Networks Lecture 14

Paolo Ciancarini

Innopolis University

March 4, 2022

Source of the material

- This lecture is based on the following resources
  - Chapter 7 (Multimedia networking) of Computer Networking: A Top Down Approach (6th edition) by Jim Kurose and Keith Ross
  - Section 2.6 (Video Streaming and Content Distribution Networks) Computer Networking: A Top Down Approach (8th edition) by Jim Kurose and Keith Ross
  - The material is aligned and add/deleted according to the need of the students.

Topic of the lecture

- Introduction to multimedia networking
- Multimedia networking applications
- Streaming stored video
- Voice-over-IP (VoIP)
- Multimedia networking protocols
- Network support for multimedia

Topic of the tutorial

- Recap of Multimedia Networking
- Questions & Answers

Topic of the lab

- Continuation of the labs

Introduction

- Multimedia technology that enables humans to use computers for processing
  - textual data,
  - audio and video,
  - still pictures, and animation.

- Today, people not only use the internet for communication but also to upload/watch videos (YouTube,TikTok), make internet calls (MS Teams, Skype and Google talk) etc.
- Streaming video—including Netflix, YouTube and Amazon Prime—account for about 80% of Internet traffic in 2020

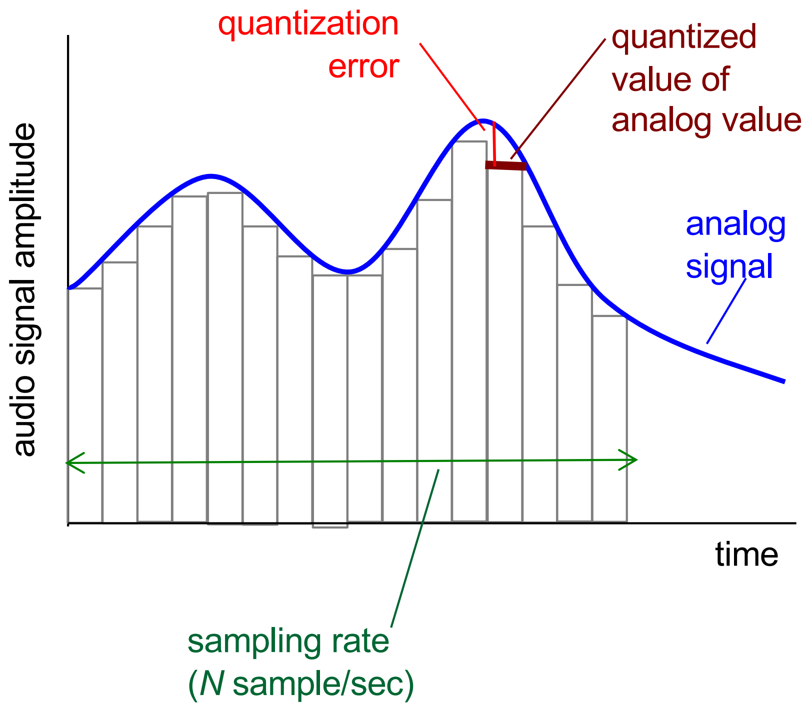
Multimedia Networking Applications

- Multimedia network application is any network application that employs audio/video.
- To understand internet multimedia applications, first we look at the characteristics of video and audio.

Multimedia: Audio

- Analog audio signal sampled at constant rate
  - Telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec

- Each sample quantized, (rounded)
  - For example: 2^8=256 possible quantized values
  - Each quantized value represented by bits
    - For example: 8 bits for 256 values



Example:

- 8,000 samples/sec,
- 256 quantized values: 64,000 bps

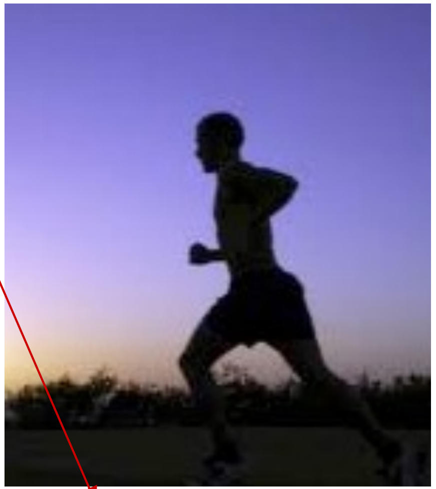
Multimedia: Video

- **Video:** sequence of images displayed at constant rate
  - For example: 24 images/sec
- **Digital image:** array of pixels
  - Each pixel represented by bits
- **Coding:** use redundancy *within* and *between* images to decrease # bits used to encode image
  - Spatial (within image)
  - Temporal (from one image to next)

Spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)

frame *i*

Temporal coding example: instead of sending complete frame at i+1, send only differences from frame i



frame *i+1*

Types of Multimedia Network Applications

- *Streaming stored* audio/video
  - *Streaming:* can begin playout before downloading entire file
  - *Stored (at server):* can transmit faster than audio/video will be rendered (implies storing/buffering at client)
    - For example: YouTube, Netflix

- *Streaming live* audio, video
  - For example: live sporting event (football)

- *Conversational* voice/video over IP
  - Interactive nature of human-to-human conversation limits delay tolerance
    - For example: Skype

Topic of the lecture

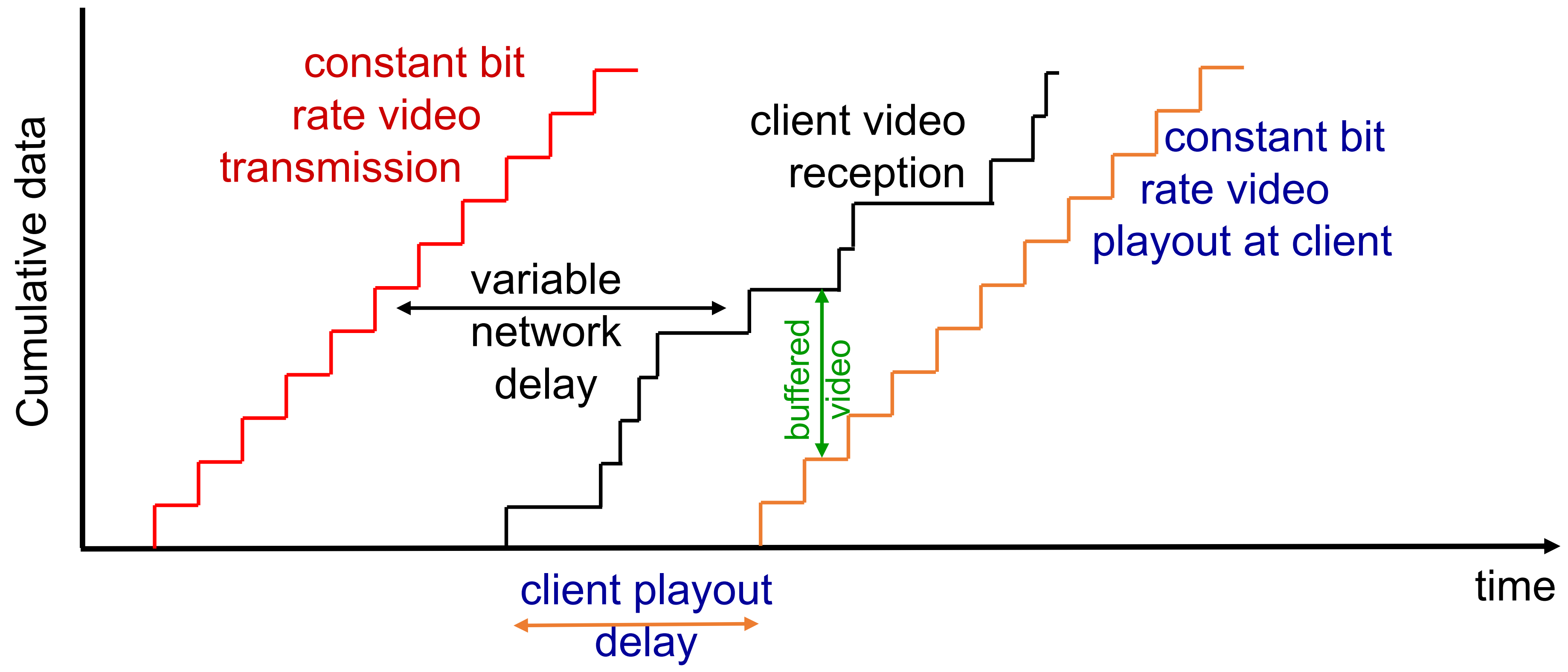
- Introduction
- Multimedia networking applications
- Streaming stored video
- Voice-over-IP
- Multimedia networking protocols
- Network support for multimedia

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.

