BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2201

COMPUTER NETWORKING

Unit 13

Multimedia Networking

Table of Contents

1. INTRODUCTION

In the previous unit, we discussed World Wide Web and the Domain Name System. In this unit, we will discuss multimedia networking. Nowadays, we are using the web not only for e-mail or text data communication but also for audio and video distribution. People use sites like YouTube to upload and distribute their own videos. Network applications like Skype and Google talk allow people to make telephone calls and also enhance those calls with video and multi-person conferencing.

We begin this unit with a discussion on presentation formatting. In the next section, we will discuss multimedia data. In the last section, we will explore multimedia applications, streaming stored audio and video.

1.1 Objectives:

After studying this unit, you should be able to:

- *Describe presentation formatting*
- *Explain lossless compression techniques*
- *Describe image representation and compression*
- *Explain audio and video compression*
- *Describe streaming stored audio and video*

2. PRESENTATION FORMATTING

One of the most common transformations of network data is from the representation used by the application program into a form that is suitable for transmission over a network and vice versa. This transformation is typically called presentation formatting. The sending program translates the data it wants to transmit from the internal representation to a message that can be transmitted over the network. That is, we can say, data is encoded in a message. On the receiving side, the application translates this arriving message into a representation that it can then process. That is, the message is decoded. Figure 13.1 illustrated the process of encoding and decoding.

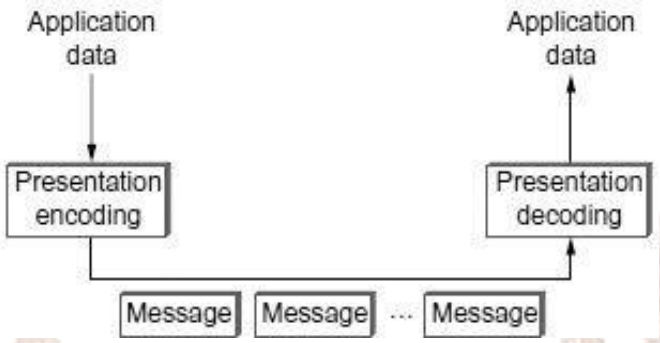


Fig 13.1: Presentation formatting involves encoding and decoding application data

Encoding is sometimes called *argument marshalling,* and decoding is sometimes called *unmarshalling.*

Taxonomy

We begin by giving simple taxonomy for argument marshalling systems.

Data types

We can classify the data types supported by an argument marshalling mechanism at three levels. At the lowest level, a marshalling system operates on some set of base types. Typically, the base types include integers, floating-point numbers, and characters. The system also supports ordinal types and Booleans. Next level are flat types such as structures and arrays. The problem with flat type argument marshalling is that compilers used to insert additional filling or line breaks between structure fields while compiling an application program. This is to align those fields on word boundaries. The marshalling system typically packs structures so that they contain no padding.

At the highest level, the marshalling system might have to deal with *complex types* such as those types that are built using pointers. A good example of a complex structure is a tree. Data structure that one program sends to another contains pointers from one structure to another. Pointers are implemented by memory address and the structure lives at a certain memory address on one machine does not mean it will live at the same address on another machine. So, a data encoder must prepare the data structure for transmission over the network.

Lowest Level:

- At the lowest level, the marshalling mechanism works on a collection of base types, at its most basic level. Integers, floating-point numbers and characters are some of the most common base types that the system uses at this level.
- A collection of base types implies that the encoding method must be capable of converting each base type from one representation to the next.

Example:

- Convert an integer from big-endian to little-endian.

Next Level

- At this level, the system uses flat types which include structures and arrays. Although the flat forms do not seem to complicate argument marshalling at the first glance, the truth is that they do.
- The problem is that the compilers used to compile application programs often add padding between the fields that make up the framework to match them on word boundaries.
- In the marshalling system, structures are typically packed without padding.

Highest Level

- At this level, the marshalling method may have to deal with the most complex form—those that are constructed with pointers.
- This means that the data structure that one program needs to transmit to another program may not be stored in a single structure, instead it may consist of pointers from one structure to another.
- A tree is an excellent example of a complex type of pointer.

Conversion strategy

After establishing the data type, the next issue is which conversion strategy the argument marshaller will use. There are two options. They are *canonical intermediate form and receiver-makes-right.* The idea behind the canonical intermediate form is to settle on an external representation for each type. The sender translates from its internal representation to this external representation before sending data and the receiving host translates from this external representation into its local representation when receiving data. For example, consider the integer data. Suppose you use the big-endian format as the external representation for integers. The sending host must translate each integer it sends into big-endian form, and the receiving host must translate big-endian integers into whatever representation it uses.

