# 5.1 Multimedia Networking Applications

- A multimedia network application as any network application that employs audio or video.
- This section provides a taxonomy of multimedia applications.
- It is useful to consider the intrinsic characteristics of the audio and video media.

## 5.1.1 Properties of Video

- Perhaps the most salient characteristic of video is its high bit rate.
- Video distributed over the Internet typically ranges from 100 kbps for low-quality video conferencing to over 3 Mbps for streaming high-definition movies.
- Among different media classes, video has comparatively high resource requirements.
- Consider three different users, each using a different Internet application.
- Our first user, Frank, is going quickly through photos posted on his friends' Facebook pages. Let's assume that Frank is looking at a new photo every 10 seconds, and that photos are on average 200 Kbytes in size (assumption that 1 Kbyte = 8,000 bits.).
- Our second user, Martha, is streaming music from the Internet ("the cloud") to her smartphone. Let's assume Martha is listening to many MP3 songs, one after the other, each encoded at a rate of 128 kbps.
- Our third user, Victor, is watching a video that has been encoded at 2 Mbps. Finally, let's suppose that the session length for all three users is 4,000 seconds (approximately 67 minutes).
- Below table compares the bit rates and the total bytes transferred for these three users.

| | Bit rate | Bytes transferred in 67 min |
|---|---|---|
| **Facebook Frank** | 160 kbps | 80 Mbytes |
| **Martha Music** | 128 kbps | 64 Mbytes |
| **Victor Video** | 2 Mbps | 1 Gbyte |

- We see that video streaming consumes by far the most bandwidth, having a bit rate of more than ten times greater than that of the Facebook and music-streaming applications.
- Another important characteristic of video is that it can be compressed, thereby trading off video quality with bit rate.
- A video is a sequence of images, typically being displayed at a constant rate, for example, at 24 or 30 images per second.
- An uncompressed, digitally encoded image consists of an array of pixels, with each pixel encoded into a number of bits to represent luminance and color. A pixel of color image will be 24-bit in size.
- There are two types of redundancy in video, both of which can be exploited by video compression.
    - *Spatial redundancy* is the redundancy within a given image. Intuitively, an image that consists of mostly white space has a high degree of redundancy and can be efficiently compressed without significantly sacrificing image quality.
    - *Temporal redundancy* reflects repetition from image to subsequent image. If, for example, an image and the subsequent image are exactly the same, there is no reason to re-encode the subsequent image; it is instead more efficient simply to indicate during encoding that the subsequent image is exactly the same.
- Of course, the higher the bit rate, the better the image quality and the better the overall user viewing experience. We can also use compression to create multiple versions of the same video, each at a different quality level.
- For example, we can use compression to create, say, three versions of the same video, at rates of 300 kbps, 1 Mbps, and 3 Mbps.
- Users can then decide which version they want to watch as a function of their current available bandwidth.
- Users with high-speed Internet connections might choose the 3 Mbps version; users watching the video over 3G with a smartphone might choose the 300 kbps version.

- Similarly, the video in a video conference application can be compressed "on-the-fly" to provide the best video quality given the available end-to- end bandwidth between conversing users.

## 5.1.2 Properties of Audio

- Digital audio (including digitized speech and music) has significantly lower bandwidth requirements than video.
- Digital audio, however, has its own unique properties that must be considered when designing multimedia network applications.
- To understand these properties, let's first consider how analog audio (which humans and musical instruments generate) is converted to a digital signal:

  • The analog audio signal is sampled at some fixed rate, for example, at 8,000 samples per second. The value of each sample is an arbitrary real number.

  • Each of the samples is then rounded to one of a finite number of values. This operation is referred to as **quantization**. The number of such finite values—called quantization values—is typically a power of two, for example, 256 quantization values.

  • Each of the quantization values is represented by a fixed number of bits. For example, if there are 256 quantization values, then each value—and hence each audio sample—is represented by one byte (8-bit). The bit representations of all the samples are then concatenated together to form the digital representation of the signal.

- This basic encoding technique is called pulse code modulation (PCM).

***Read for a better understanding…***

*As an example, if an analog audio signal is sampled at 8,000 samples per second and each sample is quantized and represented by 8 bits, then the resulting digital signal will have a rate of 64,000 bits per second.*

*For playback through audio speakers, the digital signal can then be converted back—that is, decoded—to an analog signal. However, the decoded analog signal is only an approximation of*

*the original signal, and the sound quality may be noticeably degraded (for example, high-frequency sounds may be missing in the decoded signal).*

- By increasing the sampling rate and the number of quantization values, the decoded signal can better approximate the original analog signal.
- Thus (as with video), there is a trade-off between the quality of the decoded signal and the bit-rate and storage requirements of the digital signal.
- Instead, as with video, compression techniques are used to reduce the bit rates of the stream.
- Human speech can be compressed to less than 10 kbps and still be intelligible.
- A popular compression technique for near CD-quality stereo music is MPEG 1 layer 3, more commonly known as MP3.
- A related standard is Advanced Audio Coding (AAC), which has been popularized by Apple.

## 5.1.3 Types of Multimedia Network Applications

- The Internet supports a large variety of useful and entertaining multimedia applications. In this subsection, we classify multimedia applications into three broad categories:
  - (i) Streaming stored audio/video
  - (ii) Conversational voice/video-over-IP
  - (iii) Streaming live audio/video.
- Each of these application categories has its own set of service requirements and design issues.

### a) Streaming Stored Audio and Video

Let's consider streaming stored video, which typically combines video and audio components.

In this class of applications, the underlying medium is prerecorded video, such as a movie, a television show, a prerecorded sporting event, or a prerecorded user generated video (such as those commonly seen on YouTube).

These prerecorded videos are placed on servers, and users send requests to the servers to view the videos on demand.

Many Internet companies today provide streaming video, including YouTube (Google), Netflix, and Hulu.

Streaming stored video has three key distinguishing features.

- *Streaming.* In a streaming stored video application, the client typically begins video play-out 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 having to download the entire video file (and incurring a potentially long delay) before playout begins.

- *Interactivity.* Because the media is prerecorded, the user may pause, reposition forward, reposition backward, fast-forward, and so on through the video content. This is referred to as interactivity

- *Continuous play-out.* Once play-out of the video begins, it should proceed according to the original timing of the recording. Therefore, data must be received from the server in time for its play-out 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).

**b) Conversational Voice- and Video-over-IP**

Real-time conversational voice over the Internet is often referred to as Internet telephony. It is also commonly called Voice-over-IP (VoIP).

Conversational video is similar, except that it includes the video of the participants as well as their voices.

Most of today's voice and video conversational systems allow users to create conferences with three or more participants (Skype, Whatsapp, etc.).

