedback-free* case, when there does not exist a back-channel between the decoder and encoder, e.g. broadcast, multicast, or point-to-point with unreliable or long-RTT backchannel.

7.4.5 Scalable Video Coding for Lossy Networks

In scalable or layered video the video is coded into a base layer and one or more enhancement layers. There are a number of forms of scalability, including temporal, spatial, and SNR (quality) scalability. Scalable coding essentially prioritizes the video data, and this prioritization effectively supports intelligent discarding of the data. For example, the enhancement data can be lost or discarded while still maintaining usable video quality. The different priorities of video data can be exploited to enable reliable video delivery by the use of unequal error protection (UEP), prioritized transmission, etc. As a result, scalable coding is a nice match for networks which support different qualities of service, e.g. DiffServ.

While scalable coding prioritizes the video data and is nicely matched to networks that can exploit different priorities, many important networks do not provide this capability. For example, the current Internet is a best-effort network. Specifically, it does not support any form of QoS, and all packets are equally likely to be lost. Furthermore, the base layer for scalable video is critically important – if the base layer is corrupted then the video can be completely lost. Therefore, there is a fundamental mismatch between scalable coding and the best-effort Internet: Scalable coding produces multiple bitstreams of differing importance, but the best-effort Internet does not treat these bitstreams differently – every packet is treated equally. This problem motivates the development of Multiple Description Coding, where the signal is coded into multiple bitstreams, each of roughly equal importance.

7.4.6 Multiple Description Video Coding

In Multiple Description Coding (MDC) a signal is coded into two (or more) separate bitstreams, where the multiple bitstreams are referred to as

multiple descriptions (MD). MD coding provides two important properties: (1) each description can be independently decoded to give a usable reproduction of the original signal, and (2) the multiple descriptions contain complementary information so that the quality of the decoded signal improves with the number of descriptions that are correctly received. Note that this first property is in contrast to conventional scalable or layered schemes, which have a base layer that is critically important and if lost renders the other bitstream(s) useless. MD coding enables a useful reproduction of the signal when *any* description is received, and provides increasing quality as more descriptions are received.

A number of MD video coding algorithms have recently been proposed, which provide different tradeoffs in terms of compression performance and error resilience [33,34,35,36]. In particular, the MD video coding system of [36,38] has the importance property that it enables repair of corrupted frames in a description using uncorrupted frames in the other description so that usable quality can be maintained even when *both* descriptions are afflicted by losses, as long as both descriptions are not *simultaneously* lost. Additional benefits of this form of MD system include high compression efficiency (achieving MDC properties with only slightly higher total bit rate than conventional single description (SD) compression schemes), ability to successfully operate over paths that support different or unbalanced bit rates (discussed next) [37], and this MD video coder also corresponds to a *standard-compatible enhancement* to MPEG-4 V2 (with NEWPRED), H.263 V2 (with RPS), and H.264/MPEG-4 Part 10 (AVC).

Multiple Description Video Coding and Path Diversity

MD coding enables a useful reproduction of the signal when any description is received -- and specifically this form of MD coding enables a useful reproduction when at least one description is received *at any point in time*. Therefore it is beneficial to increase the probability that at least one description is received correctly at any point in time. This can be achieved by combining MD video coding with a path diversity transmission system [38], as shown in Figure 4, where different descriptions are explicitly transmitted over different network paths (as opposed to the default scenarios where they would proceed along a single path). Path diversity enables the end-to-end video application to effectively see a virtual channel with improved loss characteristics [38]. For example, the application effectively sees an average path behavior, which generally provides better performance than seeing the behavior of any individual random path. Furthermore, while any network path may suffer from packet loss, there is a much smaller probability that all of the multiple paths *simultaneously* suffer from losses. In other words, losses on different paths are likely to be uncorrelated. Furthermore, the path diversity transmission system and the MD coding of [36,38] complement each other to improve the effectiveness of MD coding: the path diversity transmission system reduces the probability that both descriptions are simultaneously lost, and the MD decoder enables recovery from losses as long as both descriptions are not simultaneously lost. A path diversity transmission system may be created in a number of ways, including by source-based routing or by using a relay infrastructure. For example, path diversity may be achieved by a relay infrastructure, where each stream is sent to a different relay placed at a strategic node in the network, and each relay performs a simple forwarding operation. This approach

