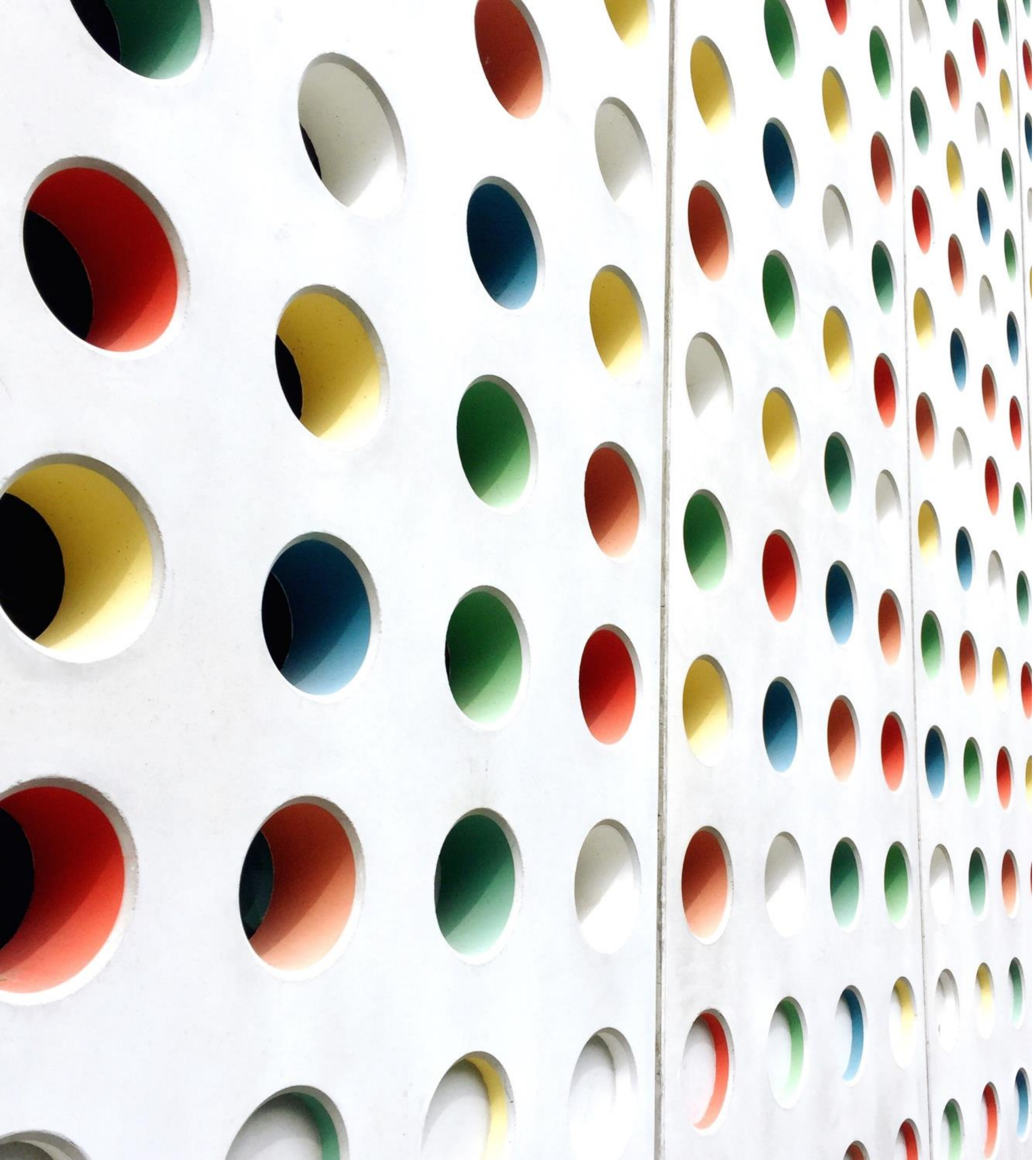


Protocols for Streaming Applications

CHANDREYEE CHOWDHURY

Need for Multimedia Protocols

Over the next few years, 90 percent of the bits carried on the Internet will be video related and consumed by more than 1 billion users

Video delivery over the public Internet is also referred to as over-the top (OTT) video streaming, since the content or the streaming service provider usually differs from the network provider

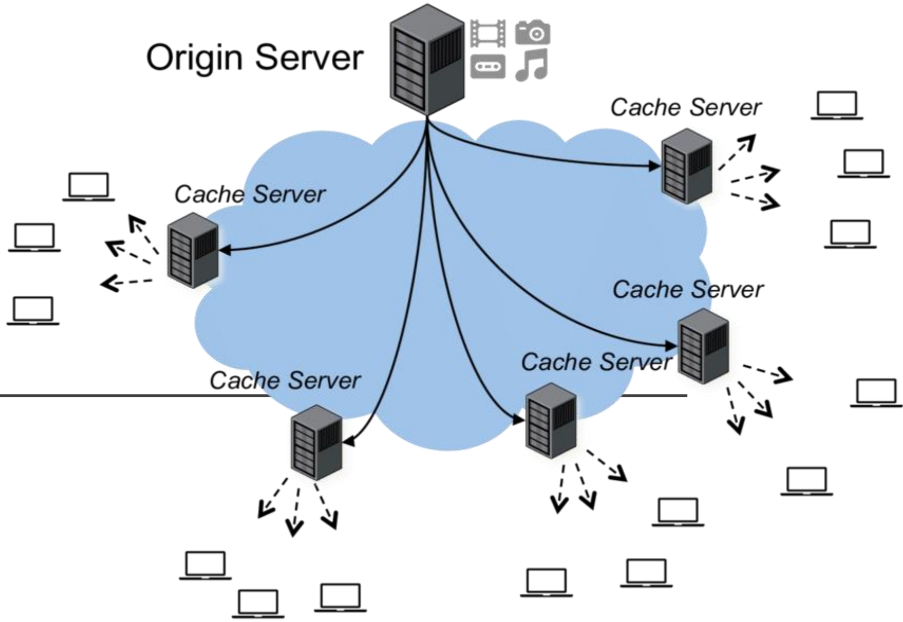
Cable and IPTV services run over managed networks for distribution because these services use multicast transport and require certain quality-of-service (QoS) features.

How to design video service to run over mostly unmanaged networks?

Introduction

- ❑ Netflix currently has over 20 million subscribers, many of whom use its streaming services on a variety of devices (www.netflix.com/ MediaCenter?id=5379)
- ❑ We can divide Internet video, also known as over-the-top (OTT) services, into distinct categories
  - ❑user-generated content from mostly amateurs
    - ❑the content served by YouTube
  - ❑professionally generated content from studios and networks to promote their commercial offerings and programming
    - ❑What you find on ABC.com or Hulu
  - ❑direct movie sales to consumers over the Internet (also referred to as electronic sell-through, or EST)
    - ❑Netflix

Application Classes



Typically sensitive to delay, but can sometimes tolerate packet loss (would cause glitches that can be concealed somewhat)

Data contains audio and video content ("continuous media"), three classes of applications:

- Streaming
- Unidirectional Real-Time
- Interactive Real-Time

These streaming technologies send the content to the viewer over a unicast connection (from a server or content delivery network [CDN]) through either a proprietary streaming protocol running on top of an existing transport protocol, mostly TCP and occasionally UDP, or the standard HTTP protocol over TCP

Application Classes (more)

Streaming

- Consumers' desire to access practically limitless amounts of content any time they want and the drop in delivery costs hastened the deployment of streaming services
    - Clients request audio/video files from servers and pipeline reception over the network and display
    - Interactive: user can control operation (similar to VCR: pause, resume, fast forward, rewind, etc.)
    - Delay: from client request until display start can be 1 to 10 seconds
    - Ubiquitous on the web

Application Classes (more)

Unidirectional Real-Time:

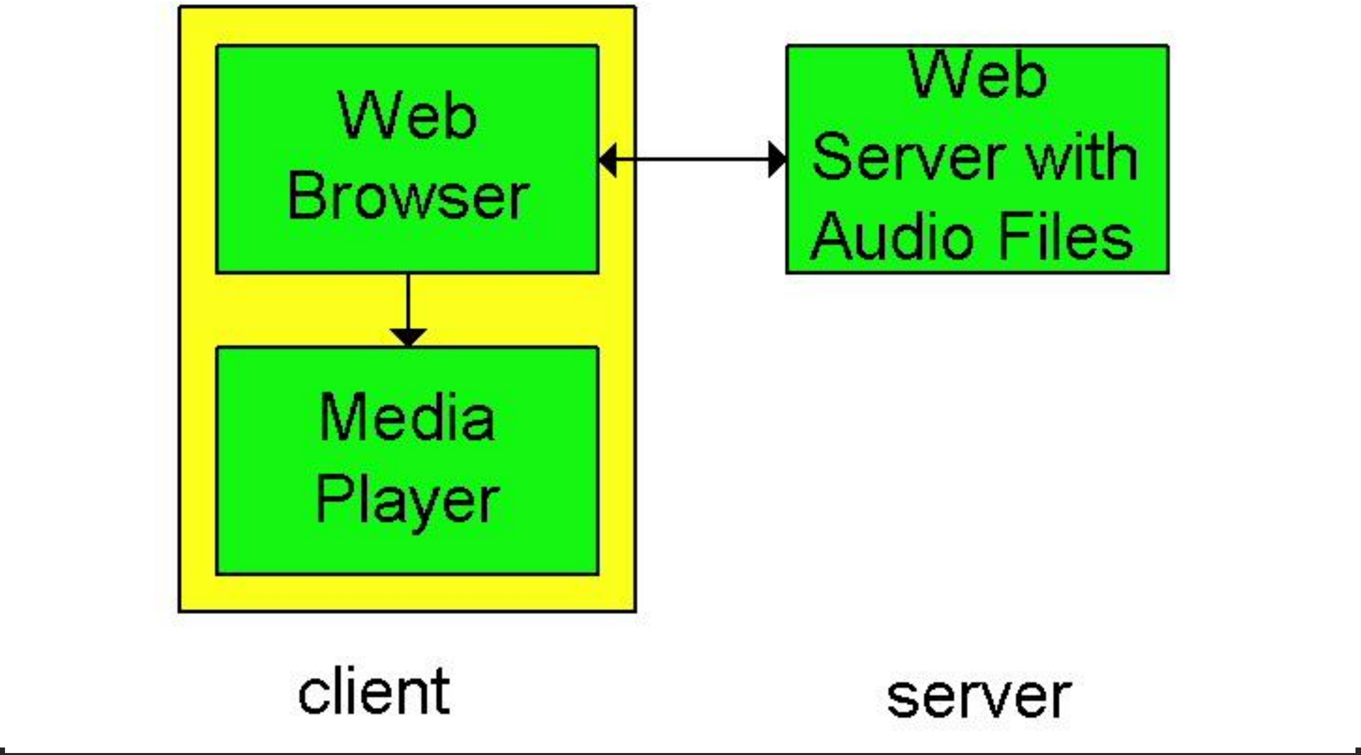
- similar to existing TV and radio stations, but delivery on the network
- Non-interactive, just listen/view

Interactive Real-Time :

- Phone conversation or video conference
- More stringent delay requirement than Streaming and Unidirectional because of real-time nature
- Video: < 150 msec acceptable
- Audio: < 150 msec good, <400 msec acceptable

Progressive Download

- Historically, progressive download, which uses HTTP over TCP, has been quite popular for online content viewing due to its simplicity



Progressive Download

❑In progressive download, the playout can start as soon as enough necessary data is retrieved and buffered.

YouTube delivered more than 2 billion videos daily with this approach

YouTube Live Streaming Ingestion Protocol is the protocol used now that supports both HTTP live streaming (HLS) and RTMPS

❑ Progressive download doesn’t offer the flexibility and rich features of streaming

❑ Before the download starts, the viewer must choose the most appropriate version

  ❑there are multiple offerings with different resolutions for the same content.

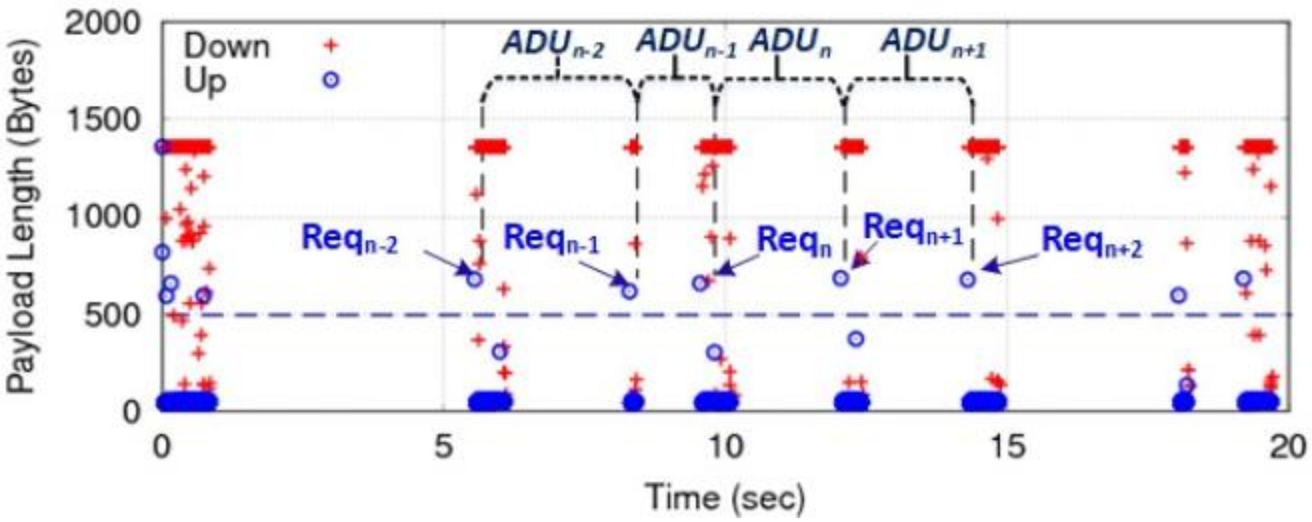
❑If there isn’t enough bandwidth for the selected version

  ❑ the viewer might experience frequent freezes and rebuffering.

  ❑ Trick modes, such as fast-forward seek/play or rewind, are often unavailable or limited.

❑Once the client fills up a minimum required buffer level, it starts playing the media while continuing to download content from the server in the background.