- A 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 Stored Video



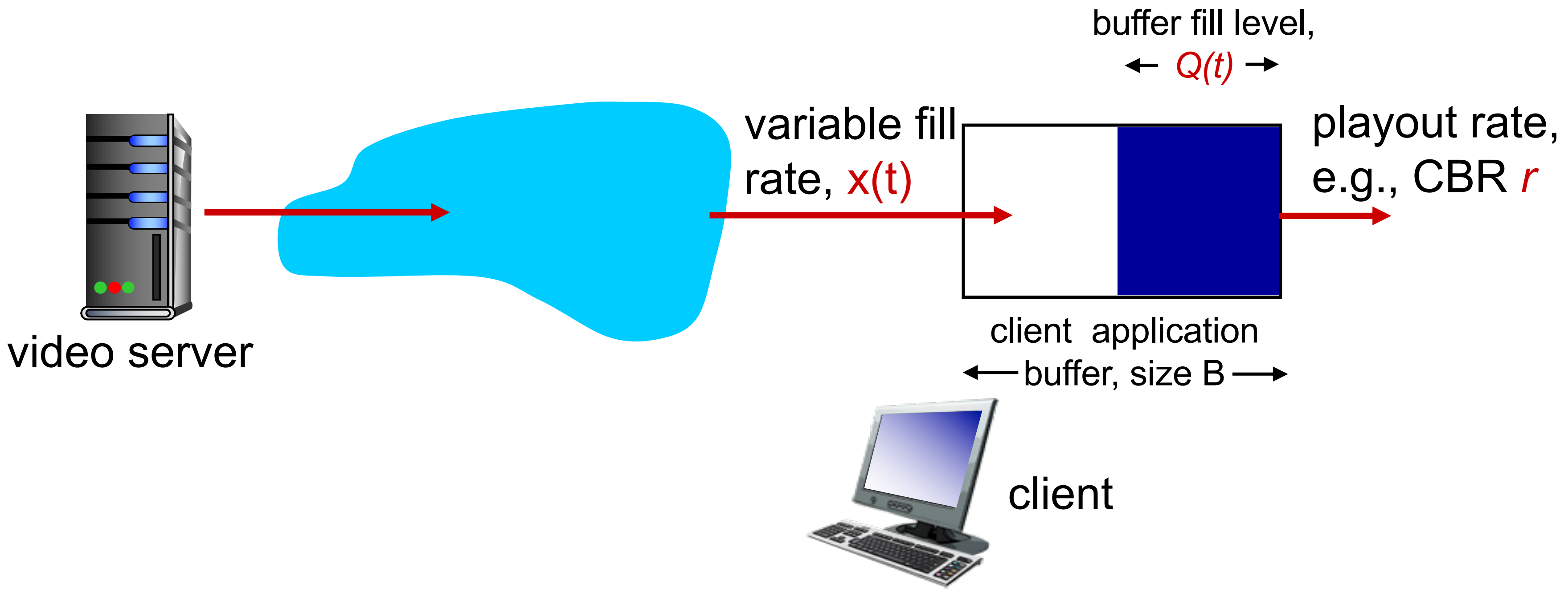
- *Client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Streaming Stored Video: Challenges

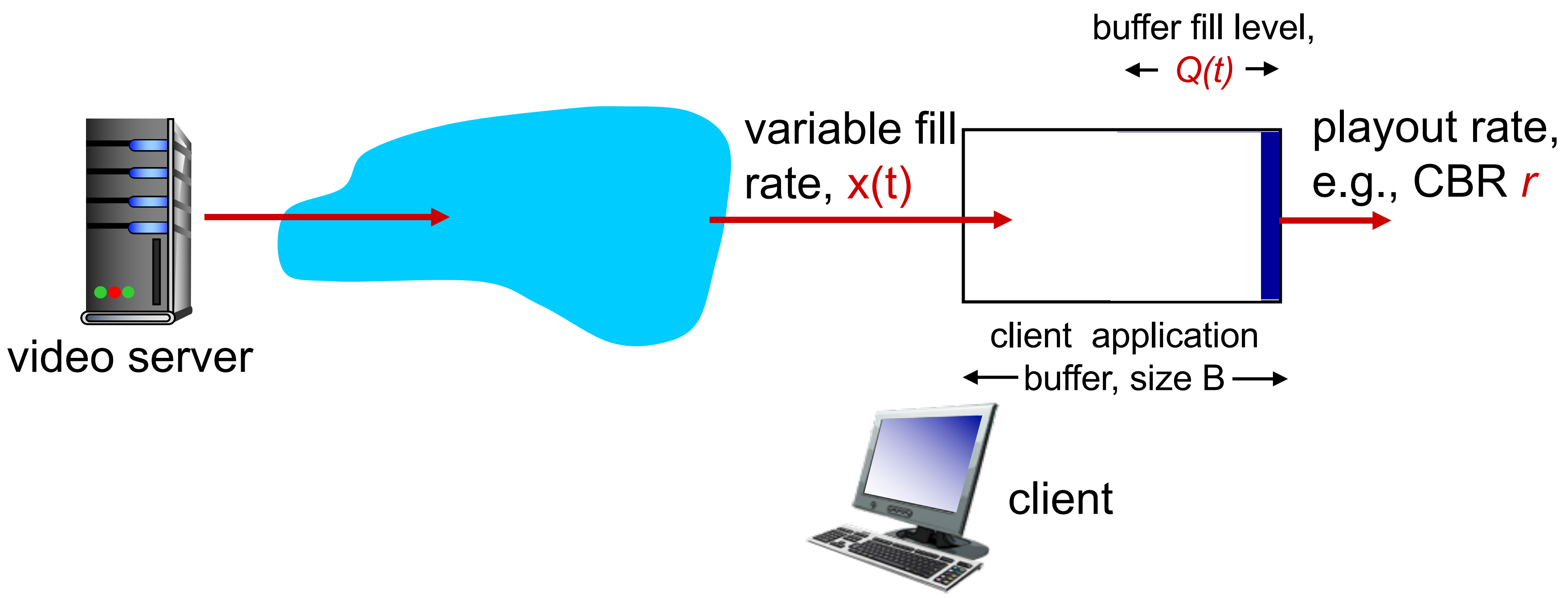
- Continuous Playout constraint: once client Playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match Playout requirements

- Other challenges:
  - Client interactivity: pause, fast-forward, rewind, jump through video
  - Video packets may be lost, retransmitted

Client-side Buffering, Playout



Client-side buffering, Playout



1. Initial fill of buffer until playout begins at t_p

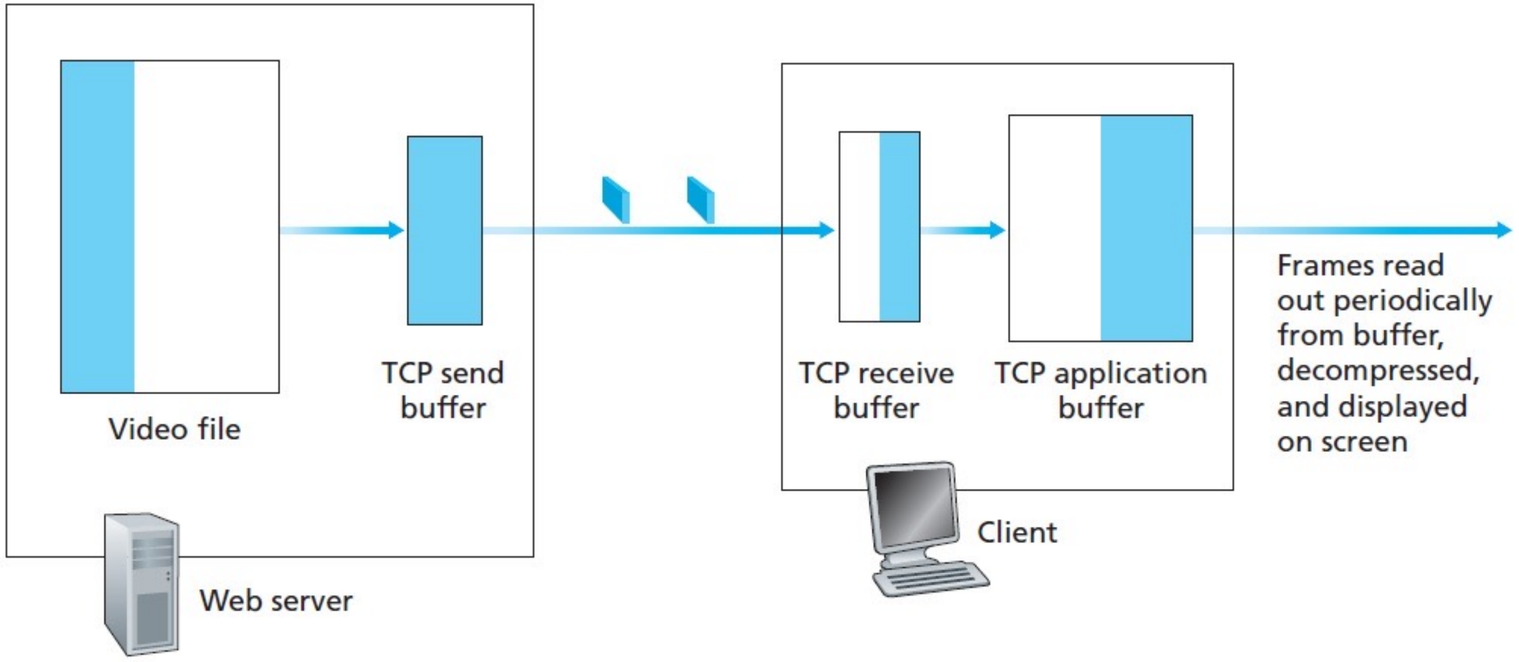
2. Playout begins at t_p,

3. Buffer fill level varies over time as fill rate x(t) varies and playout rate r is constant

Advantages of client-Side Buffering

- Client side buffering can absorb variations in server-to-client delays

- If the server-to-client bandwidth briefly drops below the video consumption rate, a user can continue to enjoy continuous playback, as long as the client application buffer does not become completely drained.