Another alternative is that the sending host will send data in its format, the receiver is then responsible for translating the data from the sender's format into its local format. In this case, every receiver must be prepared to convert data from all other machine architectures. This is known as an N-by-N solution. That is, each of the N machine architectures must be able to handle all N architectures. But, in a system that uses a canonical intermediate form, each host needs to know how to convert between its representation and a single other external representation. The drawdrawbackthese method is that there are not many representations for the various base classes. That is, we can say N is not that large. Also, it is common that two machines of the same type can communicate with each other. In this situation, it is a waste to translate data from that architecture's representation into some foreign external representation, only to have to translate the data back into the same architecture's representation on the receiver.

Another option can also be used, that is to use receiver-makes-right if the sender knows that the destination has the same architecture and the sender would use some canonical intermediate form if the two machines use different architectures. The sender will get information about receiver's architecture either from a name server or by first using a simple test case to see if the appropriate result occurs.

Tags

In argument marshalling, the issue is how the receiver knows what kind of data is contained in the message it receives. There are two common approaches. They are *tagged* and *untagged* data. The tagged approach is more spontaneous. A tag is any additional information included in a message beyond the practical representation of the base types. Tags help the receiver decode the message. Several possible tags might be included in a message. For instance, each data might be added with a *type tag*. A type tag indicates that the value that follows is an integer or a floating-point number etc. Another tag is the *length tag* which is used to indicate the number of elements in an array or the size of an integer. Another example is an *architecture tag*, which is used in conjunction with the *receiver-makes-right* strategy to specify the architecture on which the data contained in the message was generated. Figure 13.2 shows how a simple 32-bit integer might be encoded in a tagged message.

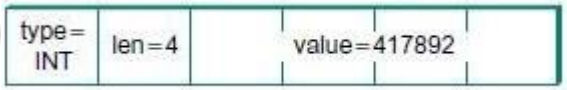


Fig 13.2: A 32-bit integer encoded in a tagged message

Another alternative encoding is without using tags. In this method, the receiver is programmed to know how to decode the data. If a remote procedure is called that takes two integers and a floating-point number as arguments, then there is no need for the remote procedure to check tags to know what it has received. It simply assumes that the message contains two integers and a float and decodes it accordingly. This assumption will work in most of these and the one place it fails is when sending variable-length arrays. In that case, a length tag is commonly used to indicate the length of the array. In the untagged method, the presentation formatting is truly ended to end. So, some intermediate agent can't interpret the message unless the data is tagged.

Stubs

A stub is a piece of code that implements argument marshalling. They are used to support remote procedure calls (RPC). On the client side, the stub marshals the procedure arguments into a message which can be transmitted using the RPC protocol. On the server, the stub converts the message back into a set of variables that can be used as arguments to call the remote procedure. Stubs can either be interpreted or compiled.

In a compilation-based approach, each procedure has a customized client and server stub. Stubs are generated by a stub compiler, based on a description of the procedure's interface. Stubs which are compiled are very efficient. In an interpretation-based approach, the system provides generic client and server stubs that have their parameters set by a description of the procedure's interface. Interpreted stubs have the advantage of being flexible because it is easy to change the descriptions. Compiled stubs are more common in practice.

SELF-ASSESSMENT QUESTIONS – 1

1. The most common transformation of network data are from the representation used by the application program into a form that is suitable for transmission over a network and *vice versa*. This transformation is known as.
2. Encoding is sometimes called, and decoding is called _____.
3. After establishing the type system, there are two conversion strategies the argument marshaller can use. They are _____ and____.
4. Any additional information included in a message beyond the practical representation of the base types is known as _____.
5. Which of the following d o e s n ' t belong to the category of a tag in a message
   a) Type tag
   b) length tag
   c) data tag
   d) architecture tag
6 _____is a piece of code thatimplements argument marshalling
7 In _____ approach, each procedure has a customized client and server stub.
8 In _____approach, the system provides generic client and server stubs.

3. MULTIMEDIA DATA

Multimedia data includes audio, video and images which constitutes the majority of traffic on the internet by many measures. In this section, we will discuss some major efforts in representing and compressing multimedia data. Compression is mainly performed for multimedia data and it is a lossy compression. The use of compression is not limited to multimedia data. We need to zip or compress data files also. This uses a typically lossless technique, because it avoids the loss of data from the data file. Lossy compression does not promise that the data received is the same as the data sent. The lossy algorithm achieves a better compression ratio than lossless.

To see the importance of compression, consider a high-definition TV screen which has 1080 x 1920 pixels, each of which has 24 bits of colour information. So, each frame is
1080 x 1920 x 24 =50Mb

So, if you want to send 24 frames per second, then it would be over 1 Gbps. Modern compression techniques can get a reasonably high-quality signal down to the range of 10Mbp. Similar compression gains apply to lower quality videos such as YouTube clips.

