7 - Multimedia Networking

Index

Multimedia Applications and QoS

Multimedia Applications

- These involve transmitting audio and video over the network, which have specific requirements such as delay, jitter (variability of packet delays within the same packet stream), and bandwidth.

Quality of Service (QoS)

- Network provides application with level of performance needed for application to function.
- Parameters include **delay** (latency), **jitter**, **rate**, and **reliability**, which are affected by various factors.

7.1 - Multimedia Networking Applications

Classes of MM applications

1. Stored streaming
2. Live streaming (Internet radio talk show, live sports events)
3. Interactive, Real-time (IP telephony, video conference, distributed interactive worlds)

Fundamental characteristics

- typically **delay sensitive**
  - End-to-end delay
  - Delay jitter
- **loss tolerant:**
  - Infrequent losses cause minor glitches
- antithesis of data
  - Loss intolerant but delay tolerant

Streaming Stored Multimedia

- In stored streaming, media is stored at source and transmitted to client
- **Streaming:** client begins playback before all data has arrived

- Implies timing constraints for data still to be transmitted: in time for play out.
- Interactivity
    - pause, rewind, and fast forward

Streaming **Live** Multimedia

- Streaming (as with streaming stored multimedia)
    - Playback buffer
    - Playback can lag tens of seconds after transmission...
    - Still have timing constraint
- Interactivity
    - Fast forward obviously impossible
    - Rewind and pause possible!

Real-Time Interactive Multimedia

- End-to-end delay requirements:
    - audio: < 150 ms good, < 400 ms acceptable
- Session initiation...

Multimedia Over Today's Internet

- Current internet offers a "best-effort service" (TCP/UDP/IP) without guarantees on delay or loss.
- Multimedia applications employ application-level techniques to mitigate these effects.

How should the Internet evolve to better support multimedia?

- **Integrated services philosophy**
    - Fundamental changes in Internet so that apps can reserve end-to-end bandwidth
    - Requires new, complex software in hosts & routers
- **Laissez-faire**
    - No major changes
    - More bandwidth when needed
    - Content distribution, application layer multicast
    - Application layer techniques
- **Differentiated services philosophy**
    - Fewer changes to Internet infrastructure, yet provide 1st and 2nd class service

Audio compression overview

- Analogue signal sampled at constant rate
- Each sample quantized
- Each quantized value represented by bits
- Receiver converts bits back to analogue signal, possibly with some quality loss

Video compression overview

- Video: sequence of images displayed at constant rate
- Digital image: array of pixels (each pixel represented by bits)

- Redundancy
    - Spatial (within image)
    - Temporal (from one image to the next)

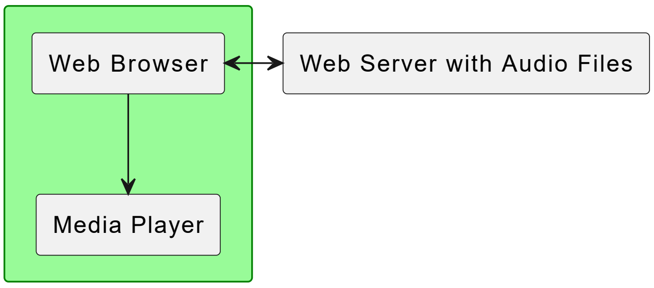
7.2 - Streaming Stored Audio and Video

Streaming Stored Multimedia

- Application-level streaming techniques for making the best out of best effort service:
    - Client-side buffering
    - Use of UDP versus TCP
    - Multiple encodings of multimedia

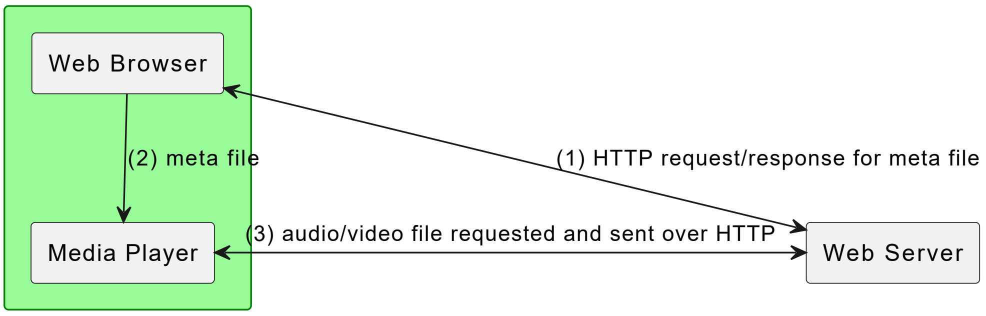
- Media Player
    - Jitter removal
    - Decompression
    - Error concealment
    - Graphical user interface with controls for interactivity

Internet multimedia: simplest approach



- Audio or video stored in file
- Files transferred as HTTP object
    - received in entirety at client
    - then passed to player
- Audio, video not streamed:
    - no "pipelining" - long delays until play out!