Streaming Stored Video

Streaming systems can be classified as:

- UDP Streaming
- HTTP Streaming
- Adaptive HTTP Streaming

UDP Streaming

- Server sends data at rate appropriate for client

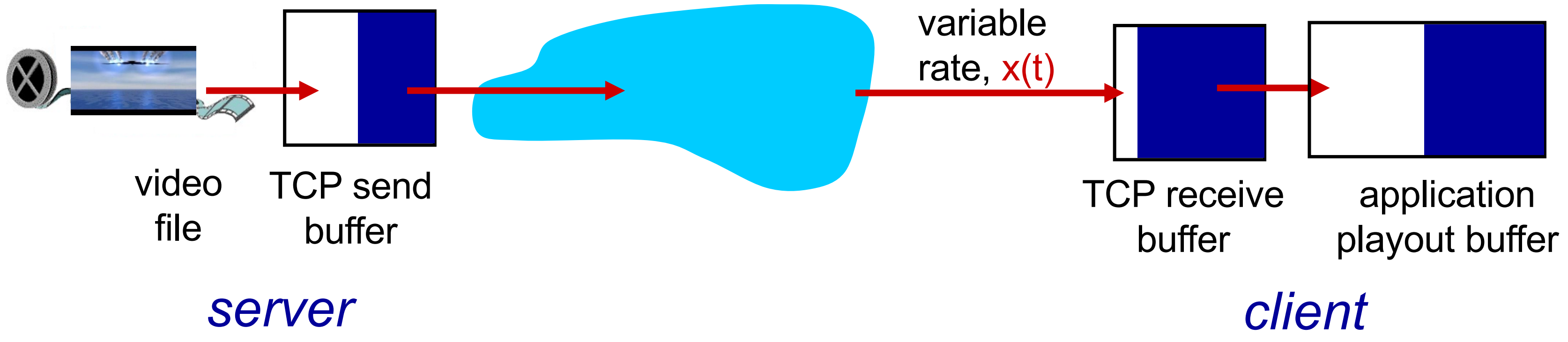
- Drawbacks of UDP Streaming
  - Its unpredictable and varying amount of available bandwidth between server and client, constant-rate UDP streaming can fail to provide continuous playout

  - It requires a media control server, such as RTSP server, to process client-to-server interactivity requests and to track client state

  - Many firewalls are configured to block UDP traffic, preventing the users behind these firewalls from receiving UDP video

Streaming multimedia: HTTP

- Multimedia file retrieved via HTTP GET
- Send at maximum possible rate under TCP



- Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- Larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

Streaming Multimedia: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

- *Server:*
  - Divides video file into multiple chunks
  - Each chunk stored, encoded at different rates
  - *Manifest file:* provides URLs for different chunks
- *Client:*
  - Periodically measures server-to-client bandwidth
  - Consulting manifest, requests one chunk at a time
    - Chooses maximum coding rate sustainable given current bandwidth
    - Can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming Multimedia: DASH

- *"Intelligence"* at client: client determines
  - *When* to request chunk (so that buffer starvation, or overflow does not occur)
  - *What encoding rate* to request (higher quality when more bandwidth available)
  - *Where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)
- DASH allows clients with different Internet access rates to stream in video at different encoding rates

Content Distribution Networks (CDN)

- ***Challenge:*** How to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- ***Option 1:*** single, large "mega-server"
- Drawbacks
  - Single point of failure
  - Point of network congestion
  - Long path to distant clients
  - Multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

Content Distribution Networks (CDN)

- *Option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*

  - *For Example (data 2022):*
    - Akamai (over 1400 locations)
    - Limelight (more than 140 points of presence)

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 server 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 sites. Instead of getting inside the access ISPs, these CDNs typically place their clusters in Internet Exchange Points (IXPs) Compared with the enter-deep design philosophy, the bring-home design typically results in lower maintenance and management overhead.

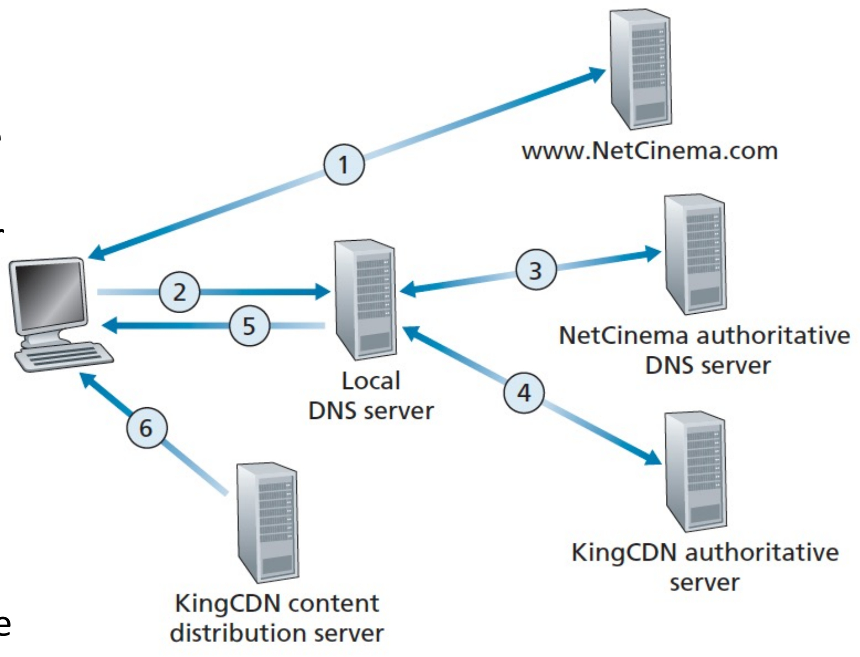
CDN Cluster Selection Strategy

- *Challenge:* how does CDN DNS select "good" CDN node to stream to client
  - Pick CDN node geographically closest to client
  - Pick CDN node with shortest delay (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)

- *Alternative:* Let *client* decide – give client a list of several CDN servers
  - Client pings servers, picks "best"
  - Netflix approach

How DNS redirects a user request to a CDN server

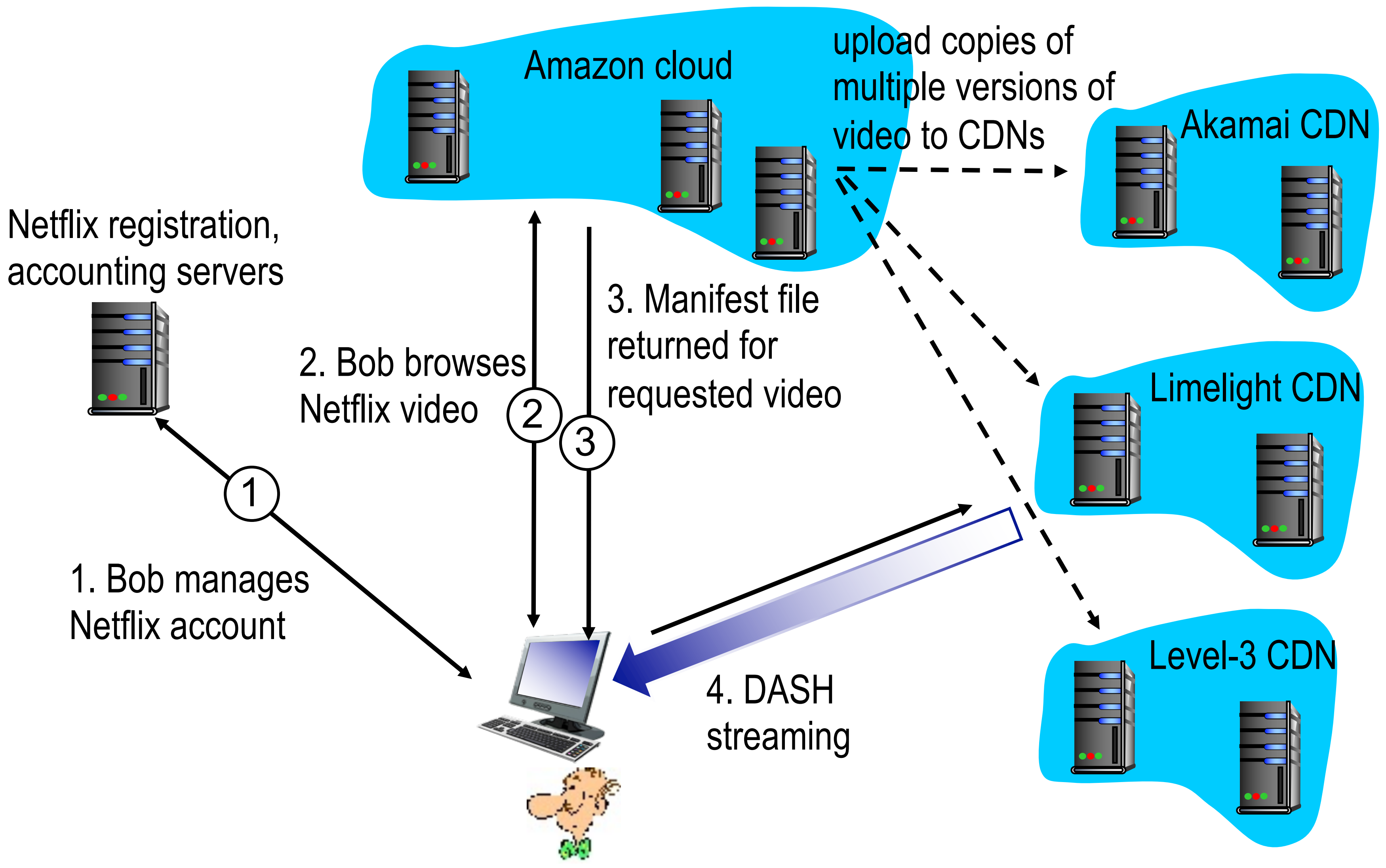
1. The user visits the Web page at NetCinema.
2. When the user clicks on the movie link the user's host sends a DNS query for video.netcinema.com.
3. The user's Local DNS Server (LDNS) relays the DNS query to an authoritative DNS server for NetCinema. To "hand over" the DNS query to KingCDN, instead 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.