3.1 Lossless Compression Techniques

Compression is indivisible from data encoding. To encode a piece of data in a set of bits, we will think about how to encode the data in the smallest set of bits possible. For example, if there is a block of data that is made up of 26 symbols A through Z, and also if all these have an equal chance of occurring in the data block, then encoding each symbol in 5 bits is the best way to do the encoding. But suppose if symbol R occurs 50% of the time, then it is better to use fewer bits to encode the R than any of the other symbols. In general, if we know the relative probability that each symbol will occur in the data, then we can assign a different number of bits to each possible symbol in a way that minimizes the number of bits it takes to encode a given block of data. This is the essential idea of *Huffman codes* (one of the important early developments in data compression).

Run Length Encoding

Run length encoding (RLE) is a compression technique. The idea behind this technique is to replace consecutive occurrences of a given symbol with only one copy of the symbol, plus a count of how many times that symbol occurs, so it is called *run length*. For example, the string AAABBCDDDD would be encoded as 3A2B1C4D.

RLE appears to be useful for compressing some classes of images. It can be used in this context by comparing adjacent pixel values and then encoding only the changes. For images that have large homogeneous regions, this technique is rather effective. For example, it is common that RLE can achieve compression ratios on the order of 8-to-1 for scanned text images. RLE works well on such files because they often contain a large amount of white space that can be removed. RLE is the key compression algorithm used to transmit faxes.

Differential Pulse Code Modulation

Another simple lossless compression algorithm is Differential Pulse Code

Modulation (DPCM). The idea here is to first output a reference symbol. And then, for each symbol in the data, output the difference between that symbol and the reference symbol. For example, using symbol A as the reference symbol, the string AABBBCCDDD would be encoded as A0011122333 because A is the same as the reference symbol, B has a difference of 1 from the reference symbol, and so on. When the differences are small, they can be encoded with fewer bits. In the above example, the range of differences, 0–3, can be represented with 2 bits each, instead of the 7 or 8 bits required by the full character. Whenever the difference becomes too large, a new reference symbol is selected.

DPCM works well than RLE for most digital imaging because it takes advantage of the fact that adjacent pixels are usually similar. Due to this correlation, the dynamic range of the differences between the adjacent pixel values can be significantly less than the dynamic range of the original image, and this range can therefore be represented using fewer bits. We have measured compression ratios of 1.5-to-1 on digital images, using DPCM. DPCM also works on audio, because adjacent samples of an audio waveform are likely to be close in value.

A somewhat similar approach, called delta encoding, simply encodes a symbol as the difference from the previous one. So, for instance, AAABBCDDDD would be represented as A001011000. Delta encoding is likely to work well for encoding images where adjacent pixels are similar. It is also possible to perform RLE after delta encoding since we might find long strings of 0s if there are many similar symbols next to each other.

Dictionary-Based Methods

The final lossless compression method we consider is the dictionary-based approach. In this, the Lempel–Ziv (LZ) compression algorithm is the best known. The UNIX compress and gzip commands use variants of the LZ algorithm.

The concept of a dictionary-based compression algorithm is to create a dictionary (table) of variable-length strings that are expected to find in the data and then replace each of these strings when it appears in the data with the corresponding index to the dictionary. That means, instead of working with individual characters in text data, we treat each word as a string and output the index in the dictionary instead of that word. For example, the word compression has an index 4978 in one particular dictionary. Suppose it is the 4978th word in */usr/share/dict/words*. To compress a body of text, each time the string compression appears, it would be replaced by 4978. Since this particular dictionary has just over 25,000 words in it, it would take 15 bits to encode the index, meaning that the string "compression" could be represented in 15 bits rather than the 77 bits required by 7-bit ASCII. This is a compression ratio of 5-to-1.

We can either define a static dictionary, preferably one that is tailored for the data being compressed. A more general solution is to adaptively define the dictionary based on the contents of the data being compressed. This method is used by LZ compression. In this case, the dictionary constructed during compression has to be sent along with the data so that the decompression half of the algorithm can use this.

3.2 Image Representation and Compression (GIF, JPEG)

In response to the need for standard representation formats and compression algorithms for digital imagery data, the ISO defined a digital image format known as *JPEG*, which is named after the Joint Photographic Experts Group that designed it. *JPEG* is the most widely used format for still images. Many techniques used in *JPEG* are also there in *MPEG* (the set of

standards for video compression and transmission created by the Moving Picture Experts Group). At the heart of the definition of JPEG format is a compression algorithm.