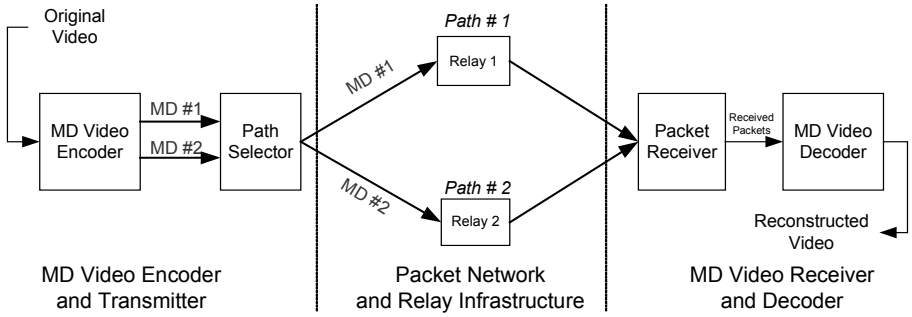

Figure 4: Multiple description video coding and path diversity for reliable communication over lossy packet networks.

corresponds to an application-specific overlay network on top of the conventional Internet, providing a service of improved reliability while leveraging the infrastructure of the Internet [38].

Multiple Description versus Scalable versus Single Description

The area of MD video coding is relatively new, and therefore there exist many open questions as to when MD coding, or scalable coding, or single description coding is preferable. In general, the answer depends crucially on the specific context, e.g. specific coder, playback delay, possible retransmits, etc. A few works shed light on different directions. [39,40] proposed MD image coding sent over multiple paths in an ad-hoc wireless network, and [41] examined MD versus scalable coding for an EGPRS cellular network. Analytical models for accurately predicting SD and MD video quality as a function of path diversity and loss characteristics are proposed in [42]. Furthermore, as is discussed in Section 10, path diversity may also be achieved by exploiting the infrastructure of a content delivery network (CDN), to create a Multiple Description Streaming Media CDN (MD-CDN) [43]. In addition, the video streaming may be performed in a channel-adaptive manner as a function of the path diversity characteristics [44,4].

7.5 Joint Source/Channel Coding

Data and video communication are fundamentally different. In data communication all data bits are equally important and *must* be reliably delivered, though timeliness of delivery may be of lesser importance. In contrast, for video communication some bits are more important than other bits, and often it is *not* necessary for all bits to be reliably delivered. On the other hand, timeliness of delivery is often critical for video communication. Examples of coded video data with different importance include the different frames types in MPEG video (i.e. I-frames are most important, P-frames have medium importance, and B-frames have the least importance) and the different layers in a scalable coding (i.e. base layer is critically important and each of the enhancement layers is of successively lower importance). A basic goal is then to exploit the differing importance of video data, and one of the motivations of joint source/channel coding is to jointly design the source coding and the channel coding to exploit this difference in importance. This has been an important area of research for many years, and the limited space here prohibits a detailed discussion, therefore we only present two illustrative examples of how error-control can be adapted based on the

importance of the video data. For example, for data communication all bits are of equal importance and FEC is designed to provide equal error protection for every bit. However, for video date of unequal importance it is desirable to have unequal error protection (UEP) as shown in Table 2. Similarly, instead of a common retransmit strategy for all data bits, it is desirable to have unequal (or priortized) retransmit strategies for video data.

Table 2. Adapting error control based on differing importance of video data: unequal error protection and unequal (prioritized) retransmission based on coded frame type.

	I-frame	P-frame	B-frame
FEC	Maximum	Medium	Minimum (or none)
Retransmit	Maximum	Medium	Can discard

8. MEDIA STREAMING PROTOCOLS AND STANDARDS

This section briefly describes the network protocols for media streaming over the Internet. In addition, we highlight some of the current popular specifications and standards for video streaming, including 3GPP and ISMA.

8.1 Protocols for Video Streaming over the Internet

This section briefly highlights the network protocols for video streaming over the Internet. First, we review the important Internet protocols of IP, TCP, and UDP. This is followed by the media delivery and control protocols.