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

Case study: Netflix

- Owns very little infrastructure, uses 3rd party services:
  - Own registration, payment servers
  - Amazon (3rd party) cloud services:
    - Netflix uploads studio master to Amazon cloud
    - Create multiple version of movie (different encodings) in cloud
    - Upload versions from cloud to CDNs
    - Cloud hosts Netflix web pages for user browsing
  - 3rd party CDNs host/stream Netflix content: Akamai, Limelight, Level-3

Case study: Netflix


Amazon cloud
upload copies of multiple versions of video to CDNs
Akamai CDN
Netflix registration, accounting servers
2. Bob browses Netflix video
2
3
3. Manifest file returned for requested video
Limelight CDN
1
1. Bob manages Netflix account
Level-3 CDN
4. DASH streaming

Topic of the lecture

- Multimedia networking applications
- Streaming *stored* video
- Voice-over-IP
- Protocols for *real-time* conversational applications
- Network support for multimedia

Voice-over-IP (VoIP)

- Involves taking analog audio signals and converting them into digital data which can be transmitted over the Internet
- Example: The sender generates bytes at a rate of 8,000 bytes per second; every 20 msecs the sender gathers these bytes into a chunk. A chunk and a special header are encapsulated in a UDP segment.
- UDP segment is sent every 20 msecs.
- When each packet manages to get to the receiver with a continuous end-to-end delay, then packets arrive at the receiver occasionally every 20 msecs.
- Receiver plays back each chunk as it is received
- **Problem:** some packets get lost and don't have the same end-to-end delay

Limitations VoIP

- *Packet loss:* IP datagram lost due to network congestion (router buffer overflow)
- *End-to-End delay:* IP datagram arrives too late for playout at receiver
  - Delays: processing, queueing in network; end-system (sender, receiver) delays
  - Typical maximum tolerable delay: 400 ms
- *Loss tolerance:* depending on voice encoding, loss concealment, packet loss rates between 1% and 10% can be tolerated

VoIP: Fixed Playout Delay

- Receiver attempts to playout each chunk exactly *q* msecs after chunk was generated.
- Chunk is timestamped at the sender at time *t*
- Receiver plays out the chunk at time $t + q$, if chunk has arrived by stipulated time.
- Packets that arrive late are lost.

VoIP: Adoptive Playout Delay

- With large initial playout delays, most packets will meet their deadlines. Hence there will be slightly loss of information.

- **Problem:** in conversational services such as VoIP, long delays may become annoying and intolerable so playout delay should be minimized so as to decrease the loss percentage.

- So as to address the above problem, there should be an estimate of the network delay and the variance of the network delay.

VoIP: Recovery from Packet Loss

- Retransmitting lost packets may not be practical in a real-time conversational application such as VoIP

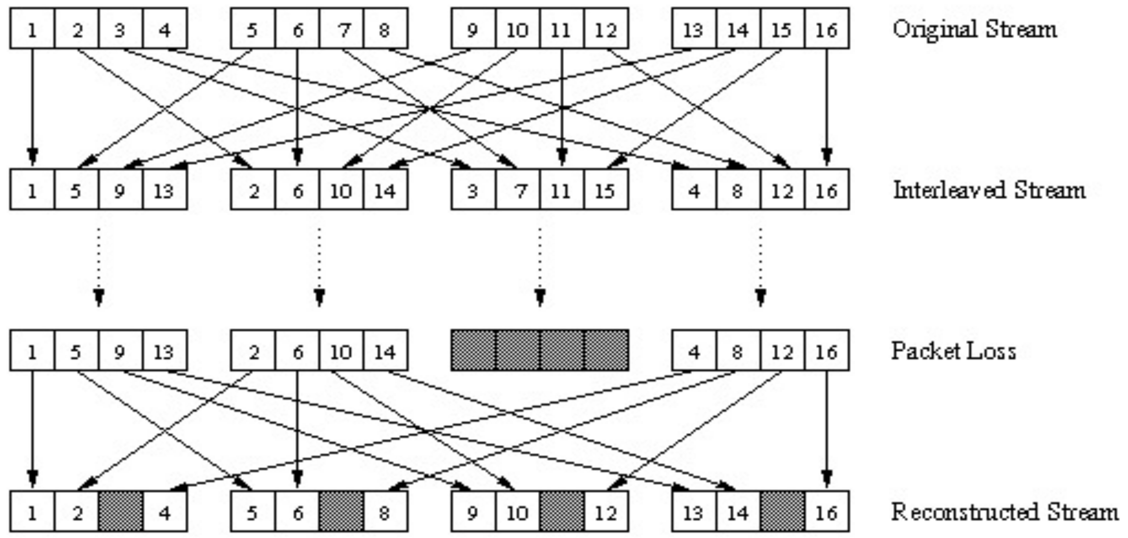
- It may not easily be accomplished on time for the conversation to remain understandable.

- *Forward Error Correction (FEC)*
  - Send enough bits to allow recovery without retransmission (recall two-dimensional parity in lecture 10)

Simple FEC

- For every group of *n* chunks, create redundant chunk by exclusive OR-ing *n* original chunks
- Send *n+1* chunks, increasing bandwidth by factor *1/n*
- Can reconstruct original *n* chunks if at most one lost chunk from *n+1* chunks, with playout delay

