CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Scheme Of Evaluation
Internal Assessment Test 3 – October.2019

Sub:	Computer Networks							**Code:**	17CS52
Date:	19 / 11 / 2019	Duration:	90mins	Max Marks:	50	**Sem:**	VI	**Branch:**	ISE

Note: Answer Any Five Questions

Question #		**Description**	**Marks Distribution**		**Max Marks**
1	a)	Discuss Mobile IP in detail. • Advertisement • Registration	5M 5M	10M	10 M
2	a)	Draw 3G architecture and interpret its components. • Diagram • Explanation	5M 5M	5M	10 M
3	a)	Describe in detail about Diffserv. • Path attributes • Selection policy	5M 5M	6M	10 M
4	a)	Compare and contrast DASH and HTTP streaming. • HTTP • DASH	5M 5M	10M	10 M
5	a)	Compare different scheduling mechanisms. • FIFO • Round Robin • WFQ • Leaky Bucket	2.5M 2.5M 2.5M 2.5M	10M	10 M
6	a)	List the features Netflix, youtube and Kankan • Features	5 M	6M	10 M
	b)	Illustrate the way of providing multiple class of service. • Methods	5M	4M	

7	a)	Discuss about per connection QOS. • Introduction • Description	2M 8M	10M	10 M

Answers

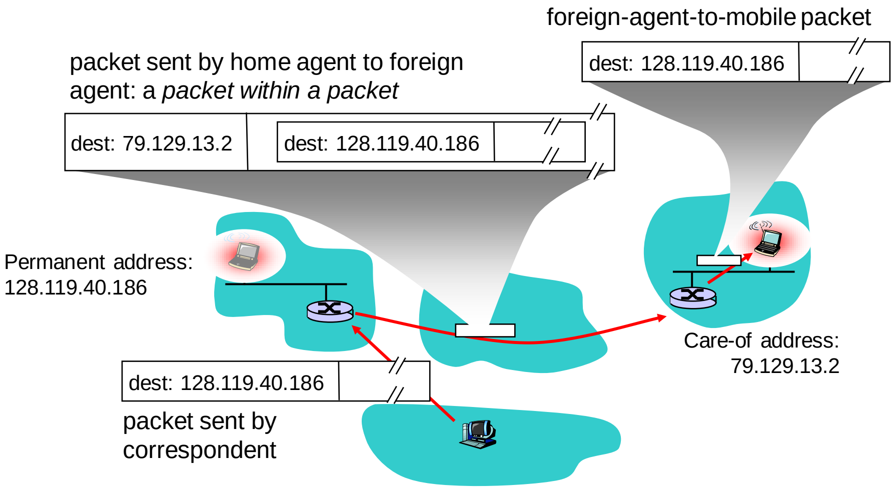
1.

5 (a)	Explain Mobile IP in detail.