Two important factors to be considered in this regard:—

a) Timing considerations

b) Tolerance of data loss

- Timing considerations are important because audio and video conversational applications are highly delay-sensitive.
- For a conversation with two or more interacting speakers, the delay from when a user speaks or moves until the action is manifested at the other end should be less than a few hundred milliseconds.
- For voice, delays smaller than 150 milliseconds are not perceived by a human listener, delays between 150 and 400 milliseconds can be acceptable, and delays exceeding 400 milliseconds can result in frustrating, if not completely unintelligible, voice conversations.
- On the other hand, conversational multimedia applications are loss-tolerant— occasional loss only causes occasional glitches in audio/video playback, and these losses can often be partially or fully concealed.
- These delay-sensitive but loss-tolerant characteristics are clearly different from those of elastic data applications such as Web browsing, e-mail, social networks, and remote login.

**c) Streaming Live Audio and Video**

- This third class of applications is similar to traditional broadcast radio and television, except that transmission takes place over the Internet.
- These applications allow a user to receive a live radio or television transmission—such as a live sporting event or an ongoing news event—transmitted from any corner of the world.
- Live, broadcast-like applications often have many users who receive the same audio/video program at the same time.
- Live distribution of video contents are accomplished using P2P networks or CDNs.
- The network must provide each live multimedia flow with an average throughput that is larger than the video consumption rate.
- Because the event is live, delay can also be an issue, although the timing constraints are much less stringent than those for conversational voice. Delays of up to ten seconds or so from when the user chooses to view a live transmission to when play-out begins can be tolerated.

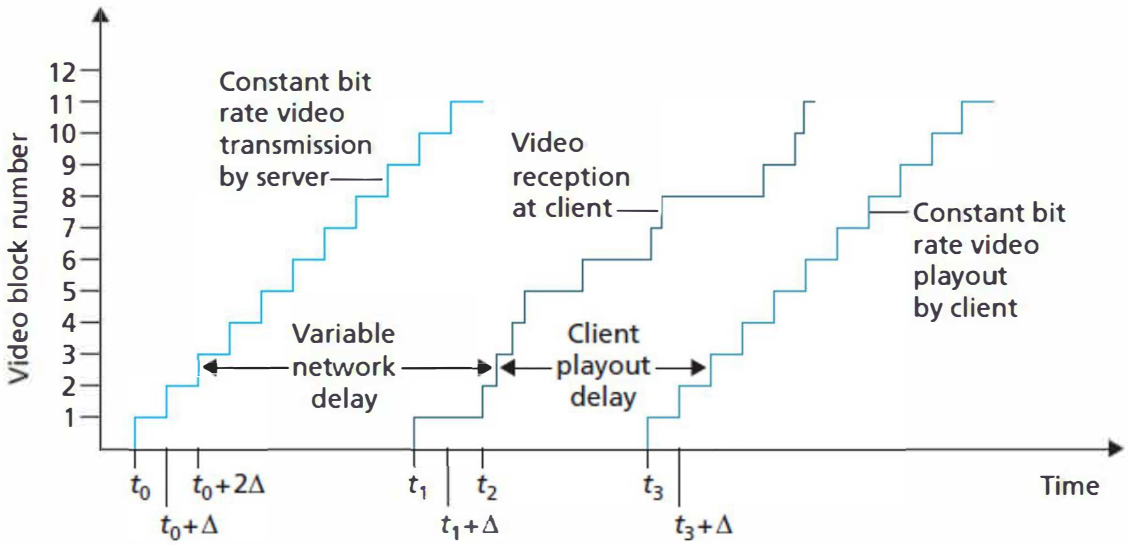
**Probable Question:** ***Explain various types of Multi-media Network Applications***

## 5.2 Streaming Stored Video

- For streaming video applications, prerecorded videos are placed on servers, and users send requests to these servers to view the videos on demand.
- The user may watch the video from beginning to end without interruption, may stop watching the video well before it ends, or interact with the video by pausing or repositioning to a future or past scene.
- Streaming video systems can be classified into three categories:
  a) UDP streaming
  b) HTTP streaming
  c) Adaptive HTTP streaming.
- Although all three types of systems are used in practice, the majority of today's systems employ HTTP streaming and adaptive HTTP streaming.
- A common characteristic of all three forms of video streaming is the extensive use of client-side application buffering to mitigate the effects of varying end-to-end delays and varying amounts of available bandwidth between server and client.
- For streaming video (both stored and live), users generally can tolerate a small several second initial delay between when the client requests a video and when video play-out begins at the client.
- Consequently, when the video starts to arrive at the client, the client need not immediately begin play-out, but can instead build up a reserve of video in an application buffer.
- Once the client has built up a reserve of several seconds of buffered-but-not-yet-played video, the client can then begin video play-out. This method is referred to as client buffering.
- There are two important advantages provided by such client buffering.
  1) Client side buffering can absorb variations in server-to-client delay.

2) User continues to enjoy the content, even if the server-to-client bandwidth briefly drops below the video consumption rate.

- The graph shown below depicts the relativity between video frames being streamed, buffered at client end and displayed to user.
- Transmission rate is constant but fluctuations in video reception rate and play-out rate is also constant.




## 5.2.1 UDP Streaming

- With UDP streaming, the server transmits video at a rate that matches the client's video consumption rate by clocking out the video chunks over UDP at a steady rate.
- For example, if the video consumption rate is 2 Mbps and each UDP packet carries 8,000 bits of video, then the server would transmit one UDP packet into its socket every (8000 bits)/(2 Mbps) = 4 msec.
- Because UDP does not employ a congestion-control mechanism, the server can push packets into the network at the consumption rate of the video without the rate-control restrictions of TCP.
- UDP streaming typically uses a small client-side buffer, big enough to hold less than a second of video.

- Another distinguishing property of UDP streaming is that, a separate control connection over which the client sends commands regarding session state changes (such as pause, resume, reposition, and so on).
- UDP streaming suffers from three significant drawbacks.
    1) Due to the unpredictable and varying amount of available bandwidth between server and client, constant-rate UDP streaming can fail to provide continuous play-out.
    2) It requires a media control server to process client-to-server interactivity requests and to track client state (e.g., the client's playout point in the video, whether the video is being paused or played, and so on) for each ongoing client session. This increases the overall cost and complexity of deploying a large-scale video-on-demand system.
    3) It cannot bypass well configured firewalls.

**Probable Question:** ***Explain UDP streaming. What are its drawbacks?***

## 5.2.2 HTTP Streaming

- In HTTP streaming, the video is simply stored in an HTTP server as an ordinary file with a specific URL.
- When a user wants to see the video, the client establishes a TCP connection with the server and issues an HTTP GET request for that URL.
- The server then sends the video file, within an HTTP response message, as quickly as possible, that is, as quickly as TCP congestion control and flow control will allow.
- On the client side, the bytes are collected in a client application buffer.
- Once the number of bytes in this buffer exceeds a predetermined threshold, the client application begins playback—specifically, it periodically grabs video frames from the client application buffer, decompresses the frames, and displays them on the user's screen.
- In particular, it is not uncommon for the transmission rate to vary in a "saw-tooth" manner associated with TCP congestion control.

