Module – 5

MULTIMEDIA NETWORKING

Multimedia Networking Applications

➔ Properties of Video

- 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.
  - Video streaming consumes most bandwidth, having a bit rate of more than ten times greater than that of the normal HTTP and music-streaming applications.
- Video can be compressed.
  - 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.
  - 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.
  - 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.

➔ Properties of Audio

- Digital audio has significantly lower bandwidth requirements than video.

- Analog audio can be converted to a digital signal using pulse code modulation with the following steps:
  - The analog audio signal is **sampled** at some fixed rate.
  - 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.
  - Each of the quantization values are **encoded** by representing with a fixed number of bits.
- PCM-encoded speech and music, however, are rarely used in the Internet. Instead, as with video, compression techniques are used to reduce the bit rates of the stream.
- A popular compression technique for near CD-quality stereo music is MPEG 1 layer 3, more commonly known as MP3.
- MP3 encoders can compress to many different rates; 128 kbps is the most common encoding rate and produces very little sound degradation.
- As with video, multiple versions of a prerecorded audio stream can be created, each at a different bit rate.

➔ Types of Multimedia Network Applications

Multimedia applications are classified into three broad categories:

(i) Streaming stored audio/video

(ii) Conversational voice/video-over-IP

(iii) Streaming live audio/video

1) Streaming Stored Audio and Video

- 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.
- By some estimates, streaming stored video makes up over 50 percent of the downstream traffic in the Internet access networks today.

Streaming stored video has three key distinguishing features.

- **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 having to download the entire video file 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. 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 playout 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 playout at the client; otherwise, users experience video frame freezing or frame skipping.

2) 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.
- Conversational voice and video are widely used in the Internet today, with the Internet companies Skype, QQ, and Google Talk boasting hundreds of millions of daily users.
- Timing considerations and tolerance of data loss are important for conversational voice and video applications.
- 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.

- 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.

3) 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.
- Today, thousands of radio and television stations around the world are broadcasting content over the Internet.
- Live, broadcast-like applications often have many users who receive the same audio/video program at the same time.
- Although the distribution of live audio/video to many receivers can be efficiently accomplished using the IP multicasting techniques, multicast distribution is more often accomplished today via application-layer multicast (using P2P networks or CDNs) or through multiple separate unicast streams.
- As with streaming stored multimedia, 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.

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:
  1. UDP streaming

2. HTTP streaming
3. 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.
- When the video starts to arrive at the client, the client need not immediately begin playout, 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 playout.
- There are two important advantages provided by such client buffering. First, client side buffering can absorb variations in server-to-client delay. Second, if the server-to-client bandwidth briefly drops below the video consumption rate, a use can continue to enjoy continuous playback, again as long as the client application buffer does not become completely drained.

➔ 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.
- 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.
- Before passing the video chunks to UDP, the server will encapsulate the video chunks within transport packets specially designed for transporting audio and video, using the Real-Time Transport Protocol (RTP).
- The client and server also maintain, in parallel, a separate control connection over which the client sends commands regarding session state changes (such as pause, resume, reposition,

and so on). The Real-Time Streaming Protocol is a popular open protocol for such a control connection.

Limitation:

- Due to the unpredictable and varying amount of available bandwidth between server and client, constant-rate UDP streaming can fail to provide continuous playout.
- It requires a media control server, such as an RTSP server, to process client-to-server interactivity requests and to track client state for each ongoing client session.
- Many firewalls are configured to block UDP traffic, preventing the users behind these firewalls from receiving UDP video.

➔ 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.

Advantages:

- The use of HTTP over TCP also allows the video to traverse firewalls and NATs more easily.
- Streaming over HTTP also obviates the need for a media control server, such as an RTSP server, reducing the cost of a large-scale deployment over the Internet.

Limitation and solution:

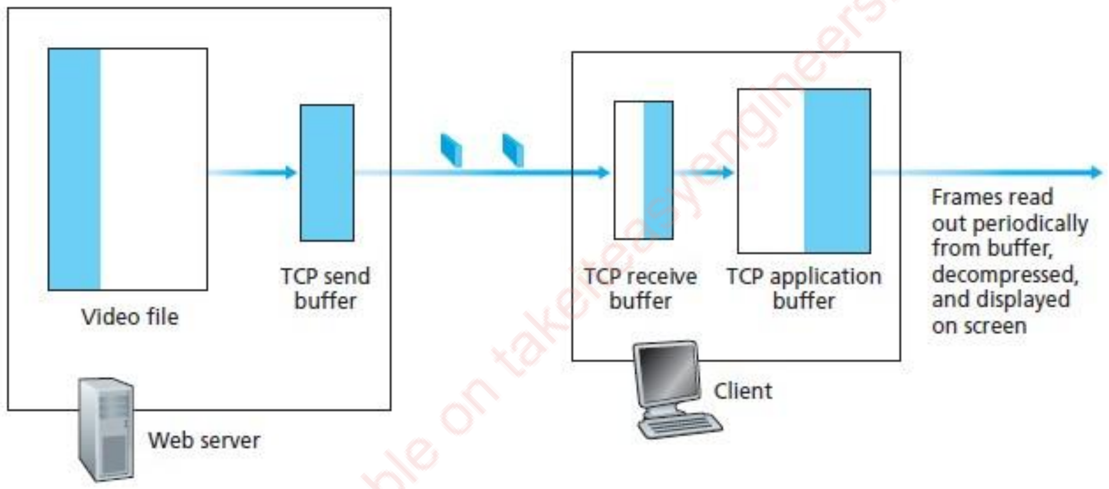
When transferring a file over TCP, the server-to client transmission rate can vary significantly due to TCP's congestion control mechanism. Packets can also be significantly delayed due to

TCP's retransmission mechanism. Because of these characteristics of TCP, it was believed that video streaming would never work well over TCP. Over time, however, designers of streaming video systems learned that TCP's congestion control and reliable-data transfer mechanisms do not necessarily preclude continuous playout when client buffering and prefetching are used.