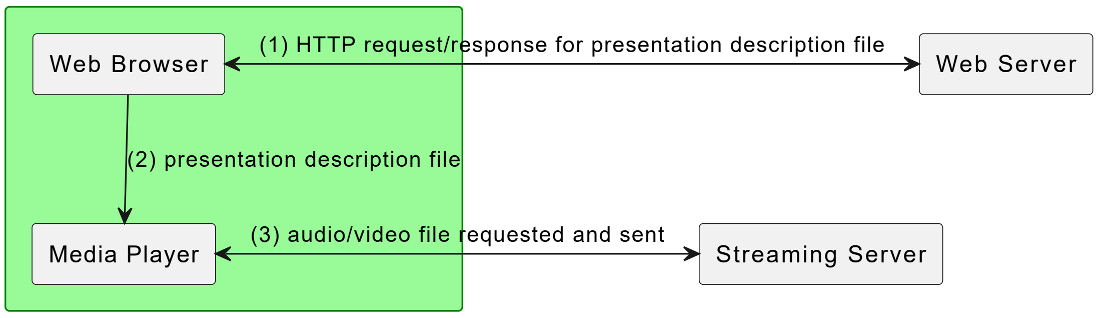
Internet multimedia: streaming approach


Web Browser
(2) meta file
(1) HTTP request/response for meta file
Media Player
(3) audio/video file requested and sent over HTTP
Web Server

- Browser GETs metafile referencing media file(s)

    - Media can be split across multiple files
- Browser launches player, passing metafile
- Player contacts server
- Server streams audio/video to player

Streaming from a streaming server


Web Browser
(1) HTTP request/response for presentation description file
Web Server
(2) presentation description file
Media Player
(3) audio/video file requested and sent
Streaming Server

- Allows for non-HTTP protocol between server, media player
- UDP or TCP for step (3)

Streaming Multimedia: Client Buffering

- Client-side buffering and play out delay compensate for network-added jitter
- Buffer size limits the amount of jitter we can absorb

Streaming Multimedia: UDP or TCP?

UDP	TCP
• Server sends at rate appropriate for client (oblivious to network congestion!) ◦ often, send rate matches encoding rate => constant rate ◦ then, fill rate = constant rate - packet loss • Short play out delay (2-5 seconds) to remove network jitter • Error recovery: time permitting	• Send at maximum possible rate under TCP ◦ subject to flow control • Fill rate fluctuates due to TCP congestion control • Larger play out delay: smooth TCP delivery rate • HTTP/TCP passes more easily through firewalls

Streaming Multimedia: client rate(s)

- Server stores and transmits multiple copies of media encoded at different rates, which allows adaptation to different client receive rate capabilities.

User Control of Streaming Media: RTSP

HTTP

- Does not target multimedia content (though there are mechanisms such as HLS and DASH)

- No commands for fast forward, etc.

Real Time Streaming Protocol (RTSP)

- Client-server application layer protocol
- User control: rewind, fast forward, pause, resume, repositioning, etc...

What RTSP doesn't do:

- Doesn't define how audio/video is encapsulated for streaming over network
- Doesn't restrict how streamed media is transported (UDP or TCP possible)
- Doesn't specify how media player buffers audio/video

RTSP messages sent out-of-band

- "out-of-band" and "in-band" channels use different port numbers
- RTSP control messages use different port numbers than media stream: out-of-band (similar to FTP)
    - port 554
- Media stream is considered "in-band"

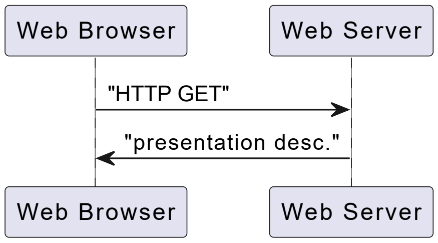
RTSP Example

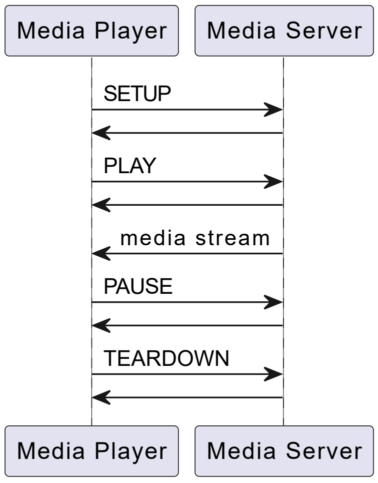
- Scenario:
    - Metafile communicated to web browser
    - Browser launches player
    - Player sets up an RTSP control connection, data connection to streaming server

Metafile Example

```
<title>Twister</title>
<session>
    <group language=en lipsync>
        <switch>
            <track type=audio
                e="PCMU/8000/1"
                src = "rtsp://audio.example.com/twister/audio.en/lofi">
            <track type=audio
                e="DVI4/16000/2" pt="90 DVI4/8000/1"
                src="rtsp://audio.example.com/twister/audio.en/hifi">
        </switch>
    <track type="video/jpeg"
    src="rtsp://video.example.com/twister/video">
    </group>
</session>
```

RTSP Operation





7.3 - Making The Best Out Of Best Effort Service

Interactive Multimedia: Internet Phone

Example

- Speaker's audio: alternating talk spurts and silent periods
  - packets generated only during talk spurts
- Application layer header is added to each chunk, encapsulated into UDP segment, and sent every 20 msec during talk spurt

Internet Phone: Packet Loss and Delay

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

Delay jitter

- End-to-end delivery of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)
- Play out must be exactly every 20 msec

Internet Phone: Fixed play out Delay

- Receiver attempts to play out each chunk exactly **q** msecs after chunk was generated

    - Chunk has time stamp **t**: play out chunk at **t+q**
    - Chunk arrives after **t+q**: data arrives too late for play out - data "lost"
- Trade-off in choosing **q**:
    - **large q:** fewer packet losses (delay losses)
    - **small q:** better interactive experience

Adaptive play out Delay (1)

- **Goal:** minimize play out delay, keeping delay losses low
- **Approach:** adaptive play out delay adjustment:
    - Estimate network delay, adjust play out delay **at the beginning of each talk spurt**
    - Silent periods may be compressed or elongated
    - Chunks still played out every 20 msec during talk spurt

```
tᵢ = timestamp of the ith packet
rᵢ = the time the ith packet is received by the player
pᵢ = the time the ith packet is played by the player
rᵢ - tᵢ = network delay for the ith packet
dᵢ = estimate of average network delay after receiving ith packet
```