Internet Protocols: TCP, UDP, IP

The Internet was developed to connect a heterogeneous mix of networks that employ different packet switching technologies. The Internet Protocol (IP) provides baseline best-effort network delivery for all hosts in the network: providing addressing, best-effort routing, and a global format that can be interpreted by everyone. On top of IP are the end-to-end transport protocols, where Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the most important. TCP provides reliable byte-stream services. It guarantees delivery via retransmissions and acknowledgements. On the other hand, UDP is simply a user interface to IP, and is therefore unreliable and connectionless. Additional services provided by UDP include checksum and port-numbering for demultiplexing traffic sent to the same destination. Some of the differences between TCP and UDP that affects streaming applications are:

- TCP operates on a byte stream while UDP is packet oriented.
- TCP guarantees delivery via retransmissions, but because of the retransmissions its delay is unbounded. UDP does not guarantee delivery, but for those packets delivered their delay is more predictable (i.e. one-way delay) and smaller.
- TCP provides flow control and congestion control. UDP provides neither. This provides more flexibility for the application to determine the appropriate flow control and congestion control procedures.
- TCP requires a back channel for the acknowledgements. UDP does not require a back channel.

Web and data traffic are delivered with TCP/IP because guaranteed delivery is far more important than delay or delay jitter. For media streaming the uncontrollable delay of TCP is unacceptable and compressed media data is usually transmitted via UDP/IP despite control information is usually transmitted via TCP/IP.

Media Delivery and Control Protocols

The IETF has specified a number of protocols for media delivery, control, and description over the Internet.

Media Delivery

The Real-time Transport Protocol (RTP) and Real-time Control Protocol (RTCP) are IETF protocols designed to *support streaming media*. RTP is designed for data transfer and RTCP for control messages. Note that these protocols do *not* enable real-time services, only the underlying network can do this, however they provide functionalities that support real-time services. RTP does not guarantee QoS or reliable delivery, but provides support for applications with time constraints by providing a standardized framework for common functionalities such as time stamps, sequence numbering, and payload specification. RTP enables detection of lost packets. RTCP provides feedback on quality of data delivery. It provides QoS feedback in terms of number of lost packets, inter-arrival jitter, delay, etc. RTCP specifies periodic feedback packets, where the feedback uses no more than 5 % of the total session bandwidth and where there is at least one feedback message every 5 seconds. The sender can use the feedback to adjust its operation, e.g. adapt its bit rate. The conventional approach for media streaming is to use RTP/UDP for the media data and RTCP/TCP or RTCP/UDP for the control. Often, RTCP is supplemented by another feedback mechanism that is explicitly designed to provide the desired feedback information for the specific media streaming application. Other useful functionalities facilitated by RTCP include inter-stream synchronization and round-trip time measurement.

Media Control

Media control is provided by either of two session control protocols: Real-Time Streaming Protocol (RTSP) or Session Initiation Protocol (SIP). RTSP is commonly used in video streaming to establish a session. It also supports basic VCR functionalities such as play, pause, seek and record. SIP is commonly used in voice over IP (VoIP), and it is similar to RTSP, but in addition it can support user mobility and a number of additional functionalities.

Media Description and Announcement

The Session Description Protocol (SDP) provides information describing a session, for example whether it is video or audio, the specific codec, bit rate, duration, etc. SDP is a common exchange format used by RTSP for content description purposes, e.g., in 3G wireless systems. It has also used with the Session Announcement Protocol (SAP) to announce the availability of multicast programs.

8.2 Video Streaming Standards and Specifications

Standard-based media streaming systems, as specified by the 3rd Generation Partnership Project (3GPP) for media over 3G cellular [45] and the Internet

Streaming Media Alliance (ISMA) for streaming over the Internet [46], employ the following protocols:

- Media encoding
  - MPEG-4 video and audio (AMR for 3GPP), H.263
- Media transport
  - RTP for data, usually over UDP/IP
  - RTCP for control messages, usually over UDP/IP
- Media session control
  - RTSP
- Media description and announcement
  - SDP

The streaming standards do not specify the storage format for the compressed media, but the MP4 file format has been widely used. One advantage of MP4 file format is the ability to include "hint tracks" that simplify various aspects of streaming by providing hints such as packetization boundaries, RTP headers and transmission times.