As we know, the digital images are made up of pixels (you might have noticed the megapixel quoted in digital camera advertisements). Each pixel represents one location in the two-dimensional grid that makes up the image, and for colour images, each pixel has some numerical value representing a colour. There are lots of ways to represent colours, referred to as *colour spaces*. The most popular one is RGB (red, green, blue). The encoding and transmission of colour images require agreement between the two ends of the color space, otherwise, a different colour image will be displayed on the receiver. So, agreeing on a colour space definition is part of the definition of any image or video format.

Consider the example of Graphical Interchange Format (GIF). GIF uses the RGB colour space and starts with 8 bits to represent each of the three dimensions of colour for a total of 24 bits. Instead of sending 24 bits per pixel, GIF first reduces 24-bit colour images to 8-bit colour images. This is done by identifying the colours used in the picture, and then picking the 256 colours that most closely approximate the colours used in the picture. These colours will be picked in such a way that no pixel has its colour changed too much.

The 256 colours are stored in a table. This table can be indexed with an 8-bit number, and the value for each pixel is replaced by the appropriate index. This is an example of lossy compression for a picture with more than 256 colours. GIF then runs an LZ variant over the result, treating common sequences of pixels as the strings that make up the dictionary which is a lossless operation. Using this approach, GIF is sometimes able to achieve compression ratios on the order of 10:1, but only when the image consists of a relatively small number of discrete colours.

The JPEG format is considerably better suited to photographic images. JPEG does not reduce the number of colours like GIF.

3.3 Video Compression (MPEG)

MPEG is named after the Moving Picture Experts Group that defined it. A moving picture is simply a succession of still images which are also called frames or pictures displayed at some video rate. These frames can be compressed using similar techniques which are used in JPEG.

For example, two successive frames in a video contain similar information if there is not much movement in the scene. Even when there is motion, there may be plenty of redundancies since a moving object may not change from one frame to the next. MPEG considers this interframe redundancy.

Frame types

MPEG takes a sequence of video frames as input and compresses them into three types of frames called, *I frame* (intrapicture), *P frame* (predicted picture) and *B frame* (bidirectional predicted picture). Each input frame is compressed into one of these three frame types. I frame can be thought of as reference frames; they are self-contained, depending on neither earlier frames nor later frames. I frames are the JPEG compressed version of the corresponding frame in the video source. P and B frames are not self-contained, they specify relative differences from some reference frame. A P frame specifies the differences from the previous I frame, while a B frame gives an interpolation between the previous and subsequent I or P frames. Figure 13.3 shows a sequence of seven video frames that after being compressed by MPEG, result in a sequence of I, P and B frames. The two I frames stand alone; each can be decompressed at the receiver independently of any other frames. The P frame depends on the preceding I frame; it can be decompressed at the receiver only if the preceding I frame also arrives. Each of the B frames depends on both the preceding I or P frame and the subsequent I or P frame. Both of these reference frames must arrive at the receiver before MPEG can decompress the B frame to reproduce the original video frame.

Since each B frame depends on a later frame in the sequence, the compressed frames are not transmitted in sequential order. Instead, the sequence "I B B P B B I" shown in Figure 13.3 is transmitted as "I P B B I B B". Also, MPEG does not define the ratio of I frames to P and B frames; this ratio may vary depending on the required compression and picture quality.

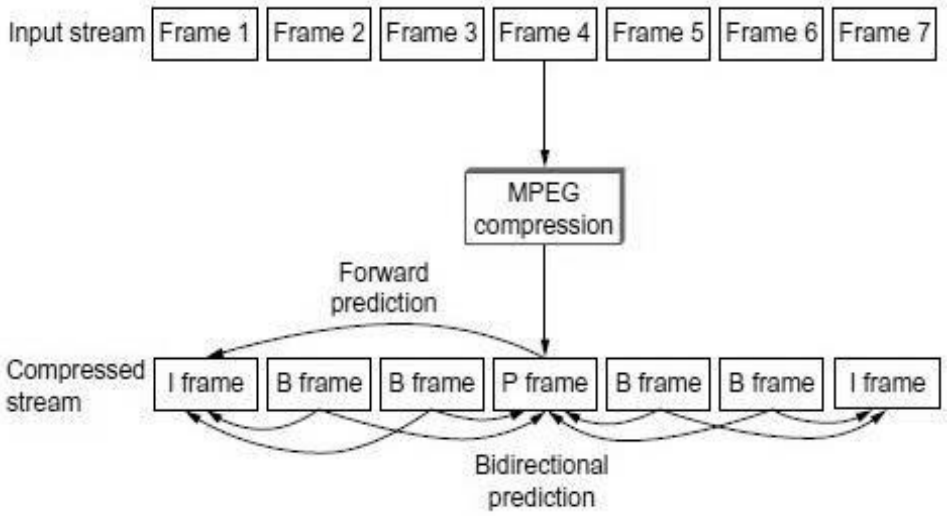


Fig 13.3: Sequence of I, P and B frames generated by MPEG

Now we will discuss the decoding of an *MPEG* stream, which is the operation that is most often implemented in networking systems today. Since MPEG coding is very expensive, it is frequently done offline. For example, in a video-on-demand system, the video would be encoded and stored on disk ahead of time. When a user wanted to watch video, the MPEG stream would then be transmitted to the viewer's machine, which would decode and display the stream in real time.