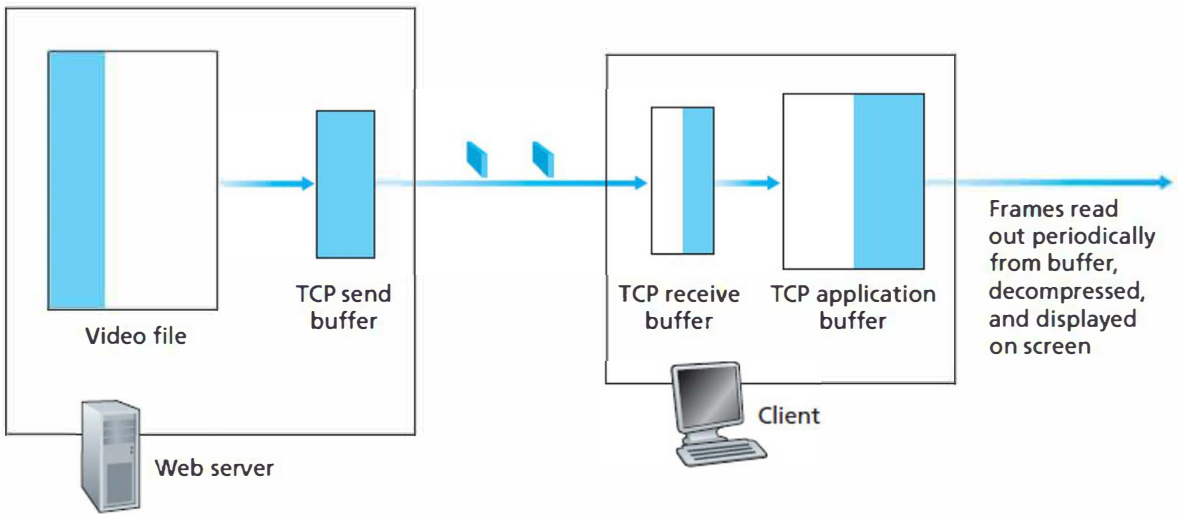
- Furthermore, packets can also be significantly delayed due to TCP's retransmission mechanism.

**Prefetching Video**

- We know that, client-side buffering can be used to mitigate the effects of varying end-to-end delays and varying available bandwidth.
- However, for streaming stored video, the client can attempt to download the video at a rate higher than the consumption rate, thereby prefetching video frames that are to be consumed in the future.
- This prefetched video is naturally stored in the client application buffer.
- Such prefetching occurs naturally with TCP streaming, since TCP's congestion avoidance mechanism will attempt to use all of the available bandwidth between server and client.
- Let's understand prefetching in depth:
- Suppose the video consumption rate is 1 Mbps but the network is capable of delivering the video from server to client at a constant rate of 1.5 Mbps.
- Then the client will not only be able to play out the video with a very small playout delay, but will also be able to increase the amount of buffered video data by 500 Kbits every second.
- In this manner, if in the future the client receives data at a rate of less than 1 Mbps for a brief period of time, the client will be able to continue to provide continuous playback due to the reserve in its buffer.

**Client Application Buffer and TCP Buffers**

- Below shown figure illustrates the interaction between client and server for HTTP streaming.
- At the server side, the portion of the video file in white has already been sent into the server's socket, while the darkened portion is what remains to be sent.
- After "passing through the socket door," the bytes are placed in the TCP send buffer before being transmitted into the Internet.

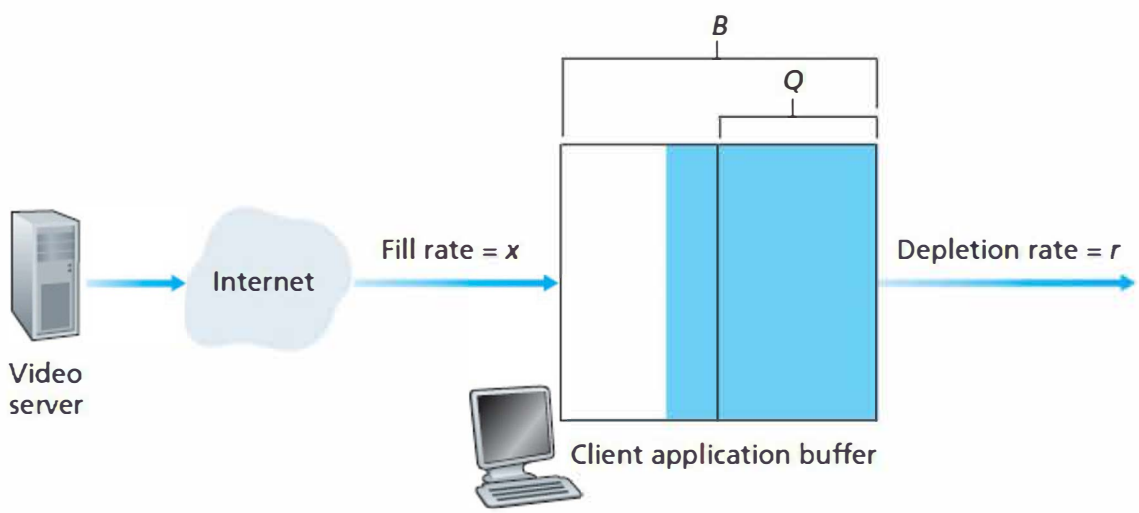



- Because the TCP send buffer is shown to be full, the server is momentarily prevented from sending more bytes from the video file into the socket.
- On the client side, the client application (media player) reads bytes from the TCP receive buffer (through its client socket) and places the bytes into the client application buffer.
- At the same time, the client application periodically grabs video frames from the client application buffer, decompresses the frames, and displays them on the user's screen.
- Note that if the client application buffer is larger than the video file, then the whole process of moving bytes from the server's storage to the client's application buffer is equivalent to an ordinary file download over HTTP—the client simply pulls the video off the server as fast as TCP will allow!
- Consider now what happens when the user pauses the video during the streaming process.
- During the pause period, bits are not removed from the client application buffer, even though bits continue to enter the buffer from the server.
- If the client application buffer is finite, it may eventually become full, which will cause "back pressure" all the way back to the server.
- Once the client receive TCP buffer becomes full, bytes can no longer be removed from the client TCP send buffer, so it also becomes full.

- Once the TCP send buffer becomes full, the server cannot send any more bytes into the socket.
- Thus, if the user pauses the video, the server may be forced to stop transmitting, in which case the server will be blocked until the user resumes the video.

**Analysis of Video Streaming**

- Let B denote the size (in bits) of the client's application buffer, and let Q denote the number of bits that must be buffered before the client application begins playout. (Of course, Q < B.)
- Let r denote the video consumption rate—the rate at which the client draws bits out of the client application buffer during playback.
- So, for example, if the video's frame rate is 30 frames/sec, and each (compressed) frame is 100,000 bits, then r = 3 Mbps.