VoIP: Recovery from Packet Loss

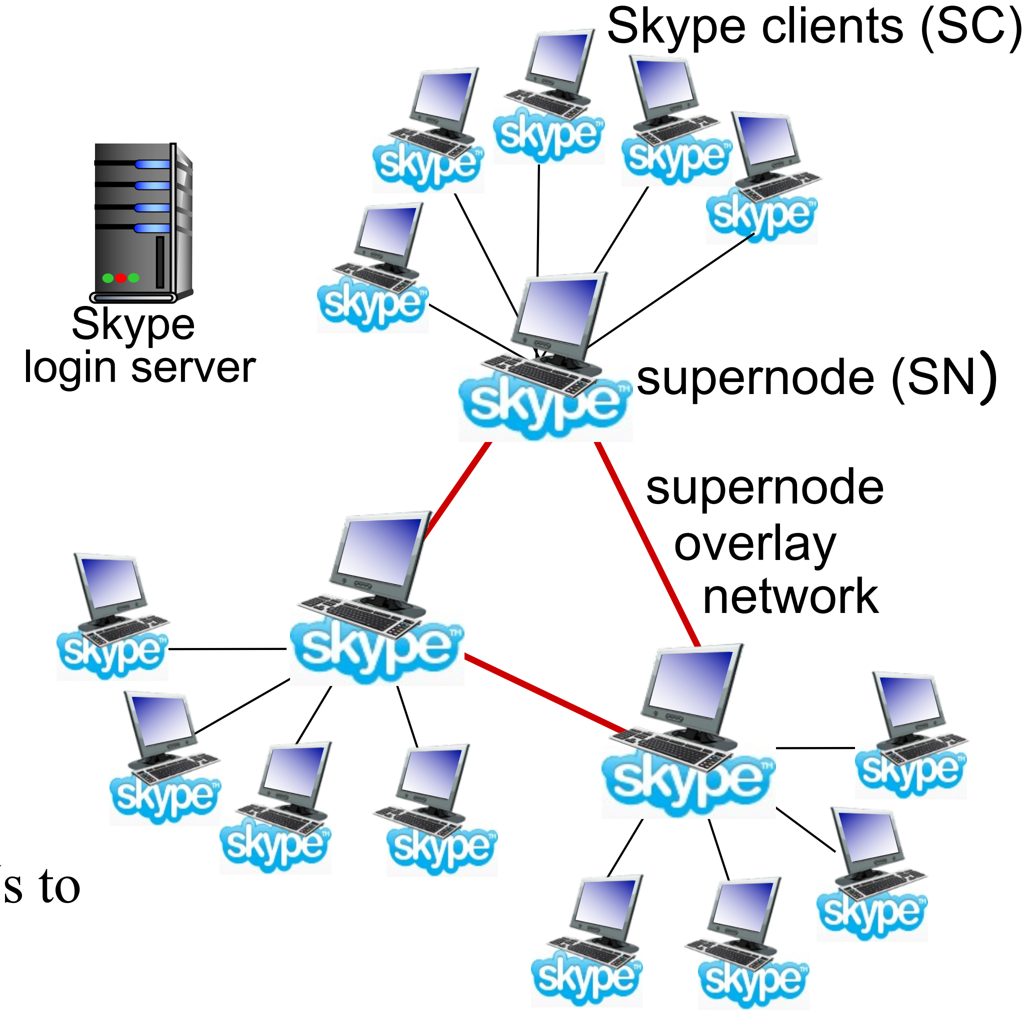


- *Interleaving to conceal loss:*
  - Audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk
  - Packet contains small units from different chunks
  - If packet lost, still have *most* of every original chunk

Voice-over-IP: Skype

- Proprietary application-layer protocol
  - Encrypted messages

- **P2P components:**
  - Clients: skype peers connect directly to each other for VoIP call
  - Super Nodes (SN): skype peers with special functions
  - Overlay network: among SNs to locate SCs
  - Login server



P2P voice-over-IP: Skype

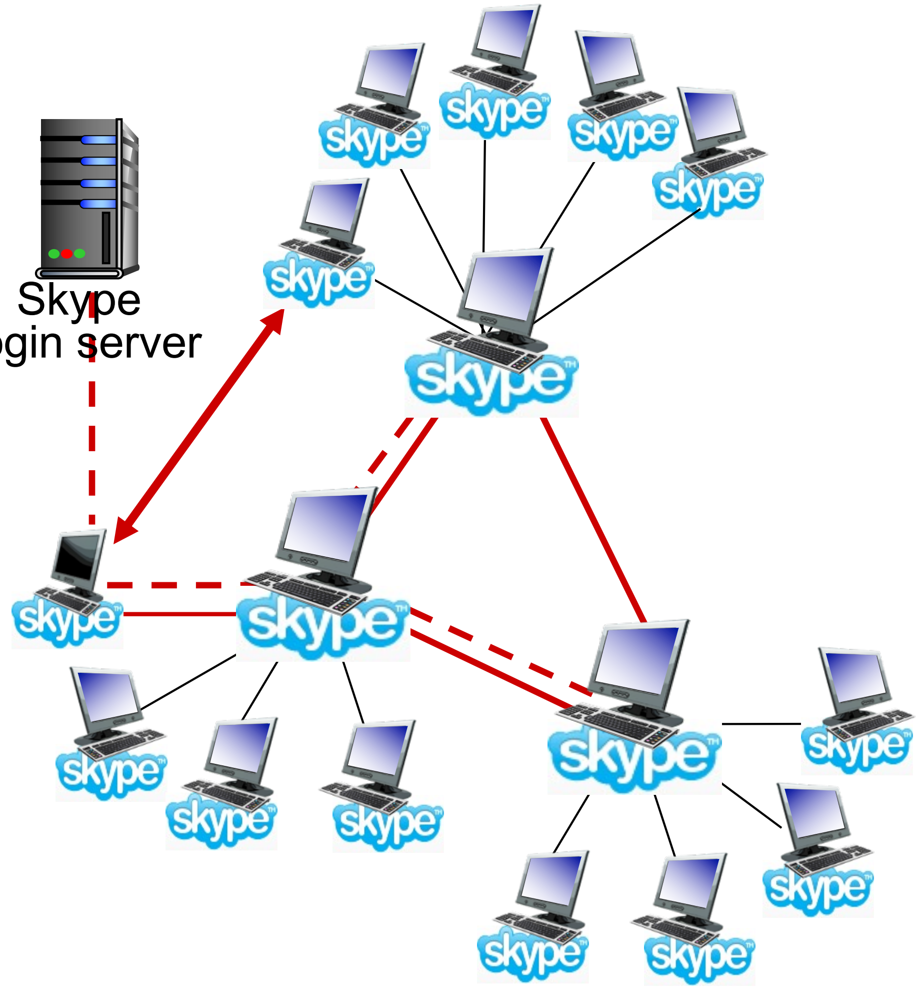
Skype Client Operation:

1. Joins skype network by contacting SN using TCP

2. Logs-in (usename, password) to centralized skype login server

3. Obtains IP address for callee from SN, SN overlay
    - or client buddy list

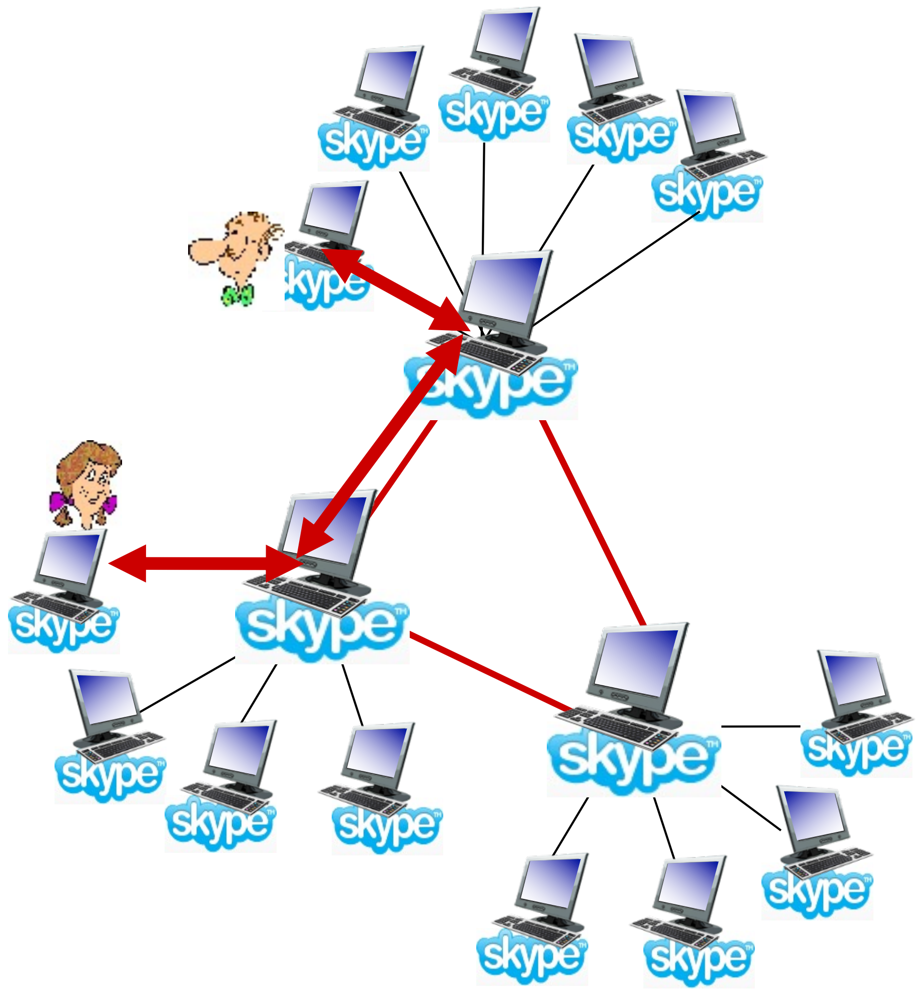
4. Initiate call directly to callee



Skype: Peers as Relays

- *Problem:* both Alice, Bob are behind "NATs"
  - NAT prevents outside peer from initiating connection to insider peer
  - Inside peer *can* initiate connection to outside

- *Relay solution: Alice, Bob maintain open connection to their SNs*
  - Alice signals her SN to connect to Bob
  - Alice's SN connects to Bob's SN
  - Bob's SN connects to Bob over open connection Bob initially initiated to his SN



