Bhanu Prakash Naredla

Student ID: 00001630571

COEN 233 (Computer Networks)

Fall 2021

Streaming Technologies

Audience

This document covers the architecture of media streaming, compression techniques, Quality of Service (QoS) and challenges in media streaming.

The reader is expected to have basic knowledge of networks, internet architecture, packet-switched network, circuit-switched network, TCP/UDP and some understanding in the field of audio, image compression techniques to have better understanding.

Table of Contents

Table of Figures

Table of Tables

1 INTRODUCTION

Streaming is a technology that allows to view content without downloading it to the device prior to viewing. Streaming of video and audio over internet in our day-to-day life has been increased drastically in the last two decades. More and more users started using applications like Skype, Zoom, YouTube, Netflix, Spotify, Amazon Prime Video, Hotstar etc. for daily entertainment, sports, and business needs. Online gaming and streaming television are being used by many users. Especially during COVID-19 pandemic classrooms, business meetings are being held virtually thus the increased usage of streaming applications.

Now, let's look at the history and evolution of streaming applications. Cost and potential of the computer hardware Before 1980s made difficult to view media in computer. But, later because of the innovations in computer hardware, more production of computer hardware made computers powerful enough to process and view media on computers using CD-ROMs, hard disk drives. Media is compressed and written to CD-ROMs or hard disk drives and decompressed before viewing the media. Streaming media seems impossible because uncompressed media requires more bandwidth which was impractical, advancement in data compression later 1990s and low cost of computer hardware made streaming media possible. By the advancement of network technology in early 2000s, end-users had access to more bandwidth which facilitated the streaming media to computer users.

Now, let's understand the high-level workflow of streaming media shown in Figure 1. The device which is producing media is called source, then the media is encoded and stored or sent to streaming Application Server (AS) to serve users over network.

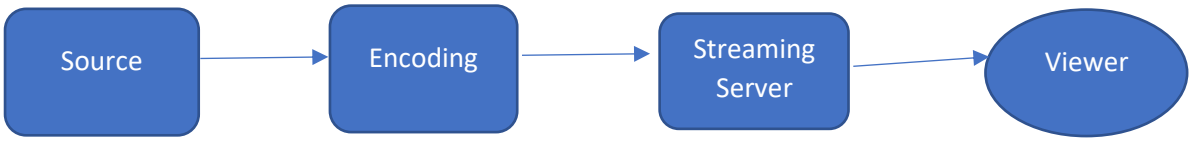

Figure 1: workflow of streaming media

In this paper we shall discuss the concepts of media streaming architecture, media compression, media compression standards, different type of streaming applications and their requirements for better quality of service, how to deal with constant-bit-rate and variable-bit-rate channels for packet delivery, how to deal with packet loss, time constraints for packet delivery to end devices, pros and cons of streaming media using different protocols, how the playout buffer is used to solve delay jitter, error control, media streaming protocols, standards and structure, multicast media streaming techniques, streaming media using content delivery networks.

2 MEDIA STREAMING ARCHITECTURE AND OVERVIEW

Streaming server is responsible for receiving compressed media i.e., compressed audio or compressed video along with compressed audio. Streaming application monitors quality of service at the application level for better user experience. We can use different protocols to send data to destination over internet which can be packet-switching network or circuit-switching network. Client receives the data and decodes it to play the media. Application layer quality of service control sends acknowledgment of data received or feedback of data received in last few seconds which can help in dynamically adapting the media bit rate.

There exist different media streaming applications which has their own specifications for providing better quality of service. For example, streaming application may be for multicast communication like group zoom meetings or may be for point-to-point communication or may be for broadcast communication like live streaming of sports, news etc. on top of this, communication channels can be static or dynamic, may use packet-switching or circuit switching network. So, the design to provide better quality of service to users of a streaming application depends on its requirements.

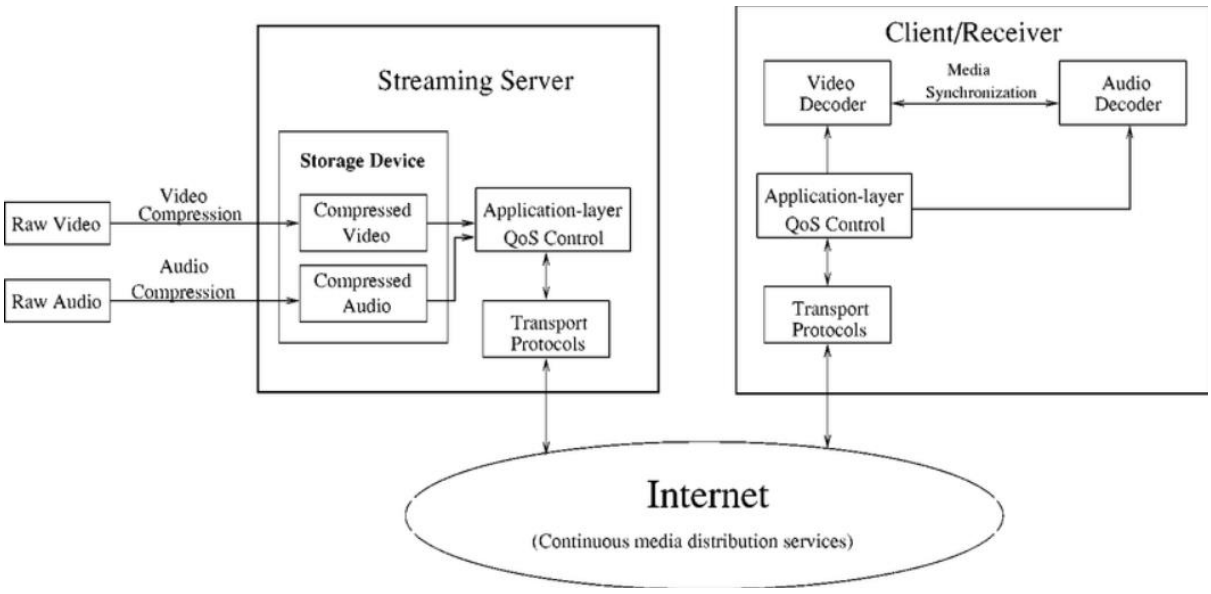

Figure 2: media streaming architecture [1]

Broadcast Communication

Broadcast communication is like one-to-many scenario, server sends one copy of data to all its clients connected to multiple access communication network. Broadcast address is used in the ipv4(Internet Protocol version 4) which is combination of network prefix and all 1's. When routers in the network see this broadcast address, sends packet to all the host in the same network. Popular example for this is broadcast television, subscription vendor should provide required bandwidth for every intended user i.e., every user watching a particular channel should receive same bitrate. One interesting problem here is, how then vendor is preventing HD(High-Definition) channel access to basic subscription users? They send HD channel data only on a particular range of slots, so that before

giving access to these HD channels to a user they check if user is allowed to access based on their subscription.