- Let's assume that the server sends bits at a constant rate xbps whenever the client buffer is not full.
- Suppose at time t = 0, the application buffer is empty and video begins arriving to the client application buffer.
- First, let's determine , the time when Q bits have entered the application buffer and playout begins. Recall that bits arrive to the client application buffer at rate x and no bits are removed

from this buffer before playout begins. Thus, the amount of time required to build up Q bits (the initial buffering delay) is $t_p = Q/x$.

- Now let's determine $t_f$, the point in time when the client application buffer becomes full.
- We first observe that if $x < r$, then the client buffer will never become full!
- Eventually the client buffer will empty out entirely, at which time the video will freeze on the screen while the client buffer waits another $t_p$ seconds to build up Q bits of video.
- Thus, when the available rate in the network is less than the video rate, playout will alternate between periods of continuous playout and periods of freezing.
- Now let's determine $t_f$ for when $x > r$.
- In this case, starting at time $t_p$, the buffer increases from Q to B at rate x - r since bits are being depleted at rate r but are arriving at rate x.

**Early Termination and Repositioning the Video**

- HTTP streaming systems often make use of the HTTP byte-range header in the HTTP GET request message, which specifies the specific range of bytes the client currently wants to retrieve from the desired video.
- This is particularly useful when the user wants to reposition (that is, jump) to a future point in time in the video.
- When the user repositions to a new position, the client sends a new HTTP request, indicating with the byte-range header from which byte in the file should the server send data.
- When the server receives the new HTTP request, it can forget about any earlier request and instead send bytes beginning with the byte indicated in the byte range request.
- When a user repositions to a future point in the video or terminates the video early, some prefetched-but-not-yet-viewed data transmitted by the server will go unwatched—a waste of network bandwidth and server resources.
- For example, suppose that the client buffer is full with B bits at some time t0 into the video, and at this time the user repositions to some instant $t > t0 + B/r$ into the video, and then watches

the video to completion from that point on. In this case, all B bits in the buffer will be unwatched and the bandwidth and server resources that were used to transmit those B bits have been completely wasted.

**Probable Question:** ***Explain HTTP streaming***

***Describe video prefetching under HTTP streaming***

***Explain how early termination and videos repositioning works under HTTP streaming***

## 5.2.3 Adaptive Streaming and DASH

- HTTP streaming has a major shortcoming:
- All clients receive the same encoding of the video, despite the large variations in the amount of bandwidth available to a client, both across different clients and also over time for the same client.
- This has led to the development of a new type of HTTP-based streaming, often referred to as Dynamic Adaptive Streaming over HTTP (DASH).
- In DASH, the video is encoded into several different versions, with each version having a different bit rate and, correspondingly, a different quality level.
- The client dynamically requests chunks of video segments of a few seconds in length from the different versions.
- When the amount of available bandwidth is high, the client naturally selects chunks from a high-rate version; and when the available bandwidth is low, it naturally selects from a low-rate version.
- The client selects different chunks one at a time with HTTP GET request messages.
- On one hand, DASH allows clients with different Internet access rates to stream in video at different encoding rates.
- Clients with low-speed 3G connections can receive a low bit-rate (and low-quality) version, and clients with fiber connections can receive a high-quality version.

- On the other hand, DASH allows a client to adapt to the available bandwidth if the end-to-end bandwidth changes during the session.
- With DASH, each video version is stored in the HTTP server, each with a different URL.
- The HTTP server also has a **manifest file,** which provides a URL for each version along with its bit rate.
- The client first requests the manifest file and learns about the various versions.
- The client then selects one chunk at a time by specifying a URL and a byte range in an HTTP GET request message for each chunk.
- While downloading chunks, the client also measures the received bandwidth and runs a **rate determination algorithm** to select the chunk to request next.
- Naturally, if the client has a lot of video buffered and if the measured receive bandwidth is high, it will choose a chunk from a high-rate version.
- And naturally if the client has little video buffered and the measured received bandwidth is low, it will choose a chunk from a low-rate version.
- DASH therefore allows the client to freely switch among different quality levels.

**Probable Question:** ***Explain the concept of DASH. How is it different from Adaptive streaming***

## 5.2.4 Content Distribution Networks

- Today, many Internet video companies are distributing on-demand multi-Mbps streams to millions of users on a daily basis. YouTube, for example, with a library of hundreds of millions of videos, distributes hundreds of millions of video streams to users around the world every day.
- Streaming all this traffic to locations all over the world while providing continuous playout and high interactivity is clearly a challenging task.

- For an Internet video company, perhaps the most straightforward approach to providing streaming video service is to build a single massive data center, store all of its videos in the data center, and stream the videos directly from the data center to clients worldwide.
- But there are three major problems with this approach.
  1) If the client is far from the data center, server-to-client packets will cross many communication links and likely pass through many ISPs, with some of the ISPs possibly located on different continents.
  2) A popular video will likely be sent many times over the same communication links. Not only does this waste network bandwidth, but the Internet video company itself will be paying its provider ISP (connected to the data center) for sending the same bytes into the Internet over and over again.
  3) A single data center represents a single point of failure—if the data center or its links to the Internet goes down, it would not be able to distribute any video streams.
- Hence Content Distribution Networks (CDNs).
- A CDN manages servers in multiple geographically distributed locations, stores copies of the videos (and other types of Web content, including documents, images, and audio) in its servers, and attempts to direct each user request to a CDN location that will provide the best user experience.
- The CDN may be a private CDN, that is, owned by the content provider itself; for example, Google's CDN distributes YouTube videos and other types of content, or may be a third-party CDN that distributes content on behalf of multiple content providers; Akamai's CDN, for example, is a third party CDN that distributes Netflix and Hulu content, among others.
- CDNs typically adopt one of two different server placement philosophies:

  • **Enter Deep.** To enter deep into the access networks of Internet Service Providers, by deploying server clusters in access ISPs all over the world. The goal is to get close to end users, thereby improving user-perceived delay and throughput by decreasing the number of links and routers between the end user and the CDN cluster from which it receives content.

• **Bring Home.** To bring the ISPs home by building large clusters at a smaller number (for example, tens) of key locations and connecting these clusters using a private high-speed network. Compared with the enter-deep design philosophy, the bring-home design typically results in lower maintenance and management overhead, possibly at the expense of higher delay and lower throughput to end users.

**CDN Operation**

- Let us no understand how content distribution networks distribute a requested content to clients.
- When a browser in a user's host is instructed to retrieve a specific video (identified by a URL), the CDN must intercept the request so that it can

  (1) determine a suitable CDN server cluster for that client at that time, and

  (2) redirect the client's request to a server in that cluster.
- Suppose a content provider, NetCinema, employs the third-party CDN company, KingCDN, to distribute its videos to its customers. On the NetCinema Web pages, each of its videos is assigned a URL that includes the string "video" and a unique identifier for the video itself; for example, Transformers 7 might be assigned http://video.netcinema.com/6Y7B23V.
- Six steps then occur, as shown in Figure 7.4:

1. The user visits the Web page at NetCinema.

2. When the user clicks on the link http://video.netcinema.com/6Y7B23V, the user's host sends a DNS query for video.netcinema.com.