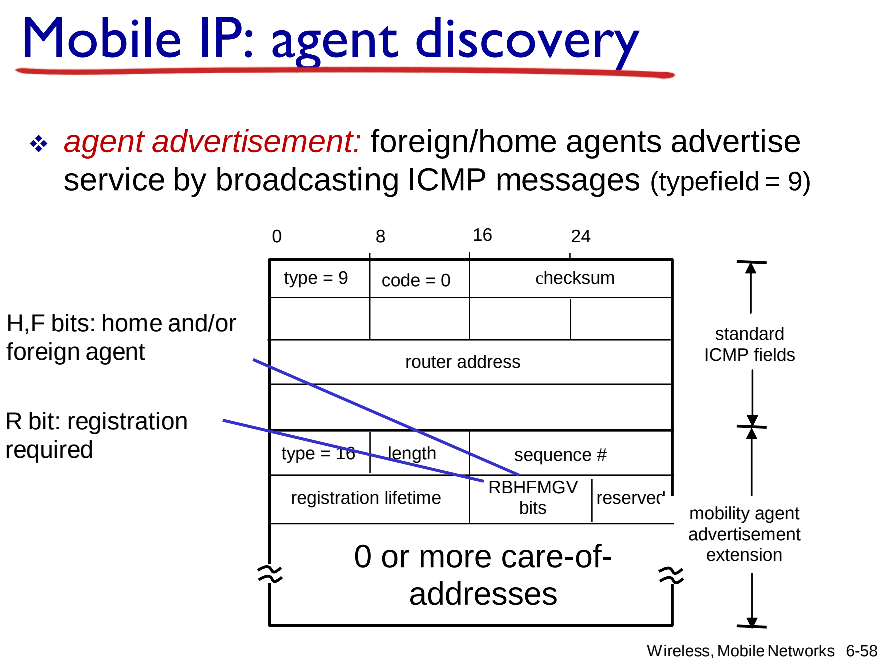
Mobile IP

- RFC 3344
- has many features we' ve seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - indirect routing of datagrams
  - agent discovery
  - registration with home agent

Mobile IP: indirect routing



foreign-agent-to-mobile packet
dest: 128.119.40.186
packet sent by home agent to foreign agent: a *packet within a packet*
dest: 79.129.13.2
dest: 128.119.40.186
Permanent address: 128.119.40.186
Care-of address: 79.129.13.2
dest: 128.119.40.186
packet sent by correspondent



Mobile IP: agent discovery

❖ *agent advertisement:* foreign/home agents advertise service by broadcasting ICMP messages (typefield = 9)

0	8	16	24
type = 9	code = 0	checksum	
router address			
type = 16	length	sequence #	
registration lifetime		RBHFMGV bits	reserved
0 or more care-of-addresses			

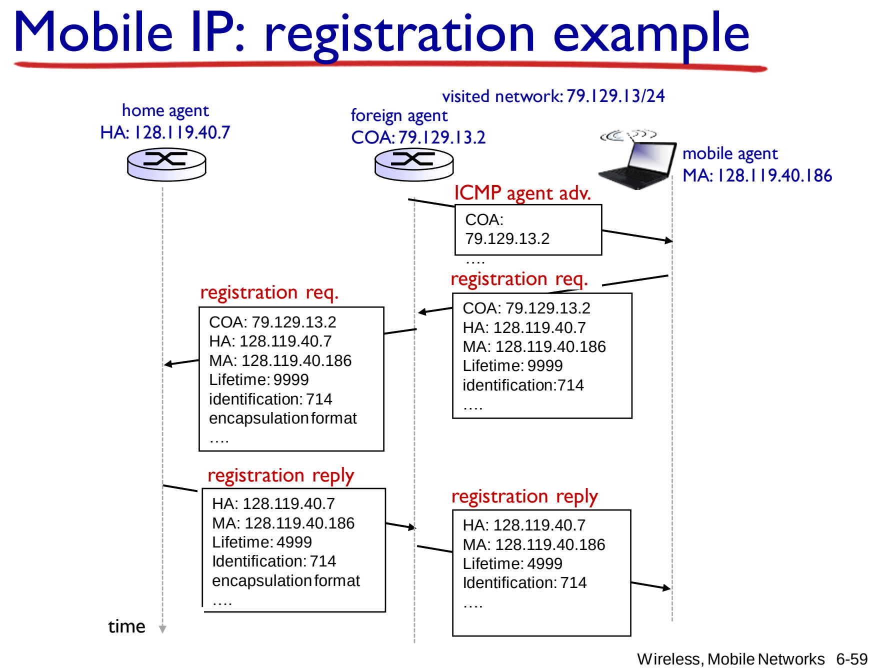
H,F bits: home and/or foreign agent

R bit: registration required

standard ICMP fields

mobility agent advertisement extension

Wireless, Mobile Networks 6-58



Mobile IP: registration example

ICMP agent adv.

COA:
79.129.13.2
….

registration req.

COA: 79.129.13.2
HA: 128.119.40.7
MA: 128.119.40.186
Lifetime: 9999
identification:714
….

registration req.