Prefetching Video:

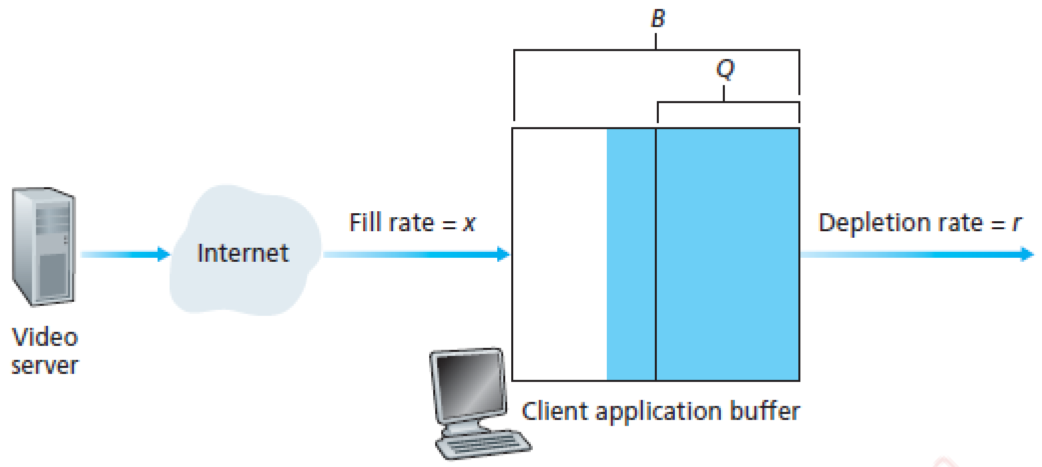
The client can attempt to download the video at a rate higher than the consumption rate, thereby prefetching video frames that are to be consumed in the future. This prefetched video is naturally stored in the client application buffer.

Client Application Buffer and TCP Buffers:



- Here 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 reads bytes from the TCP receive buffer 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.
- 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.

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.
- Let r denote the video consumption rate—the rate at which the client draws bits out of the client application buffer during playback.
- Let's assume that the server sends bits at a constant rate x whenever the client buffer is not full.
- If x < r (that is, if the server send rate is less than the video consumption rate), then the client buffer will never become full.
- When the available rate in the network is more than the video rate, after the initial buffering delay, the user will enjoy continuous playout until the video ends.

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 byterange request.

➔ Adaptive Streaming and DASH

Shortcoming of HTTP Streaming:

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.

Solution: **DASH**

- In DASH - Dynamic Adaptive Streaming over HTTP, 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.
- 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.
- By dynamically monitoring the available bandwidth and client buffer level, and adjusting the transmission rate with version switching, DASH can often achieve continuous playout at the best possible quality level without frame freezing or skipping.

<u>Content Distribution Networks</u>

- Streaming stored video to locations all over the world while providing continuous playout and high interactivity is clearly a challenging task.

- For an Internet video company, the most straightforward approach to providing streaming video service is to build a single massive data center which stores all of its videos in the data center, and stream the videos directly from the data center to clients worldwide.
- But this approach faces some problems:
  - Single massive date center is single point of failure
  - It leads long path to distant clients
  - It may create network congestion.
  - 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.
- In order to meet the challenge of distributing massive amounts of video data to users distributed around the world, almost all major video-streaming companies make use of 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.
- The CDN may alternatively be a third-party CDN that distributes content on behalf of multiple content providers; for example, Akamai's CDN 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:** One philosophy, pioneered by Akamai, is 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:** A second design philosophy, taken by Limelight and many other CDN companies, is to bring the ISPs home by building large clusters at a smaller number (for

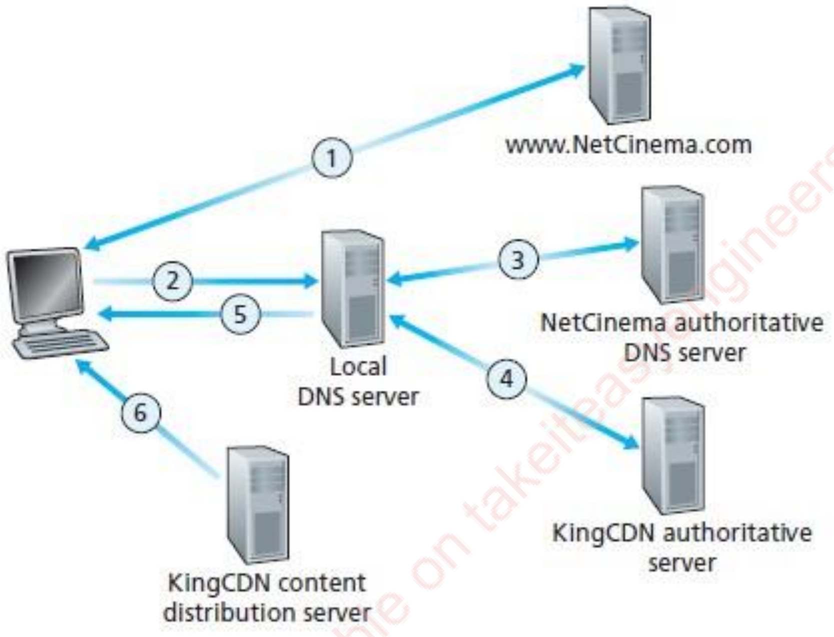
example, tens) of key locations and connecting these clusters using a private high-speed network. Instead of getting inside the access ISPs, these CDNs typically place each cluster at a location that is simultaneously near the PoPs of many tier-1 ISPs

➔ CDN Operation

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.