YouTube Case Study



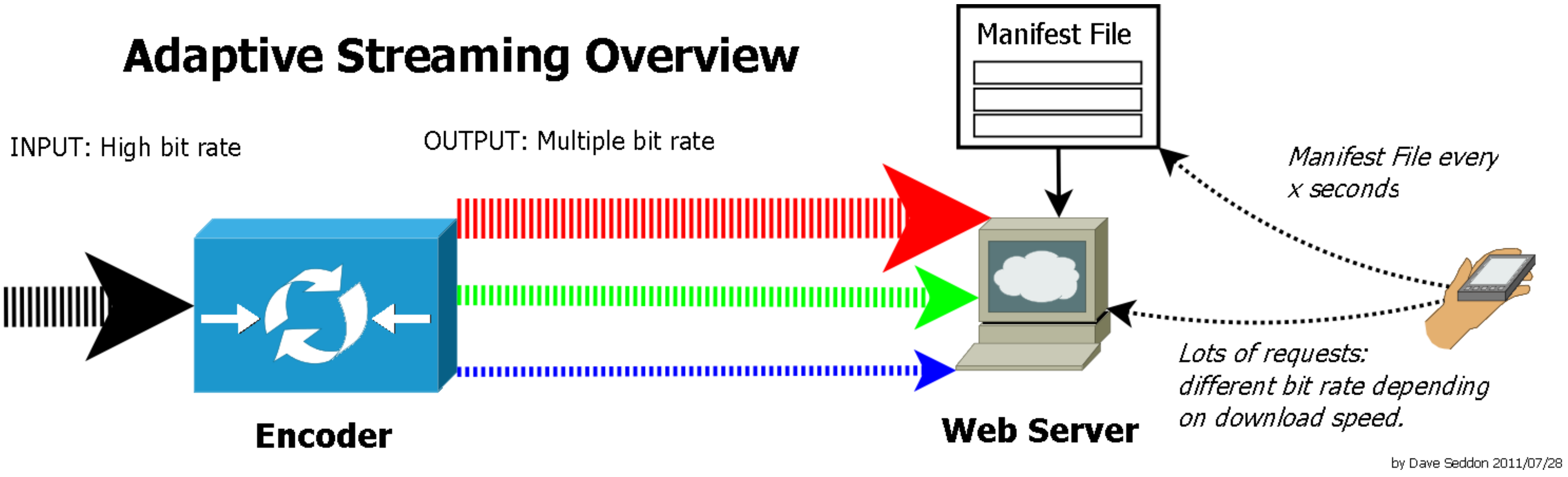
- For streaming with progressive downloading, each video session consists of two phases: startup and steady state.
- In startup, the video session downloads rapidly to fill the client player’s playout buffer as fast as possible. Once the buffer is filled, the video session enters steady state and pauses downloading, resuming only when the playout buffer depletes.
- In both progressive downloading and HAS, the video player sends an HTTP request to the server to request the next video segment.

F. Li, J. W. Chung, and M. Claypool. 2018. Silhouette: Identifying YouTube Video Flows from Encrypted Traffic. In Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '18). ACM, New York, NY, USA, 19-24.

Adaptive Streaming

- Adaptive streaming is a hybrid of progressive download and streaming
- It is used to address the shortcomings of progressive download while preserving the simplicity of it.
- the adaptive streaming client sends HTTP request messages to retrieve particular segments of the content from an HTTP server and then renders the media while the content is being transferred.
- On the other hand, these segments are short, enabling the client to download only what's necessary and use trick modes much more efficiently, giving the impression that the client is streaming

Adaptive Streaming



Adaptive Streaming

- ❑ Short duration segments (for example, MPEG4 file fragments) are available at multiple bitrates, corresponding to
  - ❑ different resolutions and quality levels
- ❑ The client can switch between different bitrates at each request

The client player strives to always retrieve the next best segment after examining a variety of parameters related to

- ◦ **available network resources**, such as available bandwidth and the state of the TCP connections;
- ◦ **device capabilities**, such as display resolution and available CPU; and
- ◦ **current streaming conditions**, such as playback buffer size.

Adaptive Streaming

❑ Today, practically any connected device supports HTTP in some form.

❑ It’s a pull-based protocol that easily traverses middleboxes, such as firewalls and NAT devices.

❑ It keeps minimal state information on the server side, which makes HTTP servers potentially more scalable

Individual segments of any content are separately cacheable as

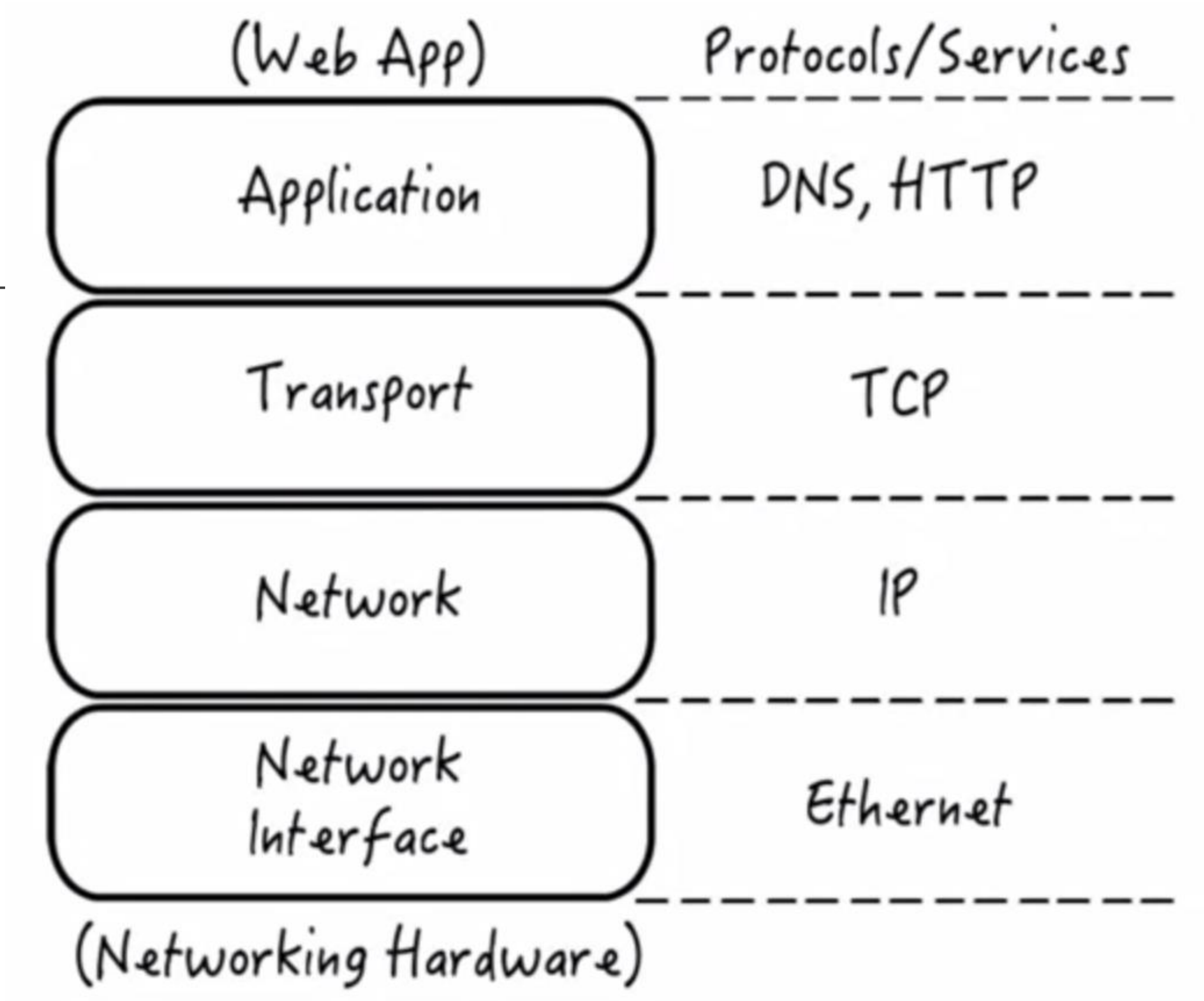
- regular Web objects using HTTP or
- any RESTful (conforming to the Representational State Transfer constraints) Web protocol
- This allows distributed CDNs to greatly enhance the scalability of content distribution

https://about.netflix.com/en/news/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience

Media Streaming

- The type of content being transferred, and the underlying network conditions usually determine the methods used for communication.
- For simple file transfer over a lossy network, the emphasis is on reliable delivery:
- added redundancy protects packets against losses,
- or retransmission can recover lost packets.
- When it comes to audio/video media delivery with real-time viewing requirements,
- The emphasis is on low latency and jitter,
- and efficient transmission;
- Occasional losses might be tolerated in this case

Protocol Stack



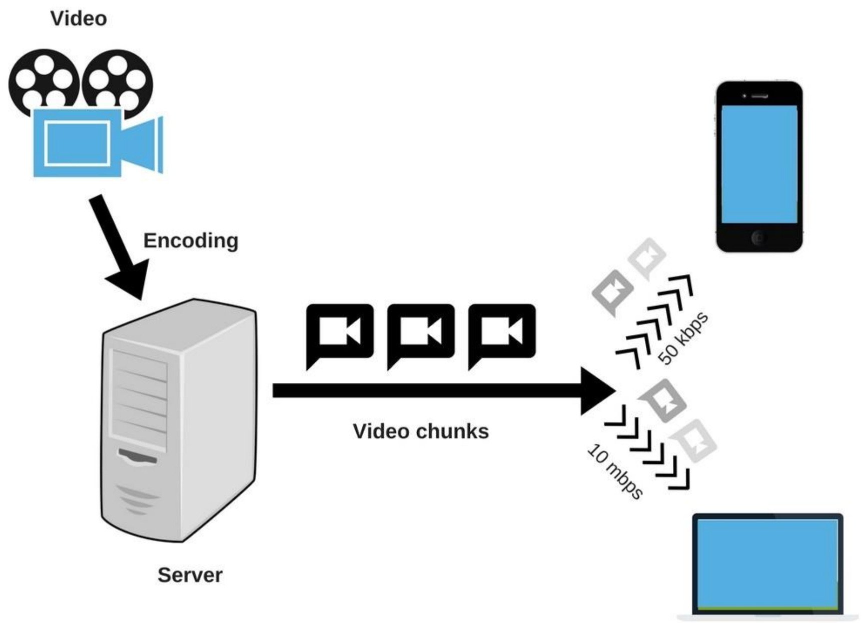
Push Protocols

When you compress a video, you typically have to tell the video compression software that it should not exceed a certain bitrate.
For example, if you tell your software to compress at 3mbps, the software will compress each second of the video such that only 3mb of data (on average) is sent to the decoder or the player.

- In conventional push-based streaming, the server transmits content at the media encoding bitrate to match the client’s media consumption rate
- client buffer levels remain stable over time
- transmission above the encoding bitrate might not be even possible for live streams in which the stream is encoded on the fly.
- if packet loss or transmission delays occur over the network, the client’s packet retrieval rate can drop
- If it drops below the client’s consumption rate, it might drain its buffer
- It results in a buffer underflow that interrupts the playback

Push based Adaptive Streaming

- When network conditions improve, the server does the opposite and switches to a higher bitrate stream — again, in multiple intermediate steps to avoid a sudden network overload.
- Provided that the stream isn’t live and the network capacity allows for a higher-bitrate transmission
- the server can also choose to send packets at a higher rate than the media encoding rate to fill up the client’s buffer.
- By dynamically monitoring available bandwidth and buffer levels and by adjusting the transmission rate via stream switching
- push-based adaptive streaming can achieve smooth playback at the best possible quality level without pauses or stuttering.



Push based Adaptive Streaming

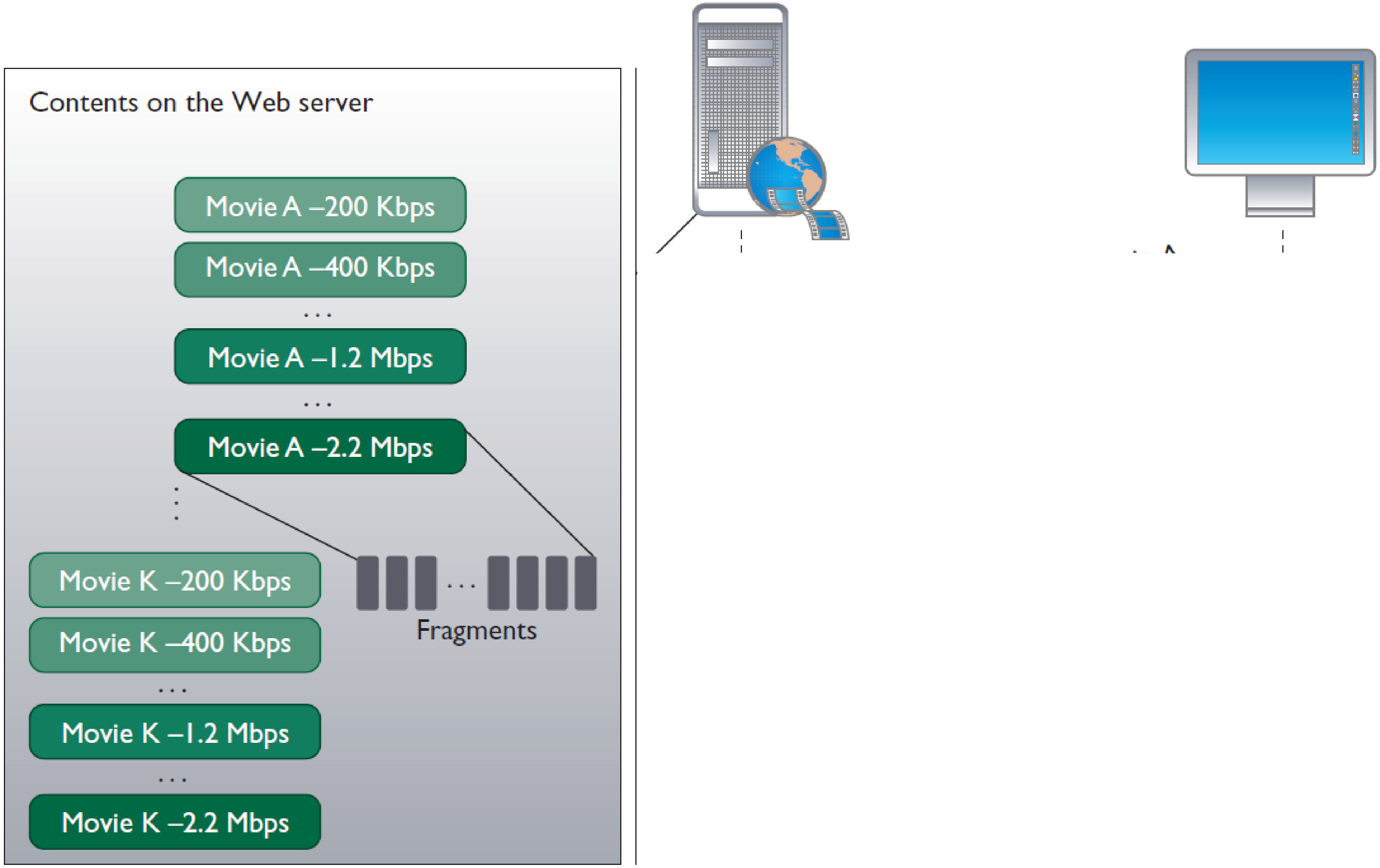
- To prevent a buffer underflow, the server can dynamically switch to a lower-bitrate stream.

- This, in turn reduces the media consumption rate at the client side and counteracts the effect of network bandwidth capacity loss.

- Because a sudden drop in the encoding bitrate can result in a noticeable visual quality degradation, this reduction should occur in multiple intermediate steps until the client's consumption rate matches or drops below its available receive bandwidth.