Dynamic estimate of average delay at receiver: $d_i = (1-u)d_{i-1} + u(r_i-t_i)$ where u is a fixed constant 0<u<1 ➔ EWMA

Adaptive play out delay (2)

- Also useful to estimate average deviation of delay, v_i: $v_i = (1-u)v_{i-1} + u|r_i - t_i - d_i|$
- Estimates d_i, v_i calculated for every received packet (but used only at start of talk spurt)
- For first packet in talk spurt, play out time is: $p_i = t_i + d_i + Kv_i$ where K is a positive constant
- Remaining packets in talk spurt are played out periodically (every 20 ms in this example)

Adaptive play out (3)

- How does receiver determine whether packet is first in a talk spurt?
    - If no loss, receiver looks at timestamps of successive packets
        - Difference of successive stamps > 20 msec → talk spurt begins
    - With loss possible, receiver must look at both timestamps and sequence numbers
        - Difference of successive stamps > 20 msec and sequence numbers without gaps → talk spurt begins
    - Some apps mark explicitly the 1st pkt in a talk spurt

Recovery from packet loss

- **Challenge:** how to recover from packet loss given small tolerable delay between original transmission and playout
- ARQ (retransmission) may be infeasible
    - each ACK/NAK takes one RTT
- Alternative: Forward Error Correction (FEC)

        - send redundant bits to allow recovery without retransmission

Recovery from packet loss (1)

Forward Error Correction (FEC): simple scheme

- For every group of n chunks create redundant chunk by XORing n original chunks
- Send out n+1 chunks, increasing bandwidth by factor 1/n
- Can reconstruct original n chunks if at most one lost chunk from n+1 chunks
- **Play out delay:** enough time to receive **all n+1 packets**
- Tradeoffs
    - Larger n means
        - Less bandwidth waste
        - Longer play out delay
        - Higher probability that 2 or more chunks will be lost

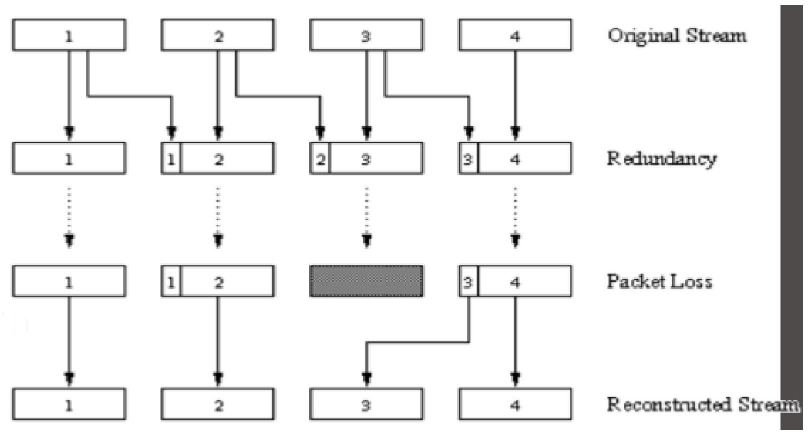
Example

- Transmitted packets
    - Original packet 1: 10100011011011011111011011100101
    - Original packet 2: 01110101000101010100101100110110
    - Extra packet (XOR):11010110011110001011110111010011

- Received packets (#2 was lost)
    - Original packet 1: 10100011011011011111011011100101
    - Extra packet: 11010110011110001011110111010011

- Packet 2 can be recovered
    - Original packet 1: 10100011011011011111011011100101
    - Extra packet: 11010110011110001011110111010011
    - XOR = packet 2: 01110101000101010100101100110110

Recovery from packet loss (2)

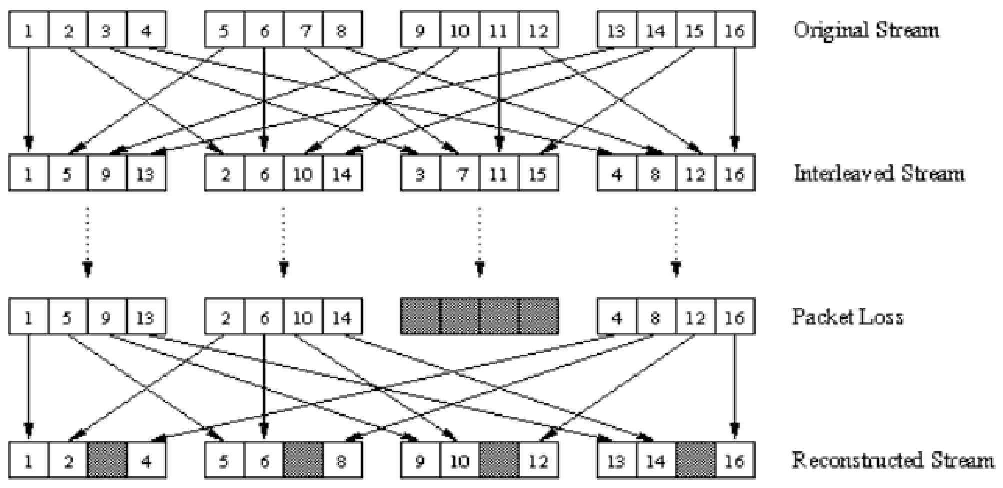


2nd FEC scheme: "piggyback lower quality stream"