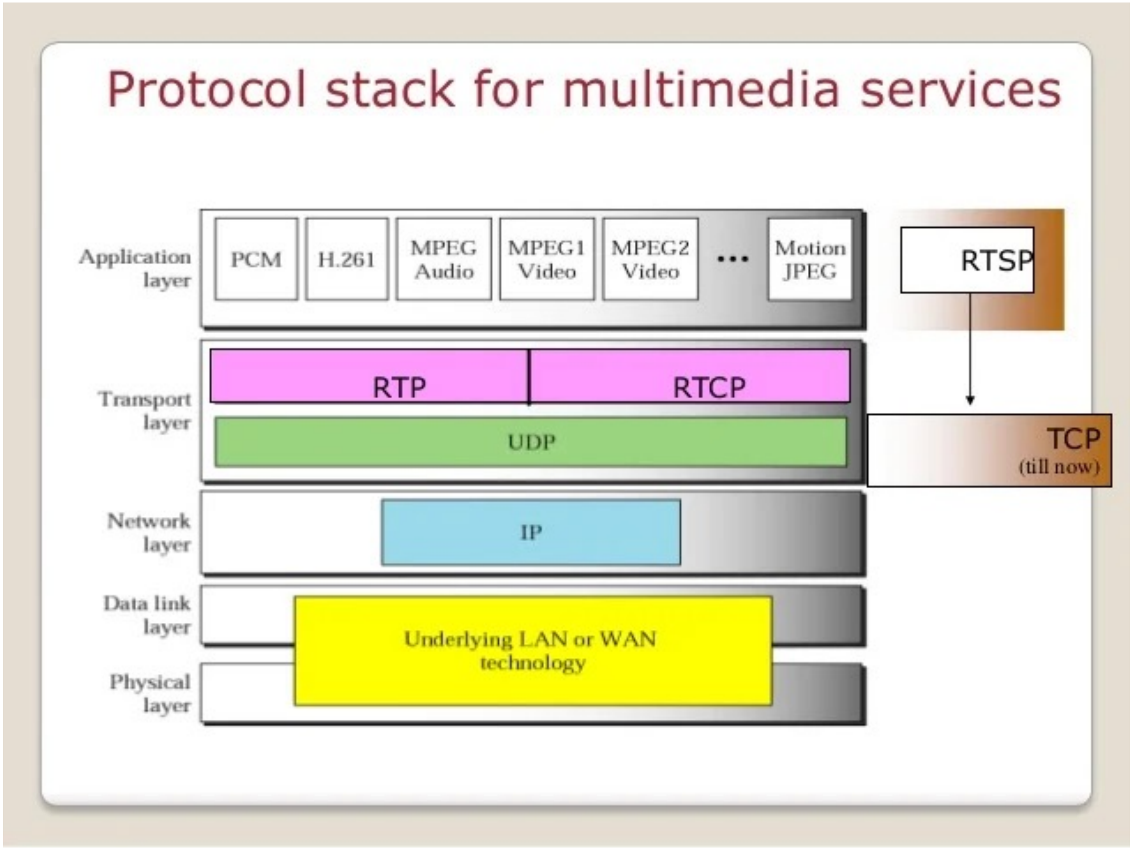
NAT: Network Address Translation

Topic of the lecture

- Multimedia networking applications
- Streaming stored video
- Voice-over-IP
- Multimedia networking protocols
- Network support for multimedia

Multimedia Networking Protocols

- There are some protocols that are used to support real time traffic over the internet.
- The following are the few protocols while many other exists
  - Real Time Protocol (RTP)
  - Real Time Control Protocol (RTCP)
  - Session Initiation protocol (SIP)
  - Real Time Streaming Protocol (RTSP)



Topic of the lecture

- Multimedia networking applications
- Streaming stored video
- Voice-over-IP
- Multimedia networking protocols
- Network support for multimedia

Dimensioning Best Effort Networks

- Approaches to improving the quality of networked multimedia

- Provide enough link capacity throughout the network
  - Such that network congestion, and its consequent packet delay and loss never occur
  - No queuing delay or loss

- Drawbacks
  - Bandwidth Provisioning
    - How much capacity to provide at network links?
  - Network Dimensioning
    - How to design a network topology?

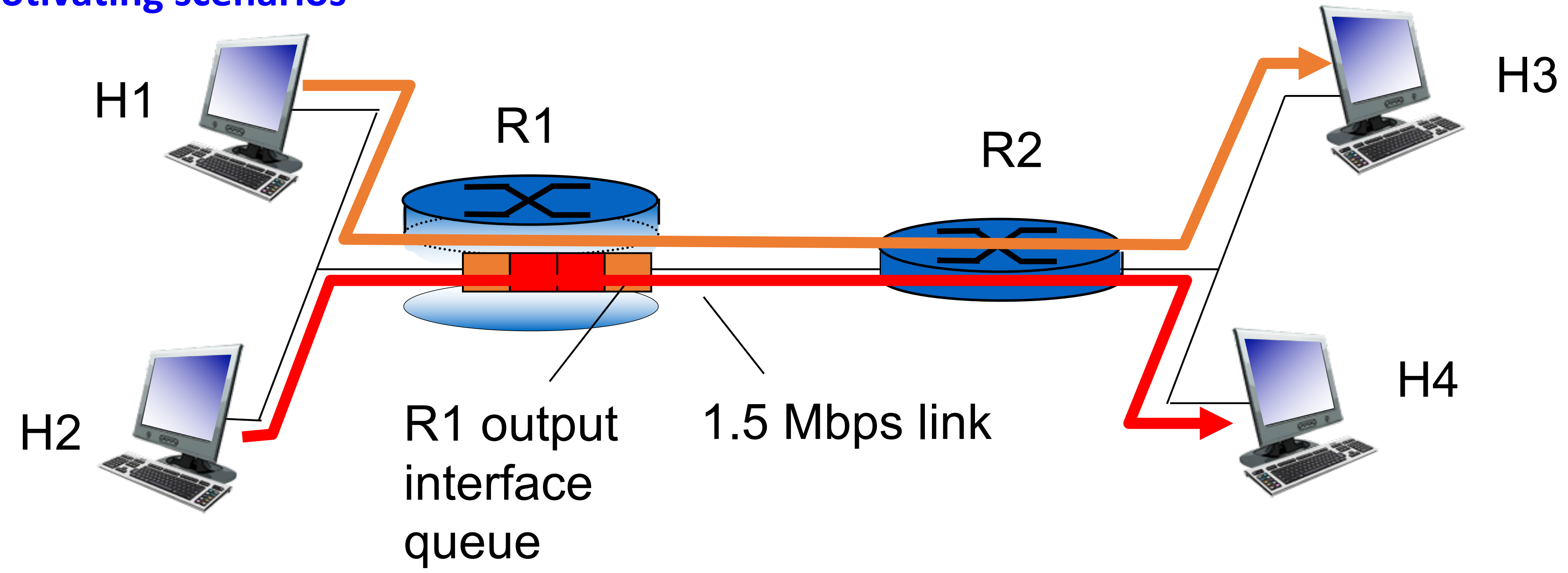
Providing Enough Capacity

- Issues to be addressed in order to predict application-level performance between two network end points
  - Models of traffic demand between network end points
  - Well-defined performance requirements
  - 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

Providing Multiple Classes of Service

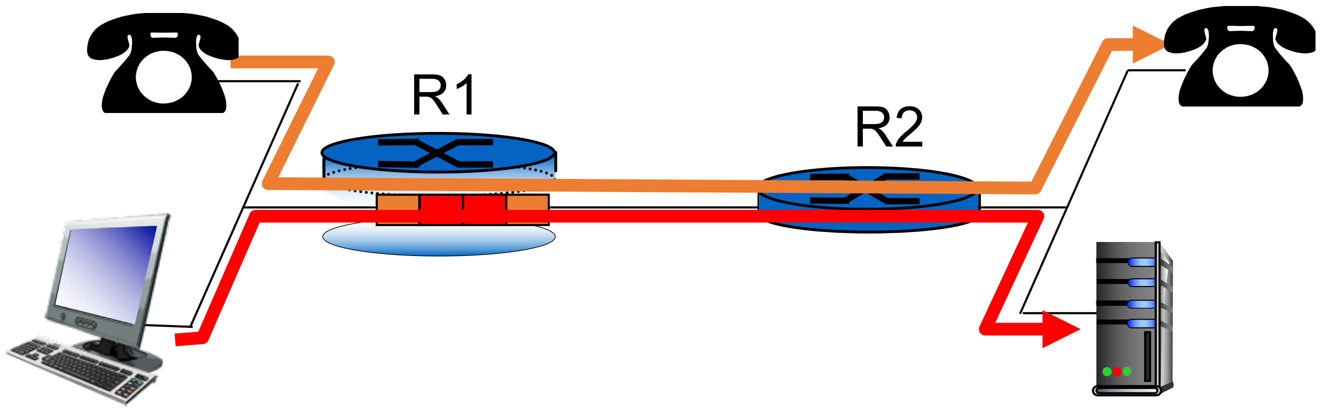
- Simplest development to the *one-size-fits-all best-effort* service model is;
  - To divide traffic into classes,
  - Provide different levels of service to these different classes of traffic.