COA: 79.129.13.2
HA: 128.119.40.7
MA: 128.119.40.186
Lifetime: 9999
identification: 714
encapsulation format
….

registration reply

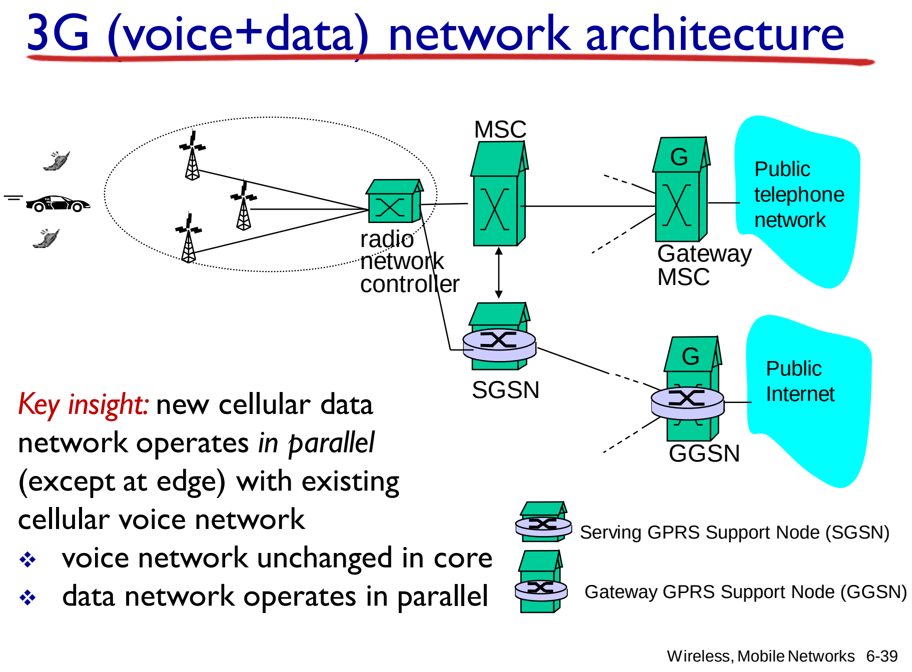
HA: 128.119.40.7
MA: 128.119.40.186
Lifetime: 4999
Identification: 714
encapsulation format
….

registration reply

HA: 128.119.40.7
MA: 128.119.40.186
Lifetime: 4999
Identification: 714
….

Wireless, Mobile Networks 6-59

2 Draw 3G architecture and explain all the components of the architecture.

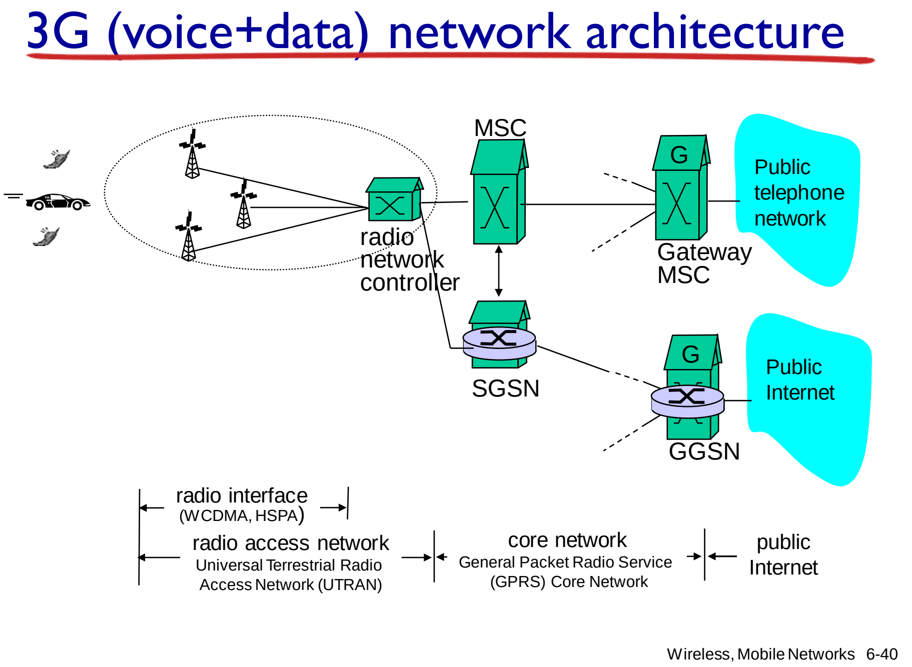


3G (voice+data) network architecture

Key insight: new cellular data network operates *in parallel* (except at edge) with existing cellular voice network

- voice network unchanged in core
- data network operates in parallel

Serving GPRS Support Node (SGSN)
Gateway GPRS Support Node (GGSN)