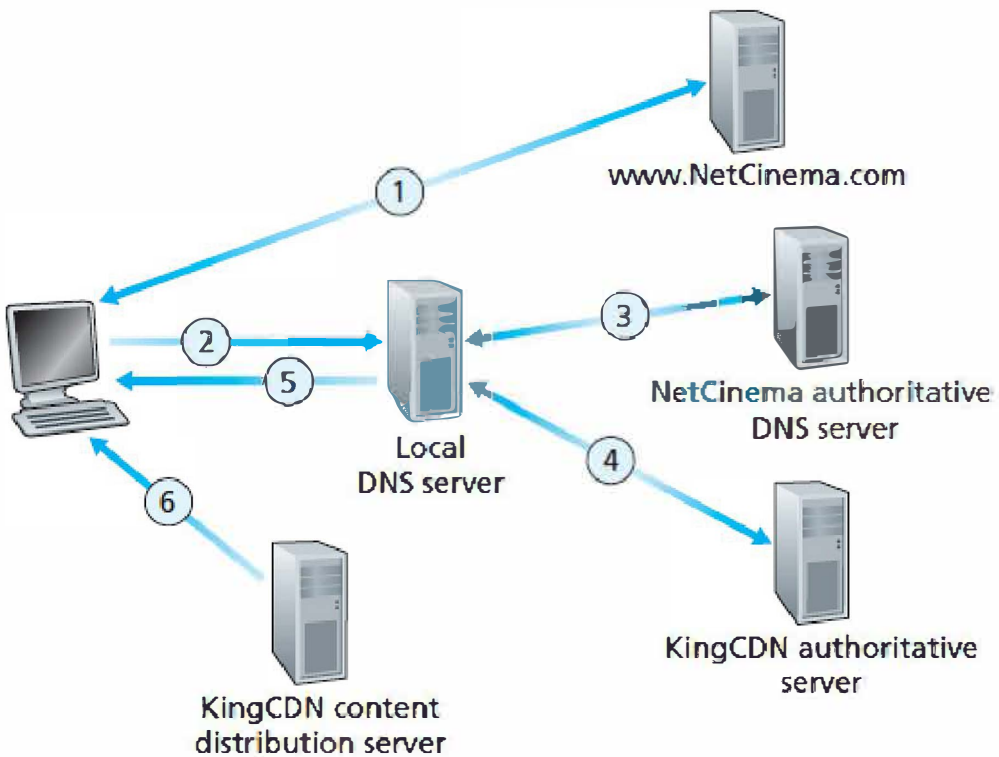
3. The user's Local DNS Server (LDNS) relays the DNS query to an authoritative DNS server for NetCinema, which observes the string "video" in the hostname video.netcinema.com. To "hand over" the DNS query to KingCDN, instead of returning an IP address, the NetCinema authoritative DNS server returns to the LDNS a hostname in the KingCDN's domain, for example, a1105.kingcdn.com.

4. From this point on, the DNS query enters into KingCDN's private DNS infrastructure. The user's LDNS then sends a second query, now for a1105.kingcdn.com, and KingCDN's DNS

system eventually returns the IP addresses of a KingCDN content server to the LDNS. It is thus here, within the KingCDN's DNS system, that the CDN server from which the client will receive its content is specified.

5. The LDNS forwards the IP address of the content-serving CDN node to the user's host.

6. Once the client receives the IP address for a KingCDN content server, it establishes a direct TCP connection with the server at that IP address and issues an HTTP GET request for the video.

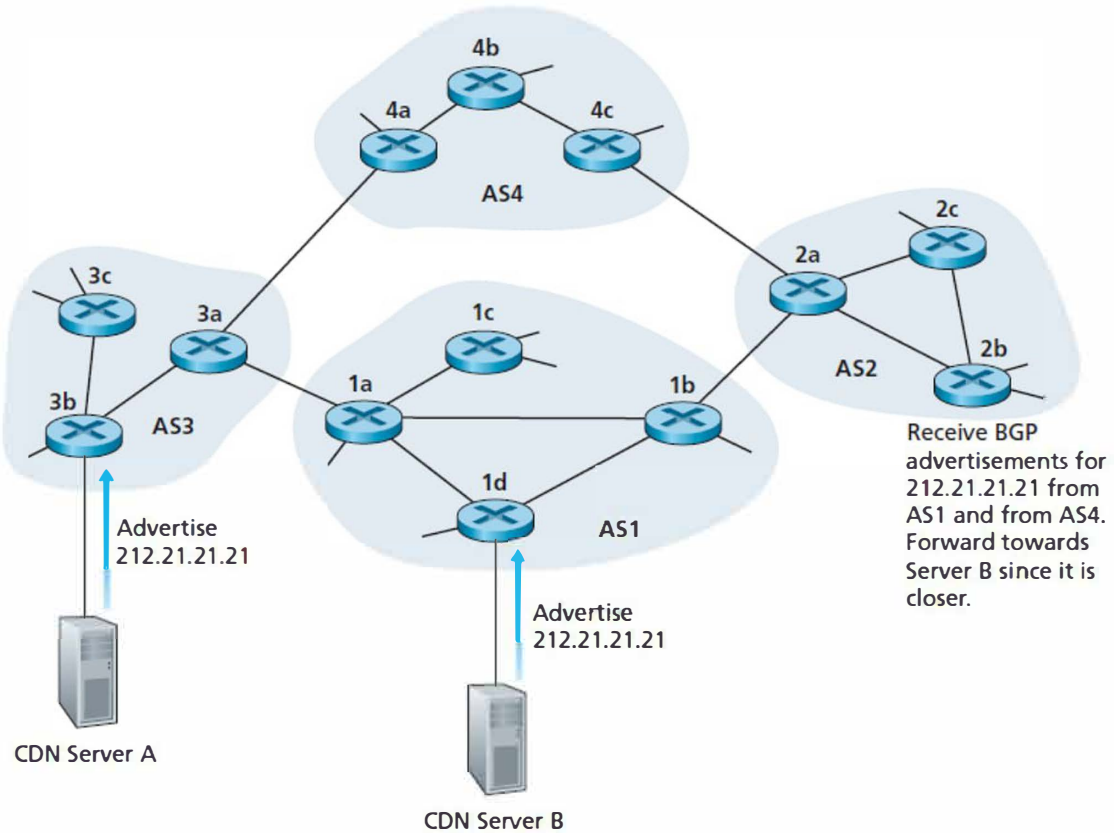



**Cluster Selection Strategies**

- At the core of any CDN deployment is a cluster selection strategy, that is, a mechanism for dynamically directing clients to a server cluster or a data center within the CDN.
- The CDN first learns the IP address of the client's LDNS server via the client's DNS lookup.
- Afterwards, the CDN needs to select an appropriate cluster based on this IP address.
- One simple strategy is to assign the client to the cluster that is geographically closest.