Point-to-Point Communication

Point-to-Point communication is like one-to-one communication. Example for this is video call of two users, here latency is the key issue because, any data that has arrived late is of no use. We can't keep sending acknowledgment of the received packets which increases traffic in the network as well as delay in receiving next packet. If there is a back channel for communication between sender and receiver, then receiver can send feedback to sender about the last few received packets and adjust the bitrate if needed accordingly.

Multicast Communication

Multicast communication is like one-to-many communication. Multicast communication is better than multiple unicast communications because, in multicast communication only one copy of data is sent to multiple users thus reducing traffic in the network.

Real-time media encoding

Applications like video conference, live streaming, interactive gaming etc. require real-time encoding. Based upon the receiver feedback or request, sender can encode data as required for the receiver. Popular content with bitrate can be cached so that server can serve the request quickly when other users request the same content and bitrate.

Pre-encoded media

In some applications media is encoded and stores locally or remotely. This has an advantage of using an efficient encoding algorithm which isn't possible in real-time because efficient encoding takes more time. This also has a disadvantage that the same pre-encoded media cannot be send to channels having different bitrates.

Interactive Applications

Interactive applications such as games have very strict time constraints. Low- latency is the key issue here, if any packet that has arrived after its expected receive time is of no use. Real-time encoding is used here thus the design of the system should consider all these key factors.

Non-interactive Applications

Non-interactive applications have very loose time constraints. Low latency isn't the key issue here. Although some non-interactive applications require real-time encoding, we may have chance to use pre-encoding technique for better quality of service. Having loose latency constraints dramatically changes the design of the system.

Variable-bit-rate Channel

When the media is encoded, the size of the frame isn't constant every time. If users require a constant quality throughout the play, then variable-bit-rate channel is required to provide it. So, to achieve this by using a variable-bit-rate channel, a buffer is used at the receiver's end to control the sequence of data received. Before starting the play of that frame, it should be available in the buffer. This buffer is called playout buffer, in the following pages we'll discuss about this in detail.

3 MEDIA COMPRESSION

Media is first compressed then compressed media is sent on demand as multiple packets over internet each at a time in sequence. Here let's discuss audio compression and standards, video compression and standards.

3.1 Audio Compression

Before discussing audio compression, let's discuss what actually is sound? sound is produced when particles collide and traverse in the form of waves. So, sound waves just consist of vibrating particles. We're able to hear sound because the vibration of particles in air causing ear drums to vibrate and converted into signals which are carried to brain nerve.

So, to store any audio on Tape drive, CD, Hard Disk etc. we should convert analog signal to digital signal. Audio is sampled at 44,100 times per second [3]. We need 2 bytes of storage to store each sample. An average audio lasting for three minutes long could take 30MB or more. In early 2000's internet bandwidth was around 56Kbps, to download an audio it would take around 10 minutes or more and also storing a greater number of audios is also a problem as it takes up more memory on storage drives. So, audio compression came into picture. With audio compression we can reduce the storage space required to store an audio down to ten times i.e., to store a 30MB audio file it would take 3MB to store same audio which is compressed. This also has another advantage which is time required to download an audio got decreased by ten times.

But, in audio compression what information in audio are we removing? Are we compromising on audio quality? The answer is NO. let's discuss what information in audio are we removing and how does that effect audio quality.

3.1.1 What information is removed in audio compression?

Knowledge of *psychoacoustics* is used to design compression algorithms. *Psychoacoustics* means how human brain interprets sounds. Human brain does auditory masking for attention to the most important sound at every time. With this information, we can decide on what information can be removed from audio without compromising on audio quality. Humans can hear audio ranging from frequency 20hz to 20kHz [3]. Any audiowave whose frequency is outside of this range is of no use for human ears, we can get rid of this information from the audio file to save some storage space.

If a loud sound is blaring over a lot of low-volume sounds, we're obviously going to focus on the loud sound. This refers to something called *simultaneous masking* [3]. With the help of this fact, we don't need to store any quiet sounds.

Another trick we can use with the same above fact is Temporal Masking. This means, if two sounds occur in very short amount of time, we're going to focus only on the loudest one.

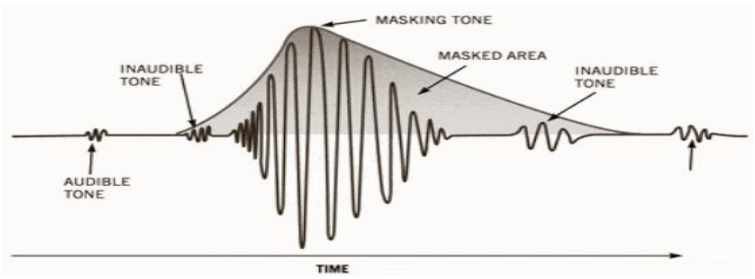

Figure 3: Temporal Masking Audio [3]

So, compression algorithms ignore quiet sound occurred just before loud sounds. Thus, we could save some storage space.

3.1.2 Working of MP3

Before understanding the working of MP3, lets understand what *bit depth* and *sample* rate are. **Bit depth** is like number of bits required to store a single sample. Audio signal consists of large number of samples. When we convert analog signal to digital signal, each sample is stored in some number of bits. **Sample rate** is the number of samples used to represent the audio wave. It's just like lower sample rate captures a smaller number of snapshots for a moment of audio. Highest sample rate has higher audio quality, lowest sample rate has lower audio quality.