(2) Redirect the client's request to a server in that cluster.


www.NetCinema.com
1
2
5
3
4
6
Local DNS server
NetCinema authoritative DNS server
KingCDN authoritative server
KingCDN content distribution server

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

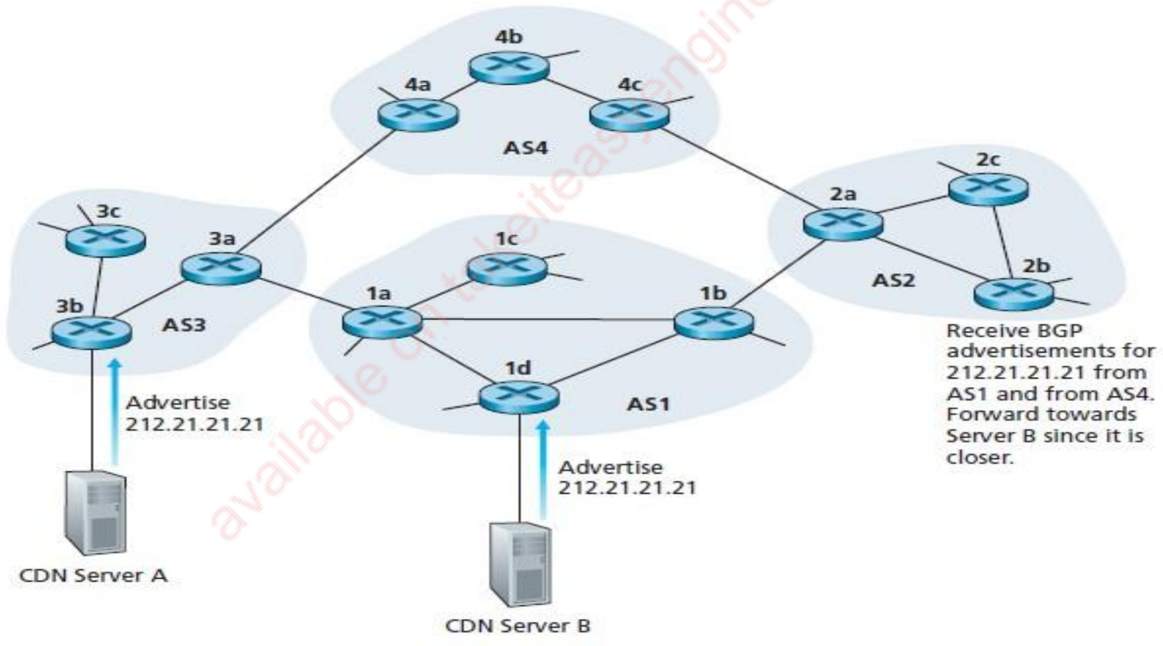
thus here, within the KingCDN's DNS system, that the CDN server from which the client will receive its content is specified.

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. If DASH is used, the server will first send to the client a manifest file with a list of URLs, one for each version of the video, and the client will dynamically select chunks from the different versions.

➔ Cluster Selection Strategies

- Cluster Selection Strategies is a mechanism for dynamically directing clients to a server cluster or a data center within the CDN.
- The CDN learns the IP address of the client's LDNS server via the client's DNS lookup. After learning this IP address, 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. Using commercial geo-location databases each LDNS IP address is mapped to a geographic location. When a DNS request is received from a particular LDNS, the CDN chooses the geographically closest cluster.
- For some clients, the solution may perform poorly, since the geographically closest cluster may not be the closest cluster along the network path.
- 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.
- 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.
- 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.
- 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.

- 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.
- Following standard operating procedures, the BGP router will then pick the "best" route to the IP address according to its local route selection mechanism.
- After this initial configuration phase, the CDN can do its main job of distributing content. When any client wants to see any video, the CDN's DNS returns the anycast address, no matter where the client is located.
- When the client sends a packet to that IP address, the packet is routed to the "closest" cluster as determined by the preconfigured forwarding tables, which were configured with BGP.


4b
4a
4c
AS4
2c
2a
AS2
2b
3c
3a
3b
AS3
1c
1a
1b
1d
AS1
Receive BGP advertisements for 212.21.21.21 from AS1 and from AS4. Forward towards Server B since it is closer.
Advertise 212.21.21.21
Advertise 212.21.21.21
CDN Server A
CDN Server B

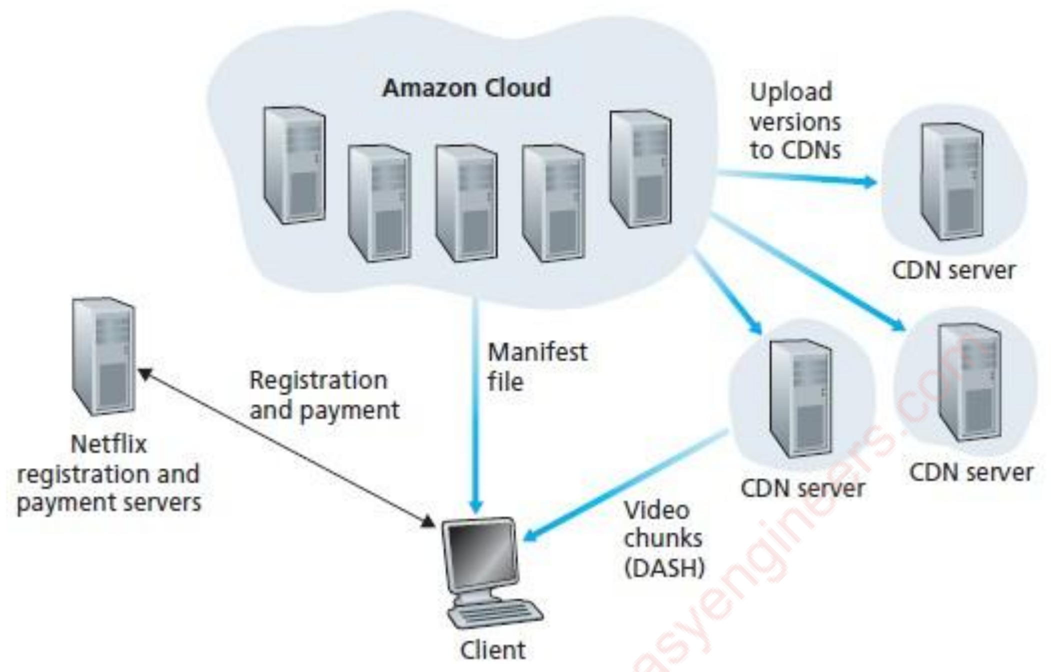
➔ Case Studies: Netflix, YouTube, and Kankan

<u>Netflix</u>

- Netflix is 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. Indeed, Netflix is an interesting example of a company

deploying a large-scale online service by renting servers, bandwidth, storage, and database services from third parties while using hardly any infrastructure of its own.