- Send lower resolution audio stream as redundant information

- Receiver can conceal non-consecutive losses
- Playout only needs to be delayed by one additional packet time
- Can also append (n-1)th and (n-2)th low-bit rate chunk

Recovery from packet loss (3)



Interleaving

- Chunks divided into smaller units
- Packet contains small units from different chunks
- If packet lost, still have most of every chunk
- No redundancy overhead, but increases play out delay

Content delivery networks (CDNs)

Content replication

- Challenging to stream large files (e.g., video) from single origin server in real time
- Solution: replicate content at hundreds of servers throughout the world
    - Content uploaded to CDN servers ahead of time
    - Placing content "close" to user avoids impairments (loss, delay) of sending content over long paths
    - CDN server typically in edge/access network

Alterative method

- No CDN distribution node
- When first user in region requests a file, local CDN node retrieves it from original server and caches it for future requests

CDN Example: **ver exemplo do slide 53**

Routing requests

- CDN creates a "map", indicating distances from leaf ISPs and CDN nodes
- When query arrives at authoritative DNS server
    - DNS server determines ISP from which query originates (from source IP address)

    - Uses "map" to determine best/closest CDN server
    - Note: using global DNS servers (Google, OpenDNS, ...) may lead to performance issues
        - Use of EDNS Client Subnet (ECS) extension solves this issue, if supported
- CDN nodes create an application-layer overlay network

Summary: Internet Multimedia - bag of tricks

- Use UDP to avoid TCP congestion control (delays) for time-sensitive traffic
- Client-side adaptive play out delay to compensate for jitter
- Server side matches stream bandwidth to available client-to-server path bandwidth
    - Choose among pre-encoded stream rates
    - Dynamic server encoding rate
- Error recovery (on top of UDP)
    - FEC, interleaving, error concealment
    - Retransmissions, time permitting
- CDN: bring content closer to clients

7.4 - Protocols for Real-time Interactive Applications RTP, RTCP, SIP

Real-time Transport Protocol (RTP)

- Specifies packet structure for audio and video data
- Runs in end systems and encapsulates RTP packets in UDP segments
- **Interoperability** between Internet phone applications
- RTP does **not** ensure QoS guarantees, and its encapsulation is only seen at end systems, **not** by intermediate routers
- Runs on top of UDP: RTP libraries provide transport-layer interface that extends UDP

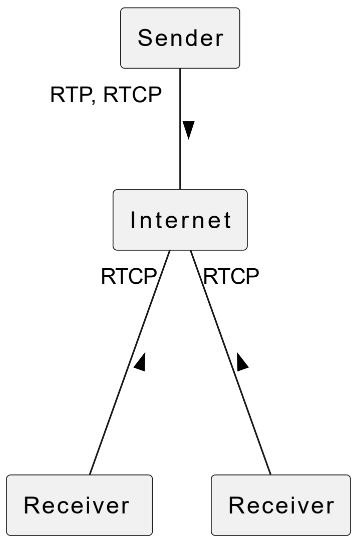
RTP Header

Payload type (7 bits)	**Sequence Number (16 bits)**	**Timestamp (32 bits long)**	**Synchronization Source Identifier (32 bits long)**	**Miscellaneous fields**
Indicates type of encoding currently being used. If sender changes encoding in middle of conference, receiver is informed via payload type field	Increments by one for each RTP packet sent. May be used to detect packet loss and to restore packet sequence	Sampling instant of first byte in this RTP data packet	Identifies source of RTP stream. Each stream in RTP session should have distinct SSRC. (Synchronization Source Identifier)	Marker bit: marks special packets, with specific meaning dependent on the payload type

Real-time Transport Control Protocol (RTCP)

- Works with RTP

- Each participant in RTP session periodically transmits RTCP control packets to all other participants
- Each RTCP packets may contain sender or receiver reports, providing statistics useful for monitoring and controlling performance
- Sessions typically use a single multicast address
- RTP and RTCP packets distinguished by their distinct port numbers
- To limit traffic, each participant reduces RTCP traffic as number of conference participants increases (bandwidth scaling)
- RTCP bandwidth scaling limits its traffic to 5% of session bandwidth, with participants adjusting transmission periods based on allocated rates



RTCP Messages

- Receiver report:
  - Fraction of packets lost, last sequence number, average inter-arrival jitter
- Sender report:
  - SSRC of RTP stream, current time, number of packets sent, number of bytes sent
- Source description:
  - E-mail address of sender, sender's name, SSRC of associated RTP stream
  - Provide mapping between the SSRC and the user/host name

Synchronization of Streams

- RTCP can synchronize different media streams within an RTP session based on timestamp and wall-clock time.

Example

- Two streams
  - Audio: 22100 samples/s
  - Video: 25 frames/s
- Last sender reports
  - For audio, matched timestamp 4044300 to 2022-02-22 16:28:03
  - For video, matched timestamp 4625 to 2022-02-22 16:28:05
- Which video frame should be played simultaneously with the audio sample with timestamp 4110600?

    - 4110600 - 4044300 = 66300 (samples)
    - 66300/22100 = 3 (seconds)
    - Current wall clock time (NTP) is 2022-02-22 16:28:03 + 3 = 2022-02-22 16:28:06
    - 2022-02-22 16:28:06 - 2022-02-22 16:28:05 = 1 (s)
    - 25 × 1 = 25 (frames)
    - 4625 + 25 = 4650
    - Answer: the frame with timestamp 4650

Jitter Calculation

- R_k - arrival time of packet k
- S_k - RTP timestamp of packet k
- 1/16 factor amortizes noise and leads to a good convergence (EWMA)
- Delay difference between packets: $D(i,j) = (R_j - S_j) - (R_i - S_i) = (R_j - R_i) - (S_j - S_i)$
- Estimated jitter: $J(i) = (1 - 1/16)\ J(i - 1) + 1/16\ |D(i - 1, i)|$

RTT Calculation

- If Rec1 sends a RR (Receiver Report) that Src1 receives at time **Trec**
- The report has:
    - **LSR** of Src1: time at which Src1 sent the last report received by Rec1
    - **DLSR**: time since Rec1 received the last report from Src1 and sent this RR
    - RTT = Trec - LSR - DLSR

Overview of...