Now as we know what bit depth and sample rate are, let's discuss how MP3 brings down the storage space by the factor of ten. Once we have compressed audio file using the Knowledge of *psychoacoustics,* we're still left with a major part of the original large audio file. Because the remaining data is still in its higher resolution. Now let's discuss how can we bring down the storage space by lowering the resolution of left-over data with totally compromising on audio quality.

MP3 is a lossy data compression technique. It lowers the bit rate of the audio to 16bits from 24bits or more [3]. This means, we use only 16bits to store a sample digitally. Are we compromising on audio quality? The answer is NO, the quality of audio totally depends on signal-to-noise ratio (SNR) and 16 bits for a sample is quiet enough to have good signal-to-noise ratio. By this, we lowered storage space required to store an audio file by 25%.

Sample rates can be as large as 96,000 or more per second. MP3 compression uses 44.1kHz sample rate [3]. This means, we're storing half the amount of data as sample rates got decreased by around 50%.

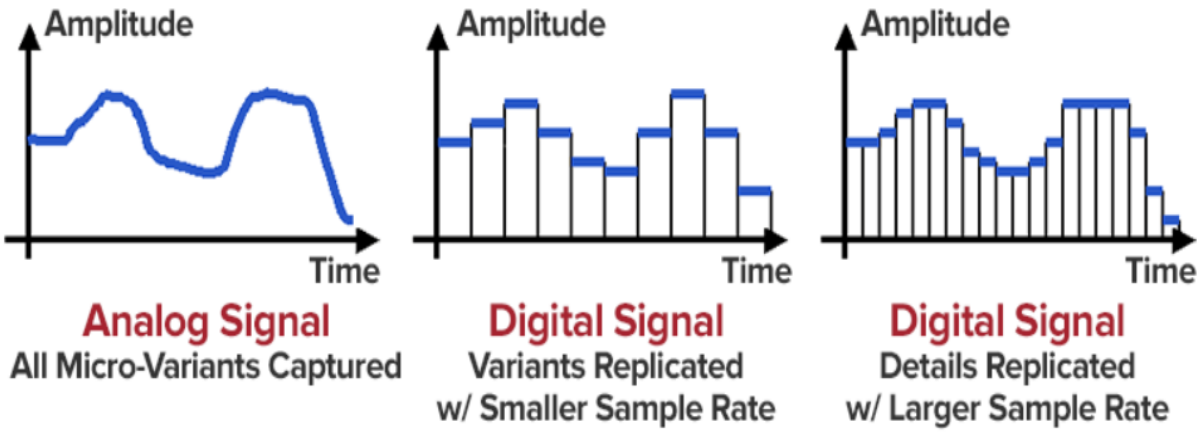

Figure 4: Audio Sample rate [3]

Now that we have lowered bit depth and sample rate without totally compromising on audio quality, MP3 sets limit to the data throughput per second like 128kbps, 192kbps, 320kbps. With the limit on bit rate, we adjust the bit depth and sample rate to match with the bit rate. Lower the bit rate lower the quality of audio, higher the bit rate higher the quality of audio. Most of the streaming services uses constant bit rate because, we can predict the bandwidth and storage requirements for the user. But some improvements have been made to improve the audio quality by coming up with

variable bit rate, variable bit rate is like uses the lower bit rate during quiet parts of the audio and uses higher bit rate at louder parts of the audio by which the audio quality is increased.

3.2 Video Compression

Video is nothing but a collection of moving images, each image in a video is called as a frame. Compression can be achieved by compressing each frame in a video using image compression techniques like JPEG. But, to achieve an optimized video compression, compressing each frame in a video isn't sufficient. So, before discussing about video compression lets discuss about image compression.

Image compression algorithm look for spatial and color redundancies in an image. It partitions an image into 8x8 pixel blocks because neighbouring pixels in an image are similar. So, rather than partitioning an image into larger pixel blocks we prefer partitioning an image into smaller pixel blocks so that pixels within each block could be more similar. After partitioning an image into 8x8 pixel blocks, we compute the 2-D Discrete Cosine Transform (DCT) for each 8x8 pixel block and DCT coefficients are obtained for each block which are sufficient to reconstruct an accurate version of the image using techniques like zigzag scanning, Huffman coding [2].

3.2.1 Video Compression Overview

Now that we know about image compression, let's discuss video compression. Video compression algorithms looks for spatial and temporal redundancies. The adjacent frames in a video are typically similar so, higher compression can be achieved comparing to compression achieved by compressing each frame in a video. So, by encoding similar data minimum number of times a greater compression can be achieved. For example, let's take a video with a constant background image. The background image in every frame need not be encoded: instead, we can encode it once and refer back to it for further frames. So, how are we using similarities in frames? We first predict the frame using the last coded frame and then make correction to this frame to obtain actual frame. As we know, video is a collection of moving frames, predicting the current frame using last encoded frame and motion estimation is called as *motion-compensated prediction* [4].

Types of coded frames:

- ➔ Intra-coded frames or I-frames
- ➔ Predictively coded frames or P-frames
- ➔ Bi-directionally coded frames or B-frames

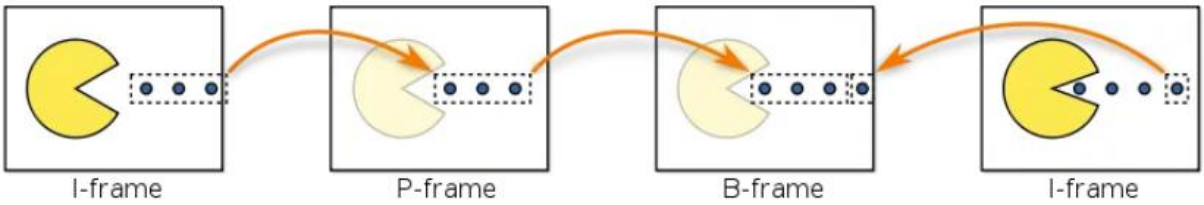

Figure 5: I-frame, P-frame and B-frame [4]

I-frame is independent of other frames and consists of all encoded data to obtain the actual image.