- Basic Architecture:



- Netflix has four major components: the registration and payment servers, the Amazon cloud, multiple CDN providers, and clients.
- In its own hardware infrastructure, Netflix maintains registration and payment servers, which handle registration of new accounts and capture credit-card payment information.
- Netflix runs its online service by employing machines (or virtual machines) in the Amazon cloud. 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. Netflix receives studio master versions of movies and uploads them to hosts in the Amazon cloud.
  - **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.
- After the client selects a CDN, the CDN leverages DNS to redirect the client to a specific CDN server.
- The client and that CDN server then interact using DASH.

<u>Youtube:</u>

- 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.
- 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. From a subset of about 50 of these locations, Google distributes YouTube video.
- 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.
- 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.
- 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

- Kankan 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.
- The Kankan design employs a tracker and its own DHT for tracking content.

Network Support for Multimedia

There exist three broad approaches towards providing network-level support for multimedia applications.

Approach	Granularity	Guarantee	Mechanisms	Complexity	Deployment to date
Making the best of best-effort service.	all traffic treated equally	none, or soft	application-layer support, CDNs, overlays, network-level resource provisioning	minimal	everywhere
Differentiated service	different classes of traffic treated differently	none, or soft	packet marking, policing, scheduling	medium	some
Per-connection Quality-of-Service (QoS) Guarantees	each source-destination flows treated differently	soft or hard, once flow is admitted	packet marking, policing, scheduling; call admission and signaling	light	little

- **Making the best of best-effort service:** The application-level mechanisms and infrastructure can be successfully used in a well-dimensioned network where packet loss and excessive end-to-end delay rarely occur. 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:** 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.
- **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.

➔ Dimensioning Best-Effort Networks

- A first approach to improving the quality of multimedia applications is through providing enough link capacity throughout the network so that network congestion, and its consequent packet delay and loss, never (or only very rarely) occurs. With enough link capacity, packets could zip through today's Internet without queuing delay or loss.

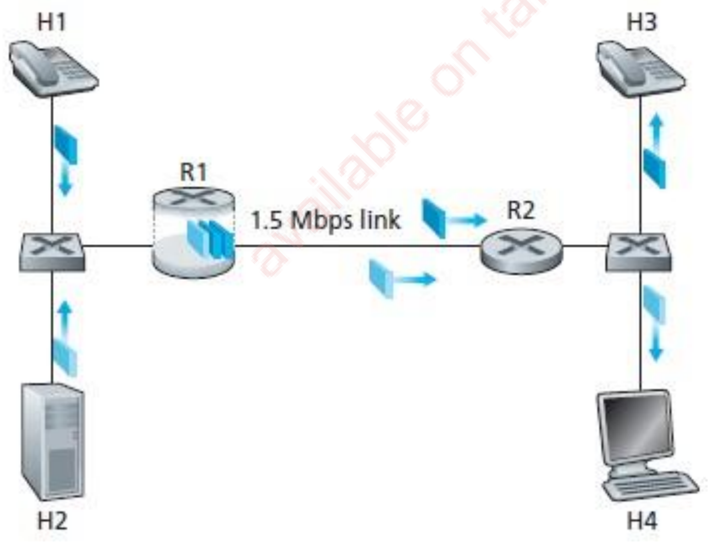
- Challenges:
  - The question of how much capacity to provide at network links in a given topology to achieve a given level of performance is often known as **bandwidth provisioning**.
  - The even more complicated problem of how to design a network topology (where to place routers, how to interconnect 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**.

➔ Providing Multiple Classes of Service

The simplest enhancement to the one-size-fits-all best-effort service in today's Internet is to divide traffic into classes, and provide different levels of service to these different classes of traffic.

The type-of-service (ToS) field in the IPv4 header can be used for this purpose.

Motivating Scenarios



Here H1 and H3 are using audio application, H2 and H4 are using HTTP web application.

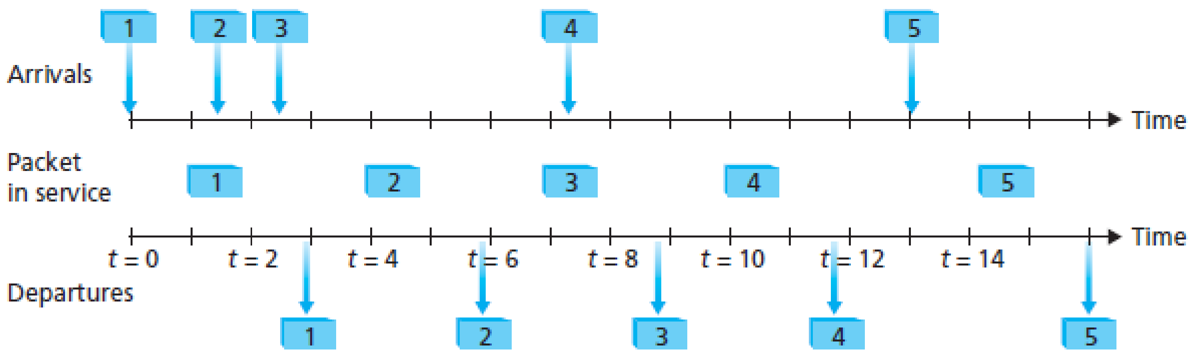
- In the best-effort Internet, the audio and HTTP packets are mixed in the output queue at R1 and (typically) transmitted in a first-in-first-out (FIFO) order.