3G (voice+data) network architecture

3HTTP Streaming (10)

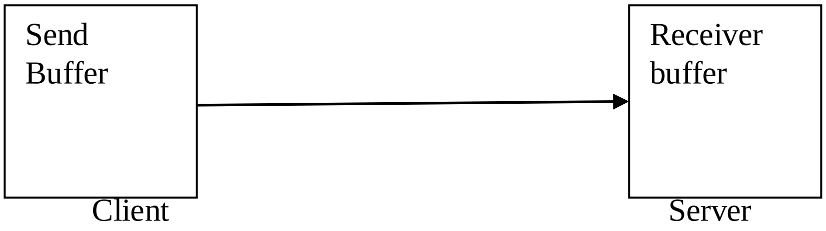
Introduction (3)

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 HTTPGET 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

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. To gain some insight into perfecting, let's take a look at a simple example. 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 : (5)

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.



Streaming stored video using HTTP Streaming

Early Termination and Repositioning the Video HTTP streaming: (2)

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.

1. UDP Streaming (5)

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. UDP streaming typically uses a small client-side buffer, big enough to hold less than a second of video. 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 scheme.

Another distinguishing property of UDP streaming is that in addition to the server- to-client video stream, 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 (RTSP) [RFC 2326], explained in some detail in the companion Web site for this textbook, is a popular open protocol for such a control connection. Although UDP streaming has been employed in many open-source systems and proprietary products, it suffers from three significant drawbacks.

First, 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. For example, consider the scenario where the video consumption rate is 1 Mbps and the server- to-client available bandwidth is usually more than 1 Mbps, but every few minutes the available bandwidth drops below 1 Mbps for several seconds. In such a scenario, a UDP streaming system that transmits video at a constant rate of 1 Mbps over RTP/UDP would likely provide a poor user experience, with freezing or skipped frames soon after the available bandwidth falls below 1 Mbps.

The second drawback of UDP streaming is that it requires a media control server, such as an RTSP server, to process client-to-server interactivity requests and to track client state (e.g., the client's play out 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. The third drawback is that many firewalls are configured to block UDP traffic, preventing the users behind these firewalls from receiving UDP video.

Adaptive streaming (5)

Although HTTP streaming has been extensively deployed in practice (for example, by YouTube since its inception), it 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.

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. This feature is particularly important for mobile users, who typically see their bandwidth availability fluctuate as they move with respect to the base stations. Comcast, for example, has deployed an adaptive streaming system in which each video source file is encoded into 8 to 10 different MPEG-4 formats, allowing the highest quality video format to be streamed to the client, with adaptation being performed in response to changing network and device conditions. 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 down- loading 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. Since a sudden drop in bit rate by changing versions may result in noticeable visual quality degradation, the bit-rate reduction may be achieved using multiple intermediate versions to smoothly transition to a rate where the client's consumption rate drops below its available receive bandwidth. When the network conditions improve, the client can then later choose chunks from higher bit-rate versions. By dynamically monitoring the available bandwidth and client buffer level, and adjusting the transmission rate with version switching, DASH can often achieve continuous play out at the best possible quality level without frame freezing or skip- ping. Furthermore, since the client (rather than the server) maintains the intelligence to determine which chunk to send next, the scheme also improves server-side scalability.

Another benefit of this approach is that the client can use the HTTP byte-range request to precisely control the amount of perfected video that it buffers locally. We conclude our brief discussion of DASH by mentioning that for many implementations, the server not only stores many versions of the video but also separately stores many versions of the audio. Each audio version has its own quality level and bit rate and has its own URL. In these implementations, the client dynamically selects both video and audio chunks, and locally synchronizes audio and video play out.

2. Content Distribution Networks
   Issues (3)
   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 [Ding 2011]. Streaming all this traffic to locations all over the world while providing continuous play out 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.

First, 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. If one of these links provides a throughput that is less than the video consumption rate, the end-to-end throughput will also be below the consumption rate, resulting in annoying freezing delays for the user. The likelihood of this happening increases as the number of links in the end-to-end path increases.

A second drawback is that 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.

A third problem with this solution is that 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. 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). ACDN 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.

• 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

Akamai takes this approach with clusters in approximately 1,700 locations. 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. Because of this highly distributed design, the task of maintaining and managing the clusters becomes challenging.