P-frame is dependent on the last I-frame and only needs to encode blocks which are unique to current frame while comparing it to the last encoded frame. In figure 5, P-frame needs to track only the dots i.e., how they are moving across the image and Pac-Man is staying at the same position as in previous frame.

B-frame is dependent on the P-frame as well as next I-frame and only requires encoding data which is unique to this frame while comparing it to P-frame and next I-frame.

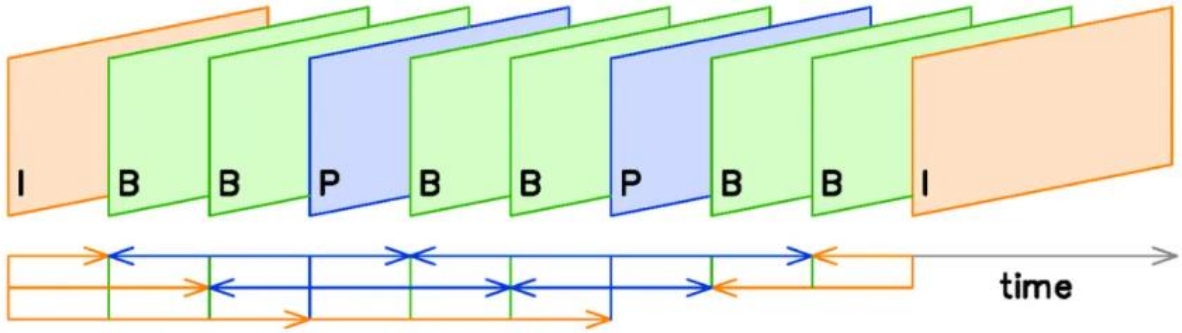

Figure 6: Intraframe encoding [4]

As I-frames are independent and contains all the information to obtain the actual image, the quality is higher compared to P-frame and B-frame. But best compression cannot be achieved using I-frames.

P-frames and b-frames predict the motion between the frames and the more accurate the prediction, higher compression is achieved. In prediction approach an image is partitioned into 16x16 blocks. We'll search for a similar block in the previous frame called as reference frame. If similar frame is found in reference frame, then that block is encoded in motion vector which describes the changes required to obtain current frame from the reference frame. Else, we consider it as a new block and encode it in current frame. There are different encoding algorithms which yields different compression. It all depends on how accurately algorithm is able to predict the current frame from the reference frames.

3.2.2 Video Compression Standards

Video compression standards ensures communication between different encoders or decoders manufactured by different companies. Without these, a device encoded a video and sent to another device may not be able to decode it. These standards are accepted by all the manufacturers and manufactures standard encoders or decoders thus interoperability between different encoders or decoders.

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

Table 1: video compression standards [2]

H.261 is designed for video conferencing in 1990 over the usage in integrated services digital network (ISDN). The bitrate is p*64 kbps where p = 1,2,3, …, 30 [2]

H.263 is designed for video conferencing to use in public switched telephone network (PSTN). The bitrate is about 33.6 kbps [2]

H.264 is the designed with improved video compression and can achieve bitrate up to 100 kbps [2]

MPEG-1 is designed to store compressed video and audio on CD-ROM, bit rate is about 1.5 Mbps [2]

MPEG-2 is designed for digital television for achieving higher bit rates, bit rate can be up to 20 Mbps [2]

4 CHALLENGES IN MEDIA STREAMING

What if media is delivered as a file download? This approach works irrespective of receiver's bandwidth. In file download approach, though the receiver has part of the media already downloaded which the user wants to view, receiver must wait until the full media is downloaded. Media can be a large file for example, videos. If the user wants to view only a part of the video, user has no other option but to wait until the whole video is downloaded. This approach isn't optimal for real-time usage. Viewers hate to wait for long duration to begin media playing.

Streaming is the optimal approach for real-time usage. As soon as the user receives part of the media, we shall decode and play the media while we're still receiving the net parts of the video.

Workflow of media streaming

Split the media into parts, send these parts each at a time in sequence to the receiver and receiver begins playing media by decodes these parts as soon as it receives while still receiving the other parts.

- ➔ Split the compressed media into packets
- ➔ Begin transfer of these packets
- ➔ Start decoding at receiver end and play media while the media is still being delivered.

Until the first few have been delivered successfully, receiver must wait. But it is acceptable as receiver need not wait until the whole media is downloaded.

There are some time constraints which should be achieved for best quality of service (QoS). These constraints totally depend on the streaming application. For example, let's take a video streaming application. The next frame yet to be played should be received and decoded before its playback time. If only a very few frames are lost or not received by the frame's playback time, only a fraction seconds of the video is lost which hardly is recognized by the viewer. But, if series of frames are lost or not received by their playback time then receiver must wait for those frames to be delivered which usually effects Quality of service. Another example is live streaming or interactive online gaming, in these types of applications receiving frame and decoding it before the frame's playback time is very crucial for quality of service. Any frame that is lost or not received before its playback time is of no use for the viewer. Frames received after its playback time is irrelevant and cannot be played to the user. But still these frames could help in predicting the current frame when using motion-compensated prediction.

Problems in media streaming

Best media streaming application has better Quality of Service (QoS) and better Quality of Service (QoS) depends on the system design of streaming application. We shall discuss media streaming over the internet, Internet does not provide guarantee on bandwidth allocation, loss rate and delay jitter which are dynamic and unknown to the internet. So, streaming media over the internet is difficult when dealing with dynamic and unknown characteristics:

- ➔ Bandwidth
- ➔ Delay Jitter
- ➔ Loss rate

In internet, bandwidth between source and destination is time varying. If sender send packets too fast than the bandwidth available, then congestion take place and the packets are dropped based on the approach used for congestion control, and the Quality of Service (QoS) drops drastically. If the sender sends packets slower than the bandwidth available, receiver cannot play the optimal bitrate media to the user. To solve this problem, we have to estimate the available bandwidth and send encoded media with appropriate bit rate to match with available bandwidth. Some more problems are added on top of this when we consider multi-cast media streaming, as the send must send one copy of data to multiple receivers. If some of the receivers got allocated more bandwidth during streaming, though they can get higher bit rate media they cannot. To solve these problems, different approaches has been used which are discussed in the following sections.