- In this scenario, a burst of packets from the Web server could potentially fill up the queue, causing IP audio packets to be excessively delayed or lost due to buffer overflow at R1.
- Solution for this is differentiating traffic class and assigning suitable priority to it.
- **Packet marking** allows a router to distinguish among packets belonging to different classes of traffic.
- Now suppose that the router is configured to give priority to packets marked as belonging to the 1 Mbps audio application. Since the outgoing link speed is 1.5 Mbps, even though the HTTP packets receive lower priority, they can still, on average, receive 0.5 Mbps of transmission service. But if the audio application starts sending packets at a rate of 1.5 Mbps or higher, the HTTP packets will starve, that is, they will not receive any service on the R1-to-R2 link.
- Therefore it is desirable to provide a degree of traffic isolation among classes so that one class is not adversely affected by another class of traffic that misbehaves.
- If a traffic class or flow must meet certain criteria, then a policing mechanism can be put into place to ensure that these criteria are indeed observed. If the policed application misbehaves, the policing mechanism will take some action so that the traffic actually entering the network conforms to the criteria.
- A complementary approach for providing isolation among traffic classes is for the link-level packet-scheduling mechanism to explicitly allocate a fixed amount of link bandwidth to each class.
- While providing isolation among classes or flows, it is desirable to use resources (for example, link bandwidth and buffers) as efficiently as possible.

➔ Scheduling Mechanisms

First-In-First-Out (FIFO)

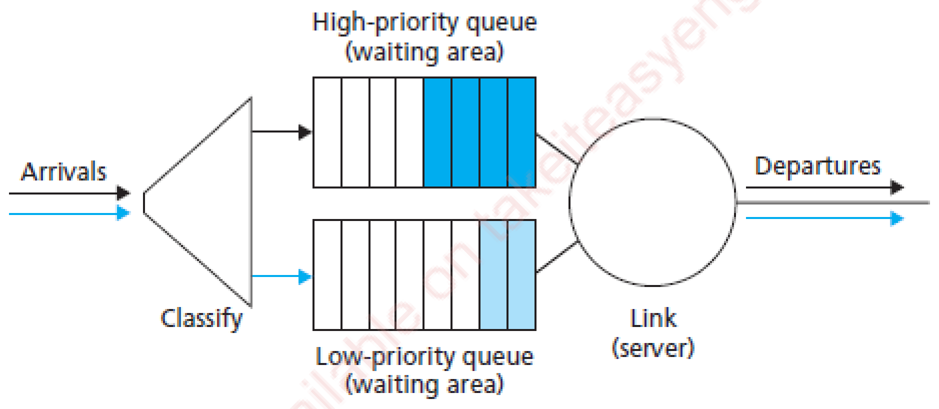
The FIFO (also known as first-come-first-served, or FCFS) scheduling discipline selects packets for link transmission in the same order in which they arrived at the output link queue.

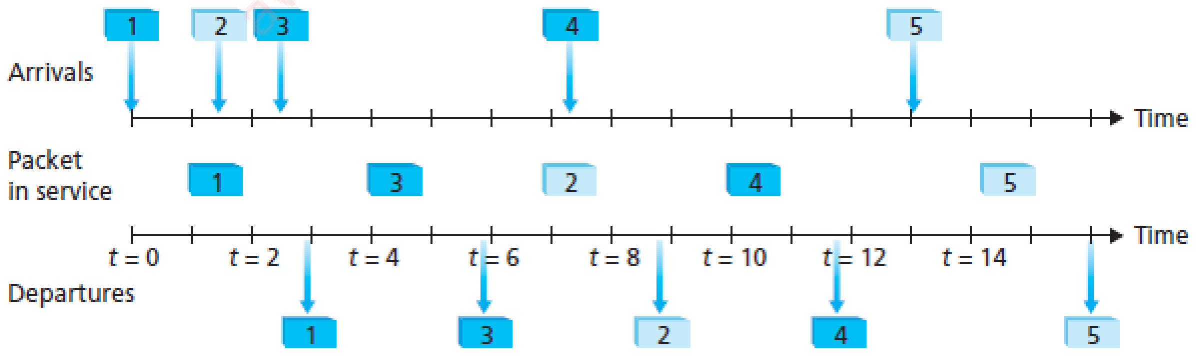


Priority Queuing

Under priority queuing, packets arriving at the output link are classified into priority classes at the output queue.

Each priority class typically has its own queue. When choosing a packet to transmit, the priority queuing discipline will transmit a packet from the highest priority class that has a nonempty queue. The choice among packets in the same priority class is typically done in a FIFO manner.


High-priority queue (waiting area)
Arrivals
Classify
Low-priority queue (waiting area)
Link (server)
Departures



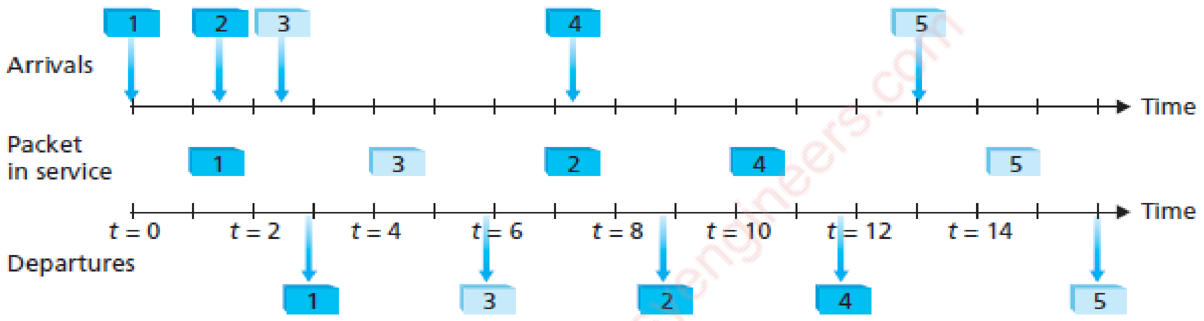
Round Robin

Under the round robin queuing discipline, packets are sorted into classes as with priority queuing.