Effectiveness and Performance

MPEG achieves a compression ratio of 90:1. In terms of the individual frame type, we can expect a compression ratio of about 30:1 for the 'I' frames while P and B frame compression ratios are typically three to five times smaller than the rates for the 'I' frame. Without first reducing the 24 bits of color to 8 bits, the achievable compression with *MPEG* is typically between 30:1 and 50:1. In recent years, processors are fast enough to keep pace with 30 frames per second video rates when decoding *MPEG* streams purely in software. Modern processors can decode *MPEG* streams of high-definition video (HDTV).

Other video encoding standards

ITU-T has defined the H series for encoding real-time multimedia data. H series includes standards for video, audio, control, and multiplexing. Within the H *series,* H.261 and H.263 are the first- and second-generation video encoding standards. Unlike early versions of

MPEG, which is targeted at bit rates on the order of 1.5Mbps, H.261 and H.263 are targeted at lower speeds. The newer H.264 standard is a part of MPEG -4 standard.

3.4 Transmitting MPEG over a Network

MPEG can be used for videos stored on disk and also videos transmitted over a stream-oriented network connection because *MPEG* defines the format of a video stream, but does not specify how this stream is broken into network packets. One advantage of MPEG format is that it gives the encoder the opportunity to change the encoding over time. It can change the frame rate, the resolution, the mix of frame types that define a GOP (Groups of Pictures), the quantization table and the encoding used for individual macroblocks (*MPEG* works in units of 16×16 macroblocks). As a result, it is possible to adapt the rate at which a video is transmitted over a network by trading picture quality for network bandwidth.

Another interesting aspect of sending an MPEG stream over the network is exactly how the stream is broken into packets. Packetization is not an issue, when sending over a TCP connection. TCP decides when it has enough bytes to send the next IP datagram. But, when using video interactively, it is rare to transmit it over TCP. If we are transmitting video using UDP, then it makes sense to break the stream at carefully selected points, such as at macroblock boundaries. This helps to limit the effects of a lost packet to a single macroblock, rather than damaging several macroblocks with a single loss.

Packetizing the stream is only the first problem in sending *MPEG* compressed video over a network. The next issue is how to deal with packet loss. If a 'B' frame is dropped by a network, then it is possible to simply replay the previous frame without seriously compromising the video. But, a lost 'I' frame has serious consequences. None of the subsequent 'B' and 'P' frames can be processed without it. So, losing an 'I' frame would result in losing multiple frames of the video. We could retransmit the missing 'I' frame, but the resulting delay would probably not be acceptable in a real¬time videoconference. Because of this reason, many videoconference applications encode video using JPEG, which is often called motion-JPEG. One solution to this problem would be the use of Differentiated Services to mark the packets containing 'I' frames with a lower drop probability than other packets. An interframe encoding that depends upon only prior frames rather than later frames is not a problem, and would work well for interactive videoconferencing.

3.5 Audio Compression (MP3)

MPEG defines a standard for compressing audio. This standard can be used to compress the audio portion of a movie or it can be used to compress stand-alone audio (for Eg: an audio CD). CD quality audio, which is the real digital representation for high-quality audio, is sampled at a rate of 44.1 KHz. Each sample is 16 bits, so a stereo audio stream results in a bit rate of *2x 44.1 x 1000 x 16 = 1.41 Mbps.*

On contrast, telephone-quality voice is sampled at a rate of 8 KHz, with 8-bit samples, resulting in a bit rate of 64 kbps. Some amount of compression is required to transmit CD-quality audio over the 128-kbps capacity of an ISDN data/voice line pair. To encode each 16-bit sample, an additional 49 bits is required for synchronization and error correction, which results in an actual bit rate of *49/16 x 1.41 Mbps = 4.32 Mbps*

MPEG addresses this requirement by defining three levels of compression as shown in table 13.1. Out of these layers, layer III, which is widely known as MP3 is the most commonly used.