- When a DNS request is received from a particular LDNS, the CDN chooses the geographically closest cluster, that is, the cluster that is the fewest kilometers from the LDNS "as the bird flies."
- Such a solution can work reasonably well for a large fraction of the clients. However, for some clients, the solution may perform poorly, since the geographically closest cluster may not be the closest cluster along the network path.
- Furthermore, a problem inherent with all DNS-based approaches is that some end-users are configured to use remotely located LDNSs, in which case the LDNS location may be far from the client's location.
- In order to determine the best cluster for a client based on the current traffic conditions, CDNs can instead perform periodic real-time measurements of delay and loss performance between their clusters and clients.
- For instance, a CDN can have each of its clusters periodically send probes (for example, ping messages or DNS queries) to all of the LDNSs around the world.
- One drawback of this approach is that many LDNSs are configured to not respond to such probes.
- An alternative to sending extraneous traffic for measuring path properties is to use the characteristics of recent and ongoing traffic between the clients and CDN servers.
- For instance, the delay between a client and a cluster can be estimated by examining the gap between server-to-client SYNACK and client-to-server ACK during the TCP three-way handshake.
- Such solutions, however, require redirecting clients to (possibly) suboptimal clusters from time to time in order to measure the properties of paths to these clusters.
- Although only a small number of requests need to serve as probes, the selected clients can suffer significant performance degradation when receiving content (video or otherwise)
- Another alternative for cluster-to-client path probing is to use DNS query traffic to measure the delay between clients and clusters.

- Specifically, during the DNS phase (within Step 4 in previous figure), the client's LDNS can be occasionally directed to different DNS authoritative servers installed at the various cluster locations, yielding DNS traffic that can then be measured between the LDNS and these cluster locations.




- In this scheme, the DNS servers continue to return the optimal cluster for the client, so that delivery of videos and other Web objects does not suffer.
- A very different approach to matching clients with CDN servers is to use IP anycast.
- The idea behind IP anycast is to have the routers in the Internet route the client's packets to the "closest" cluster, as determined by BGP.
- As shown in above figure, during the IP-anycast configuration stage, the CDN company assigns the same IP address to each of its clusters, and uses standard BGP to advertise this IP address from each of the different cluster locations.

- When a BGP router receives multiple route advertisements for this same IP address, it treats these advertisements as providing different paths to the same physical location (when, in fact, the advertisements are for different paths to different physical locations).
- Following standard operating procedures, the BGP router will then pick the "best" (for example, closest, as determined by AS-hop counts) route to the IP address according to its local route selection mechanism.
- This approach has the advantage of finding the cluster that is closest to the client rather than the cluster that is closest to the client's LDNS.

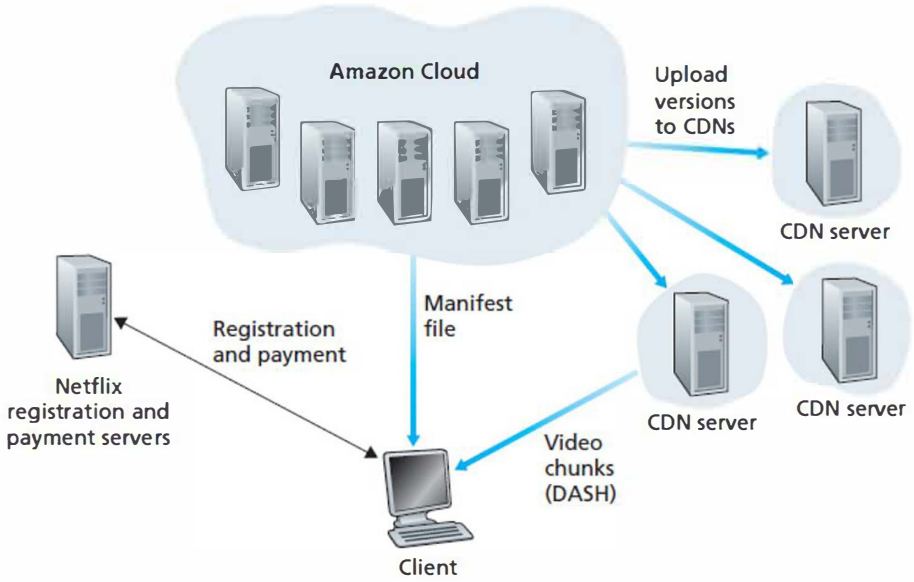
**Probable Question:** ***Explain CDNs. What are the major problems in it? How CDN works?***

## 5.2.5 Case Studies: Netflix, YouTube, and Kankan

- We conclude our discussion of streaming stored video by taking a look at three highly successful large-scale deployments: Netflix, YouTube, and Kankan.

**Netflix**

- Generating almost 30 percent of the downstream U.S. Internet traffic in 2011, Netflix has become the leading service provider for online movies and TV shows in the United States.
- In order to rapidly deploy its large-scale service, Netflix has made extensive use of third-party cloud services and CDNs.
- Below figure shows the basic architecture of the Netflix video-streaming platform.
- It has four major components:
  - the registration and payment servers
  - the Amazon cloud
  - multiple CDN providers
  - clients.




Some of the functions taking place in the Amazon cloud include:

• *Content ingestion*. Before Netflix can distribute a movie to its customers, it must first ingest and process the movie.

• *Content processing*. The machines in the Amazon cloud create many different formats for each movie, suitable for a diverse array of client video players running on desktop computers, smartphones, and game consoles connected to televisions. A different version is created for each of these formats and at multiple bit rates, allowing for adaptive streaming over HTTP using DASH.

• *Uploading versions to the CDNs*. Once all of the versions of a movie have been created, the hosts in the Amazon cloud upload the versions to the CDNs.

- The Web pages for browsing the Netflix video library are served from servers in the Amazon cloud.
- When the user selects a movie to "Play Now," the user's client obtains a manifest file, also from servers in the Amazon cloud.
- The manifest file includes a variety of information, including a ranked list of CDNs and the URLs for the different versions of the movie, which are used for DASH playback.

- The ranking of the CDNs is determined by Netflix, and may change from one streaming session to the next. Typically the client will select the CDN that is ranked highest in the manifest file.

**YouTube**

- With approximately half a billion videos in its library and half a billion video views per day, YouTube is indisputably the world's largest video-sharing site.
- YouTube began its service in April 2005 and was acquired by Google in November 2006.
- As with Netflix, YouTube makes extensive use of CDN technology to distribute its videos.
- Unlike Netflix, however, Google does not employ third-party CDNs but instead uses its own private CDN to distribute YouTube videos.
- Google has installed server clusters in many hundreds of different locations.
- Google uses DNS to redirect a customer request to a specific cluster.
- Most of the time, Google's cluster selection strategy directs the client to the cluster for which the RTT between client and cluster is the lowest; however, in order to balance the load across clusters, sometimes the client is directed (via DNS) to a more distant cluster.
- Furthermore, if a cluster does not have the requested video, instead of fetching it from somewhere else and relaying it to the client, the cluster may return an HTTP redirect message, thereby redirecting the client to another cluster.
- YouTube employs HTTP streaming.
- YouTube often makes a small number of different versions available for a video, each with a different bit rate and corresponding quality level.
- As of 2011, YouTube does not employ adaptive streaming (such as DASH), but instead requires the user to manually select a version.
- In order to save bandwidth and server resources that would be wasted by repositioning or early termination, YouTube uses the HTTP byte range request to limit the flow of transmitted data after a target amount of video is prefetched.