Fragmentation for Streaming

- ❑ The media content is divided into short-duration media segments (also called fragments)
- ❑ Each of it is encoded at various bitrates and can be decoded independently.
- ❑ When the client plays the fragments back to back, it can seamlessly reconstruct the original media stream
- ❑ Although media fragment structures differ among implementations, the basic principle for fragment construction is the same.



Contents on the Web server

Movie A –200 Kbps

Movie A –400 Kbps

. . .

Movie A –1.2 Mbps

. . .

Movie A –2.2 Mbps

⋮

Movie K –200 Kbps

Movie K –400 Kbps

. . .

Movie K –1.2 Mbps

. . .

Movie K –2.2 Mbps

Fragments

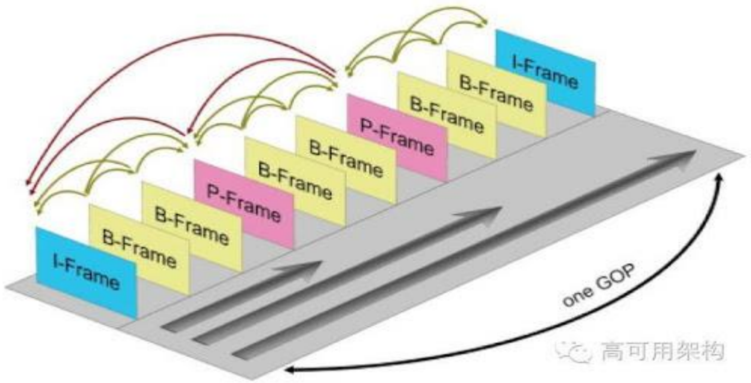
Audio Frames

❑ Each audio frame usually consists of constant duration audio samples in the milliseconds range

❑ each frame is usually decodable on its own for common audio codecs

❑ Therefore, one can easily stuff audio data into a media fragment by combining a sufficient number of audio frames to match the fragment duration.

Video Frames



□For video, on the other hand, the frames aren't necessarily independently decodable due

to temporal prediction commonly applied between the frames.

□For video, the partitioning for fragment construction is performed at the group of pictures (GoP) boundary

□In video coding, a GoP is a frame sequence that starts with an intra-coded frame (I-frame) that can be decoded independently

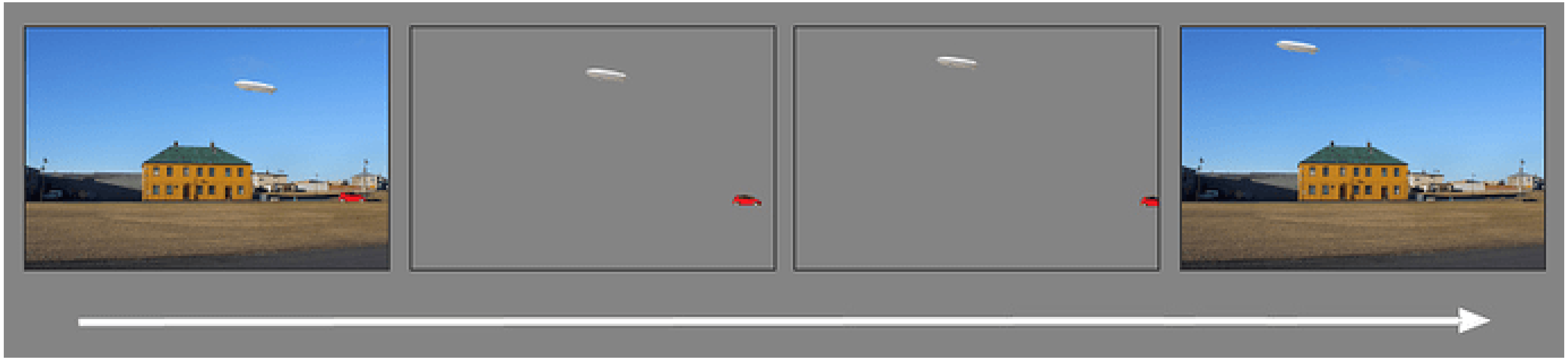
□ It is followed by predicted frames that depend on other frames.

□ If the predicted frames within a GoP depend only on the frames within that same GoP, we call it a closed GoP — otherwise, it's an open GoP.

□For a closed GoP, the encoder simply adjusts the number of frames in the GoP such that the total duration of its frames is equal to the desired fragment duration.

□if the fragment duration is large compared to a typical GoP size, we'll want to pack more than one GoP into a fragment

Video Frames



The typical GOP size for broadcast applications is 30 frames.
IPTV uses typically more than 30 frames and streaming video often uses more than 300 frames.

Pull based adaptive streaming- **Microsoft Smooth Streaming**

❑ In Smooth Streaming, all fragments with the same bitrate are stored in a single MP4 file.

❑ Therefore, each available bitrate has a separate file.

❑ Fragments are usually two seconds long and typically contain a single GoP

Movie Fragments

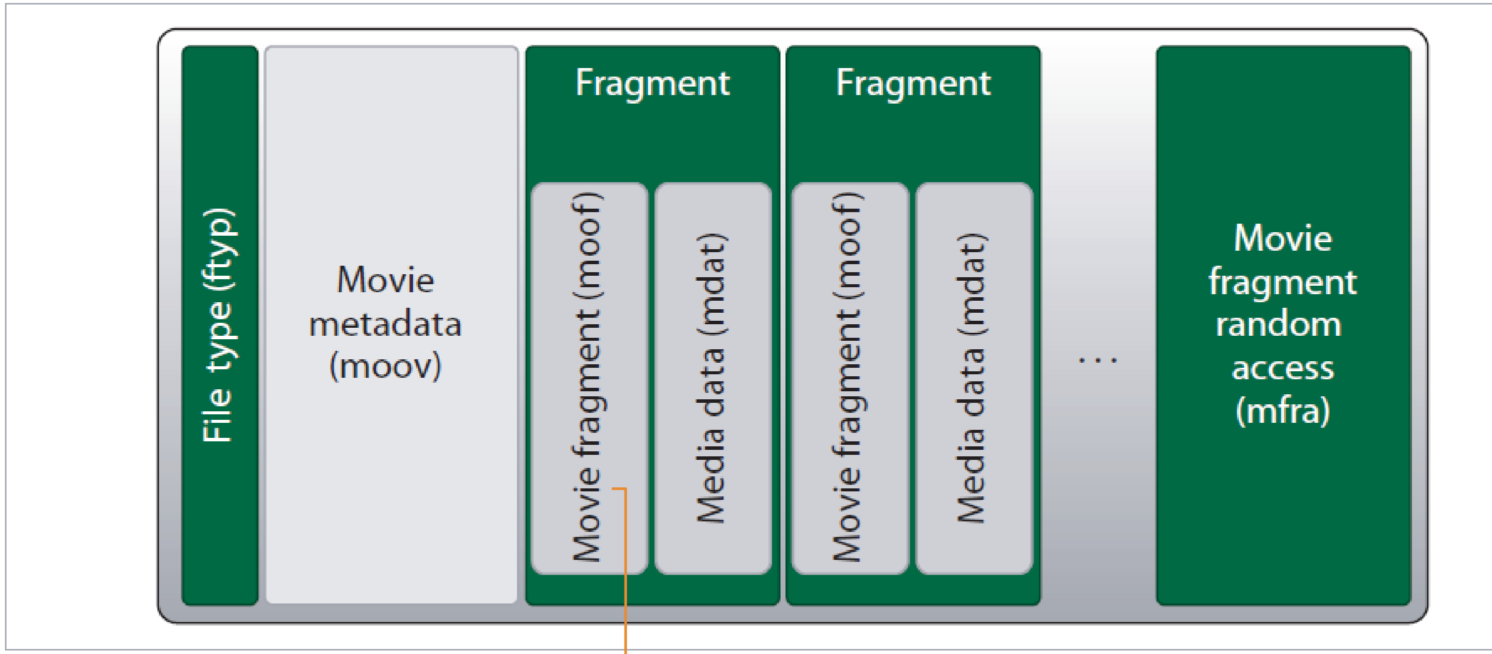
- ❑A fragmented MP4 file is composed of a hierarchical data structure.
- ❑The most basic building block of this hierarchy is called a box; it can contain audio/video data and metadata.
- ❑Each box type is designated by a four-letter identifier.

Media Fragments

❑The file starts with an ftyp box that describes the version information for the specifications with which the file complies, followed by

❑a moov box that describes the media tracks available in the file.

❑ The audio/video media data for a single fragment is contained within a box of type mdat.



In a fragmented MP4 container structure, an mdat box is immediately preceded by a box of type moof, which contains metadata for that fragment.

Signaling info, metadata

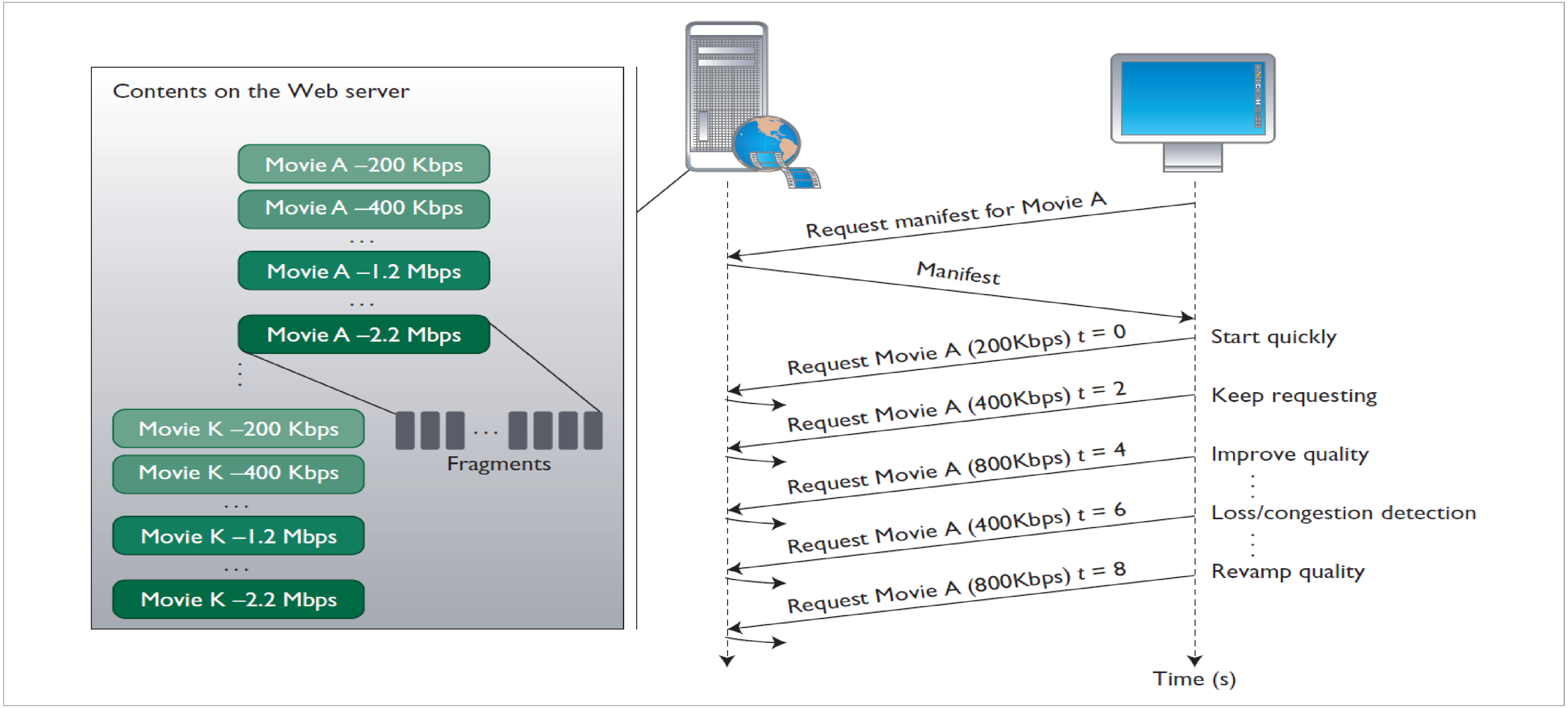
sequence counter for the fragment, the number of samples within the fragment, and each sample's duration.

Microsoft Smooth Streaming

- ❑For the client to request a fragment with a specific bitrate and start time, it first needs to know which fragment(s) are available on the server.
- ❑This information is communicated to the client at the beginning of the session via a client-side manifest file.
- ❑In addition to bitrates, this file describes codecs, video resolutions and other auxiliary information
- ❑The client uses the information in this file to make HTTP GET requests to the server for the individual fragments.
- ❑Each fragment is downloaded via a unique HTTP request-response pair.
- ❑The HTTP request message header contains two pieces of information: the bitrate and the requested fragment’s time offset

- ❑The server looks up the bitrate information it receives from the client in the manifest file and
- ❑determines the corresponding MP4 file to search

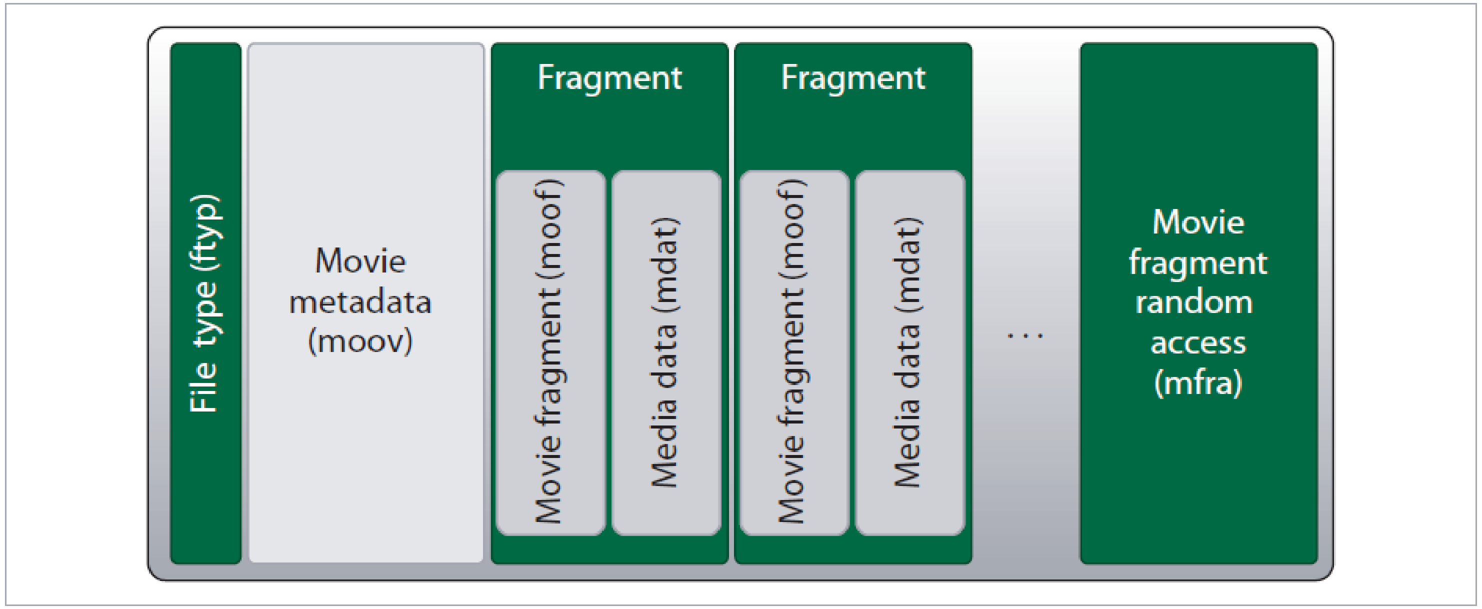
- ❑When the server gets an HTTP encapsulated request from the client for a particular media fragment, it first needs to determine which MP4 file to search for that fragment.
- ❑This information is contained in another manifest file


Contents on the Web server
Movie A –200 Kbps
Movie A –400 Kbps
...
Movie A –1.2 Mbps
...
Movie A –2.2 Mbps
Movie K –200 Kbps
Movie K –400 Kbps
...
Movie K –1.2 Mbps
...
Movie K –2.2 Mbps
Fragments
Request manifest for Movie A
Manifest
Request Movie A (200Kbps) t = 0
Start quickly
Request Movie A (400Kbps) t = 2
Keep requesting
Request Movie A (800Kbps) t = 4
Improve quality
Request Movie A (400Kbps) t = 6
Loss/congestion detection
Request Movie A (800Kbps) t = 8
Revamp quality
Time (s)

Microsoft Smooth Streaming

- An MP4 file also contains boxes of type mfra that contain a fragment’s location and presentation time. The media server uses the time offset information it receives from the client to find a match with a fragment index in the MP4 file.