Motivating scenarios



Scenario: Mixed HTTP and VoIP

- Example: 1Mbps VoIP, HTTP share 1.5 Mbps link.
  - HTTP bursts can congest router, cause audio loss
  - Want to give priority to audio over HTTP

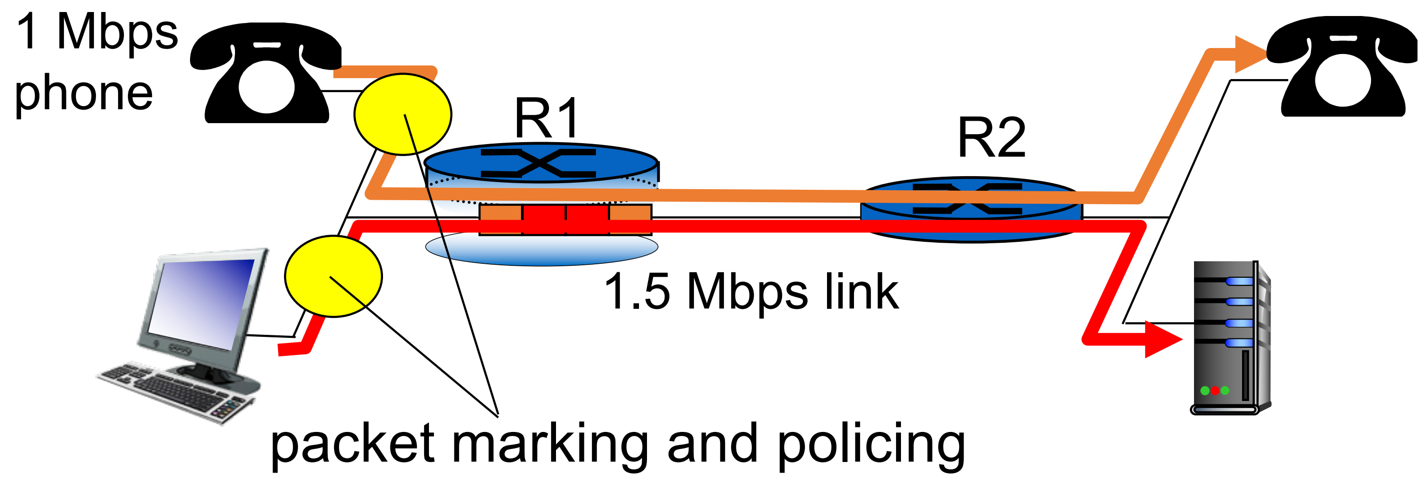


Principle 1

Packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

Scenario: Mixed HTTP and VoIP

- What if applications misbehave (VoIP sends higher than declared rate)
  - Policing: force source adherence to bandwidth allocations
- *Marking*, *policing* at network edge

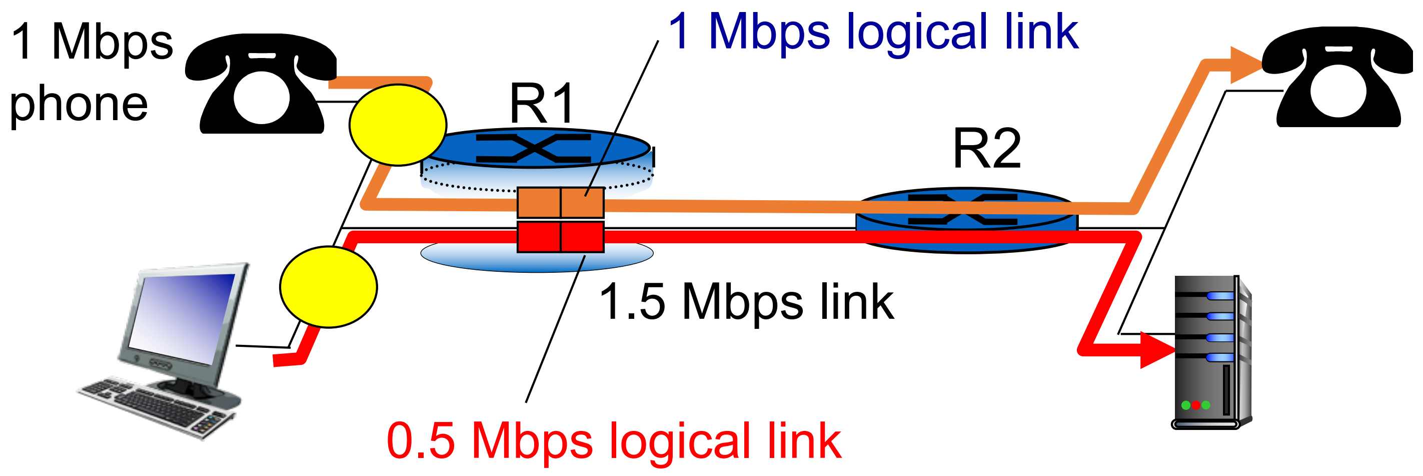


Principle 2

Provide protection (isolation) for one class from others

Scenario: Mixed HTTP and VoIP

- Allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation

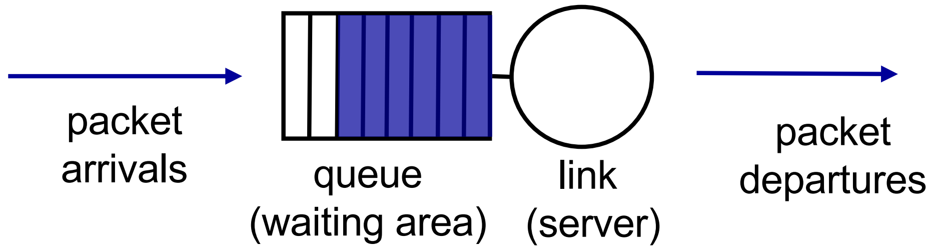


Principle 3

While providing isolation, it is desirable to use resources as efficiently as possible

Scheduling Mechanisms

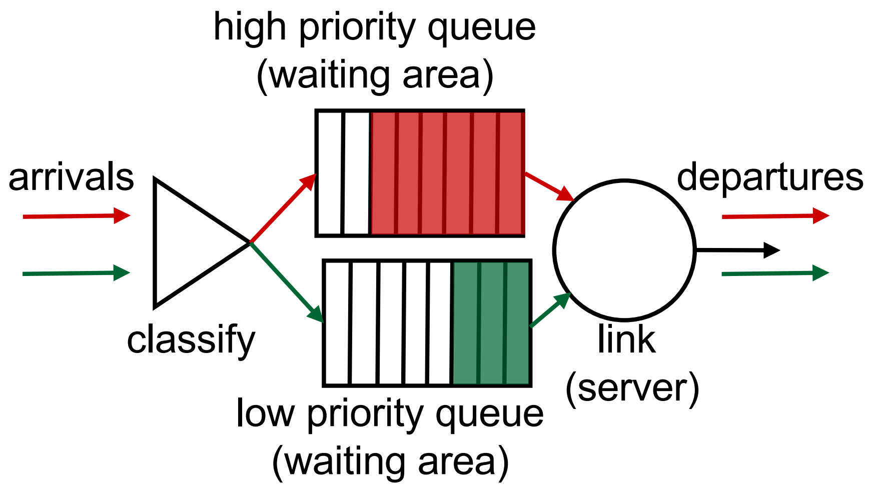
- *Scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - *Packet-discarding policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority basis
    - *random:* drop/remove randomly

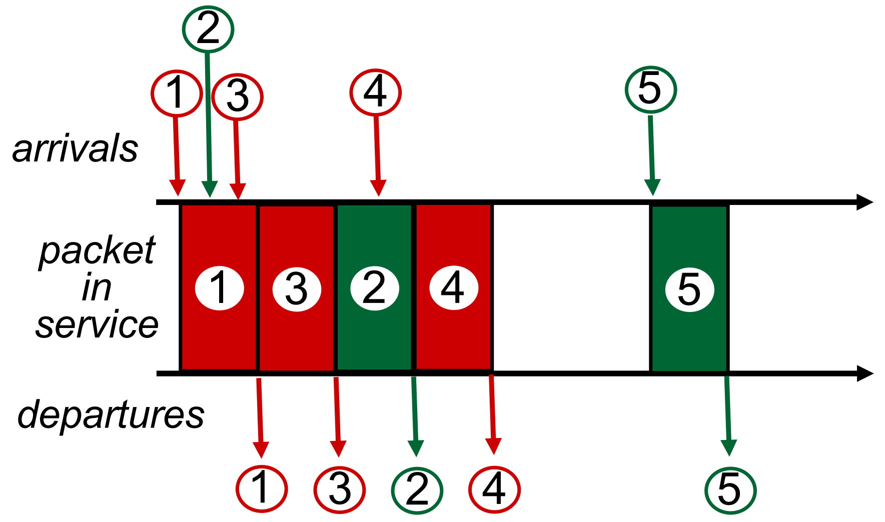


Scheduling Policies: Priority

Priority scheduling: send highest priority queued packet

- Multiple *classes*, with different priorities
  - Class may depend on marking or other header info, e.g. IP source/destinaction, port numbers, etc.