9. ADDITIONAL VIDEO STREAMING TOPICS

MULTICAST

Multicast or one-to-many communication has received much attention in the last few years due to the significant bandwidth savings it promises, and the challenges it presents. Consider the multicast extension of the Internet, or IP multicast, as an example. When multiple clients are requesting the same media stream, IP multicast reduce network resource usage by transmitting only one copy of the stream down shared links, instead of one per session sharing the link. Nevertheless, besides the many practical difficulties in supporting IP multicast for the wide-area Internet, the basic properties of multicast communication present a number of challenges to streaming media systems. First and foremost is the problem of heterogeneity: different receivers experience different channel conditions and may have conflicting requirements, e.g. in terms of maximum bit-rate that can be supported, and the amount of error protection needed. Heterogeneity is typically solved by using multiple multicast to provide choices for the receivers. For instance, it is possible to establish different multicasts for different ranges of intended bit-rates [47]. Alternatively, the different multicasts can contain incremental information [22, 48]. A second challenge that multicast presents is the more restricted choice for error control. While retransmission has been the error control mechanism of choice for many streaming applications, its applicability in multicast has been limited by a number of challenges. Using IP multicast for retransmission, for instance, requires that both the retransmission request and the actual retransmission be transmitted to all the receivers in the multicast, an obviously inefficient solution. Even when retransmissions are handled by unicast communication, scalability concerns still remain, since a single sender will have to handle the requests of potentially many receivers.

END-TO-END SECURITY AND TRANSCODING

Encryption of media is an effective tool to protect content from eavesdroppers. Transcoding at intermediate nodes within a network is also important technique to adapt compressed media streams for particular client capabilities or network conditions. Nevertheless, network transcoding poses a serious threat to the end-to-end security because transcoding encrypted

streams generally requires decrypting the stream, transcoding the decrypted stream, and then re-encrypting the result. Each transcoding node presents a possible breach to the security of the system. This problem can be overcome by Secure Scalable Streaming (SSS) which enables downstream transcoding without decryption [49,50]. SSS uses jointly designed scalable coding and progressive encryption techniques to encode and encrypt video into secure scalable packets that are transmitted across the network. These packets can be transcoded at intermediate, possibly untrusted, network nodes by simply truncating or discarding packets and without compromising the end-to-end security of the system. The secure scalable packets have unencrypted headers that provide hints, such as optimal truncation points, which the downstream transcoders use to achieve rate-distortion (R-D) optimal fine-grain transcoding *across* the encrypted packets.

STREAMING OVER WIRED AND WIRELESS LINKS

When the streaming path involves both wired and wireless links, some additional challenges evolve. The first challenge involves the much longer packet delivery time with the addition of a wireless link. Possible causes for the long delay include the employment of FEC with interleaving. For instance, round-trip propagation delay in the 3G wireless system is in the order of 100 ms even before link-level retransmission. With link-level retransmission, the delay for the wireless link alone can be significant. The long round-trip delay reduces the efficiency of a number of end-to-end error control mechanisms: the practical number of end-to-end retransmissions is reduced, and the effectiveness of schemes employing RPS and NewPred is also reduced. The second challenge is the difficulty in inferring network conditions from end-to-end measurements. In high-speed wired networks, packet corruption is so rare that packet loss is a good indication of network congestion, the proper reaction of which is congestion control. In wireless networks however, packet losses may be due to corruption in the packet, which calls for stronger channel coding. Since any end-to-end measurements contain aggregate statistics across both the wired and wireless links, it is difficult to identify the proper cause and therefore perform the proper reaction.

10. STREAMING MEDIA CONTENT DELIVERY NETWORKS

The Internet has rapidly emerged as a mechanism for users to find and retrieve content, originally for webpages and recently for streaming media. content delivery networks (CDNs) were originally developed to overcome performance problems for delivery of static web content (webpages). These problems include network congestion and server overload, that arise when many users access popular content. CDNs improve end-user performance by caching popular content on edge servers located closer to users. This provides a number of advantages. First, it helps prevent server overload, since the replicated content can be delivered to users from edge servers. Furthermore, since content is delivered from the closest edge server and not from the origin server, the content is sent over a shorter network path, thus reducing the request response time, the probability of packet loss, and the total network resource usage. While CDNs were originally intended for static web content, recently, they are being designed for delivery of streaming media as well.

10.1 Streaming Media CDN Design

A streaming media CDN is a CDN that is explicitly designed to deliver streaming media, as opposed to static webpages. Streaming media CDN design and operation is similar in many ways to conventional (webpage) CDN design and operation. For example, there are three key problems that arise in general CDN design and operation. The first is the server placement problem: Given N servers, where should these servers be placed on the network? The second problem is the content distribution problem: On which servers should each piece of content be replicated? The third problem is the server selection problem: For each request, which is the optimal server to direct the client to for delivery of the content?