- Once the server locates the fragment in the file, it sends the contained moof box followed by the mdat box, and these make up the fragment on the wire



Apple HTTP Live Streaming

- ❑ Follows a different approach for fragment (referred to as media segment in the implementation) storage, which is based on the ISO/IEC 13818-1 MPEG2 Transport Stream file format
- ❑ As opposed to Smooth Streaming, each media segment is stored in a separate transport stream file
- ❑ A stream segmentation process reads a continuous transport stream file and divides it into equal-duration segments of 10 seconds by default
- ❑Longer fragments also allow better compression because they have more temporal redundancy.
  - ❑ it also reduces the granularity at which fragment-switching decisions can be made

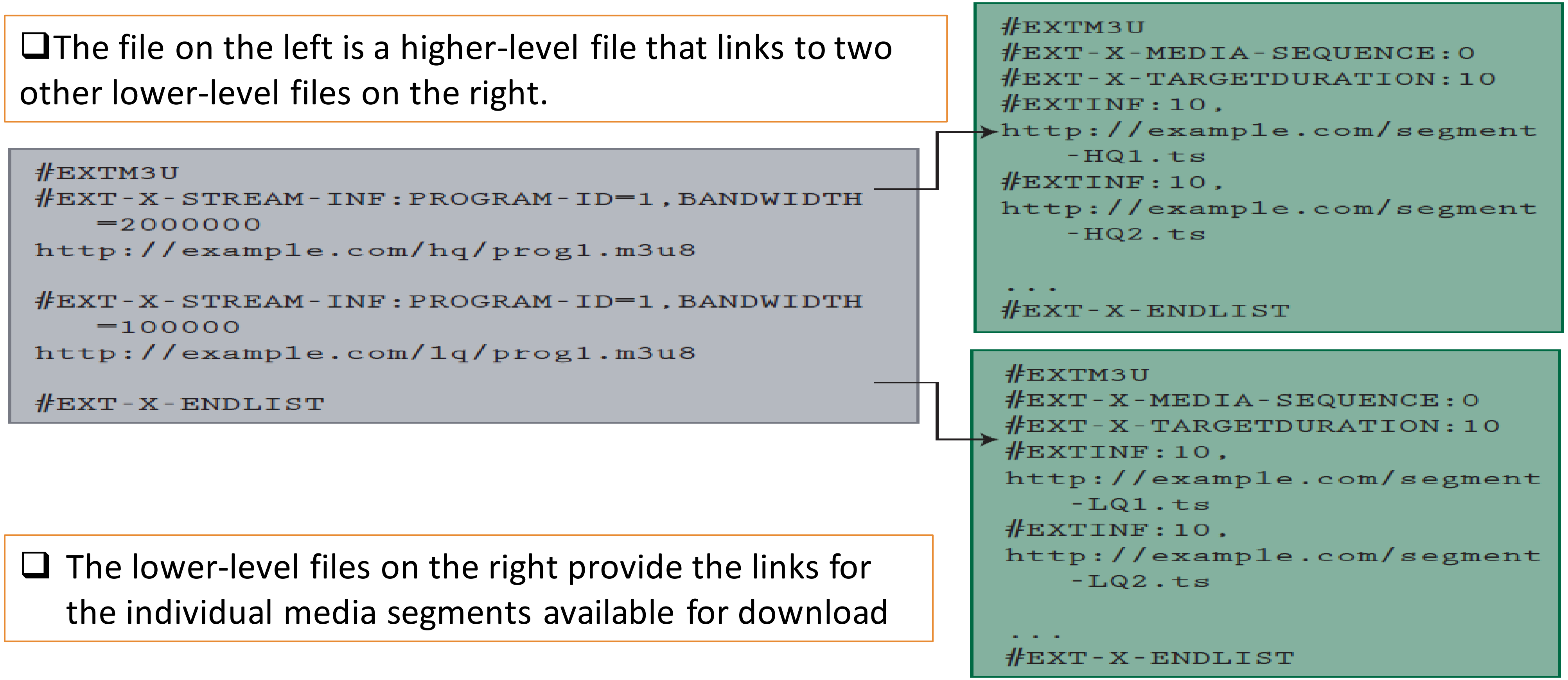
Manifest Files

❑ a client side manifest file keeps a record of the media segments available for the client.

❑ The manifest file format is an extension to the MP3 playlist file standard;

Apple HTTP Live Streaming

❑The file on the left is a higher-level file that links to two other lower-level files on the right.



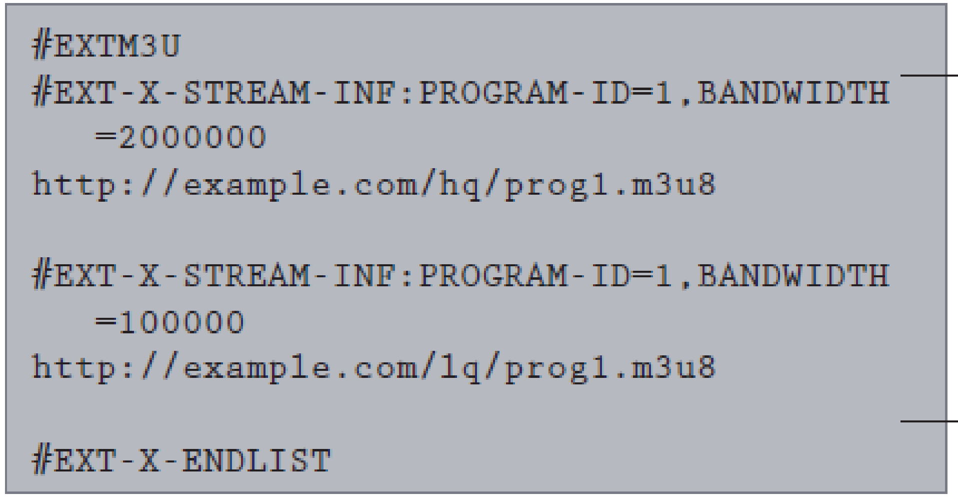
```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH
    =2000000
http://example.com/hq/prog1.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH
    =100000
http://example.com/lq/prog1.m3u8

#EXT-X-ENDLIST
```

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:10
#EXTINF:10,
http://example.com/segment
    -HQ1.ts
#EXTINF:10,
http://example.com/segment
    -HQ2.ts

...
#EXT-X-ENDLIST
```

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:10
#EXTINF:10,
http://example.com/segment
    -LQ1.ts
#EXTINF:10,
http://example.com/segment
    -LQ2.ts

...
#EXT-X-ENDLIST
```

❑ The lower-level files on the right provide the links for the individual media segments available for download

Live Streaming contd



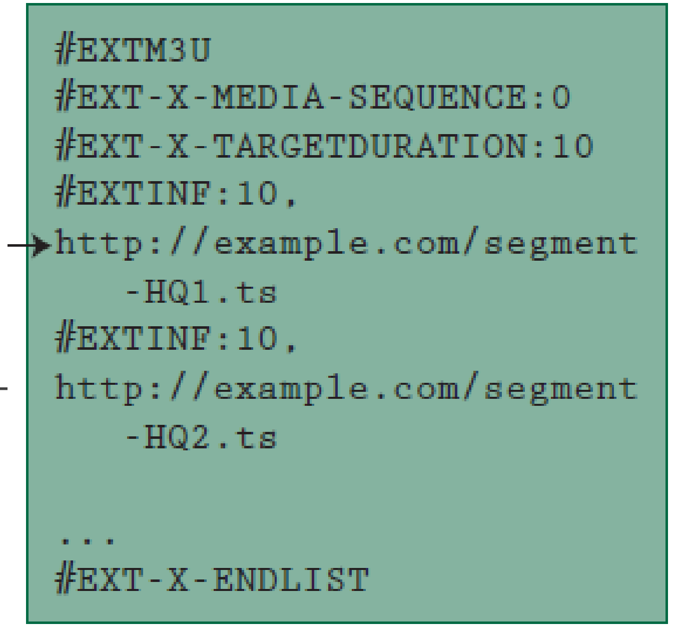
```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH
   =2000000
http://example.com/hq/prog1.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH
   =100000
http://example.com/lq/prog1.m3u8

#EXT-X-ENDLIST
```

- The manifest files are organized in a two level hierarchy.
- Each file starts with an EXTM3U tag that distinguishes it from an ordinary MP3 playlist.
- The link for each lower-level file is specified in a URI that’s always preceded by an EXT-X-STREAM-INF tag

In the example, the stream has two alternate encodings for each segment

- one low quality (100 Kbps) encoding and
- One high-quality (2 Mbps) encoding

Live Streaming contd

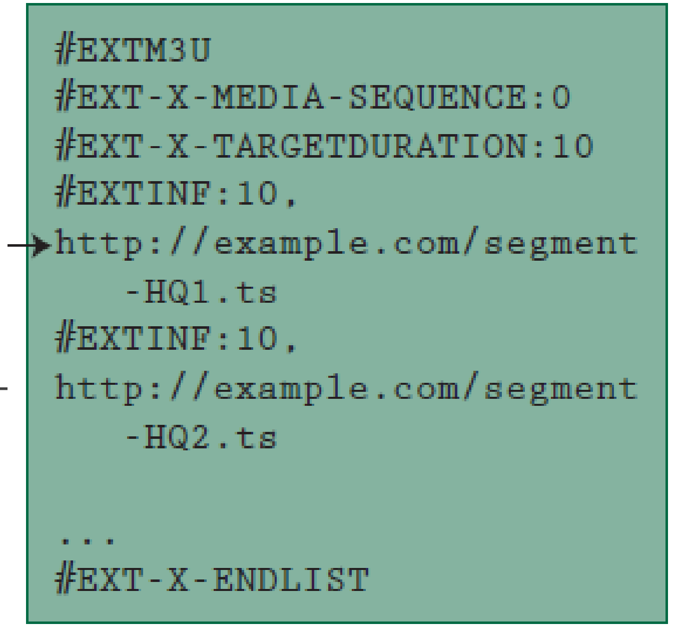


```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:10
#EXTINF:10,
http://example.com/segment
    -HQ1.ts
#EXTINF:10,
http://example.com/segment
    -HQ2.ts


...
#EXT-X-ENDLIST
```

- Following the EXTM3U tag is an EXT-X-MEDIA-SEQUENCE tag that lists the sequence number of the first segment in the playlist
- Each segment typically has a unique sequence number
- The URIs for the individual segments are marked with an EXTINF tag
- This tag has an attribute separated by a semicolon that indicates the duration of the corresponding media segment.
- For fixed-duration streams, EXTX-ENDLIST marks the end of the playlist.
- Live Streams
- This tag doesn’t exist in live streams where the playlist can grow dynamically.

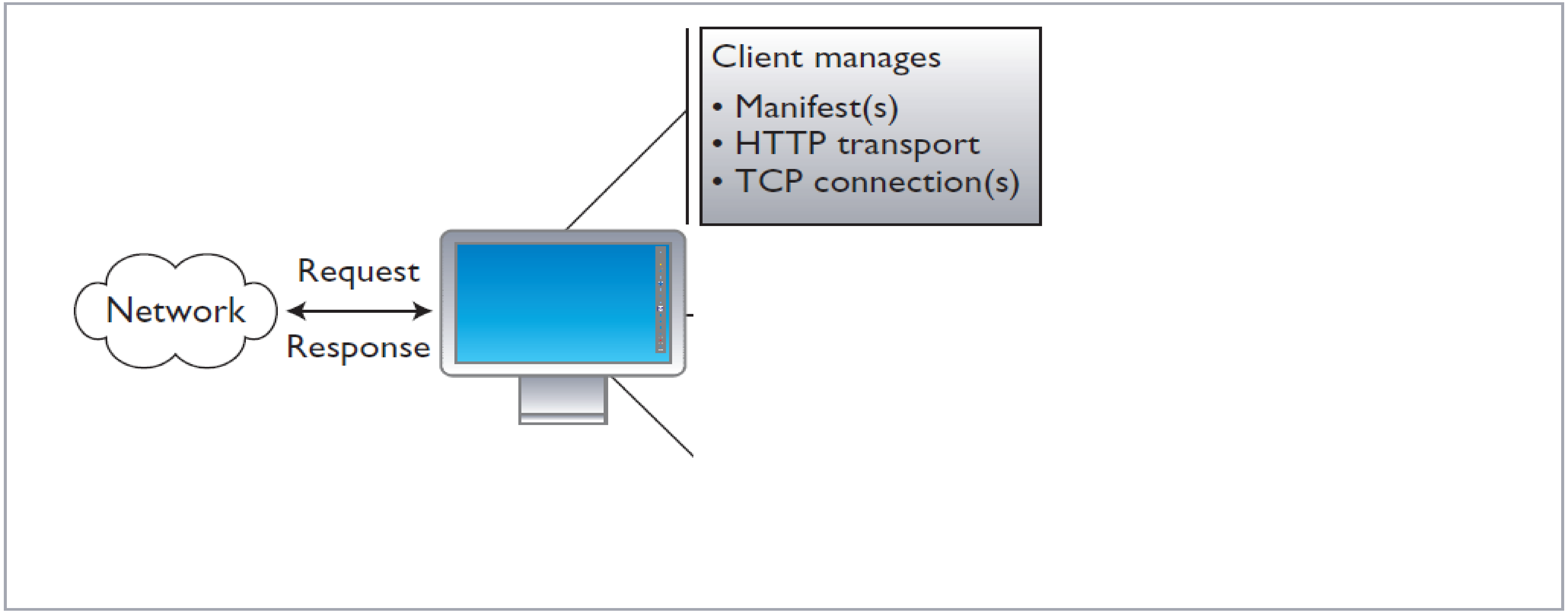
Live Streams

- □ In the case of live streams, the client must periodically refetch an updated manifest
- □ The period for refetching the manifest depends on whether the manifest file has changed since the last time it was reloaded.
- □ If the manifest has changed, the period is the duration of the last media segment in the playlist (which is specified by the EXTINF tag).
- □ If the manifest hasn’t changed, then the period is a multiple of the duration specified by the EXT-X-TARGETDURATION tag.
- □ This tag indicates the maximum EXTINF value of any media file that can be added to playlist
- □ This is constant throughout the manifest file.



```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:10
#EXTINF:10,
http://example.com/segment
   -HQ1.ts
#EXTINF:10,
http://example.com/segment
   -HQ2.ts


...
#EXT-X-ENDLIST
```