RTSP	**RTP**	**RTCP**
Controls the playing of a multimedia stream (play, pause, etc.)	Transports multimedia packets (with associated metadata)	Sends reports regarding RTP media flows (from receivers and senders)

SIP: Session Initiation Protocol

- Internet-based communication for calls and video conferences
- Identification by names or email addresses instead of phone numbers
- Accessibility to the callee regardless of location or device

SIP Services

- Mechanisms for call setup and management, and determining callee's IP address
- Supports adding new media streams and changing encoding during calls, inviting others, and transferring/holding calls

Setting up a call to known IP address

- Alice's SIP invite message indicates her port number, IP address, encoding she prefers to receive
- Bob's 200 OK message indicates his port number, IP address, preferred encoding
- SIP messages can be sent over TCP or UDP

- Default SIP port number is 5060
- Media can be sent over RTP or some other protocol, SIP doesn't care

Example of SIP message

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

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

Notes:

- HTTP-like message syntax
- sdp = session description protocol
- Call-ID is unique for every call
- IP address for receiving data: 167.180.112.24
- Port for receiving: 38060
- Media types supported: 0
- Here we don't know Bob's IP address. Intermediate SIP servers are needed
- Alice sends and receives SIP messages using SIP default port 5060
- Alice specifies in Via (header that SIP client sends) that it receives SIP messages over UDP

SDP (Session Description Protocol)

- Used to negotiate media streams used, destination addresses, ports for each stream, payload types, and to negotiate for broadcast sessions.

Name translation and user location

- Resolving callee's IP address based on name or email are services provided by SIP registrar and proxy servers

SIP Registrar

- When Bob starts SIP client, it sends a SIP REGISTER message to Bob's registrar server
- Register Message:

  ```
  REGISTER sip:domain.com SIP/2.0
  Via: SIP/2.0/UDP 193.64.210.89
  From: sip:bob@domain.com
  To: sip:bob@domain.com
  Expires: 3600
  ```

SIP Proxy

- Alice sends INVITE message to her proxy server
  - Contains address sip:bob@domain.com
- Proxy responsible for routing SIP messages to callee, possibly through multiple proxies
- Callee sends response back through the same set of proxies (in reverse order)
  - Using Via: headers
- Proxy returns SIP response message to Alice containing Bob's IP address

SIP components and examples: **ver slides 83-89**

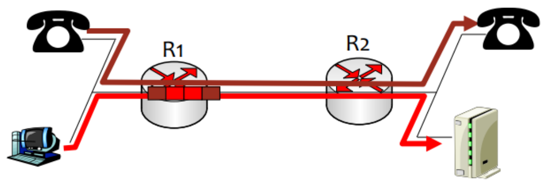
SIP and QoS

- SIP does **not** provide QoS
- However **INVITE SDP** fields may indicate QoS requirements
  - Resources may be acquired before RING
  - These are pre-conditions in SDP for the call
- Exemplo: **ver slide 91**

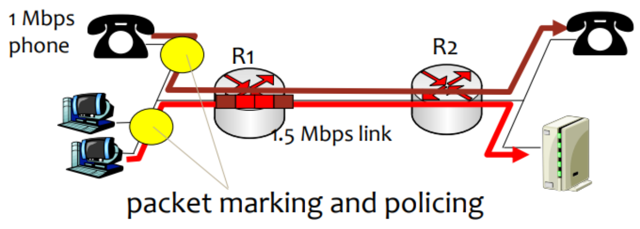
7.5 - Providing Multiple Classes of Service

Providing Multiple Classes of Service

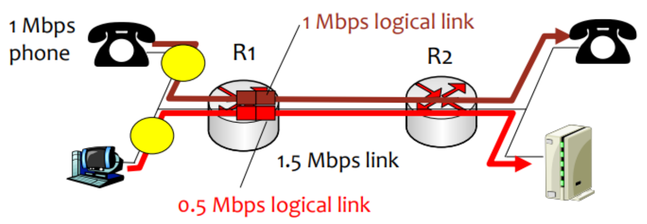
- Thus far: making the best of best effort service
- **Alternative:** implement multiple classes of service, treating different traffic classes differently
- **Granularity:** differential service among multiple classes, not among individual connections
- History: ToS bits



> **Principle 1:** Packet marking for router to distinguish between different classes, new router policy to treat packets accordingly



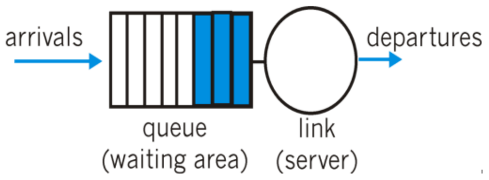
> **Principle 2:** Provide protection (isolation) for one class from others



> **Principle 3:** While providing isolation, it is desirable to use resources as efficiently as possible

Scheduling Policies

- **Scheduling:** choose next packet to send on link
- **FIFO (first in first out) scheduling:** send in order of arrival to queue
  - **Discard policy:** tail drop, head drop, priority, random