Many aspects of a streaming media CDN are also quite different from a conventional CDN. For example, client-server interaction for a conventional (webpage) CDN involves a short-lived (fraction of a second) HTTP/TCP session(s). However, a streaming session generally has a long duration (measured in minutes) and uses RTSP and RTP/UDP. While congestion and packet loss may lead to a few second delay in delivering a webpage and is often acceptable, the corresponding effect on a streaming media session would be an interruption (stall or visual artifacts) that can be highly distracting. Clearly, delay, packet loss, and any form of interruption can have a much more detrimental effect on video streaming then on static webpage delivery. In addition, in a conventional CDN each piece of content (webpage) is relatively small, on the order of 10's of kilobytes, and therefore it can be replicated in its entirety on each chosen server. However, streaming media, such as movies, have a long duration and require a significant amount of storage, on the order of megabytes or gigabytes, and therefore it is often not practical or desirable to replicate an entire video stream on each chosen server. Instead, the video can be partitioned into parts, and only a portion of each video is cached on each server. There are many interesting problems related to caching of video, e.g. see [51] and references therein.

Two other capabilities that are important for streaming media CDN, and are of lesser importance for a conventional CDN for webpage distribution, are multicast and server hand-off. Multicast is clearly a highly desirable capability for streaming of popular media. While wide-area IP Multicast is currently not available in the Internet, a streaming media CDN can be explicitly designed to provide this capability via application-layer multicast: the infrastructure in the streaming media CDN provide an overlay on the Internet and are used as the nodes for the multicast tree. Communication between nodes employs only simple ubiquitous IP service, thereby avoiding the dependence of IP multicast. Another important capability is hand-off between streaming servers. Because streaming media sessions are long lived, it is sometimes required to perform a midstream hand-off from one streaming server to another. This functionality is not required for webpage delivery where the sessions are very short in duration. Furthermore, when the streaming session involves transcoding, mid-stream hand-off of the transcoding session is also required between servers [52].

10.2 Multiple Description Streaming Media CDN (MD-CDN)

CDNs have been widely used to provide low latency, scalability, fault tolerance, and load balancing for the delivery of web content and more

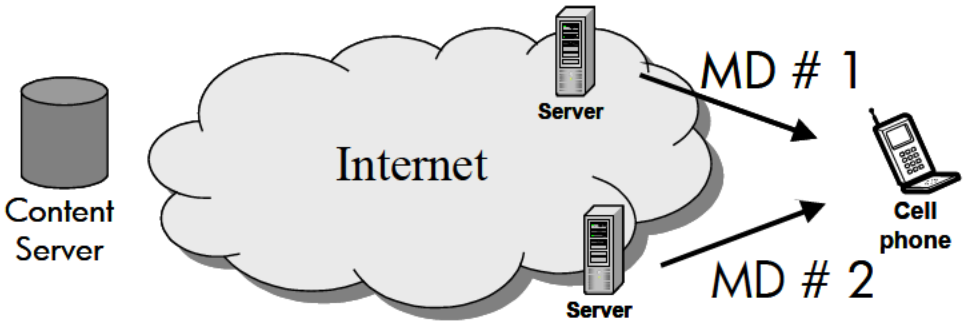

Figure 5: An MD-CDN uses MD coding and path diversity to provide improved reliability for packet losses, link outages, and server failures.

recently streaming media. Another important advantage offered by streaming media CDNs is their distributed infrastructure. The distributed infrastructure of a CDN can be used to explicitly achieve *path diversity* between each client and multiple nearby edge servers. Furthermore, appropriately coupling MD coding with this path diversity can provide improved reliability to packet losses, link outages, and server failures. This system is referred to as a *Multiple Description Streaming Media Content Delivery Network* [43] or an MD-CDN for short.

An MD-CDN operates in the following manner: (1) MD coding is used to code a media stream into multiple complementary descriptions, (2) the different descriptions are distributed across different edge servers in the CDN, (3) when a client requests a media stream, it is directed to multiple nearby servers that host complementary descriptions, and (4) the client simultaneously receives the different complementary descriptions through different network paths from different servers. That is, the existing CDN infrastructure is exploited to achieve path diversity between multiple servers and each client. In this way, disruption in streaming media occurs only in the less likely case when simultaneous losses afflict both paths. This architecture also reaps the benefits associated with CDNs, such as reduced response time to clients, load balancing across servers, robustness to network and server failures, and scalability to number of clients.