- YouTube processes each video it receives, converting it to a YouTube video format and creating multiple versions at different bit rates. This processing takes place entirely within Google data centers.

**Kankan**

- We just saw that for both the Netflix and YouTube services, servers operated by CDNs (either third-party or private CDNs) stream videos to clients.
- Netflix and YouTube not only have to pay for the server hardware (either directly through ownership or indirectly through rent), but also for the bandwidth the servers use to distribute the videos.
- Given the scale of these services and the amount of bandwidth they are consuming, such a "client-server" deployment is extremely costly.
- Kankan has an entirely different approach for providing video on demand over the Internet at a large scale---one that allows the service provider to significantly reduce its infrastructure and bandwidth costs.
- This approach uses P2P delivery instead of client-server (via CDNs) delivery. P2P video delivery is used with great success by several companies in China, including Kankan (owned and operated by Xunlei), PPTV (formerly PPLive), and PPs (formerly PPstream).
- Kankan, currently the leading P2P-based video-on-demand provider in China, has over 20 million unique users viewing its videos every month.
- At a high level, P2P video streaming is very similar to BitTorrent file downloading.
- When a peer wants to see a video, it contacts a tracker (which may be centralized or peer-based using a DHT) to discover other peers in the system that have a copy of that video. This peer then requests chunks of the video file in parallel from these other peers that have the file.
- Different from downloading with BitTorrent, however, requests are preferentially made for chunks that are to be played back in the near future in order to ensure continuous playback.

**Probable Question:** ***Write a note on (a) NetFlix (b) Youtube (c) Kankan***

# 5.3 Network Support for Multimedia

- So far we understood how application-level mechanisms such as client buffering, prefetching, adapting media quality to available bandwidth, adaptive playout, and loss mitigation techniques can be used by multimedia applications to improve a multimedia application's performance.
- We must know what network mechanisms are provided to support multimedia content delivery.
- Three broad approaches towards providing network-level support for multimedia applications.

  • *Making the best of best-effort service.* Effort designed for a well-dimensioned network leads to lower packet loss and rare occurrence of excessive end-to-end delay. When demand increases are forecasted, the ISPs deploy additional bandwidth and switching capacity to continue to ensure satisfactory delay and packet-loss performance.

  • *Differentiated service.* It's been envisioned that different types of traffic (for example, as indicated in the Type-of-Service field in the IP4v packet header) could be provided with different classes of service, rather than a single one-size-fits-all best-effort service.

  With differentiated service, one type of traffic might be given strict priority over another class of traffic when both types of traffic are queued at a router.

  For example, packets belonging to a real-time conversational application might be given priority over other packets due to their stringent delay constraints.

  • *Per-connection Quality-of-Service (QoS) Guarantees*. With per-connection QoS guarantees, each instance of an application explicitly reserves end-to-end bandwidth and thus has a guaranteed end-to-end performance.

- A **hard guarantee** means the application will receive its requested quality of service (QoS) with certainty.
- A **soft guarantee** means the application will receive its requested quality of service with high probability.

**Probable Question:** ***Define hard and soft guarantee. What are the approaches made for network support to Multimedia?***

***Read for a better understanding…***

*For example, if a user wants to make a VoIP call from Host A to Host B, the user's VoIP application reserves bandwidth explicitly in each link along a route between the two hosts. But permitting applications to make reservations and requiring the network to honor the reservations requires some big changes. First, we need a protocol that, on behalf of the applications, reserves link bandwidth on the paths from the senders to their receivers. Second, we'll need new scheduling policies in the router queues so that per-connection bandwidth reservations can be honored. Finally, in order to make a reservation, the applications must give the network a description of the traffic that they intend to send into the network and the network will need to police each application's traffic to make sure that it abides by that description. These mechanisms, when combined, require new and complex software in hosts and routers.*

## 5.3.1 Dimensioning Best-Effort Networks

- Performance requirements for any network applications are—low end-to-end packet delay, delay jitter, and loss—and the fact that packet delay, delay jitter, and loss occur whenever the network becomes congested.
- Providing enough link capacity throughout the network so that network congestion, and its consequent packet delay and loss, never (or only very rarely) occurs would definitely improve the performance.
- With enough link capacity, packets could zip through today's Internet without queuing delay or loss.
- The question of how much capacity to provide at network links in a given topology to achieve a given level of performance is often known as **bandwidth provisioning**.
- The even more complicated problem of how to design a network topology (where to place routers, how to interconnect routers with links, and what capacity to assign to links) to achieve a given level of end-to-end performance is a network design problem often referred to as **network dimensioning.**

- Both bandwidth provisioning and network dimensioning are complex topics.
- However, that the following issues must be addressed in order to predict application- level performance between two network end points, and thus provision enough capacity to meet an application's performance requirements.

  • *Models of traffic demand between network end points.* Models may need to be specified at both the call level (for example, users "arriving" to the network and starting up end-to-end applications) and at the packet level (for example, packets being generated by ongoing applications). Note that workload may change over time.

  • *Well-defined performance requirements.* For example, a performance requirement for supporting delay-sensitive traffic, such as a conversational multimedia application, might be that the probability that the end-to-end delay of the packet is greater than a maximum tolerable delay be less than some small value.

  • *Models to predict end-to-end performance for a given workload model, and techniques to find a minimal cost bandwidth allocation that will result in all user requirements being met.* Here, researchers are busy developing performance models that can quantify performance for a given workload, and optimization techniques to find minimal-cost bandwidth allocations meeting performance requirements.

**Probable Question:** ***What are the issues to be addressed in order to provide best effort networks? Explain***

## 5.3.2 Providing Multiple Classes of Service

- Simplest strategy to guarantee QOS is to divide traffic into classes, and provide different levels of service to these different classes of traffic.
- For example, an ISP might well want to provide a higher class of service to delay-sensitive Voice-over-IP or teleconferencing traffic (and charge more for this service!) than to elastic traffic such as email or HTTP.
- It's important to note that such differential service is provided among aggregates of traffic, that is, among classes of traffic, not among individual connections.

## Real-time Transport Protocol

VoIP application appends header fields to the audio chunks before passing them to the transport layer. These header fields include sequence numbers and timestamps. Since most multimedia networking applications can make use of sequence numbers and timestamps, it is convenient to have a standardized packet structure that includes fields for audio/video data, sequence number, and timestamp, as well as other potentially useful fields

RTP is real-time interactive protocols. RTP can be used for transporting common formats such as PCM, ACC, and MP3 for sound and MPEG and H.263 for video. It can also be used for transporting proprietary sound and video formats.