However, rather than there being a strict priority of service among classes, a round robin scheduler alternates service among the classes.

In the simplest form of round robin scheduling, a class 1 packet is transmitted, followed by a class 2 packet, followed by a class 1 packet, followed by a class 2 packet, and so on.

A work-conserving round robin discipline that looks for a packet of a given class but finds none will immediately check the next class in the round robin sequence.

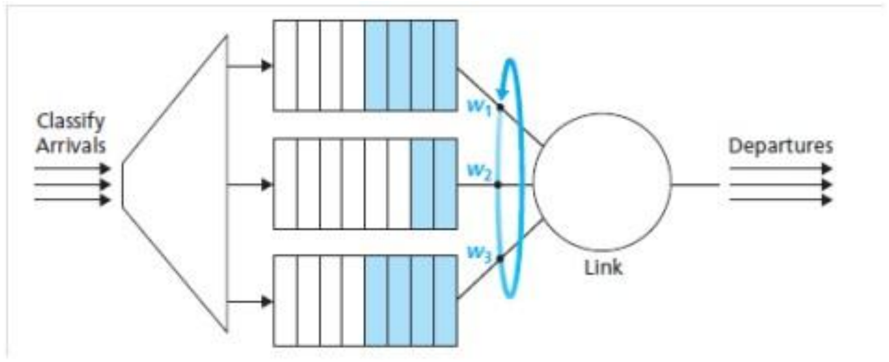


Weighted Fair Queuing (WFQ)

A generalized abstraction of round robin queuing that has found considerable use in QoS architectures is weighted fair queuing (WFQ) discipline.

Here arriving packets are classified and queued in the appropriate per-class waiting area. As in round robin scheduling, a WFQ scheduler will serve classes in a circular manner—first serving class 1, then serving class 2, then serving class 3, and then (assuming there are three classes) repeating the service pattern.

WFQ is also a work-conserving queuing discipline and thus will immediately move on to the next class in the service sequence when it finds an empty class queue.

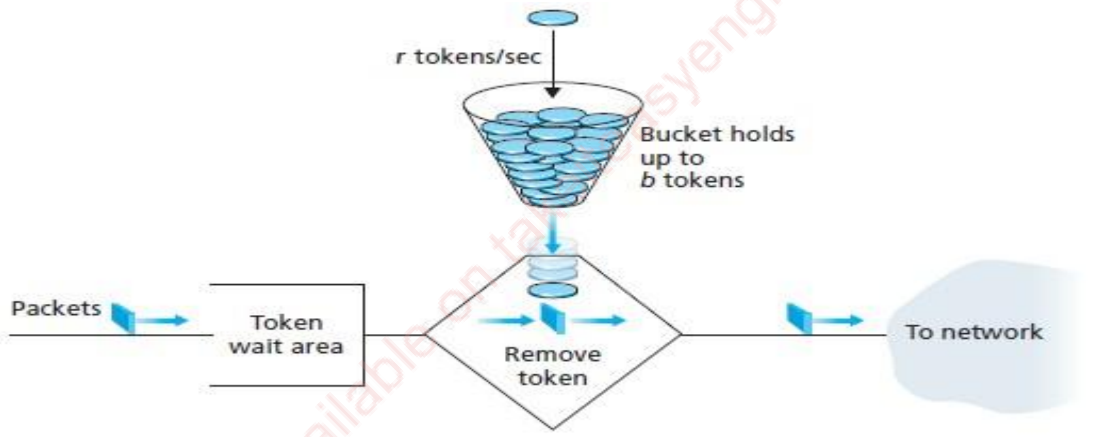


➔ Policing: The Leaky Bucket

Three important policing criteria:

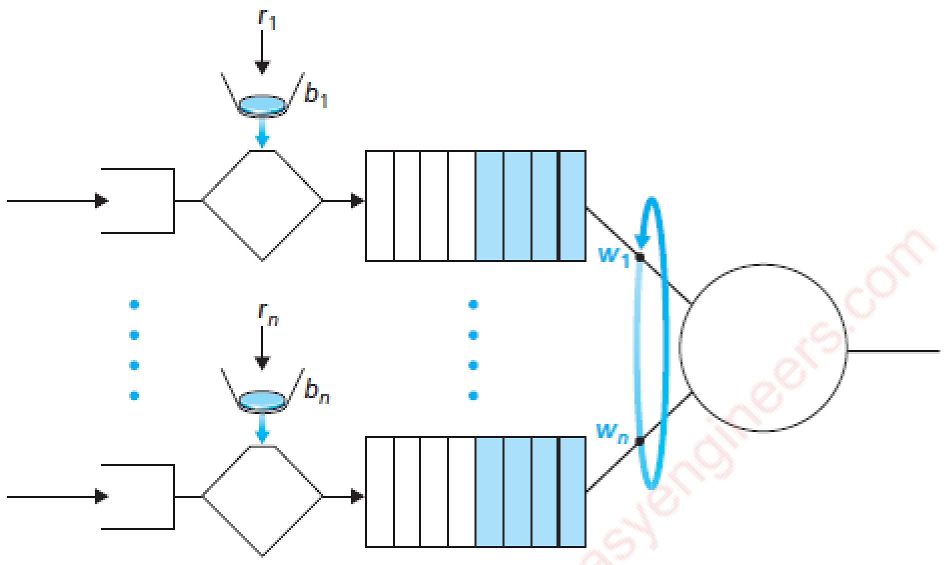
- **Average rate:** The network may wish to limit the long-term average rate (packets per time interval) at which a flow's packets can be sent into the network. A crucial issue here is the interval of time over which the average rate will be policed.
- **Peak rate:** While the average-rate constraint limits the amount of traffic that can be sent into the network over a relatively long period of time, a peak-rate constraint limits the maximum number of packets that can be sent over a shorter period of time.
- **Burst size:** The network may also wish to limit the maximum number of packets (the "burst" of packets) that can be sent into the network over an extremely short interval of time.