Further information about MD-CDN design and operation is available in [43]. Other related work include distributing MD coded data in peer-to-peer networks [53], streaming a conventional single description stream to a single client from multiple servers [54], and the use of Tornado codes and multiple servers to reduce download time for bulk data (not video) transfer [55].

11. Summary

Video communication over packet networks has witnessed much progress in the past few years, from download-and-play to various adaptive techniques, and from direct use of networking infrastructure to the design and use of overlay architectures. Developments in algorithms and in compute, communication and network infrastructure technologies have continued to change the landscape of streaming media, each time simplifying some of the current challenges and spawning new applications and challenges. For example, the emergence of streaming media CDNs presents a variety of

conceptually exciting and practically important opportunities to not only mitigate existing problems, but create new applications as well. The advent of high bandwidth wireless networking technologies calls for streaming media solutions that support not only wireless environments, but user mobility as well. Possible QoS support in the Internet, on the other hand, promises a more predictable channel for streaming media applications that may make low-bandwidth low-latency streaming over IP a reality. Therefore, we believe that video streaming will continue to be a compelling area for exploration, development, and deployment in the future.

REFERENCES

[1] M.-T. Sun and A. Reibman, eds, *Compressed Video over Networks*, Marcel Dekker, New York, 2001.

[2] G. Conklin, G. Greenbaum, K. Lillevold, A. Lippman, and Y. Reznik, "Video Coding for Streaming Media Delivery on the Internet," *IEEE Trans. Circuits and Systems for Video Technology*, March 2001.

[3] D. Wu, Y. Hou, W. Zhu, Y.-Q. Zhang, and J. Peha, "Streaming Video over the Internet: Approaches and Directions", *IEEE Transactions on Circuits and Systems for Video Technology,* March 2001.

[4] B. Girod, J. Chakareski, M. Kalman, Y. Liang, E. Setton, and R. Zhang, "Advances in Network-Adaptive Video Streaming", *2002 Tyrrhenian Inter. Workshop on Digital Communications,* Sept 2002.

[5] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications,* New Jersey, Prentice-Hall, 2002.

[6] www.realnetworks.com

[7] www.microsoft.com/windows/windowsmedia

[8] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, April, 1991.

[9] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures,* Boston, Massachusetts: Kluwer Academic Publishers, 1997.

[10] J. Apostolopoulos and S. Wee, ``Video Compression Standards'', *Wiley Encyclopedia of Electrical and Electronics Engineering,* John Wiley & Sons, Inc., New York, 1999.

[11] "Video codec for audiovisual services at px64 kbits/s", ITU-T Recommendation H.261, Inter. Telecommunication Union, 1993.

[12] "Video coding for low bit rate communication", ITU-T Rec. H.263, Inter. Telecommunication Union, version 1, 1996; version 2, 1997.

[13] ISO/IEC 11172, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s." International Organization for Standardization (ISO), 1993.

[14] ISO/IEC 13818. "Generic coding of moving pictures and associated audio information." International Organization for Standardization (ISO), 1996.

[15] ISO/IEC 14496. "Coding of audio-visual objects." International Organization for Standardization (ISO), 1999.

[16] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Trans. Communications,* Vol. 28 No. 4, April 1980, pp 553-574.

[17] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM,* August 1988.

[18] M. Mathis et al., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Communications Review*, July 1997.

[19] J. Padhye et al., "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation," *IEEE/ACM Trans. Networking,* April 2000.

[20] W. Tan and A. Zakhor, "Real-time Internet Video using Error-Resilient Scalable Compression and TCP-friendly Transport Protocol," *IEEE Trans. on Multimedia,* June 1999.

[21] S. Floyd et al., "Equation-based Congestion Control for Unicast Applications," *ACM SIGCOMM,* August 2000.

[22] S. McCanne, V. Jacobsen, and M. Vetterli, "Receiver-driven layered multicast", *ACM SIGCOMM,* Aug. 1996.

[23] S. Wee, J. Apostolopoulos and N. Feamster, "Field-to-Frame Transcoding with Temporal and Spatial Downsampling," *IEEE International Conference on Image Processing*, October 1999.

[24] P. White, "RSVP and Integrated Services in the Internet: A Tutorial," *IEEE Communications Magazine,* May 1997.

[25] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Media Playout for Low Delay Video Streaming over Error-Prone Channels," preprint, to appear *IEEE Trans. Circuits and Systems for Video Technology*.

[26] Y. Wang and Q. Zhu, "Error control and concealment for video communications: A review," *Proceedings of the IEEE*, May 1998.