What is Delay Jitter? The variation of delay in end-to-end transmission of packet is called as delay jitter. At the receiver's end, media has to be decoded and displayed at a constant rate to play the media without any jerks, if any frames have occurred after its playback time, then the user may observe jerks in media viewing. This problem is solved using playout buffer which has also added extra delay in media playing. We shall discuss about playout buffer in the following sections.

Packet losses is one of the fundamental problems in internet, but it's one of the crucial problems when coming to media streaming applications. Because it directly effects the Quality of Service (QoS). Error in received packet is also considered as the same. If a packet is lost, we have no other option but retransmit or leave the packet. But if there is error in the received packet, we can use some error correction techniques to resolve error like

- ➔ Forward Error Correction
- ➔ Retransmission
- ➔ Error concealment

5 RATE CONTROL FOR DYNAMIC BANDWIDTHS

5.1 Using TCP for media streaming

There are many advantages when using TCP for media streaming. Rate control is achieved as receiver acknowledges received packets thus even in time varying bandwidths rate control is achieved. But there is a major issue we should be concerned about is we can't keep sending acknowledgement for every packet we received. Because it keeps the connection busy by utilizing the bandwidth for sending acknowledgement thus the throughput is decreased. Thus, most probably a frame cannot be received by the receiver by frame's playback time which is very crucial in streaming applications as discussed in above sections. So, better QoS cannot be achieved using TCP.

5.2 Using UDP for media streaming

Most of the streaming applications used UDP for content delivery. Streaming applications rely on UDP's packet delivery because UDP doesn't provide acknowledgement for the packet delivery or acknowledgement for error packet for retransmission. UDP doesn't provide rate control. As weren't sending acknowledgement for the packets, we're utilising the bandwidth optimally. These characteristics made UDP to use in real-time streaming applications. As the bandwidths are time-varying in packet network, we're still left with rate control problem. If somehow, we can achieve rate control while using UDP for packet delivery, then it's the best method to implement in real-time streaming applications.

5.3 Multi-coded stream

In this approach multiple bit rates of the same video are pre-encoded and stored at the sender or server end. Receivers can choose a bit rate for the streaming depending on its own bandwidth to the network [2]. Advantages of using this approach are, not much computation is needed to switch between different bit rates of media and no re-encoding is needed as we have done in *Transcoding* which is discussed in the following sections. But this approach still has some disadvantages. More storage space is required to store different copies of the media at different bit rates which increases the implementation cost in real life. As we discussed multicast streaming introduced more problems on top of challenges in media streaming. Let's discuss some techniques for multicast streaming using multi-coded stream.

5.3.1 Receiver switching while using multiple stream

In multicast streaming same copy of the media is sent to many numbers of receivers. But here we're storing some well-known bit rate pre-encoded so that the receiver can choose the corresponding bit rate based on their network speed [5]. So, source sends each bit rate media stream to a different group address and joining or leaving the group is up to the receiver. Receiver has to make their own choice and shift between the different ip-group addresses for different bit rate of media. Some more optimizations can be made on top of this by getting feedback reports less frequently form each stream.

5.4 Transcoding

Receiver can match its bit rate by re-encoding the media to the bit rate needed. Problem with re-encoding is lower quality is achieved when compared to encoded media directly to the desired bit rate because, encoding the media and decoding it lower a fraction of media quality but again encoding and decoding lowers media quality significantly. Additionally, re-encoding requires a lot of computation, and we should be doing that for every frame to match the desired bit rate which made this approach hard to use for real-time streaming applications. But, with some advancement in this approach we can use re-encode the frame to the desired bit rate without decoding and while working on the encoded frame. As most of the coders and decoders use similar compression algorithms, this is possible. Now, let's discuss how transcoding is used in multicast streaming.

5.4.1 Source rate adaption with single stream

In multicast streaming same copy of the media is sent to many numbers of receivers. In source rate adaption with single stream the source adjusts the bit rate depending on receiver's feedback. After receiving the feedback if source needs to adjust its bit rate, transcoding is used to match the desired bit rate. RTP/RTCP quality reports can also be the feedback [5]. This approach has many drawbacks, firstly how are we going to get feedback from the receivers and synchronize with them? And second is, if one of the receivers has lower bandwidth comparing with other receivers in multicast

then every other receiver has to suffer from media quality as all the group members going to receive the b1it rate which is acceptable by the receiver with lower bandwidth.

5.5 Layered compression

Layered compression is the most optimal way of encoding the media for non-interactive media streaming applications i.e., media should already be pre-encoded and stored. It consists of multiple layers of encoded frames, base layer stores all the important information of media i.e., with just the base layer, we can play the media with lowest bit rate possible. Multiple layers are like base layer and enhancement layers on top of the base layers, enhancement layers merging with below layers gives higher bit rate media. H.263, MPEG-2 and MPEG-4 are the commonly used compression standards which support layered encoding [2]. But the major disadvantage of this approach is time required to encode media is very high and lot of computation is needed. Let's also discuss how this is used in multicast streaming.

5.5.1 Layered multicast Streaming

Layered compression technique is used in layered multicast streaming, set of cumulative layers with each having different media bit rates are sent to different ip-multicast groups. Joining or leaving any multicast group is left up to the receiver. Receiver first joins the base layer and then tries to move forward with enhancement layers.

Layered encoding approaches are also used in router managed techniques. In this technique based on the congestion, router can process or drop the layer thus filtering the media to achieve highest bit rate for the current bandwidth. With time varying bandwidth, media bit rate changes accordingly.