RTP typically runs on top of UDP. The sending side encapsulates a media chunk within an RTP packet, then encapsulates the packet in a UDP segment, and then hands the segment to IP
The receiving side extracts the RTP packet from the UDP segment, then extracts the media chunk from the RTP packet, and then passes the chunk to the media player for decoding and rendering.

RTP does not even guarantee delivery of packets or prevent out-of-order delivery of packets. Indeed, RTP encapsulation is seen only at the end systems. Routers do not distinguish between IP datagrams that carry RTP packets and IP datagrams.

RTP packets are not limited to unicast applications. They can also be sent over one-to-many and many-to-many multicast trees. For a many-to-many multicast session, all of the session's senders and sources typically use the same multicast group for sending their RTP streams. RTP multicast streams belonging together, such as audio and video streams emanating from multiple senders in a video conference application, belong to an **RTP session.**

### RTP Packet Header Fields


- ***Sequence number field.*** The sequence number field is 16 bits long. The sequence number increments by one for each RTP packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.
- ***Timestamp field.*** The timestamp field is 32 bits long. It reflects the sampling instant of the first byte in the RTP data packet. The receiver can use timestamps to remove packet jitter introduced in the network and to provide synchronous playout at the receiver. The timestamp is derived from a sampling clock at the sender
- ***Synchronization source identifier (SSRC).*** The SSRC field is 32 bits long. It identifies the source of the RTP stream. Typically, each stream in an RTP session has a distinct SSRC. The SSRC is not the IP address of the sender, but instead is a number that the source assigns randomly when the new stream is started

**Session Initiation Protocol (SIP)**

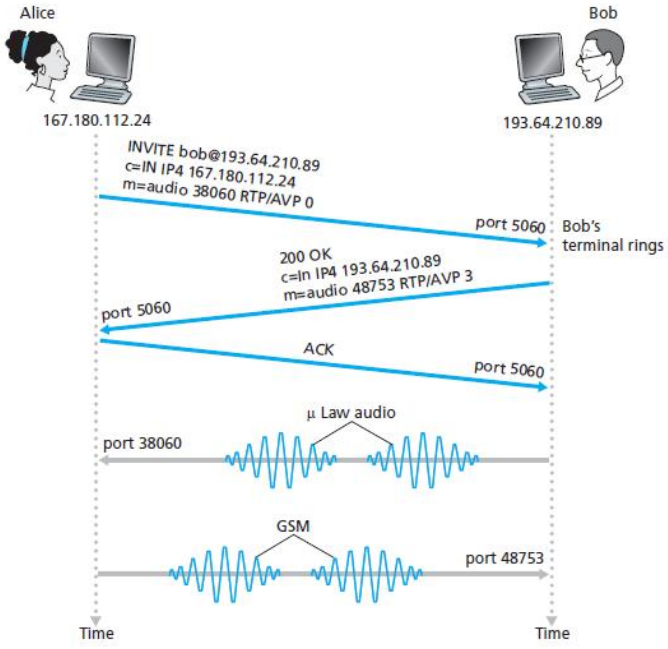
- It provides mechanisms for establishing calls between a caller and a callee over an IP network. It allows the caller to notify the callee that it wants to start a call. It allows the participants to agree on media encodings. It also allows participants to end calls.

- It provides mechanisms for the caller to determine the current IP address of the callee. Users do not have a single, fixed IP address because they may be assigned addresses dynamically (using DHCP) and because they may have multiple IP devices, each with a different IP address.

- It provides mechanisms for call management, such as adding new media streams during the call, changing the encoding during the call, inviting new participants during the call, call transfer, and call holding.

**Setting Up a Call to a Known IP Address:**

1. Alice is at her PC and she wants to call Bob, who is also working at his PC. Alice's and Bob's PCs are both equipped with SIP-based software for making and receiving phone calls.
2. This INVITE message is sent over UDP to the well-known port 5060 for SIP. (SIP messages can also be sent over TCP.) The INVITE message includes an identifier for Bob
3. (bob@193.64.210.89), an indication of Alice's current IP address, an indication that Alice desires to receive audio, which is to be encoded in format AVP 0 (PCM encoded μ-law) and encapsulated in RTP, and an indication that she wants to receive the RTP packets on port 38060.




4. This response SIP message is also sent to the SIP port 5060. Bob's response includes a 200 OK as well as an indication of his IP address, his desired encoding and packetization for reception, and his port number to which the audio packets should be sent.
5. Note that in this example Alice and Bob are going to use different audio-encoding mechanisms: Alice is asked to encode her audio with GSM whereas Bob is asked to encode his audio with PCM μ - law. After receiving Bob's response, Alice sends Bob an SIP acknowledgment message.
6. Bob will encode and packetize the audio as requested and send the audio packets to port number 38060 at IP address 167.180.112.24. Alice will also encode and packetize the audio as requested and send the audio packets to port number 48753 at IP address 193.64.210.89.

**SIP Messages:**

- The INVITE line includes the SIP version, as does an HTTP request message. Whenever an SIP message passes through an SIP device (including the device that originates the message), it attaches a via header, which indicates the IP address of the device.

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

- The SIP message includes a from header line and a to header line. The message includes a Call-ID, which uniquely identifies the call (similar to the message-ID in e-mail). It includes a Content-Type header line, which defines the format used to describe the content contained in the SIP message. It also includes a Content-Length header line, which provides the length in bytes of the content in the message.
- A carriage return and line feed, the message contains the content. In this case, the content provides information about Alice's IP address and how Alice wants to receive the audio.

**Name Translation and User Location**

- Alice needs to obtain the IP address of the device that the user bob@domain.com is currently using. To find this out, Alice creates an INVITE message that begins with INVITE bob@domain.com SIP/2.0 and sends this message to an **SIP proxy**.
- The proxy will respond with an SIP reply that might include the IP address of the device that bob@domain.com is currently using. Alternatively, the reply might include the IP address of Bob's voicemail box, or it might include a URL of a Web page.

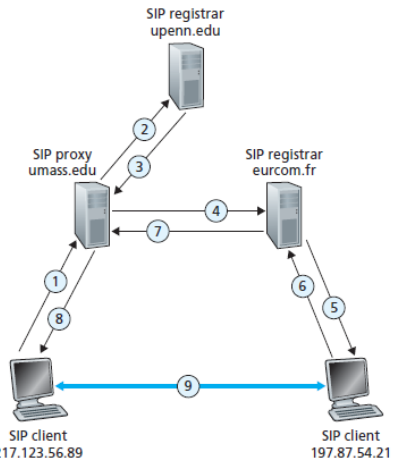



Figure 7.13 ♦ Session initiation, involving SIP proxies and registrars

- SIP user has an associated registrar. Whenever a user launches an SIP application on a device, the application sends an SIP register message to the registrar, informing the registrar of its current IP address. Alice's SIP proxy server obtains Bob's current IP address.
- Proxy server simply needs to forward Alice's INVITE message to Bob's registrar/proxy. The registrar/proxy could then forward the message to Bob's current SIP device. Finally, Bob, having now received Alice's INVITE message, could send an SIP response to Alice.