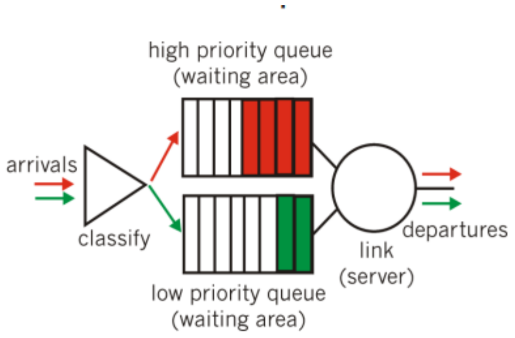


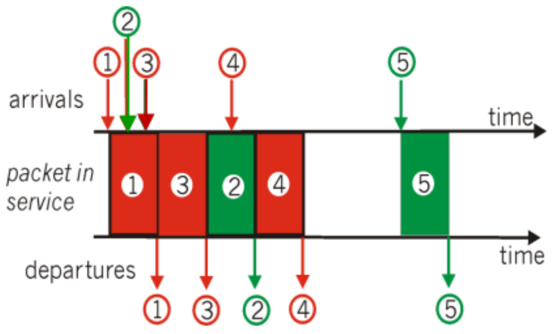
Scheduling and Policing Mechanisms: https://netlab.ulusofona.pt/rc/book/6-multimedia/6_06/index.htm

Prioritiy

Priority scheduling

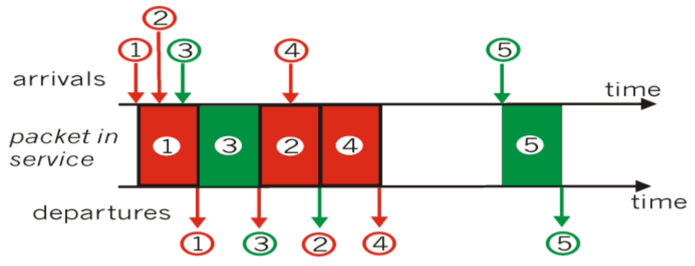
- Transmit highest priority queued packet
- Multiple classes with different priorities
  - Class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc...
  - Non-preemptive: transmission of a lower priority packet is not interrupted





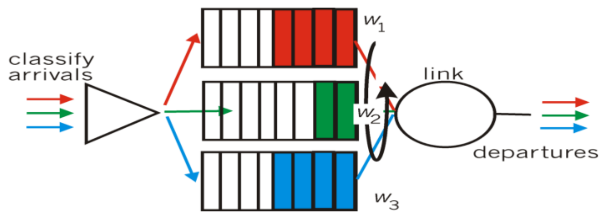
Round Robin

- Round Robin scheduling:
  - Multiple classes
  - Cyclically scan class queues, serving one packet from each class (if available)



WFQ

- Weighted Fair Queuing:
  - An improved variant of Weighted Round Robin
  - Each class gets weighted amount of service in each cycle

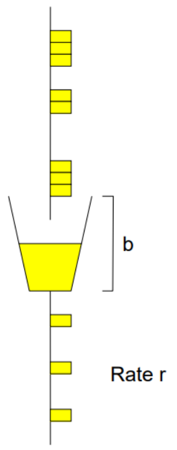


Traffic Shaping and Policing

- Goal: limit traffic to not exceed declared parameters (traffic profile) through **(long term) average rate**, **peak rate**, **(max.) burst size**.
- To enforce the profile, traffic **shaping** delays out-of-profile packets, while traffic **policing** drops them.

Leaky Bucket

- Parameters: Bucket size (b) and Rate (r)
- Packets have limited exit rate of R, where b/r is the max queuing delay
  - Bursts exceeding b lead to packet loss
  - For packets of size L, the inter-departure time is L/r

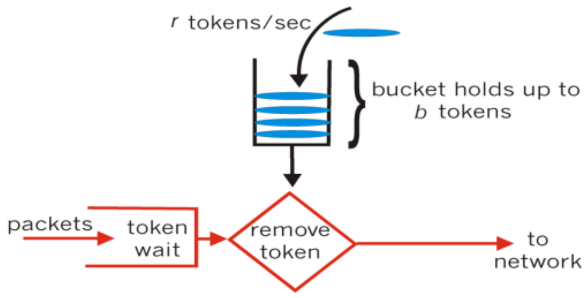


Leaky Bucket

Token Bucket

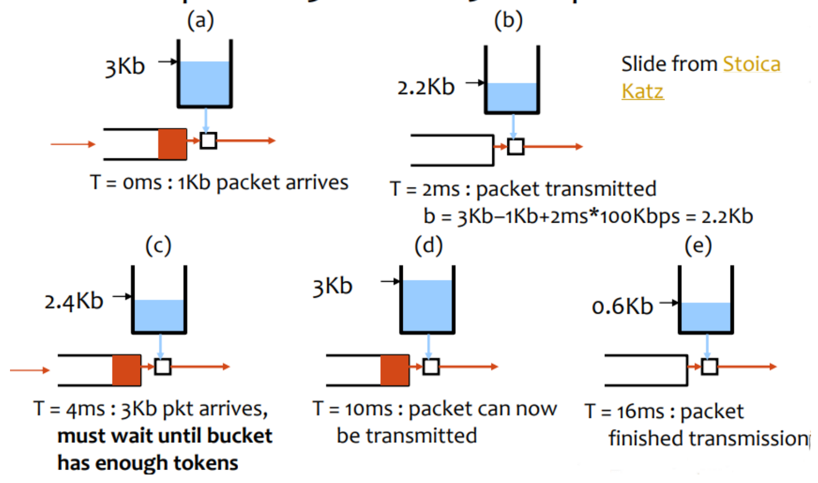
Limit input to specified Burst Size and Average Rate