Functional Diagram of Client side Pull based Adaptive Streaming

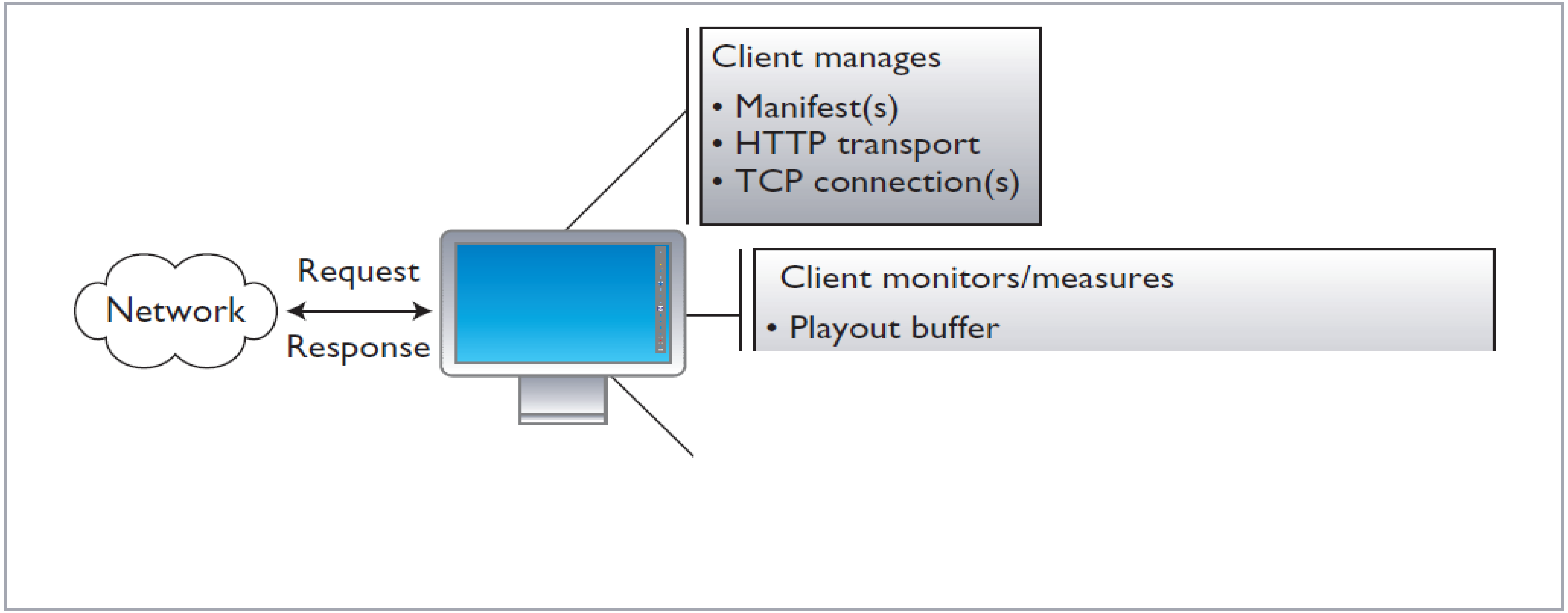


Client manages

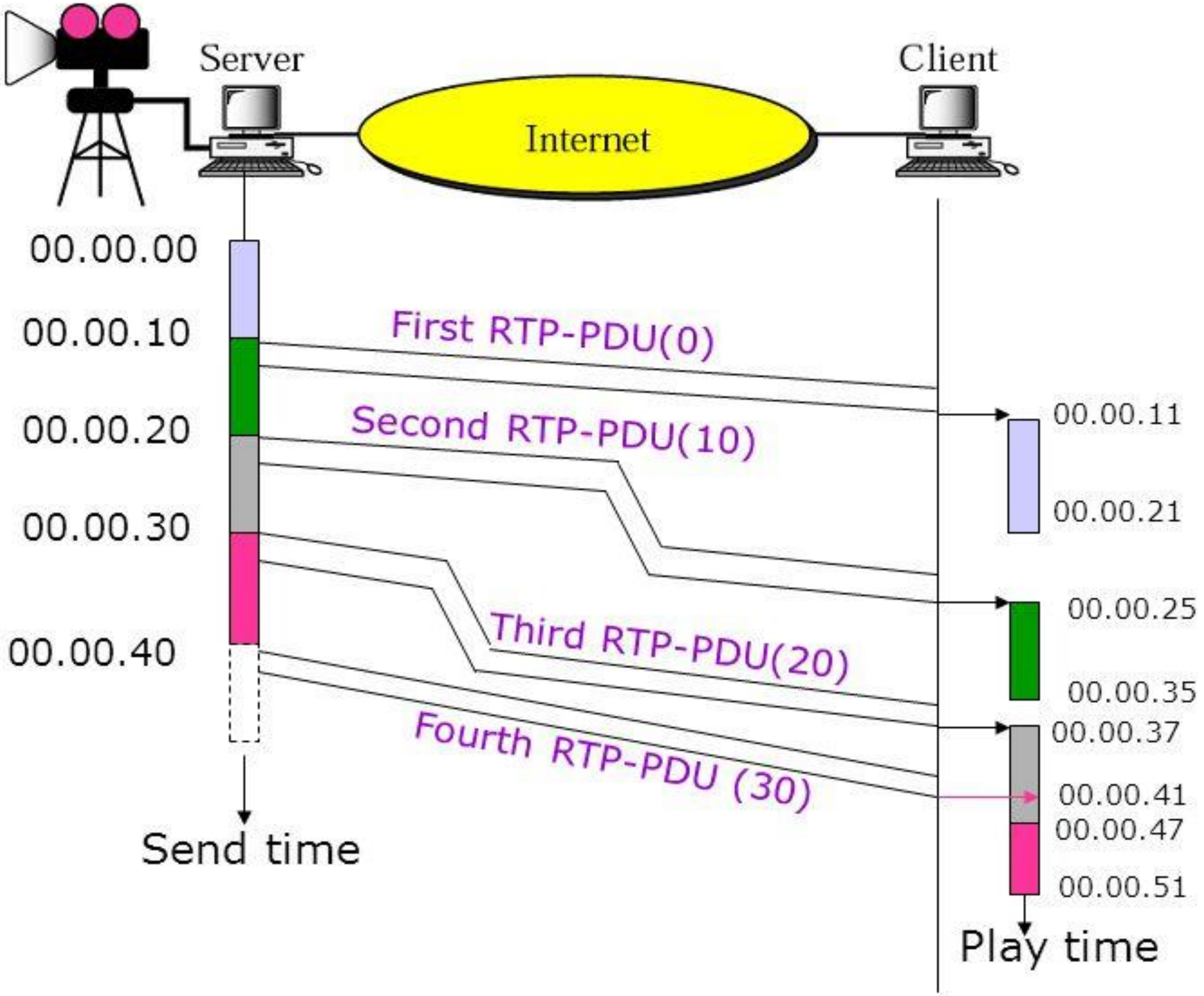
- Manifest(s)
- HTTP transport
- TCP connection(s)

- At minimum, the server provides standard responses to HTTP GET requests.
- The client gets a manifest file that identifies files containing media presentations at alternative bitrates.
- The client needs a file, called a client-side manifest or a playlist, to map fragment requests to specific files or to map byte ranges or time offsets to files.
- In some adaptive streaming schemes, a similar file on the server translates client requests.

Functional Diagram of Client side Pull based Adaptive Streaming



Jitter (contd.)



Client Side Pull Features

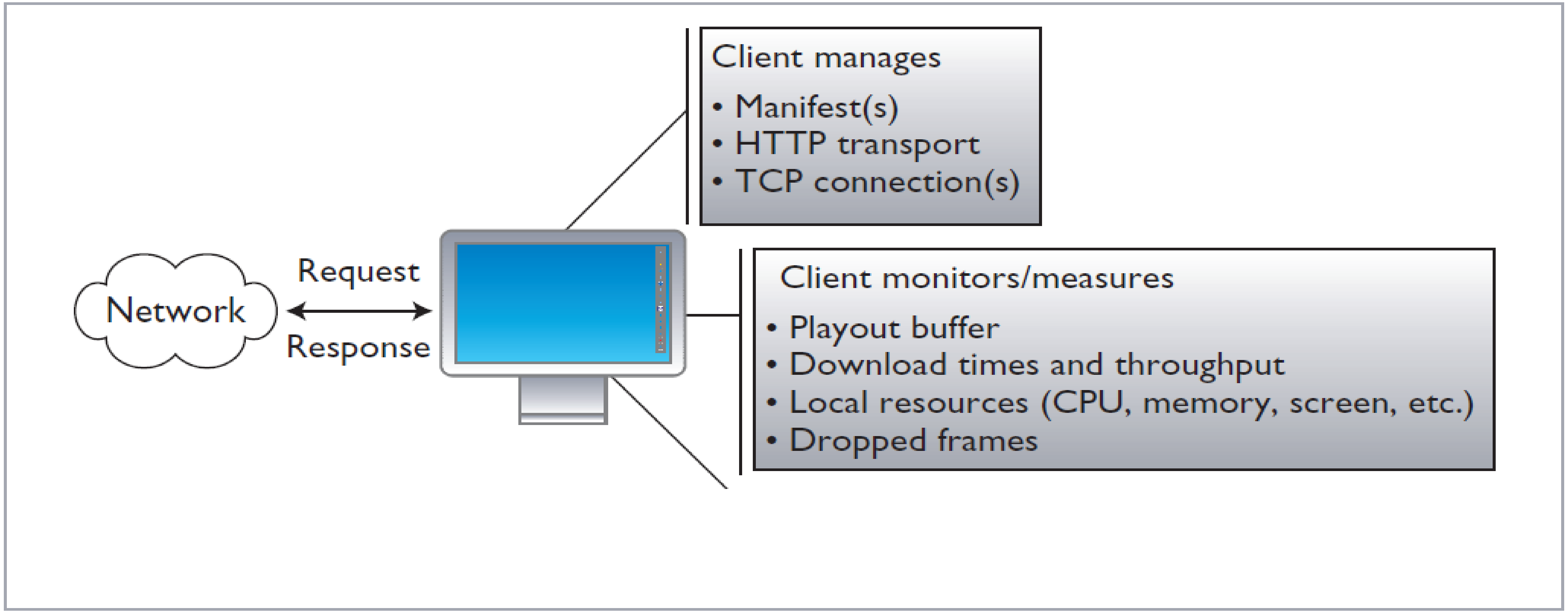
❑The client acquires media from fragments of a file over one or more connections according to the playout buffer state and other conditions

❑Playout buffer management is a basic function required on any adaptive streaming client,

❑It is used to select fragments from a file at a particular bitrate in response to buffer state and potentially other variables.

Typically, an adaptive streaming client keeps a playout buffer of several seconds (between five and 30).

Functional Diagram of Client side Pull based Adaptive Streaming



Client manages

- Manifest(s)
- HTTP transport
- TCP connection(s)

Client monitors/measures

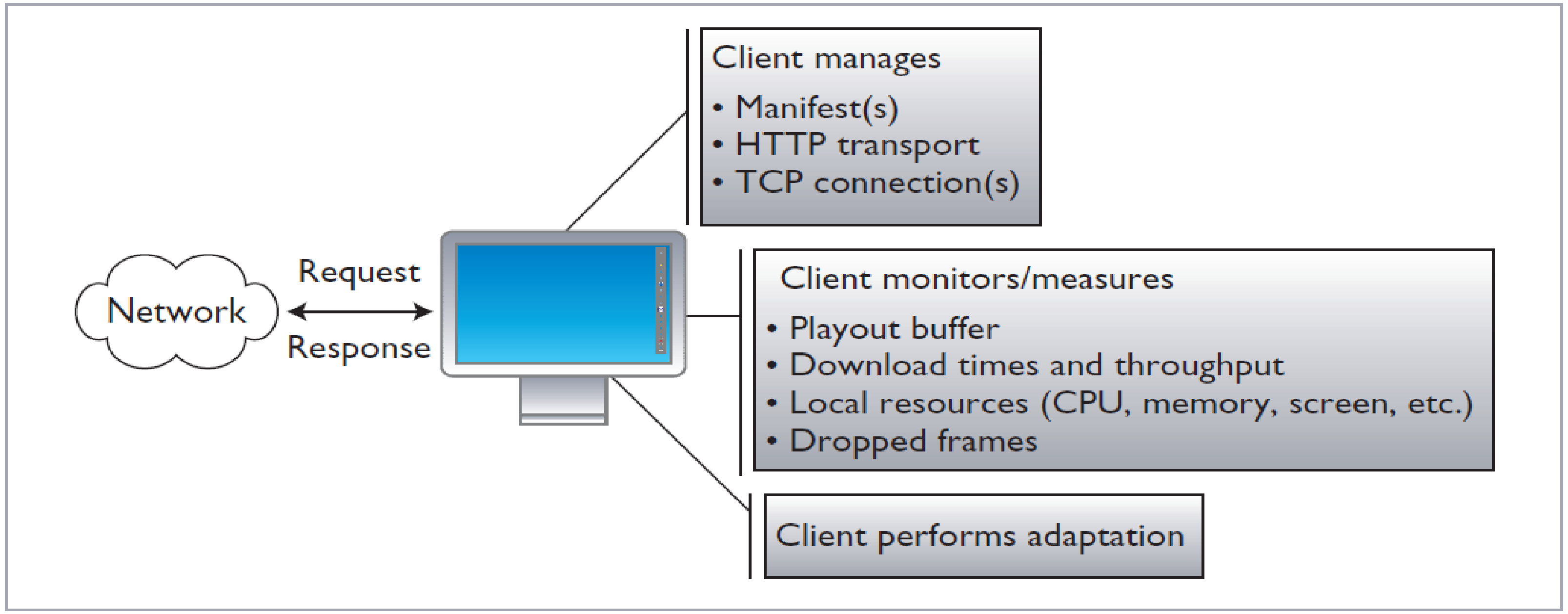
- Playout buffer
- Download times and throughput
- Local resources (CPU, memory, screen, etc.)
- Dropped frames

Client Side Pull Features

- ❑GET requests use a single TCP connection by default
- ❑ some adaptive streaming implementations support using multiple concurrent TCP connections for
- ❑requesting multiple fragments at the same time or for pulling audio and video in parallel.
- ❑The client is preconfigured to request a content at a certain bitrate, or profile, based on the result of network tests or a simple configuration script.
- ❑When a profile is selected and the client finds the URI associated with it in the manifest,
- ❑it establishes one or more connections to the server

Client Side Pull Features

- ❑ As the client monitors its buffer, it might choose to upshift to a higher-bitrate profile or downshift to a lower one, depending on how much the rate of video transport does or does not exceed the playout rate
- ❑ adaptive streaming products open a new connection when upshifting or downshifting to a new profile.