To support its vast array of cloud services—including search, gmail, calendar, YouTube video, maps, documents, and social networks—Google has deployed an extensive private network and CDN infrastructure. Google's CDN infrastructure has three tiers of server clusters:

• Eight "mega data centers," with six located in the United States and two located in Europe [Google Locations 2012], with each data center having on the order of 100,000 servers. These mega data centers are responsible for serving dynamic (and often personalized) content, including search results and gmail messages.

• About 30 "bring-home" clusters , with each cluster consisting on the order of 100–500 servers. The cluster locations are distributed around the world, with each location typically near multi- ple tier-1 ISP PoPs. These clusters are responsible for serving static content, including YouTube videos

• Many hundreds of "enter-deep" clusters with each cluster located within an access ISP. Here a cluster typically consists of tens of servers within a single rack. These enter-deep servers perform TCP splitting and serve static content including the static portions of Web pages that embody search results. All of these data centers and cluster locations are networked together with Google's own private network, as part of one enormous AS. When a user makes a search query, often the query is first sent over the local ISP to a nearby enter-deep cache, from where the static content is retrieved; while providing the static content to the client, the nearby cache also forwards the query over Google's private network to one of the mega data centers, from where the personalized search results are retrieved. For a YouTube video, the video itself may come from one of the bring-home caches, whereas portions of the Web page surrounding the video may come from the nearby enter-deep cache, and the advertisements surrounding the video come from the data centers. In summary, except for the local ISPs, the Google cloud services are largely provided by a network infrastructure that is independent of the public Internet.

• Bring Home.(3) 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, for example, within a few miles of both AT&T and Verizon PoPs in a major city. 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.

Once its clusters are in place, the CDN replicates content across its clusters. The CDN may not want to place a copy of every video in each cluster, since some videos are rarely viewed or are only popular in some countries. In fact, many CDNs do not push videos to their clusters but instead use a simple pull strategy: If a client requests a video from a cluster that is not storing the video, then the cluster retrieves the video (from a central repository or from another cluster) and stores a copy locally while streaming the video to the client at the same time. Similar to Internet caches when a cluster's storage becomes full, it removes videos that are not frequently requested.

CDN Operation Having identified the two major approaches toward deploying a CDN, let's now dive down into the nuts and bolts of how a CDN operates. 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 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.

Steps: (2)

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 authorita- tive 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 IPaddresses 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 IPaddress 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 : (2) 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. As we just saw, 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. CDNs generally employ proprietary cluster selection strategies. E.g: BGP

3. Scheduling mechanisms

First-In-First-Out (FIFO): (3)

Packets arriving at the link output queue wait for transmission if the link is currently busy transmitting another packet. If there is not sufficient buffering space to hold the arriving packet, the queue's packet-discarding policy then determines whether the packet will be dropped (lost) or whether other packets will be removed from the queue to make space for the arriving packet. In our discussion below, we will ignore packet discard. When a packet is completely transmitted over the out- going link (that is, receives service) it is removed from the queue. 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: (3)

Under priority queuing, packets arriving at the output link are classified into priority classes at the output queue. A packet's priority class may depend on an explicit marking that it carries in its packet header (for example, the value of the ToS bits in an IPv4 packet), its source or destination IP address, its destination port number, or other criteria. 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 (that is, has packets waiting for transmission). The choice among packets in the same priority class is typically done in a FIFO manner.

Round Robin and Weighted Fair Queuing (WFQ): (4)

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 so-called work-conserving queuing discipline will never allow the link to remain idle whenever there are packets (of any class) queued for transmission. 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.

4. A. Leaky bucket
   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. A flow whose average rate is limited to 100 packets per second is more constrained than a source that is limited to 6,000 packets per minute, even though both have the same average rate over a long enough interval of time. For example, the latter constraint would allow a flow to send 1,000 packets in a given second-long interval of time, while the former constraint would disallow this sending behavior.
   • 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 con- straint limits the maximum number of packets that can be sent over a shorter period of time. Using our example above, the network may police a flow at an average rate of 6,000 packets per minute, while limiting the flow's peak rate to 1,500 packets per second. • 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. In the limit, as the interval length approaches zero, the burst size limits the number of packets that can be instantaneously sent into the network. Even though it is physically impossible to instantaneously send multiple packets into the network (after all, every link has a physical transmission rate that cannot be exceeded!), the abstraction of a maximum burst size is a useful one.
   The leaky bucket mechanism is an abstraction that can be used to characterize these policing limits.