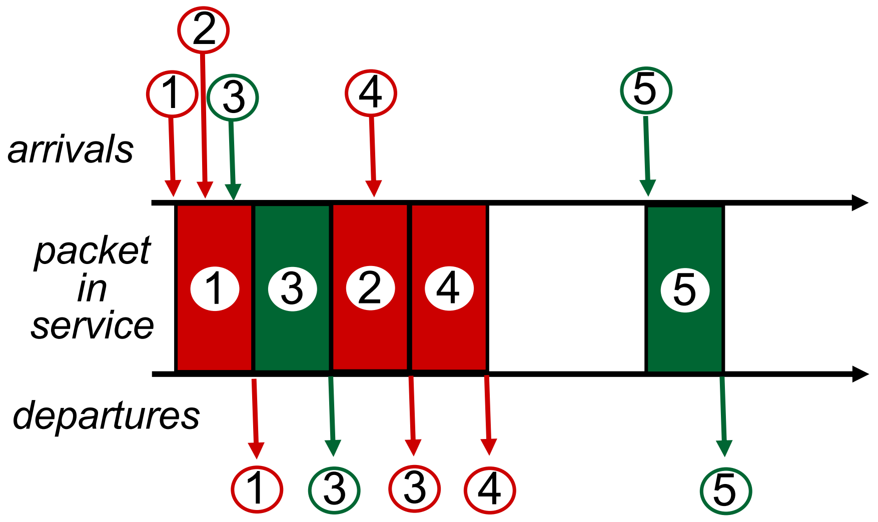

high priority queue (waiting area)
arrivals
departures
classify
link (server)
low priority queue (waiting area)



Scheduling policies: Round Robin

Round Robin (RR) scheduling:

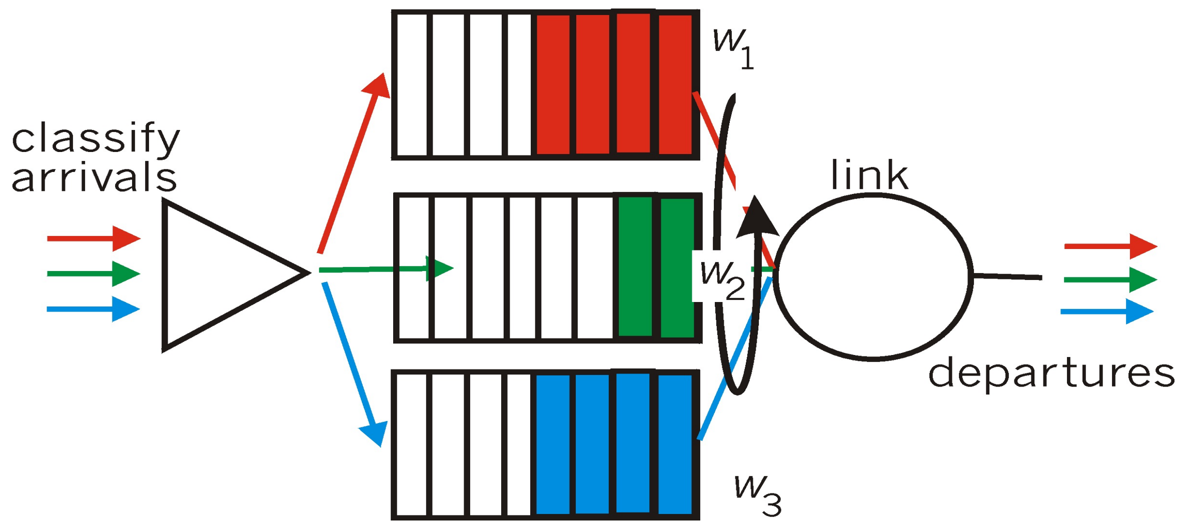
- Multiple classes
- Cyclically scan class queues, sending one complete packet from each class (if available)



Scheduling Policies: Still more

Weighted Fair Queuing (WFQ):

- Generalized Round Robin
- Each class gets weighted amount of service in each cycle



Policing Mechanisms

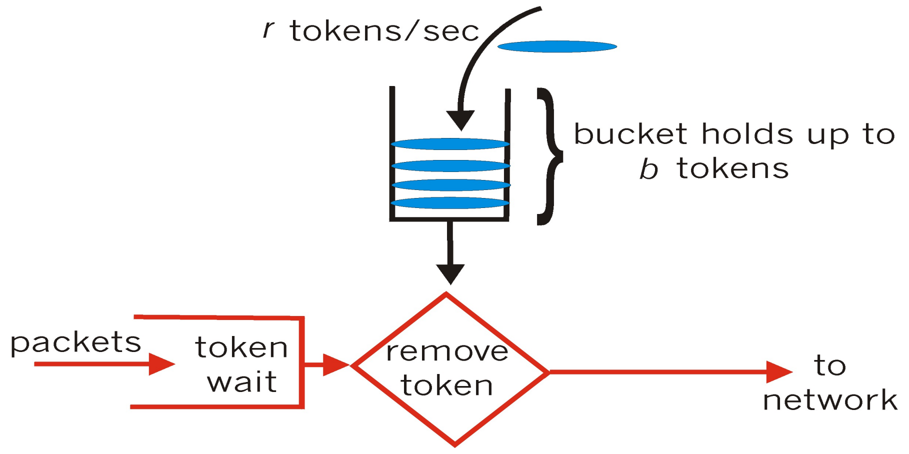
Goal: Limit traffic to not exceed declared parameters

Three common-used criteria:

- ***Average rate:*** How many packets can be sent per unit time (in the long run)
  - Crucial question: The interval of time over which the average rate will be policed.
- ***Peak rate:*** Limit the number of packets that can be sent over a shorter period of time
- ***Burst size:*** Limits the max number of Packets (the “burst” of packets) that can be sent into the network over an extremely short interval of time.

Policing Mechanisms: The Leaky Bucket

Token bucket: Limit input to specified *burst size* and *average rate*



- Bucket can hold b tokens
- Tokens generated at rate *r token/sec* unless bucket full

The Leaky Bucket - Implementation

- 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.
- The maximum burst size for a leaky-bucket policed flow is b packets.
- 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 $rt + b$.

Differentiated Services (Diffserv)

- Want "qualitative" service classes
  - "Behaves like a wire"
  - Relative service distinction: Platinum, Gold, Silver

- *Scalability:* Simple functions in network core, relatively complex functions at edge routers (or hosts)
  - signaling, maintaining per-flow router state difficult with large number of flows

- Don't define service classes, provide functional components to build service classes

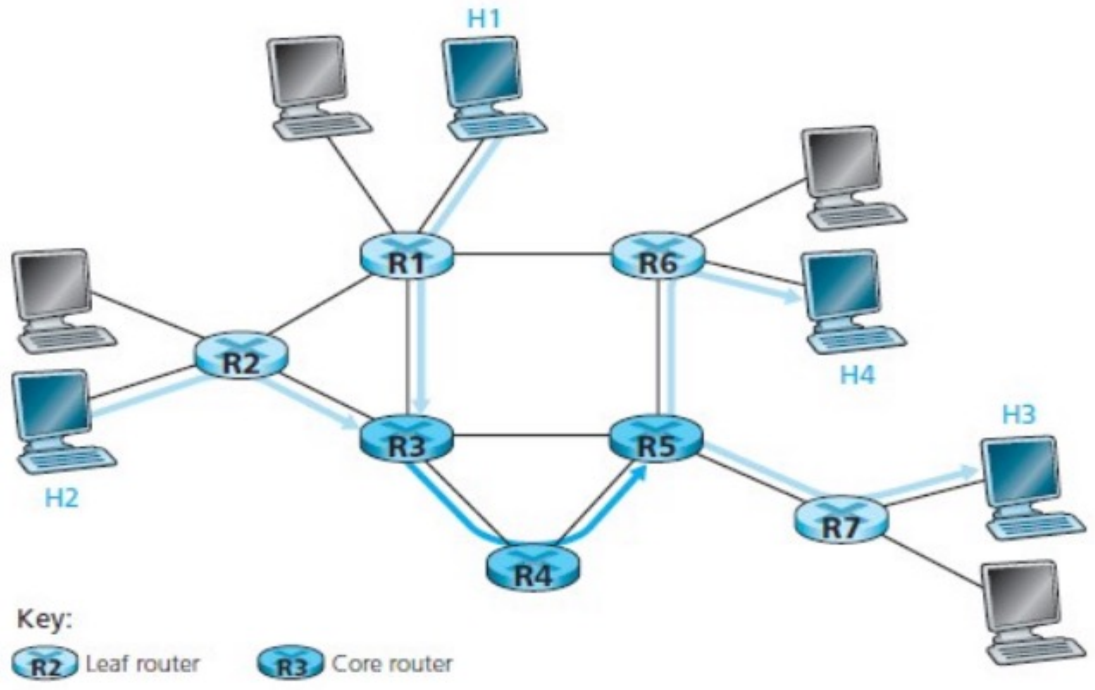
Diffserv Building Philosophies

- **Enter Deep:** This philosophy is to enter deep into the access networks of Internet Service Providers, by deploying server clusters in access ISPs all over the world. (Akamai)

- **Bring Home:** A second design philosophy is to bring the ISP’s home by building large clusters at a smaller number of key locations and connecting these clusters using a private high-speed network. (es. Limelight Networks)

Diffserv Architecture

- The packet is forwarded onto its next hop according to the per-hop behavior (PHB) associated with that packet's class.
- Buffering and scheduling based on marking at edge.
- Preference given to in-profile packets over out-of-profile packets



Summary

- Multimedia networking applications
- Streaming stored video
- Voice-over-IP
- Multimedia networking protocols
- Network support for multimedia