Table 13.1: MP3 Compression Rates

Coding	Bit Rates	Compression Factor
Layer I	384 kbps	4
Layer II	192 kbps	8
Layer III	128 kbps	12

To achieve these compression ratios, MP3 uses techniques that are similar to those used by MPEG to compress video. First, it splits the audio stream into some number of frequencies sub bands. Second, each sub band is broken into a sequence of blocks. Finally, each block is transformed using a modified DCT (discrete cosine transform) algorithm, then it is quantized, and then Huffman encoded, just as for MPEG video.

The trick to MP3 is how many sub bands it chooses to use and how many bits it allocates to each sub band. After compression, the sub bands are packaged into fixed-size frames, and a header is attached. This header includes synchronization information, as well as the bit allocation information needed by the decoder to determine how many bits are used to

encode each sub band. These audio frames can then be interleaved with video frames to form a complete *MPEG stream.*

SELF-ASSESSMENT QUESTIONS – 2

9. Compression which is commonly used for multimedia data is
10. _____. The idea behind Run Length Encoding is to replace consecutive occurrences of a given symbol with only one copy of the symbol, plus a count of how many times that symbol occurs. State True or False.
    a. True (b) false
11. The idea in _____ is to first output a reference symbol and then, for each symbol in the data, to output the difference between that symbol and the reference symbol.
12. The Lempel–Ziv (LZ) compression algorithm is a_____ approach.
13. Digital image format, JPEG is named after _____that designed it.
14. MPEG is created by _____.

4. STREAMING STORED AUDIO AND VIDEO

Here, we focus on streaming stored video which typically combines video and audio components. Streaming stored audio is very similar to streaming stored video, even though the bit rates are typically much lower. Consider the case where the underlying medium is a pre-recorded video, such as a movie, a television show, a pre-recorded sporting event, or a pre-recorded user generated video.

Pre-recorded videos are placed on servers and users can send requests to the servers to view the videos on demand. You tube is an example of an Internet company which provides streaming video. Today, over 50 percent of the downstream traffic in the Internet access networks is made up by streaming stored video. Streaming stored video has three key distinguishing features. They are:

- Streaming: In a streaming stored video application, the client typically begins video playout within a few seconds after it begins receiving the video from the server. This means that the client will be playing out from one location in the video while at the same time receiving later parts of the video from the server. This technique, known as ***streaming,*** avoids downloading the entire video file before playout begins.
- Interactivity: Since the media is pre-recorded, the user may pause, reposition forward, reposition backward, fast-forward, and so on through the video content. The time from when the user makes such a request until the action manifests itself at the client should be less than a few seconds for acceptable responsiveness.
- Continuous playout: once the playout of the video starts, it should proceed according to the original timing of the recording. Therefore, data must be received from the server in time for its playout at the client. Otherwise, users experience video frame freezing (when the client waits for the delayed frames) or frame skipping (when the client skips over delayed frames).

We can say, the most important performance measure for streaming video is average throughput. In order to provide continuous playout, the network must provide an average throughput to the streaming application that is at least as large as the bit rate of the video itself. By using buffering and prefetching we can provide continuous playout even when the throughput varies, as long as the average throughput remains above the video rate.

For many streaming video applications, pre-recorded video is stored on, and streamed from, a Content Distribution Network (CDN) rather than from a single data centre. There are also many peer-to-peer video streaming applications for which the video is stored on users' hosts (peers), with different chunks of video arriving from different peers that may spread around the globe.

In the case of streaming video applications, pre-recorded videos are placed on servers and user can send request to these servers to view the video whenever they want. Users can watch the entire video or can stop watching in the middle, or interact with video by pausing and repositioning to a previous or future scene. Streaming video systems can be classified into three categories such as UDP streaming, HTTP streaming and adaptive HTTP streaming. Today, the majority of systems use HTTP streaming and adaptive HTTP streaming.

For streaming a video, the user generally can tolerate a small initial delay between the request and the beginning of the video play. When the video starts to arrive at the client, the client can reserve video in an application buffer and play later after several seconds. Two advantages are there due to such client buffering. First, this buffering can absorb variations in server-to-client delay. Another advantage is that, if the server to client bandwidth drops below video consumption rate, then also a user can continue to enjoy continuous playback.