An alternative technology called Scalable Video Coding (SVC) lets clients choose media streams appropriate for underlying network conditions and their decoding capabilities.

Media Streaming Protocols

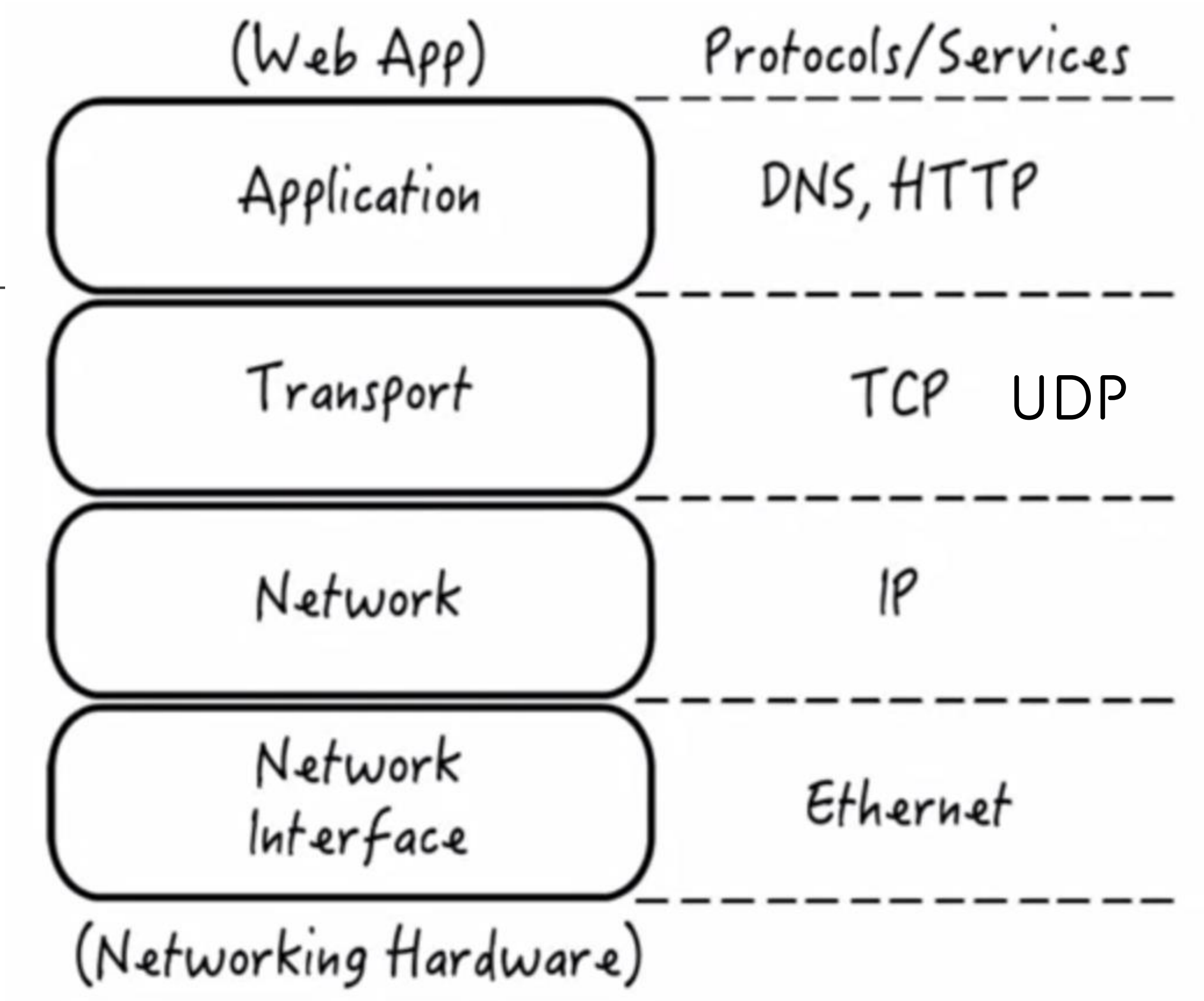
- Push Protocols
- once a server and a client establish a connection, the server streams packets to the client
- until the client stops or interrupts the session
- the server maintains a session state with the client and listens for commands from the client regarding session-state changes

Real-time Streaming Protocol (RTSP)

Push-based streaming protocols generally utilize Real-time Transport Protocol (RTP)

RTP runs on UDP

Protocol Stack



Push Protocols

- This lets the server push packets to the client at a bitrate that depends on an application-level client/server implementation rather than the underlying transport protocol
- This makes RTP a nice fit for low-latency and best effort media transmission.

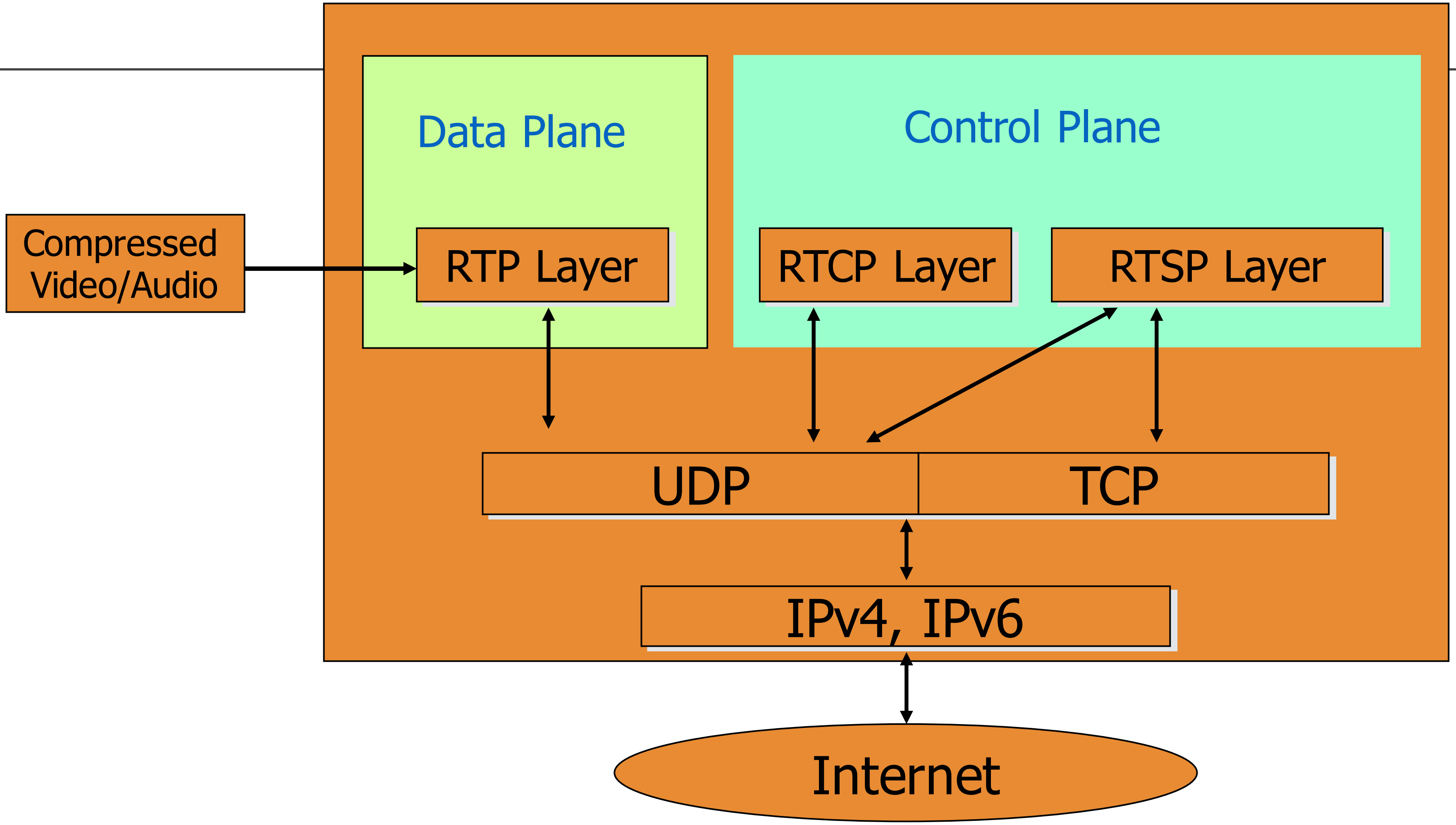
QoS Requirements

- Limited delay and jitter
- Jitter
  - fluctuation of end-to-end delay
  - It can change the original spacing between the multimedia packets
    - A frame should not be played as and when received
    - It should be buffered
- No support for congestion
  - the sender needs specific feedback from the receiver at the higher layer for responding to congestion

Motivation

- Need characteristics different from TCP but more than UDP
- Different coding schemes are used for a variety of multimedia applications
  - Need interoperability
  - Two independently developed RTP based applications should talk to each other
  - Instead of imposing a given voice/video encoding scheme for both the applications, RTP allows them to negotiate to fix the encoding technique.
- RTP enables the recipient of the packet stream to extract the timing relationships among the received packets
  - Need for synchronization among various media-voice and video at a conference
  - Identify packet loss and notify this to the application so that the application may deal with it appropriately
  - Need to indicate GoP

Protocol stack for media streaming

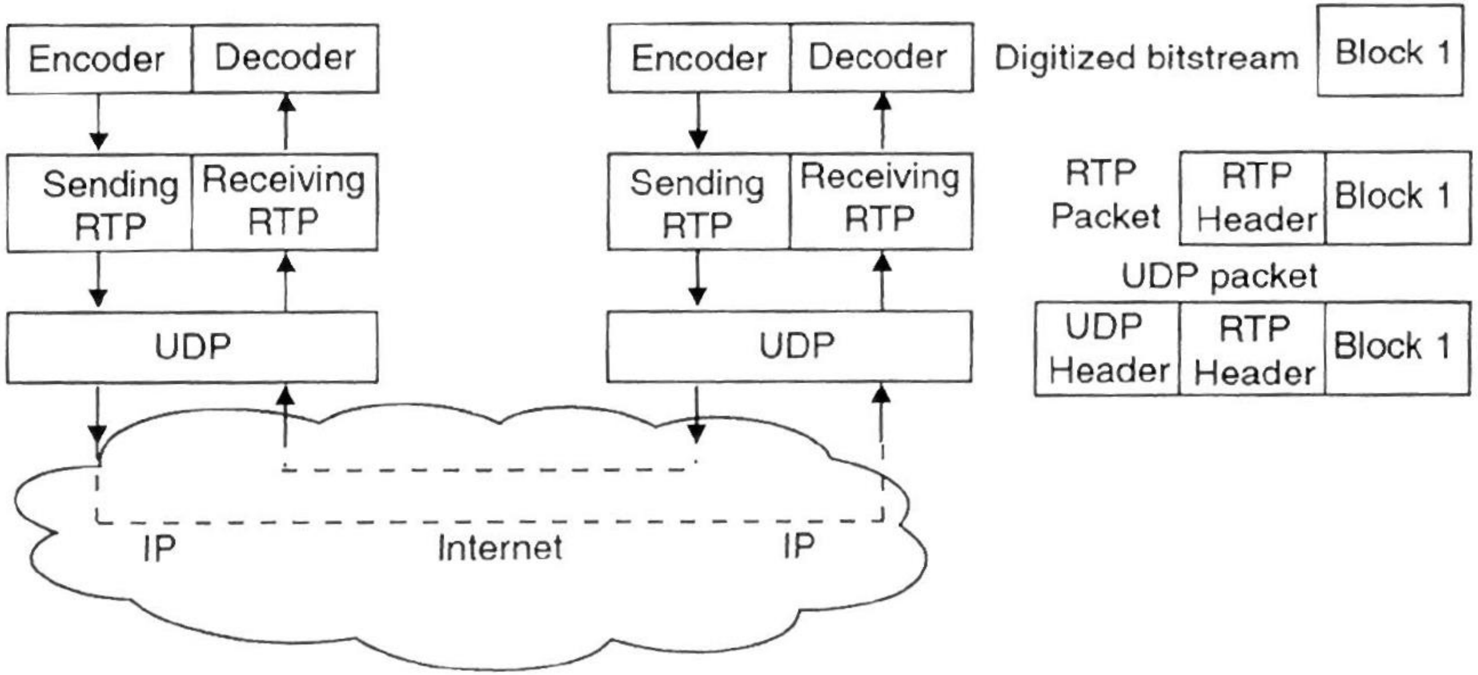


Real Time Protocol (RTP)

- RTP logically extends UDP
  - sits between UDP and application
  - end-to-end transport functions suitable for real-time audio/video applications over multicast and unicast network services
  - implemented as an application library
  - RTP uses a temporary even-numbered UDP port
  - The IETF standard for RTP defines two protocols-RTP and RTCP
  - RTP is used for the exchange of multimedia data
  - RTCP is the control part and is periodically used to obtain feedback regarding the quality of transmission

RTP

- What does it do?
  - Framing
  - Multiplexing
  - Synchronization
  - Feedback (RTCP)

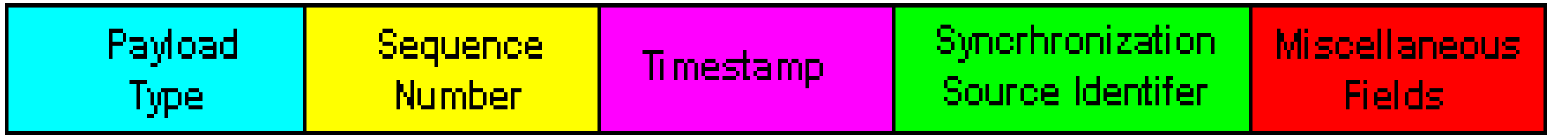


Real-time Protocol - RTP

- RTP is a data transfer protocol and RTCP is a control protocol.
- RTP is made following the architectural principle of Application Level Framing
  - Intelligence should be placed in the applications while the network should be kept simple
- For each class of applications, RTP defines a profile and one or more payload formats
- Besides sender and receiver, RTP defines two other roles-translator and mixer
- At a video conference, some participants may have lower access bandwidth
  - The translator could convert the stream to audio and video formats that require less bandwidth
- Mixers are used to combine multiple source streams into one
  - A video mixer combines images of individual persons to one video stream to simulate a group scene