The leaky bucket mechanism is an abstraction that can be used to characterize these policing limits.



- A leaky bucket consists of a bucket that can hold up to b tokens.
- Tokens are added to this bucket as follows. New tokens, which may potentially be added to the bucket, are always being generated at a rate of r tokens per second.
- If the bucket is filled with less than b tokens when a token is generated, thenewly generated token is added to the bucket; otherwise the newly generated token is ignored, and the token bucket remains full with b tokens.
- Suppose that before a packet is transmitted into the network, it must first remove a token from the token bucket. If the token bucket is empty, the packet must wait for a token.

- Because there can be at most b tokens in the bucket, the maximum burst size for a leaky-bucket policed flow is b packets. Furthermore, because the token generation rate is r, the maximum number of packets that can enter the network of any interval of time of length t is rt + b.
- Leaky Bucket + Weighted Fair Queuing = Provable Maximum Delay in a Queue



➔ Diffserv

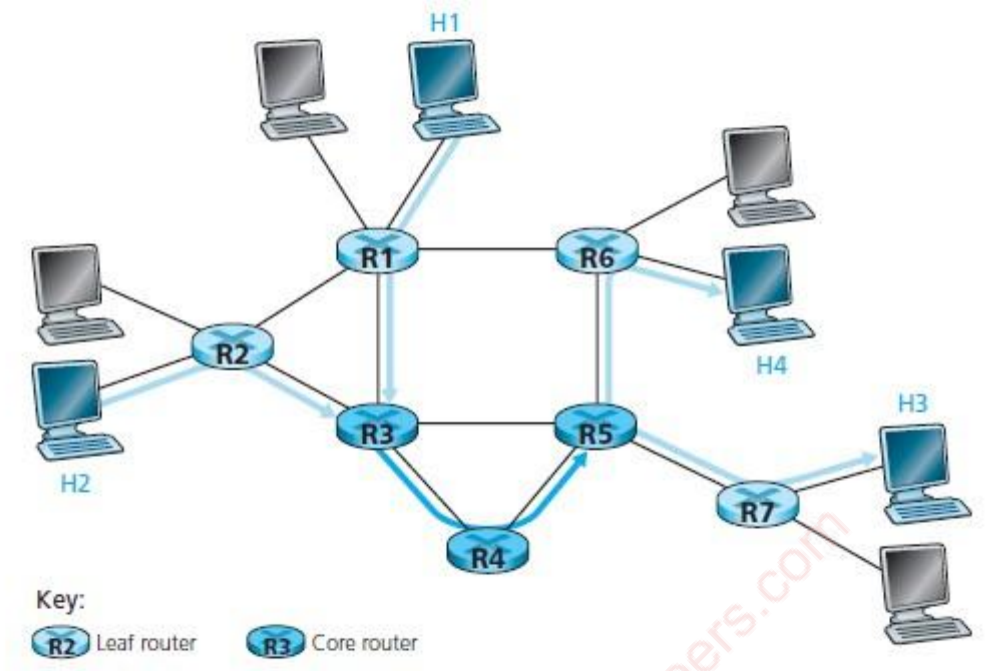
Diffserv provides service differentiation—that is, the ability to handle different classes of traffic in different ways within the Internet in a scalable manner.

The need for scalability arises from the fact that millions of simultaneous source-destination traffic flows may be present at a backbone router.

The Diffserv architecture consists of two sets of functional elements:

1) Edge functions: packet classification and traffic conditioning. At the incoming edge of the network arriving packets are marked. The mark that a packet receives identifies the class of traffic to which it belongs. Different classes of traffic will then receive different service within the core network.

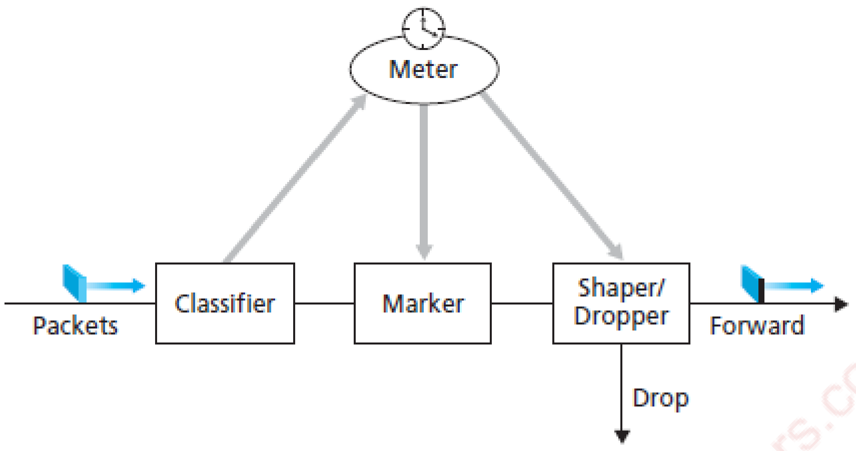
2) Core function: forwarding. When a DS-marked packet arrives at a Diffserv capable router, the packet is forwarded onto its next hop according to the so-called per-hop behavior (PHB) associated with that packet's class. The per-hop behavior influences how a router's buffers and link bandwidth are shared among the competing classes of traffic.



- Packets arriving to the edge router are first classified. The classifier selects packets based on the values of one or more packet header fields (for example, source address, destination address, source port, destination port, and protocol ID) and steers the packet to the appropriate marking function.
- In some cases, an end user may have agreed to limit its packet-sending rate to conform to a declared traffic profile. The traffic profile might contain a limit on the peak rate, as well as the burstiness of the packet flow.
- As long as the user sends packets into the network in a way that conforms to the negotiated traffic profile, the packets receive their priority marking and are forwarded along their route to the destination.
- On the other hand, if the traffic profile is violated, out-of-profile packets might be marked differently, might be shaped (for example, delayed so that a maximum rate constraint would be observed), or might be dropped at the network edge.
- The role of the metering function, is to compare the incoming packet flow with the negotiated traffic profile and to determine whether a packet is within the negotiated traffic profile.
- The second key component of the Diffserv architecture involves the per-hop behavior (PHB) performed by Diffserv-capable routers. PHB is rather cryptically, but carefully, defined as "a

description of the externally observable forwarding behavior of a Diffserv node applied to a particular Diffserv behavior aggregate”.