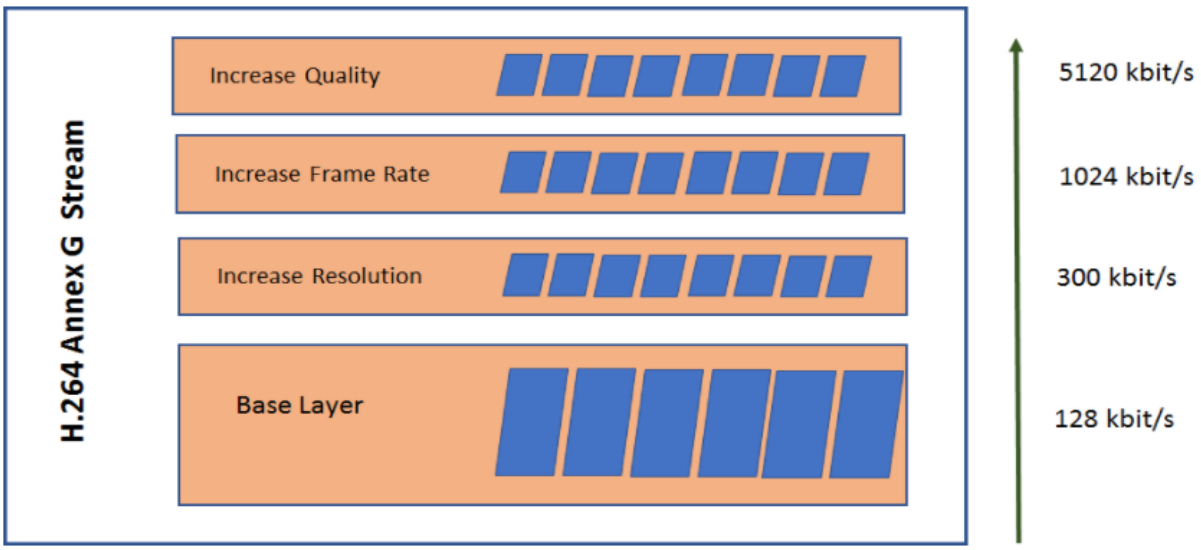

Figure 7: layered encoding [6]

The codec shown in the figure is H.264 Annex G which is Scalable Video Codec (SVC). With increase in the number of layers, bitrate gradually increases[2].

6 HOW PLAYOUT BUFFER SOLVES DELAY JITTER

A frame may be sent as multiple packets. There is no guarantee that the order of packets sent by the sender will be received in the same order and packet delay, due to variations in network conditions. As we already discussed that delay jitter will cause jerkiness in media playback. This problem is solved using playout buffer, which orders the packets and once all the packets of a frame are received before the frame's playback time, it will then decode the frame and plays media. Now, let's discuss how the playout buffer works.

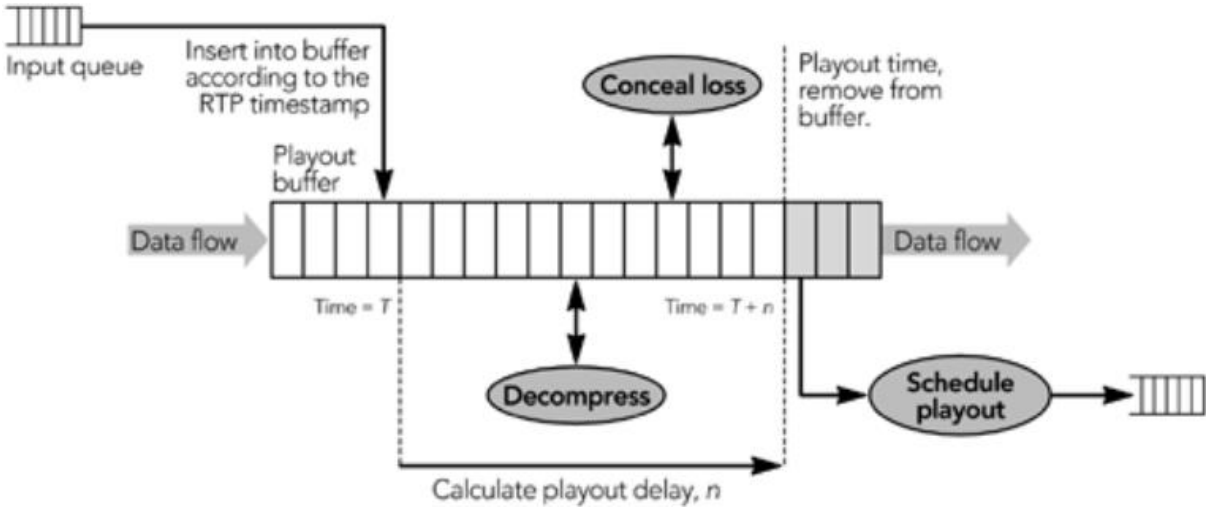

Figure 8: playout buffer [7]

Each packet received will be present in the input queue, each packet from front of the queue is extracted and placed in the playout buffer which is sorted based on the timestamp fetched from packet header, by doing this we can group the packets of a frame together based on timestamp in the packet header and when all the packets of a frame are received , we can decode it before it's playback time for viewing.

As there are multiple packets from each frame, the RTP marker bit is set for the last packet of the frame [7]. By receiving the frame with the RTP marker set, we cannot conclude that all the packets of a frame are received Successfully. So, highest unique sequence number is used instead of RTP marker, by using highest sequence number we can check if all the packets of a frame are received or not [7]. Once all the frame with sequence number lower than the highest sequence number with same timestamp are received it's up to the receiver when to decode the frame. Frame can be decoded as soon as it receives or can be kept in the buffer and decode it at the last moment. If user has more storage space to store decoded frames, then decoding as soon as the frame arrived could be the better option.

With playout buffer there some other advantages too, if any error is found in any frame and if we're not able to predict and correct the frame, then we can think of retransmission request if it could meet the playout time deadline. Like, if UDP is used for transport then, we shall send request to server for retransmission of frame. Also, as we know the bandwidth is time varying. Let's consider a case where the bandwidth become too low only for short period of time, playout buffer could sustain the media playing until the last frame received already in the buffer.