RTP

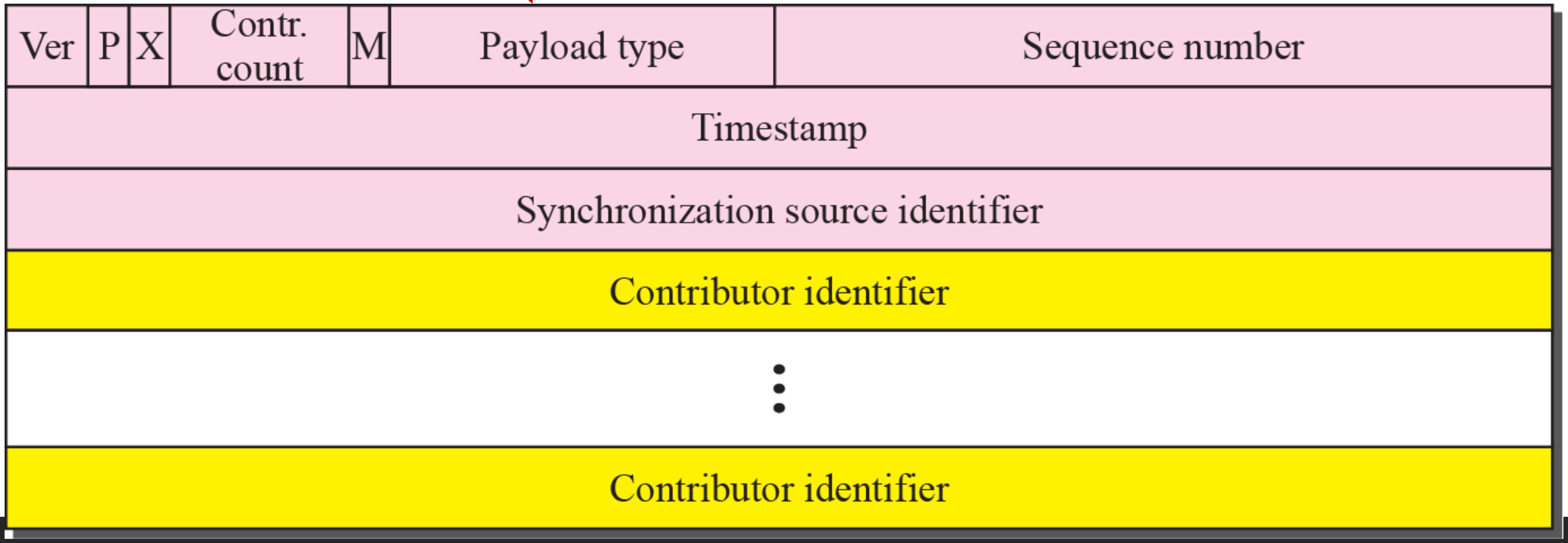
- RTP provides services for
  - **Payload type identification**: Identify which kind of information is being transmitted, RTP provides 128 possible different types of encoding; eg MPEG2 video, etc.
  - **Sequencing**: Reassemble the stream and detect packet loss.
    - Application intelligence decides
      - Ignore packet loss
      - Change encoding to cope with reduced bandwidth
  - **Timestamping**: Assure synchronization. Also used for jitter calculation
  - **Source identification:** Provide a means for the receiver to distinguish different sources.
    - It indicates where the data is combined
    - Or source of data if there is only one source



RTP Header

- **Payload Type**: 7 bits, providing 128 possible different types of encoding; eg PCM, MPEG2 video, etc.
- **Sequence Number**: 16 bits; used to detect packet loss
- The initial value could be set randomly

- **Timestamp**: 32 bytes; gives the sampling instant of the first audio/video byte in the packet; used to remove jitter introduced by the network
- **Synchronization Source identifier (SSRC)**: 32 bits; an id for the source of a stream; assigned randomly by the source



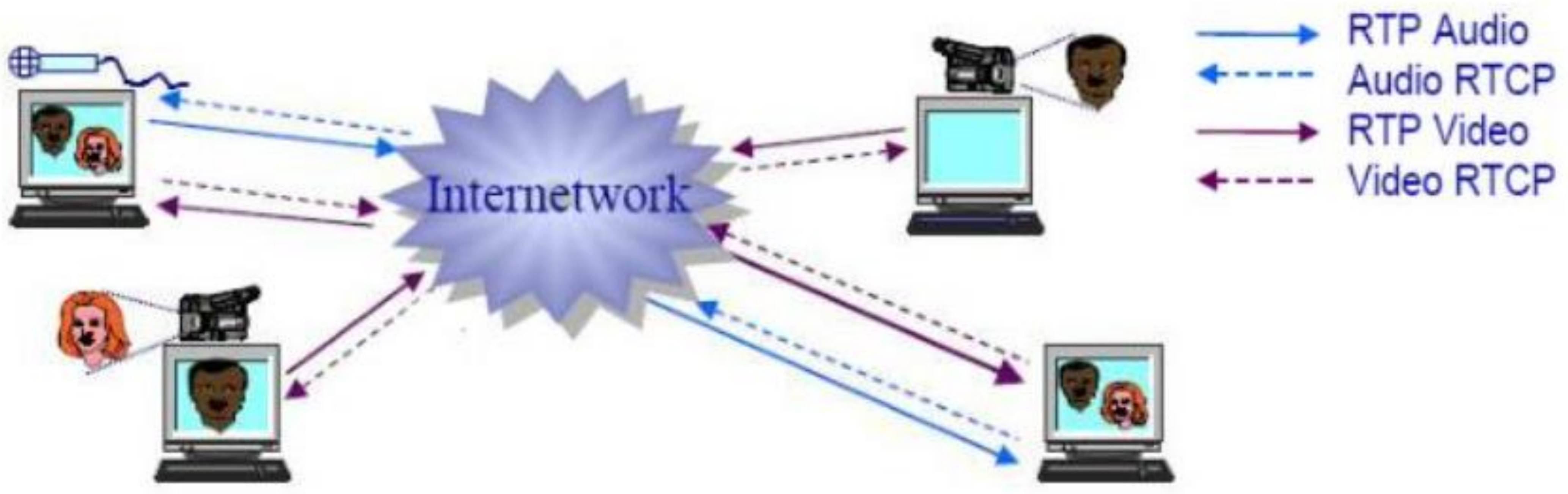
Timestamp vs. Sequence No

- Timestamps relate packets to real time
  - Timestamp value sampled from a media specific clock
- Sequence number relates packets to other packets
- Example of silent audio –
  - Consider audio data type
  - What is sent during silence?
    - Not sending anything
  - Why might this cause problems?
    - Other side needs to distinguish between loss and silence
  - Receiver uses timestamps and sequence no. to figure out what happened

RTP

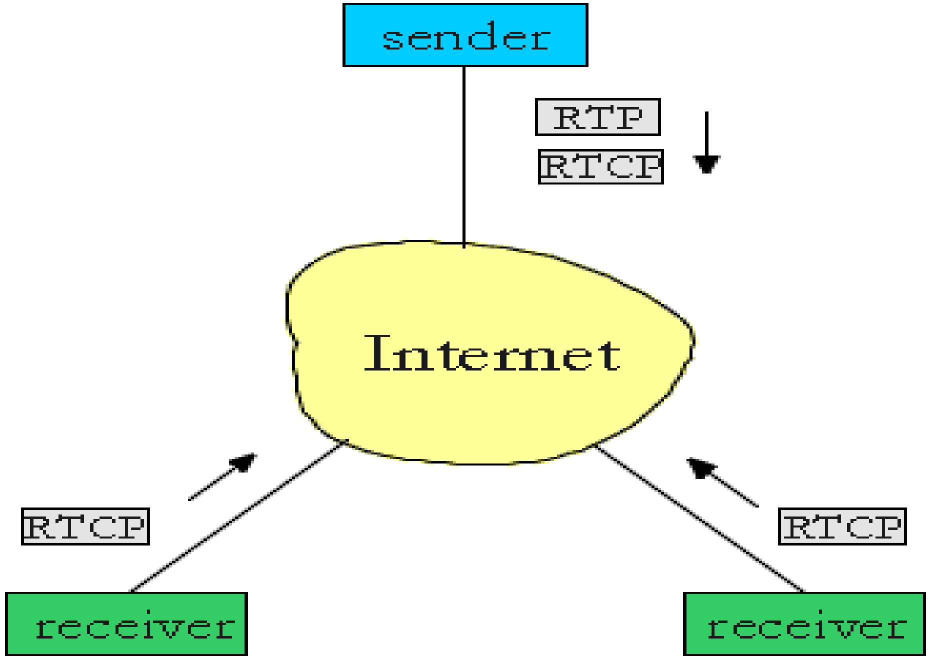
- RTP does not provide
  - Quality of service
  - Reliability in packet delivery
  - Security

RTP and RTCP



RTP Control Protocol (RTCP)

- Used in conjunction with RTP to exchange control information and reports between the sender and the receiver.
- Control connection is held over a different channel than the RTP.
- Uses an odd-numbered UDP port number that follows the port number selected for RTP.
- Reports can be *Receiver reception*, *Sender*, and *Source description.*
- Reports contain statistics such as the number of packets sent, number of packets lost, inter-arrival jitter
- Multiple RTCP packets can be concatenated by translators/mixers



RTP Control Protocol (RTCP)

- RTCP provides
  - **QoS Feedback**: In form of sender reports/receiver reports. Senders adjust transmission rate based on reports.
  - The compression ratio can be reduced (poor resolution) or improved (better resolution) depending on channel bandwidth
  - **Participant Identification**: Human-friendly source identification.
  - **Control Packets Scaling**: Typically, limit the RTCP bandwidth to 5% of the session bandwidth, divided between the sender reports (25%) and the receivers reports (75%)
  - **Minimal Session Control Information**: Advanced control functions must be implemented in a higher level protocol.
- Types of RTCP packets:
  - Sender report packet,
  - Receiver report packet,
  - Source Description RTCP Packet,
  - Goodbye RTCP Packet and
  - Application Specific RTCP packets.
- RTCP itself does not provide any flow encryption or authentication. SRTCP protocol can be used for that purpose.

- Five RTCP packets
  - SR sender reports

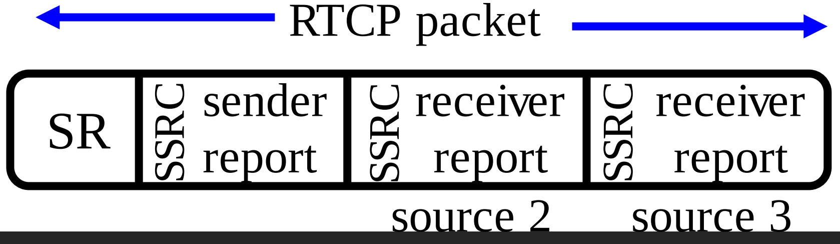
    tx and rx statistics from active senders
  - RR receiver reports

    rx statistics from other participants, or from

    active senders
  - SDES source description, e.g. name (including CNAME), email-address, telephone number and address of the owner or controller of the source
  - BYE explicit leave
  - APP application specific extensions

RTCP packets

- SR packet includes
  - SSRC of sender - identify source of data
  - NTP timestamp when report was sent
  - RTP timestamp corresponding RTP time
  - packet count - total number sent
  - octet count - total number sent
  - followed by zero or more receiver report…
  - example:



- RR packet includes
  - SSRC of source - identify the source to which this RR block pertains
  - fraction lost since previous RR (SR) sent (=int(256*lost/expected))
  - Cumulative # of packets lost -- long term loss
  - highest seq # received -- compare losses
  - interarrival jitter smoothed interpacket distortion
  - LSR time when last SR heard (timestamp from the last sender report received)
  - DLSR delay since last SR (delay since last sender report received)

Calculation of Round-trip delay

D = A-LSR-DLSR

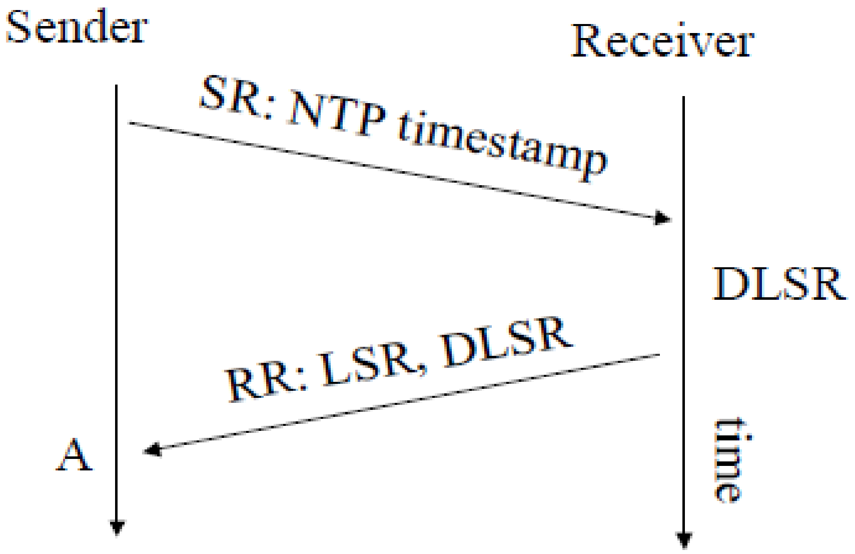
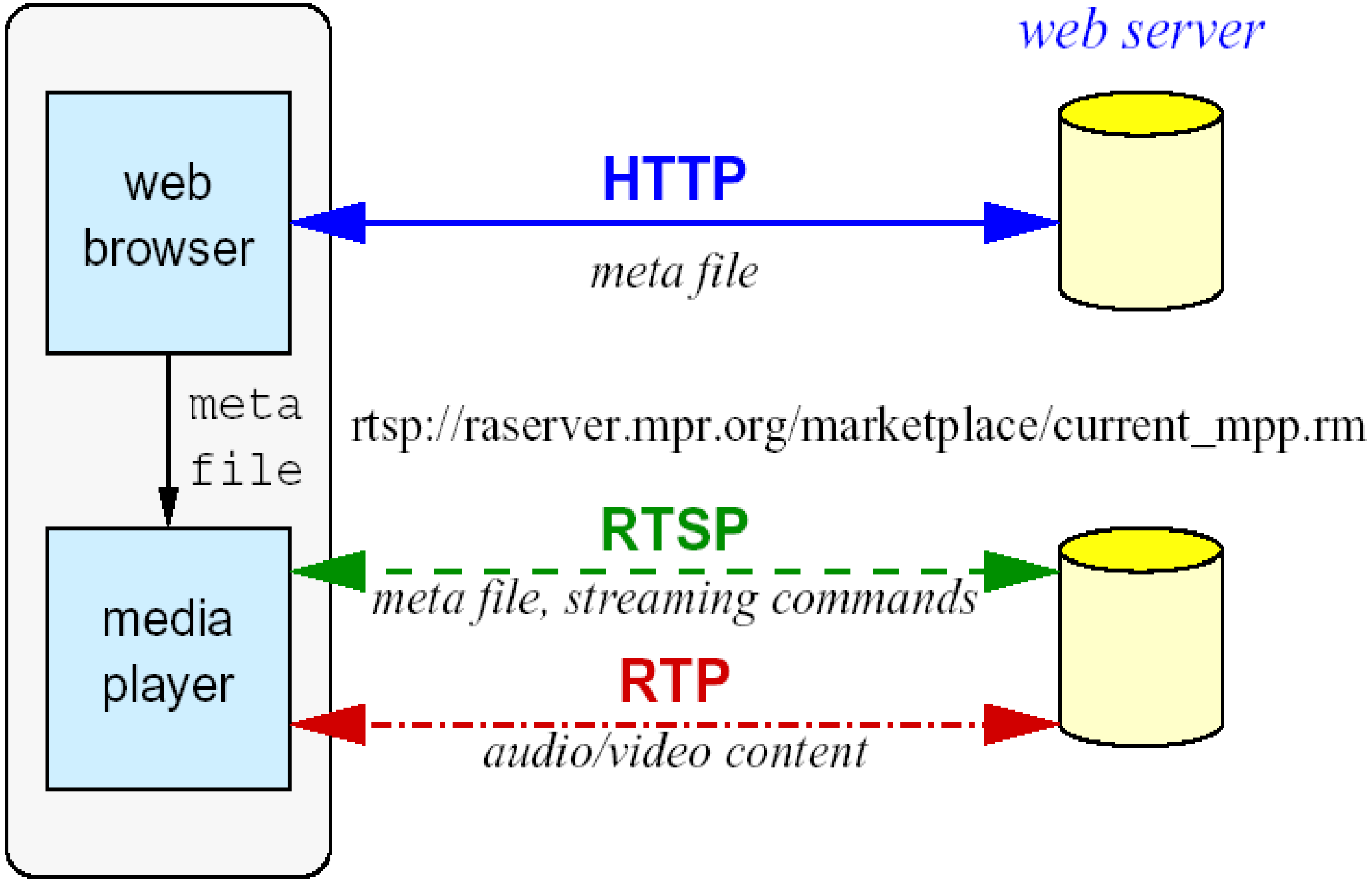


Figure 1. Calculation of round-trip delay.