Functionality of a Media Player :

Decompression:

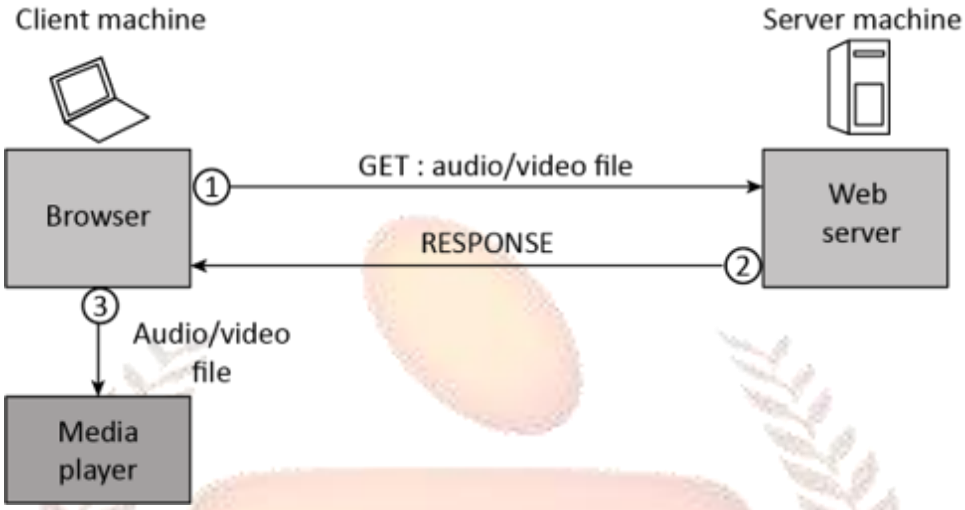
To save disc space and network bandwidth, audios and videos are mostly compressed.

Jitter removal:

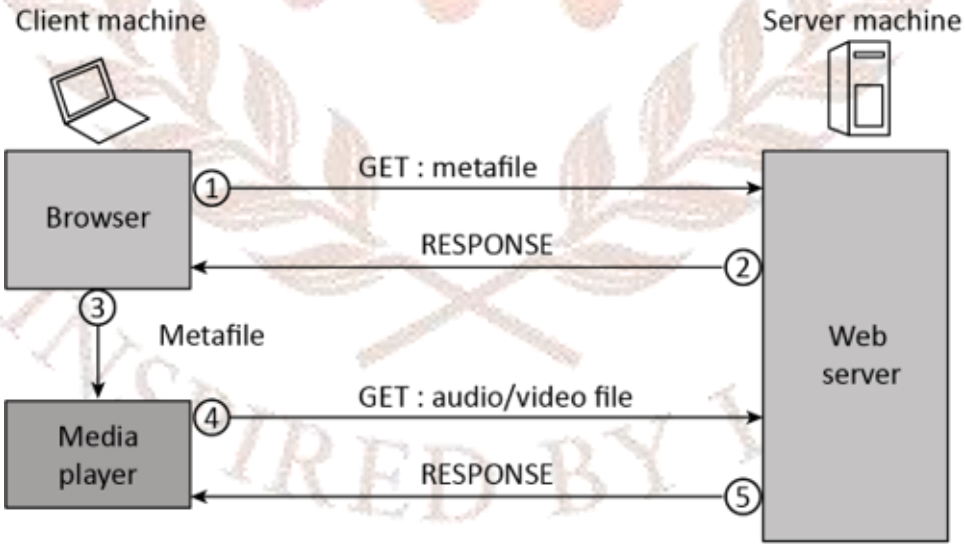
The variability of packet source-to-destination delays within the same packet stream. This is known as packet jitter.

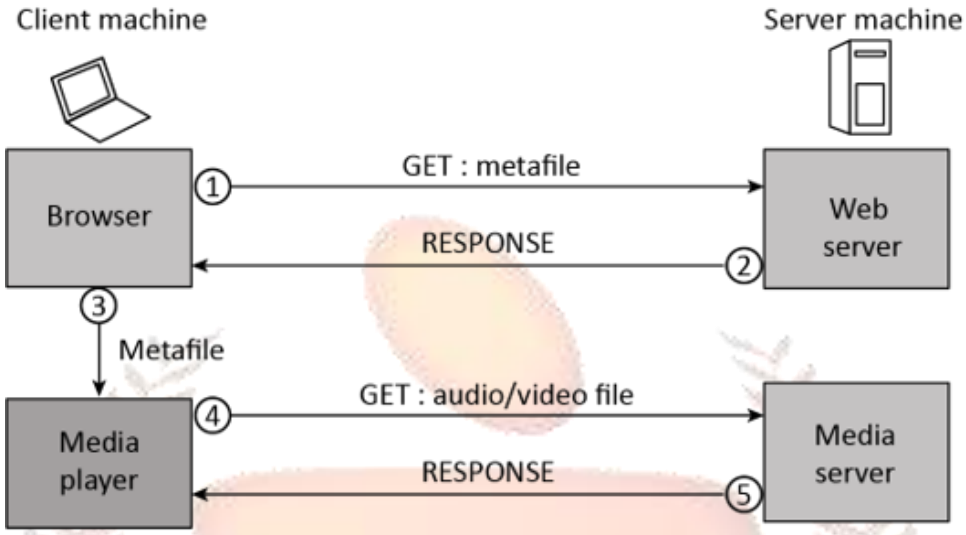
Error correction:

A portion of packets in the packet stream may be lost due to unpredictably high internet traffic. This can be corrected using this functionality.