Playout buffer adds extra delay to start media playing or resume. But it is used in every streaming application because it solved the major delay jitter problem, and the additional delay is also accepted by the users rather than experiencing jerkiness in media playback.

7 ERROR CONTROL

As we are dealing with encoded frames in media streaming, every bit is crucial to decode and reconstruct the original media. Any system that doesn't deal with error control in media streaming cannot proving best Quality of Service (QoS). There are some different approaches on how the error control is achieved

- ➔ Retransmission
- ➔ Forward error correction
- ➔ Error concealment

7.1 Retransmission

In this approach the receiver sends the request to sender using the backchannel to retransmit the packets that has error. This approach uses the bandwidth efficiently because, we're only using bandwidth for requesting retransmission, actually retransmission and not wasting bandwidth on sending acknowledgement for every packet. The extra time we're spending here for error packet is Round Trip Time (RTT) i.e., from receiver to source for requesting retransmission and from source to receiver for actual retransmission. This approach isn't practical in multicast streaming and broadcast streaming as we cannot have backchannel from every receiver to source. Some further improvement can be made at the receiver end by requesting for retransmission only those packets which can successfully be transmitted before the frame's playback time by using packet scheduling algorithms.

7.2 Forward error correction

In Forward Error Correction (FEC) some extra redundant packets are added with the original packets to transmission for error correction [2]. Basically, this approach uses block code like Reed Solomon which take N packets as input and gives N+K packets as output. This approach has both advantages and disadvantages, advantage is that we no longer need to use back channel for feedback or retransmission thus making it practical for multicast streaming and broadcast streaming too. But the major disadvantage is that we're placing some extra redundant packets which eats up some bandwidth. We're transmitting redundant packets every time even though the packets received at the received end are correct by this the throughput is decreased.

7.3 Error Concealment

This approach basically predicts the lost blocks in a frame using spatial or temporal interpolation [2]. Thus, hiding the error as it never happened. This approach is mainly used in video encoding as discussed in the *media encoding* section. Let's take an example to understand this, if a single block of a frame is lost, we can simply assume that the block has zero amplitude and that block is displayed as a block filled with black color but, this can be easily identified by the user which is not preferred. Else, we can estimate the missing block by extrapolating the neighbouring blocks, this approach is called *Spatial interpolation.* This approach doesn't exactly recover the missing block but, it is preferred rather than showing a black block.

Another approach is, we can simply replace the current block with the same block number of the last successful frame, this approach is called *temporal Interpolation*. This approach works fine only when there isn't much movement between the frames which is most of the cases. But still if there is high movement between the frames then this is irrelevant. There is another approach used for predicting blocks when there is motion between the frames and is called as *Motion-Compensated temporal interpolation* which uses neighbouring blocks motion vector to predict the current missing block [2]. Motion-compensated temporal interpolation is the best method for error concealment over other two approaches.

Receiving Error frames can cause the decoder to lose the track of bit stream and thus don't know where to start decoding the frame again. This is called *loss of bitstream synchronization*, let's discuss some approaches to solve this problem.

7.4 Solving Loss of Bitstream Synchronization Problem

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 [2]. If there is a single bit error in the Variable Length Codeword (VLC), we cannot say where the next codeword would start and end as the codeword length is variable. Thus, the decoder has lost its sync with the bitstream and the whole bitstream data is unusable.

This problem is not seen when using Fixed Length Codeword (FLC) because, even if there is an error in the codeword, we know where the next codeword starts because the codeword length is static in this case. Thus, the decoder can start decoding it from the next starting location of codeword thus only the current frame bitstream is unusable. But best compression is achieved only when using VLC, so FLC isn't much appreciated.

Let's discuss different techniques used to resynchronize decoder with the bitstream while using VLC.

7.5 Resynchronization Markers

In this approach resync markers are placed at some unique locations in the bitstream. Resync markers are distinct from all the codewords and multiple copies of codeword placed to a side. So that when the decoder lost its sync with the bit stream, decoder can find for next resync marker and start decoding the frames. But the real challenge is where to place those resync markers to have minimal frames lost as well as not much overhead is added to packets by adding too many resync markers.

7.6 Using Reversible Variable Length Codewords (RVLCs)

RVLC are just as same a VLC but has the extra property, they can be decoded uniquely in the backward direction too. When the decoder lost its sync with the bit stream, it will go to the next resync marker and starts decoding in backward direction until it finds an error. Thus, partial data of a frame can also be utilised for media playback. Nevertheless, RVLC are typically less efficient than VLC [2].

8 STREAMING PROTOCOLS

The Internet Protocol (IP) provides the routing of packets, provides IP addresses which are mandatory for a packet to transmit in IP-network which are used by everyone. On top of that there are transport protocols which we'll discuss which are most relevant to media streaming.