   Leaky Bucket + Weighted Fair Queuing = Provable Maximum Delay in a Queue

   b. Diffserv (5)
   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. We'll see shortly that this need is met by placing only simple functionality within the network core, with more complex control operations being implemented at the network's edge. Let's begin with the simple network . The Diffserv architecture consists of two sets of functional elements:
   • Edge functions: packet classification and traffic conditioning. At the incoming edge of the network (that is, at either a Diffserv-capable host that generates traffic or at the first

Diffserv-capable router that the traffic passes through), arriving packets are marked. More specifically, the differentiated service (DS) field in the IPv4 or IPv6 packet header is set to some value.

The definition of the DS field is intended to supersede the earlier definitions of the IPv4 type- of-service field and the IPv6 traffic class fields.

Different classes of traffic will then receive different service within the core network.

• 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. A crucial tenet of the Diffserv architecture is that a router's per- hop behavior will be based only on packet markings, that is, the class of traffic to which a packet belongs. An analogy might prove useful here. At many large-scale social events (for example, a large public reception, a large dance club or discothèque, a concert, or a football game), people entering the event receive a pass of one type or another: VIP passes for Very Important People; over-21 passes for people who are 21 years old or older (for example, if alcoholic drinks are to be served); backstage passes at concerts; press passes for reporters; even an ordinary pass for the Ordinary Person. These passes are typically distributed upon entry to the event, that is, at the edge of the event. It is here at the edge where computationally intensive operations, such as paying for entry, checking for the appropriate type of invitation, and matching an invitation against a piece of identification, are performed. Furthermore, there may be a limit on the number of people of a given type that are allowed into an event. If there is such a limit, people may have to wait before entering the event. Once inside the event, one's pass allows one to receive differentiated service at many locations around the event—a VIP is provided with free drinks, a better table, free food, entry to exclusive rooms, and fawning service. Conversely, an ordinary person is excluded from certain areas, pays for drinks, and receives only basic service. In both cases, the service received within the event depends solely on the type of one's pass. Moreover, all people within a class are treated alike. 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. The traffic profile might contain a limit on the peak rate, as well as the burstiness of the packet flow, as we saw previously with the leaky bucket mechanism. 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 pro- file 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 actual decision about whether to immediately remark, forward, delay, or drop a packet is a policy issue deter- mined by the network administrator and is not specified in the Diffserv architecture. So far, we have focused on the marking and policing functions in the Diffserv architecture. 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"

5. -Connection Quality-of-Service (QoS) Guarantees (10)
Introduction (2)
Under certain scheduling disciplines, such as priority scheduling, the lower classes of traffic are essentially "invisible" to the highest-priority class of traffic. With proper network dimensioning, the highest class of service can indeed achieve extremely low packet loss and delay—essentially circuit-like performance. But can the network guarantee that an ongoing flow in a high-priority traffic class will continue to receive such service throughout the flow's duration using only the mechanisms.

Even with classification and marking, isolation of flows, and sharing of unused bandwidth (of which there is none), this is clearly a losing proposition. 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. This is such an unacceptably low QoS that both audio applications are completely unusable; there's no need even to transmit any audio packets in the first place. telephone network) can- not be allocated to the call, the call is blocked (prevented from entering the network) and a busy signal is returned to the user. In our example, there is no gain in allowing a flow into the network if it will not receive a sufficient QoS to be considered usable. Indeed, there is a cost to admitting a flow that does not receive its needed QoS, as network resources are being used to support a flow that provides no utility to the end user.
By explicitly admitting or blocking flows based on their resource requirements, and the source requirements of already-admitted flows, the network can guarantee that admitted flows will be able to receive their requested QoS. Implicit in the need to provide a guaranteed QoS to a flow is the need for the flow to declare its QoS requirements. This 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. This then is our fourth insight into the mechanisms needed to pro- vide QoS.
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).
• Resource reservation.(3)
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.(3) 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. (2)

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