- Bucket can hold b tokens
- Tokens generated at rate r token/sec unless bucket full
- Over **any** interval of length t, the number of packets admitted is always ≤ (r.t + b)

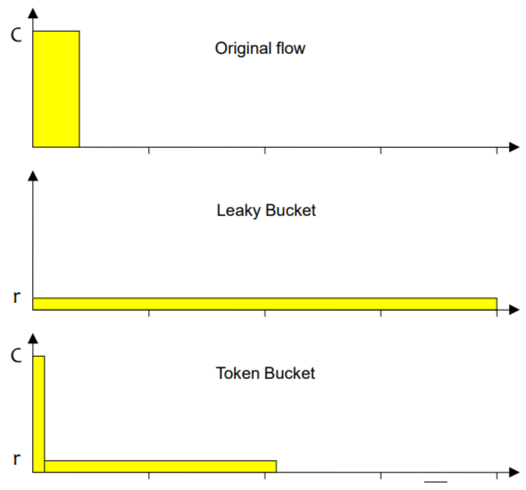


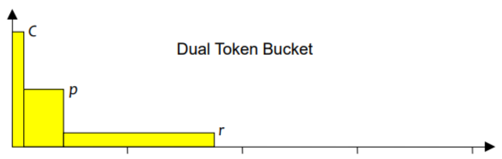
Example

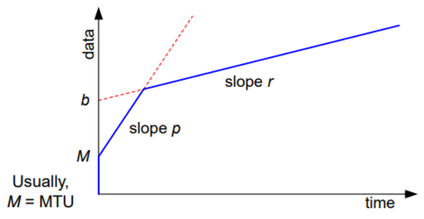
- r = 100 Kbps b = 3 Kb R = 500 Kbps


(a)
3Kb
T = 0ms : 1Kb packet arrives
(b)
2.2Kb
T = 2ms : packet transmitted
b = 3Kb–1Kb+2ms*100Kbps = 2.2Kb
Slide from Stoica Katz
(c)
2.4Kb
T = 4ms : 3Kb pkt arrives, must wait until bucket has enough tokens
(d)
3Kb
T = 10ms : packet can now be transmitted
(e)
0.6Kb
T = 16ms : packet finished transmission

Traffic shape



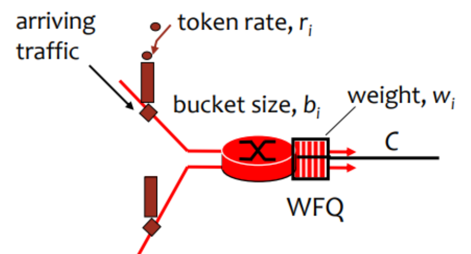


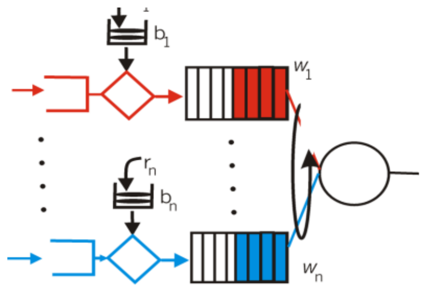


- Equivalent to two token buckets in series: (r,b) and (M,p)
- Blue curve is upper bound starting from **any** instant

Token Bucket + WFQ → QoS Guarantee

- Token bucket and WFQ combine to provide guaranteed upper bound on delay, i.e., QoS guarantee!





available rate for flow *i*: $\frac{w_i}{\sum_j w_j} C$

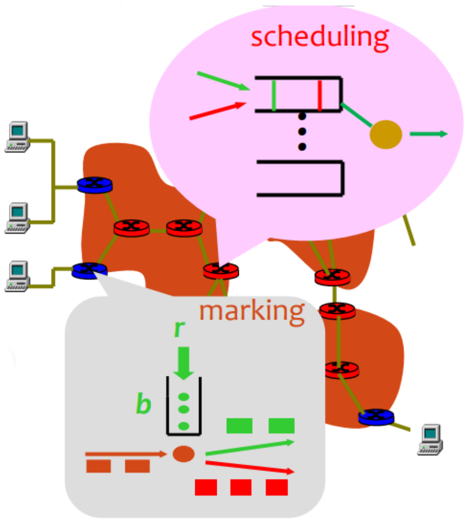
max delay for in-profile packets of flow *i*: $\frac{b_i}{C \cdot w_i / \sum_j w_j} + \frac{L_{\max}}{C}$

IETF Differentiated Services

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

- Scalability:
    - Simple functions in network core and more complex functions at edge routers/hosts
- Does not define service classes, but provides functional components to build service classes

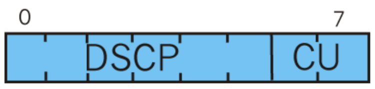
Diffserv Architecture



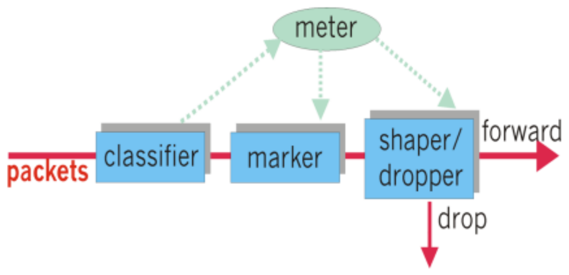
- Edge router:
    - Per flow traffic management
    - Marks packets as in-profile and out-of-profile
        - **Class-based marking:** packets of different classes marked differently
        - **Intra-class marking:** conforming portion of flow marked differently from the non-conforming one
- Core router:
    - Per class traffic management
    - Buffering and scheduling based on marking at edge
    - Preference given to in-profile packets