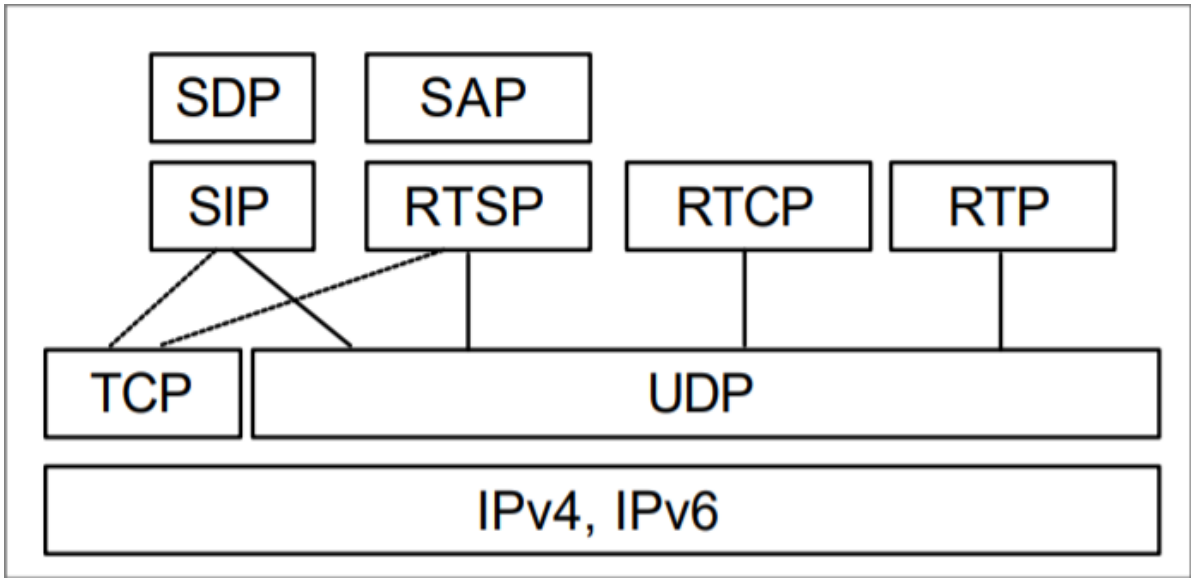

Figure 9: protocol stack used for media streaming [5]

As we already discussed in above sections that UDP for content delivery and TCP for control message is the best way to use for media streaming i.e., encoded media is delivered via UDP/IP and control information is delivered via TCP/IP.

Real-time Transport Protocol (RTP) and Real-time Control Protocol (RTCP) are designed by IETF to specially support media streaming. RTP is used for content transfer and RTCP for control messages. RTP is just like UDP, it doesn't provide guaranteed delivery of packet or guaranteed Quality of Service (QoS). RTP supports media steaming by providing functionalities such as sequence number, payload information and time stamp. RTCP is used in conjunction with RTP to send the feedback to the sender like number of packets lost, delay, RTT etc. RTCP sends feedback packets to the sender less frequently which may use up to 5% of the bandwidth [5]. These feedback messages are used by the sender to change bit rate to match with available bandwidth. It can also be used to get information about the other receivers in multicast or broadcast streaming.

Session Initiation Protocol (SIP) and Real-Time Streaming protocol (RTSP) are for maintaining and controlling media streaming sessions [5]. SIP is implemented at application layer which can establish new sessions, terminate current sessions, attach new session for the current session like group video calls. RTSP is very similar to SIP, but it adds extra functionalities such as play, seek, pause, and record.

Session Description Protocol (SDP) provides information which describes a session like, whether it is audio or video, bit rate of the media, duration of the media, also specifies codec.

Session Announcement Protocol (SAP) is an announcement protocol used to announce multicast streaming sessions or broadcast streaming sessions.

9 FUTURES OF STREAMING TECHNOLOGIES

There is lot of research going on in the field of fiber optics and satellite communication. As of now, fiber optic cables are relatively much faster than satellite communication. But satellite communication can reach people in the remote areas or to any location on the earth very efficiently. Establishing numbers of fiber optic cables to and from many different locations on earth would cost a huge amount of money.

As of now, we can use about 1600 different wavelengths in a single fiber optic cable which has diameter hardly as a single human hair. With the extensive research going on fiber optics, I guess we can use 10,000 different wavelengths in the future thus making fiber optic channel more reliable and we can assume that the delay for transmission is of no time. Speed of satellite communication can never reach the speed of fiber optic communication but, if enough number of satellites are placed in earth's orbit, we can achieve higher speed which may be possible in future.

Given a choice, we can use fiber optic channel communication if the receiver has access to fiber optic channel or else we can use satellite communication. With the help of these two emerging communications, streaming media can reach to every location on earth and can be reliable. Packet retransmissions becomes easier if that's the case.

Moreover, we have seen cost of storage systems in the past decade dropped drastically. Assuming it still gets lower in future, we can have multiple servers at many different locations on the earth which helps receiver to have faster access to the data as the distance a packet to be travelled can be reduced.

Also, with the ongoing research on media encoding and decoding techniques, in future we may be able to have best media encoding techniques which helps to resolve the block errors in a media frame more optimally with spatial and temporal interpolation. All these innovations can make a media streaming application more reliable and can provide best Quality of Service (QoS).

Gradually everyone is choosing media streaming applications over television as media streaming applications provides latest and wide categorized content to the users. Moreover, media streaming companies are investing in production to make their own content which is unique to each media streaming platform. Like amazon is premiering latest movies on their platform. With the help of user's data, streaming applications are providing more personalized ads whereas, television can personalize only on the viewer's location. With all these advantages, television may not sustain in near future.

Acronyms

AS	Application Server
DCT	Discrete Cosine Transform
FEC	Forward Error Correction
FLC	Fixed Length Codeword
HD	High Definition
IP	Internet Protocol
ISDN	Integrated Services Digital Network
MC	Motion Compensated
PSTN	Public Switch Telephone Network
QoS	Quality of Service
RTT	Round trip Time
RVLC	Reversible Variable Length Codewords
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
RTCP	Real-time Transport Control Protocol
SNR	Signal-to-Noise Ratio
SVC	Scalable Video Code
SIP	Session Initiation Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
VLC	Variable Length Codeword

References

I. https://www2.deloitte.com/global/en/pages/technology-media-and-telecommunications/articles/gx-future-of-tv-video.html
II. https://www.techtarget.com/searchnetworking/definition/Real-Time-Transport-Protocol
III. https://www.wowza.com/blog/udp-vs-tcp
IV. https://en.wikipedia.org/wiki/Real-time_Transport_Protocol

[1] Sachin Yadav, Ranjeeta Yadav, Shailendra Mishra, “Comparison between Centralized & Decentralized Overlay Networks for Media Streaming”, 2010.

[2] John G. Apostolopoulos, Wai-tian Tan, Susie J. Wee, "Video Streaming: Concepts, Algorithms, and Systems", pp. 2, 2002.

[3] https://ledgernote.com/blog/q-and-a/how-does-mp3-compression-work/

[4] https://www.maketecheasier.com/how-video-compression-works/

[5] Hatem Bettahar, " Tutorial on Multicast Video Streaming Techniques", 2005.

[6] https://medium.com/@vsachdeva/zoom-video-conf-tool-at-scale-e86289c290b8

[7] https://flylib.com/books/en/4.245.1.52/1/