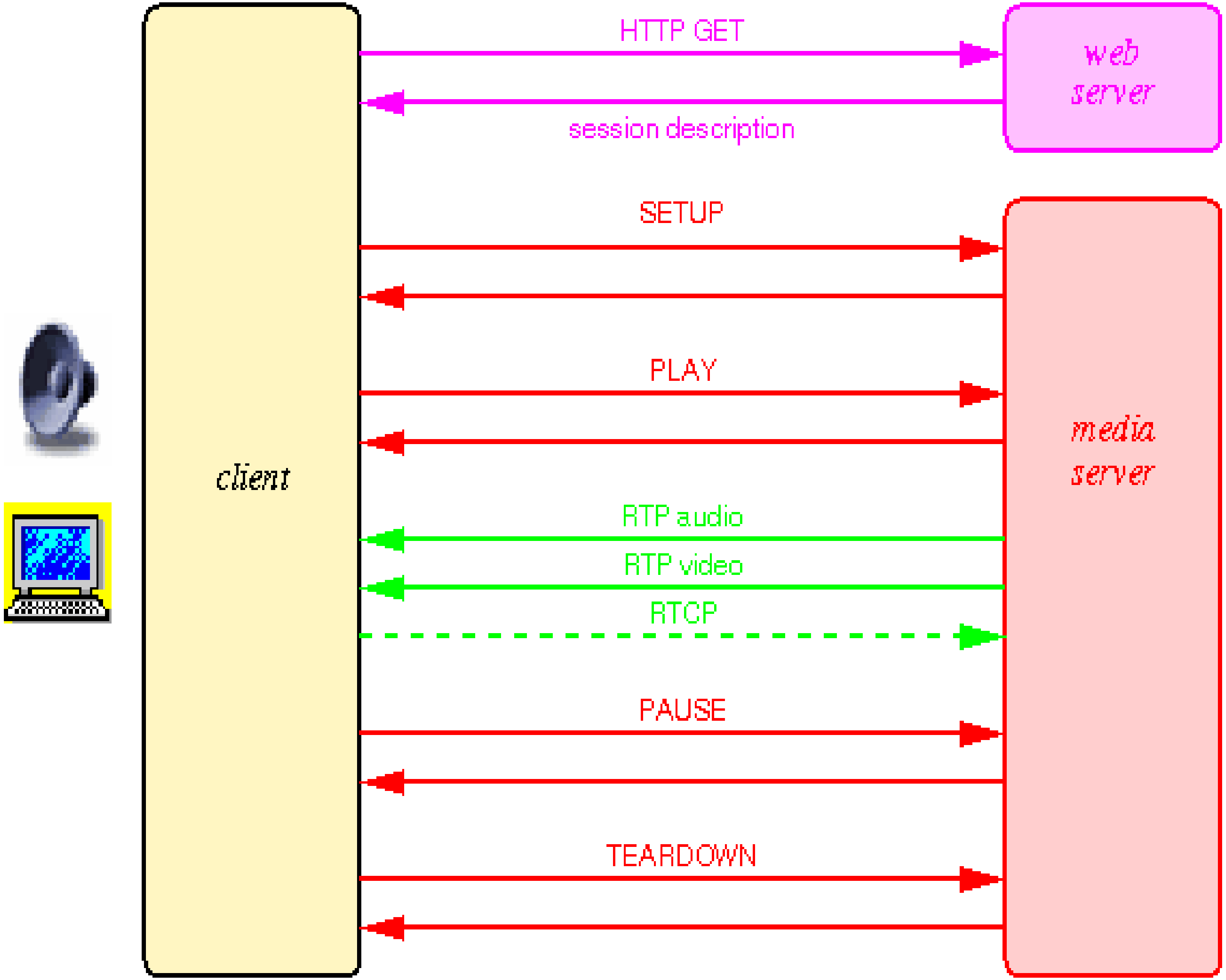
Real Time Streaming Protocol (RTSP)

- Supports VCR-like control operations
  - User controls operations like rewind, fast forward, pause, resume, etc…
  - It acts as a network remote control for multimedia servers
- Out-of-band protocol (uses two connections, one for control messages (Port 554) and one for media stream)
- RFC 2326 permits use of either TCP or UDP for the control messages connection, sometimes called the RTSP Channel
- Meta file is communicated to web browser which then launches the Player
- Player sets up an RTSP connection for control messages in addition to the connection for the streaming media
  - Retrieves requested media.
  - Adds media to an existing session.



It uses RTP as the underlying data delivery protocol

RTSP is a two-way protocol (in contrast RTP is a one-way protocol) used to send live or stored streams from the server to the client



HTTP GET

session description

web server

SETUP

PLAY

RTP audio

RTP video

RTCP

PAUSE

TEARDOWN

client

media server

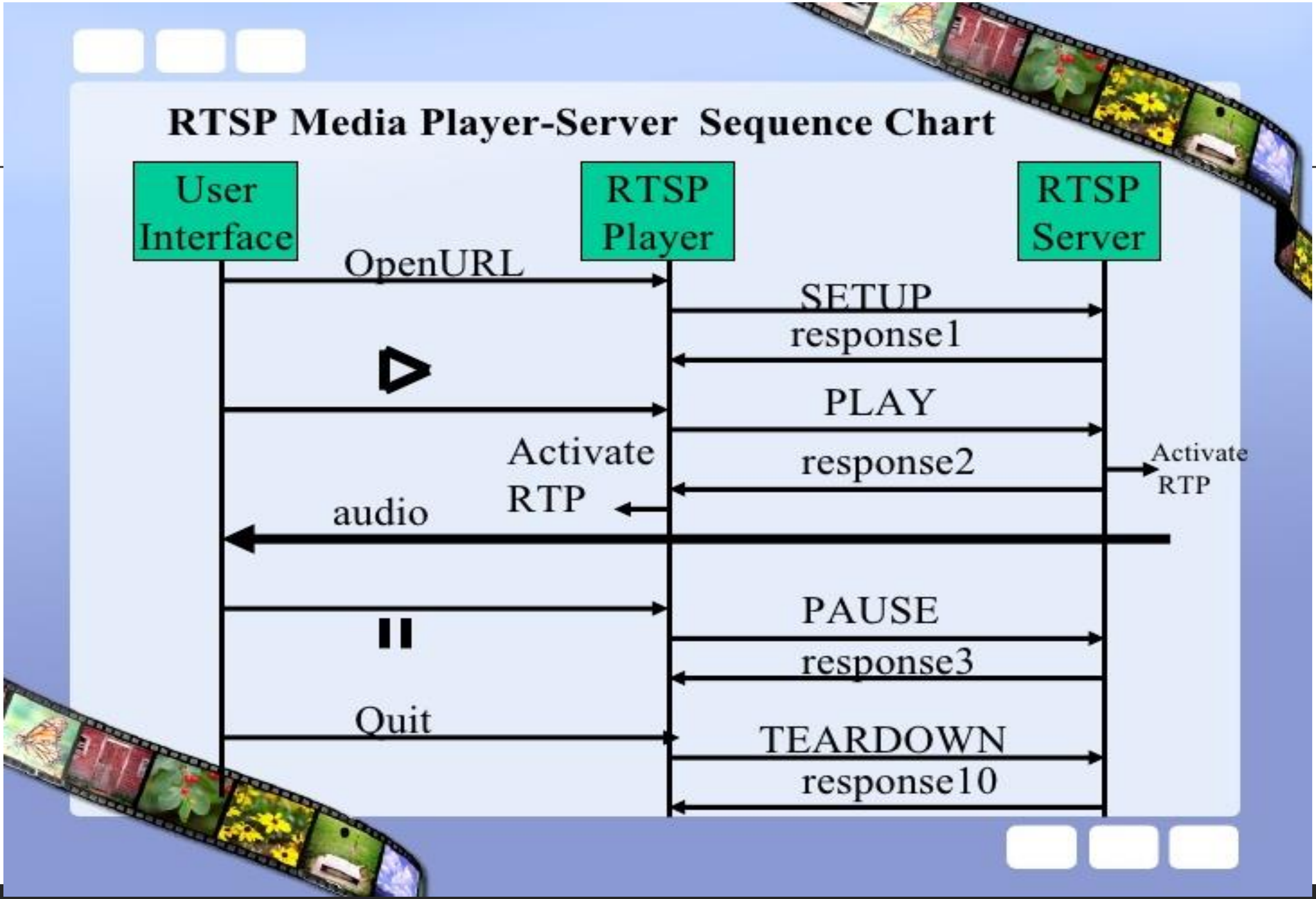
RTSP Protocol design

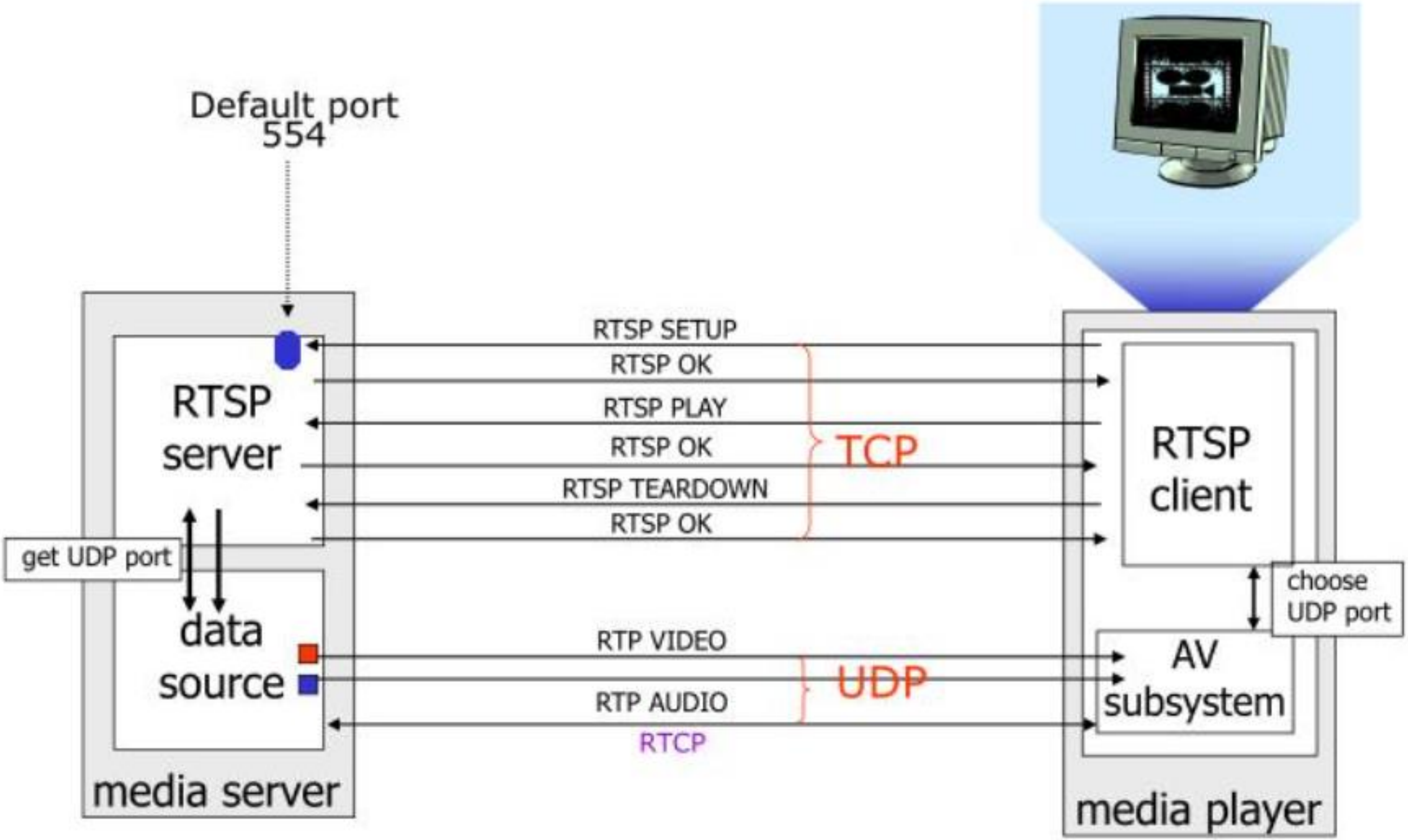
- text-based protocol
- transport protocol independent
  - chooses the optimum delivery channel to the client. For instance, if UDP cannot be used (some corporate firewalls will not pass UDP), the streaming server has to offer a choice of delivery protocols – multicast UDP or TCP to suit different clients.
- supports any session description (sdp, xml, etc.)
- similar design as HTTP with some differences
  - e.g. both the client and the server can issue requests during interaction
  - server maintains a « session state » (HTTP is a stateless protocol)
- data carried out-of-band
- works either with unicast or multicast

RTSP Methods

- Major methods
  - SETUP: server allocates resources for a stream and starts an RTSP session
  - PLAY: starts data tx on a stream
  - PAUSE: temporarily halts a stream
  - TEARDOWN: free resources of the stream, no RTSP session on server any more
- Additional methods
  - OPTIONS: get available methods
  - ANNOUNCE: change description of media object
  - DESCRIBE: get low level description of media object
  - RECORD: server starts recording a stream
  - REDIRECT: redirect client to new server
  - SET_PARAMETER: device or encoding control

RTSP Media Server Sequence Diagram




Default port
554
RTSP
server
get UDP port
data
source
media server
RTSP SETUP
RTSP OK
RTSP PLAY
RTSP OK
RTSP TEARDOWN
RTSP OK
TCP
RTP VIDEO
RTP AUDIO
UDP
RTCP
RTSP
client
choose
UDP port
AV
subsystem
media player

Conclusion

- ❑_Spring supports real time multimedia content delivery

https://www.youtube.com/watch?v=OmaF0WkBH2g

SPRING I/O 2022

THE CONFERENCE

https://2022.springio.net/