Classification and Conditioning

- Packet is marked in the Type of Service (ToS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Services Code Point (DSCP) and determine Per Hop Behaviour (PHB) that the packet will receive
- 2 bits "currently unused"



- May be desirable to limit traffic injection rate of some class:
    - User declares traffic profile (e.g., rate, burst size)
    - Traffic metered, shaped if non-conforming



Forwarding (PHB)

- Per Hop Behaviour (PHB) results in a different observable (measurable) forwarding performance behaviour
- PHB does not specify what mechanisms to use to ensure required PHB performance behaviour

PHBs developed:

- **Expedited Forwarding:** packet departure rate of a class equals or exceeds a specified rate

    - Logical link with a minimum guaranteed rate
- **Assured Forwarding:** 4 classes of traffic
    - Each guaranteed minimum amount of bandwidth
    - Each with three drop preference partitions

7.6 - Providing QoS Guarantees

Principles for QoS Guarantees (more)

> **Principle 4:** Admission Control: flow declares its needs; network may block call (e.g., busy signal) if it cannot meet needs

QoS guarantee scenario

- **Resource reservation**
    - Call setup, signalling (RSVP)
    - Traffic profile & QoS declaration
    - Per-element admission control

IETF Integrated Services

- Architecture for providing QoS guarantees in IP networks for individual application sessions
- Resource reservation - routers maintain state info of allocated resources and QoS requirements
- Admit/reject new call setup requests

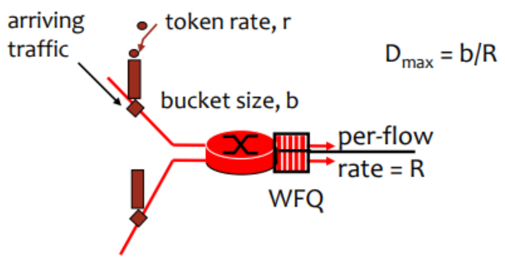
> **Question:** Can newly arriving flow be admitted with performance guarantees while not violating QoS guarantees made to already admitted flows?

Admission Control

- Arriving session must:
    - Declare its QoS requirement
        - **R-spec:** defines the QoS being requested
    - Characterize traffic it will send into network
        - **T-spec:** defines traffic profile
    - Use a signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
        - **RSVP**

Intserv QoS: Service models

- **Guaranteed service**
    - Worst case traffic arrival: token-bucket-policed source
    - Simple (mathematically provable) bound on delay
- **Controlled load service**
    - Flows receive a quality of service closely approximating best-effort from that same network element when lightly loaded



Signalling on the Internet

connectionless (stateless) forwarding by IP routers	+	best effort service	=	no network signaling protocols in initial IP design

- **New requirement:** reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- **RSVP:** Resource Reservation Protocol
  - "...allow users to communicate requirements to network in robust and efficient way." i.e., signaling!

RSVP Design Goals

1. Accommodate **heterogeneous receivers** (different bandwidth along paths)
2. Accommodate different applications **with different resource requirements**
3. Make **multicast a first class service**, with adaptation to multicast group membership
4. **Leverage existing multicast/unicast routing**, adapting to changes in underlying unicast, multicast routes
5. **Control protocol overhead** to grow (at worst) linear in # receivers
6. **Modular design** for heterogeneous underlying technologies

RSVP: does **not**...

- Specify how resources are to be reserved
  - Rather: a mechanism for communicating needs
- Determine routes packets will take
  - That's the job of routing protocols
  - Signaling decoupled from routing
- Interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

RSVP: overview of operation

- **Receivers join a multicast group**
  - Done outside of RSVP
  - Senders need not join group
- **Sender-to-network signaling**
  - <u>Path message:</u> make sender presence known to routers
  - Path teardown: delete sender's path state from routers
- **Receiver-to-network signaling**
  - <u>Reservation message:</u> reserve resources from sender(s) to receiver
  - Reservation teardown: remove receiver reservations
- **Network-to-end-system signalling**
  - Path error
  - Reservation error

IntServ and DiffServ

- Complementary
  - DiffServ: aggregation per client/user/user-group/application, ISP oriented
  - IntServ: per flow, app oriented
- Integration possible
  - IntServ reservation with DiffServ "flows"

	DiffServ	**IntServ**
Signaling	Network	Application
Granularity	Aggregate (Class)	Flow
Mechanism	Packet Class (other possible)	Dst Addr, protocol, port
Scope	Between networks, E2E	End-to-end

Comments on QoS solutions Usage

- All ISPs on the packets' path must agree on the E2E commitment
  - So that the selling QoS ISP can provide the paying customer
- Couple with Network dimensioning
  - If there isn't capacity, QoS will not magically work
- Billing procedures
  - Need to bill customers based on traffic volume (and differentiate among the traffic)
- Delay is mostly due to
  - Access rates; Router hops → Service quality for paying and best-effort clients similar?

Side Note: Deep Packet Inspection

- Uses information up to Application layer
- Can differentiate based on all information
- Used by ISPs
- Hardware implemented
  - Needs to work at wire speed

Chapter 7: Summary

- **Principles**
  - Classify multimedia applications
  - Identify network services applications need
  - Making the best of best effort service
- **Protocols and Architectures**
  - Specific protocols for best-effort
  - Mechanisms for providing QoS
  - Architectures for QoS
    - Multiple classes of service
    - QoS guarantees, admission control