[27] N. Färber, B. Girod, and J. Villasenor, "Extension of ITU-T Recommendation H.324 for error-resilient video transmission," *IEEE Communications Magazine*, June 1998.

[28] R. Talluri, "Error-resilient video coding in the ISO MPEG-4 standard," *IEEE Communications Magazine*, June 1998.

[29] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media", *IEEE Trans. on Multimedia*, submitted Feb. 2001.

[30] D. Clark and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," *ACM SIGCOMM,* September 1990.

[31] B. Girod and N. Färber, "Feedback-based error control for mobile video transmission", *Proceedings of the IEEE*, October 1999.

[32] S. Fukunaga, T. Nakai, and H. Inoue, "Error resilient video coding by dynamic replacing of reference pictures", *GLOBECOM*, Nov. 1996.

[33] S. Wenger, "Video Redundancy Coding in H.263+", *Workshop on Audio-Visual Services for Packet Networks*, September 1997.

[34] V. Vaishampayan and S. John, "Interframe balanced-multiple-description video compression", *IEEE Inter Conf. on Image Processing,* Oct.1999.

[35] A. Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple description coding for video using motion compensated prediction", *IEEE Inter. Conf. Image Processing ,* October 1999.

[36] J. Apostolopoulos, "Error-resilient video compression via multiple state streams", *Proc. International Workshop on Very Low Bitrate Video Coding (VLBV'99),* October 1999.

[37] J. Apostolopoulos and S. Wee, "Unbalanced Multiple Description Video Communication Using Path Diversity", *IEEE International Conference on Image Processing,* October 2001.

[38] J. Apostolopoulos, "Reliable Video Communication over Lossy Packet Networks using Multiple State Encoding and Path Diversity," *Visual Communications and Image Processing,* January 2001.

[39] N. Gogate and S. Panwar, "Supporting video/image applications in a mobile multihop radio environment using route diversity", *Proceedings Inter. Conference on Communications,* June 1999.

[40] N. Gogate, D. Chung, S.S. Panwar, and Y. Wang, "Supporting image/video applications in a mobile multihop radio environment using route diversity and multiple description coding," Preprint.

[41] A. Reibman, Y. Wang, X. Qiu, Z. Jiang, and K. Chawla, "Transmission of Multiple Description and Layered Video over an (EGPRS) Wireless Network", *IEEE Inter. Conf. Image Processing*, September 2000.

[42] J. Apostolopoulos, W. Tan, S. Wee, and G. Wornell, "Modeling Path Diversity for Multiple Description Video Communication," *IEEE Inter. Conference on Acoustics, Speech, and Signal Processing*, May 2002.

[43] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On Multiple Description Streaming with Content Delivery Networks," *IEEE INFOCOM*, July 2002.

[44] Y. Liang, E. Setton and B. Girod, "Channel-Adaptive Video Streaming Using Packet Path Diversity and Rate-Distortion Optimized Reference Picture Selection", to appear *IEEE Fifth Workshop on Multimedia Signal Processing,* Dec. 2002.

[45] www.3gpp.org

[46] www.isma.tv

[47] S. Cheung, M. Ammar and X. Li, "On the use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution," *IEEE INFOCOM*, March 1996.

[48] W. Tan and A. Zakhor, "Video Multicast using Layered FEC and Scalable Compression," *IEEE Trans. Circuits and Systems for Video Technology*, March 2001.

[49] S. Wee, J. Apostolopoulos, "Secure Scalable Video Streaming for Wireless Networks", *IEEE International Conference on Acoustics, Speech, and Signal Processing,* May 2001.

[50] S. Wee, J. Apostolopoulos, "Secure Scalable Streaming Enabling Transcoding without Decryption," *IEEE International Conference on Image Processing,* October 2001.

[51] Z. Miao and A. Ortega, "Scalable Proxy Caching of Video under Storage Constraints," *IEEE Journal on Selected Areas in Communications*, to appear 2002.

[52] S. Roy, B. Shen, V. Sundaram, and R. Kumar, "Application Level Hand-off Support for Mobile Media Transcoding Sessions", *ACM NOSSDAV,* May, 2002.

[53] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," *ACM NOSSDAV*, May 2002.

[54] T. Nguyen and A. Zakhor, "Distributed video streaming over internet" SPIE Multimedia Computing and Networking 2002, January 2002.

[55] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads," *IEEE INFOCOM,* 1999.