- A PHB can result in different classes of traffic receiving different performance.

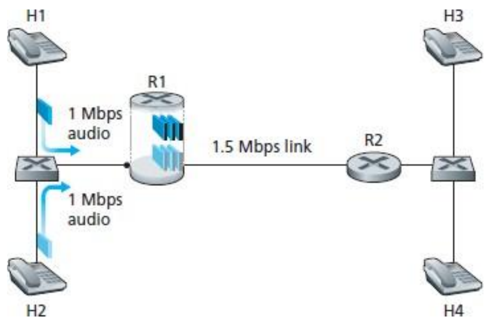


- The **expedited forwarding** PHB specifies that the departure rate of a class of traffic from a router must equal or exceed a configured rate.
- The **assured forwarding** PHB divides traffic into four classes, where each AF class is guaranteed to be provided with some minimum amount of bandwidth and buffering.

<u>Per-Connection Quality-of-Service (QoS) Guarantees: Resource Reservation and Call Admission</u>

Consider two 1 Mbps audio applications transmitting their packets over the 1.5 Mbps link. The combined data rate of the two flows (2 Mbps) exceeds the link capacity.

There is simply not enough bandwidth to accommodate the needs of both applications at the same time. If the two applications equally share the bandwidth, each application would lose 25 percent of its transmitted packets.



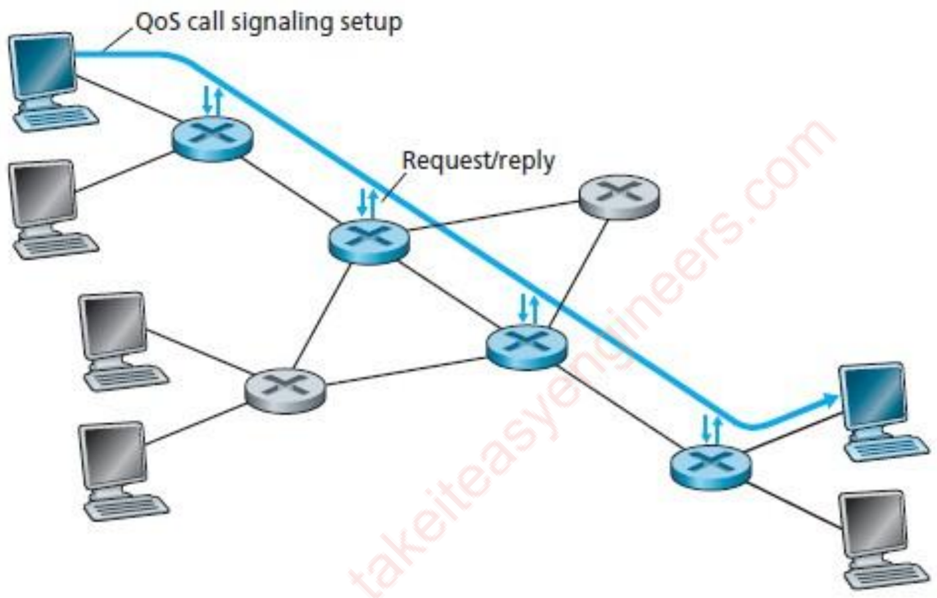
If sufficient resources will not always be available, and QoS is to be guaranteed, a call admission process is needed in which flows declare their QoS requirements and are then either admitted to the network (at the required QoS) or blocked from the network (if the required QoS cannot be provided by the network).

The process of having a flow declare its QoS requirement, and then having the network either accept the flow (at the required QoS) or block the flow is referred to as the call admission process.

Resource reservation: The only way to guarantee that a call will have the resources (link bandwidth, buffers) needed to meet its desired QoS is to explicitly allocate those resources to the call—a process known in networking parlance as resource reservation. Once resources are reserved, the call has on-demand access to these resources throughout its duration, regardless of the demands of all other calls. If a call reserves and receives a guarantee of x Mbps of link bandwidth, and never transmits at a rate greater than x, the call will see loss- and delay-free performance.

Call admission: If resources are to be reserved, then the network must have a mechanism for calls to request and reserve resources. Since resources are not infinite, a call making a call admission request will be denied admission, that is, be blocked, if the requested resources are not available. Such a call admission is performed by the telephone network—we request resources when we dial a number. If the circuits (TDMA slots) needed to complete the call are available, the circuits are allocated and the call is completed. If the circuits are not available, then the call is blocked, and we receive a busy signal. A blocked call can try again to gain admission to the network, but it is not allowed to send traffic into the network until it has successfully completed the call admission process. Of course, a router that allocates link bandwidth should not allocate more than is available at that link. Typically, a call may reserve only a fraction of the link's bandwidth, and so a router may allocate link bandwidth to more than one call. However, the sum of the allocated bandwidth to all calls should be less than the link capacity if hard quality of service guarantees are to be provided.

Call setup signaling: The call admission process described above requires that a call be able to reserve sufficient resources at each and every network router on its source-to-destination path to ensure that its end-to-end QoS requirement is met. Each router must determine the local resources required by the session, consider the amounts of its resources that are already committed to other ongoing sessions, and determine whether it has sufficient resources to satisfy the per-hop QoS requirement of the session at this router without violating local QoS guarantees made to an already-admitted session. A signaling protocol is needed to coordinate these various activities—the per-hop allocation of local resources, as well as the overall end-to-end decision of whether or not the call has been able to reserve sufficient resources at each and every router on the end-to-end path. The RSVP protocol was proposed for this purpose within an Internet architecture for providing qualityof- service guarantees.


QoS call signaling setup
Request/reply