- You can download a compressed audio/video file as a text file.
- To download the file, the client browser uses HTTP and sends a GET message.
- The web server sends the compressed file to the browser, which then plays the file using a helper application called a media player.
- Before you can play the file, you must download it completely.





SELF-ASSESSMENT QUESTIONS – 3

15. The client will be playing out from one location in the video while at the same time receiving later parts of the video from the server. This technique is known as _____.
16. For many streaming video applications, pre-recorded video is stored on, and streamed from a _____.
17. Streaming video systems can be classified into three categories such as _____, _____and _____.

5. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- The most common transformations of network data are from the representation used by the application program into a form that is suitable for transmission over a network and vice versa. This transformation is typically called presentation formatting.
- Compression which is commonly used for multimedia data is lossy compression.
- Run length encoding (RLE) is a compression technique. The idea behind this technique is to replace consecutive occurrences of a given symbol with only one copy of the symbol, plus a count of how many times that symbol occurs, so it is called run length.
- Based on the need for standard representation formats and compression algorithms for digital imagery data, the ISO defined a digital image format known as JPEG, which is named after the Joint Photographic Experts Group that designed it.
- MPEG is named after the Moving Picture Experts Group that defined it.
- A moving picture is simply a succession of still images which are also called frames or pictures displayed at some video rate.
- MPEG takes a sequence of video frames as input and compresses them into three types of frames called, I frames (intrapicture), P frames (predicted picture) and B frames (bidirectional predicted picture).
- The client will be playing out from one location in the video while at the same time receiving later parts of the video from the server. This technique is known as streaming.

6. TERMINAL QUESTIONS

1. Explain presentation formatting.
2. Differentiate between Run length encoding and Differential Pulse Code Modulation.
3. Describe Dictionary-Based Methods.
4. Explain video compression techniques.
5. Describe streaming stored audio and video.

7. ANSWERS

Self-Assessment Questions

1. Presentation formatting
2. Argument marshalling, Unmarshalling
3. Canonical intermediate form, receiver-makes-right
4. Tag
5. (c) data tag
6. Stub
7. Compilation-based
8. Interpretation- based
9. Lossy compression
10. (a) True
11. Differential pulse code modulation (DPCM)
12. Dictionary based approach
13. Joint photographic experts group
14. Moving picture experts group
15. Streaming
16. Content distribution network
17. UDP streaming, HTTP streaming, Adaptive HTTP streaming

Terminal Questions

1. One of the most common transformations of network data is from the representation used by the application program into a form that is suitable for transmission over a network and *vice versa*. This transformation is typically called *presentation formatting*... (Refer section 2 for more details).

2. Run length encoding (RLE) is a compression technique. The idea behind this technique is to replace consecutive occurrences of a given symbol with only one copy of the symbol, plus a count of how many times that symbol occurs, so it is called *run length*. Another simple lossless compression algorithm is Differential Pulse Code Modulation (DPCM). The idea here is to first output a reference symbol and then, for each symbol in the data, to output the difference between that symbol and the reference symbol. (Refer section 3.1 for more details).
3. The concept of a dictionary-based compression algorithm is to create a dictionary (table) of variable-length strings that is expected to find in the data and then to replace each of these strings when it appears in the data with the corresponding index to the dictionary. (Refer section 3.1 for more details).
4. MPEG is named after the Moving Picture Experts Group that defined it. A moving picture is simply a succession of still images which is also called frames or pictures displayed at some video rate. These frames can be compressed using similar techniques which is used in JPEG. (Refer section 3.3 for more details).
5. Streaming stored audio is very similar to streaming stored video, even though the bit rates are typically much lower. Consider the case where the underlying medium is a pre-recorded video, such as a movie, a television show, a pre-recorded sporting event, or a pre-recorded user generated video. (Refer section 4 for more details).

References

- Andrew S Tanenbaum, David J.Wetherall, *"Computer Networks,"* Fifth edition.
- Larry L. Peterson, Bruce S. Davie, *"Computer Networks- a Systems Approach,"* Fifth edition.
- James F. Kurose, Keith W.Ross, *"Computer Networking-A top-down approach,"* Sixth edition.
- Behrouz A.Forouzan, Sophia Chung Fegan, *"Data Communication and Networking,"* Fourth edition.
- William Stallings, *"Computer Networking With Internet Protocols and Technology,"* Third edition.