ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Multimedia Technology

Section # 19: Streaming

Instructor: George Xylomenos

Department: Informatics

Contents

- Classic media streaming
- RTSP
- Adaptive HTTP streaming
- MPEG DASH

Classic media streaming

Class: Multimedia Technology, **Section # 19:** Streaming

Instructor: George Xylomenos, **Department:** Informatics

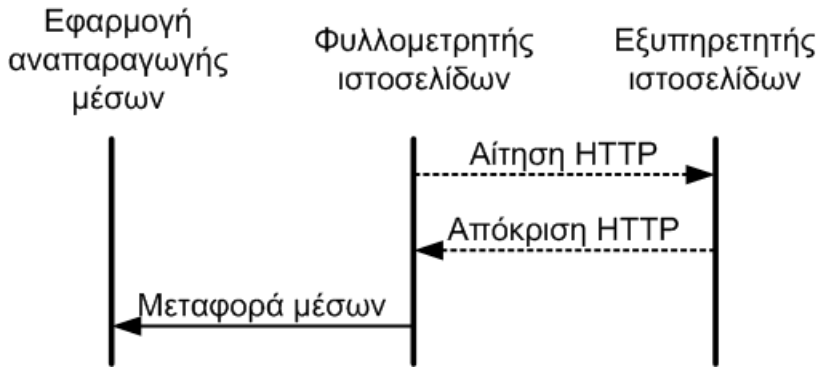
Streaming components (1 of 2)

- Media streaming
  - Media playback in parallel with media reception
    - As opposed to downloading the entire stream first
  - Can be implemented with various protocols
    - Originally RTP over UDP/IP
- Media server
  - Hosts the media files
  - Corresponds with the media player
  - Exchanges control and sync info

Streaming components (2 of 2)

- Web server
  - First point of contact with the user
  - Directs the user to the media
- Media player
  - Decompresses data
  - Buffers data to hide jitter
  - Recovers from losses (retransmission or FEC)
  - May be embedded in the web browser

From web to media server (1 of 4)

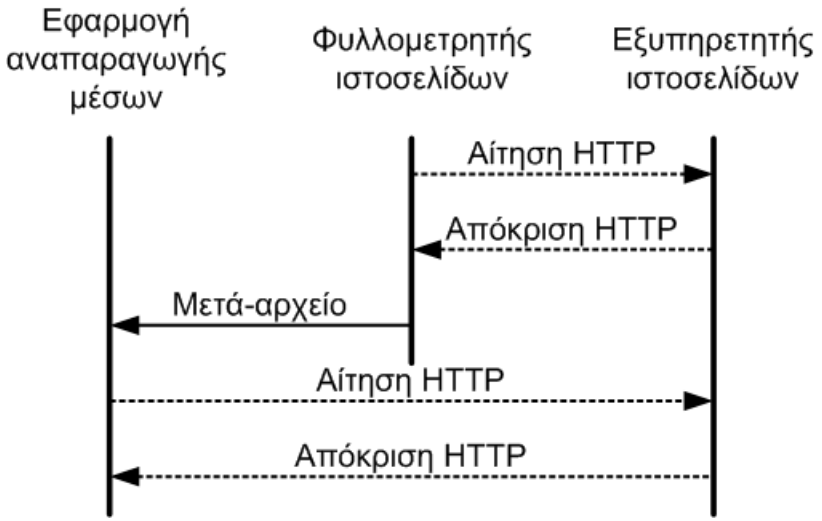


- Getting media from a web page
  - Treats media as any other file
  - Media stored on the web server
- HTTP-based
  - Requests for pages and objects
  - Responses with pages and objects

From web to media server (2 of 4)

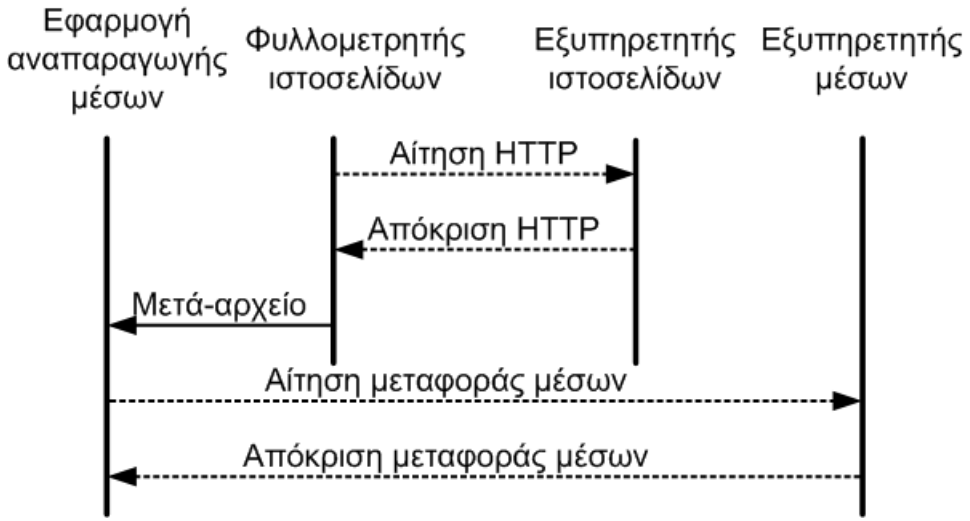
- Communication via web browser
  - Media player does not directly talk to server
  - High delay until media get to the player
  - Use of inappropriate protocol
    - Web uses HTTP over TCP
    - TCP is fully reliable
    - Transmission rate is variable
    - No way to bypass TCP behavior

From web to media server (3 of 4)



- Communication with media player
  - Metafile indicates type and address of media
    - Passed from browser to media player
    - The player can now bypass the web browser
  - Data still exchanged over HTTP and TCP

From web to media server (4 of 4)



- Communication with media server
  - The web server only hosts the metafile
  - The media player talks directly to the server
    - Using info from the metafile
  - Can switch to more appropriate protocol

RTSP

Class: Multimedia Technology, **Section # 19:** Streaming

Instructor: George Xylomenos, **Department:** Informatics

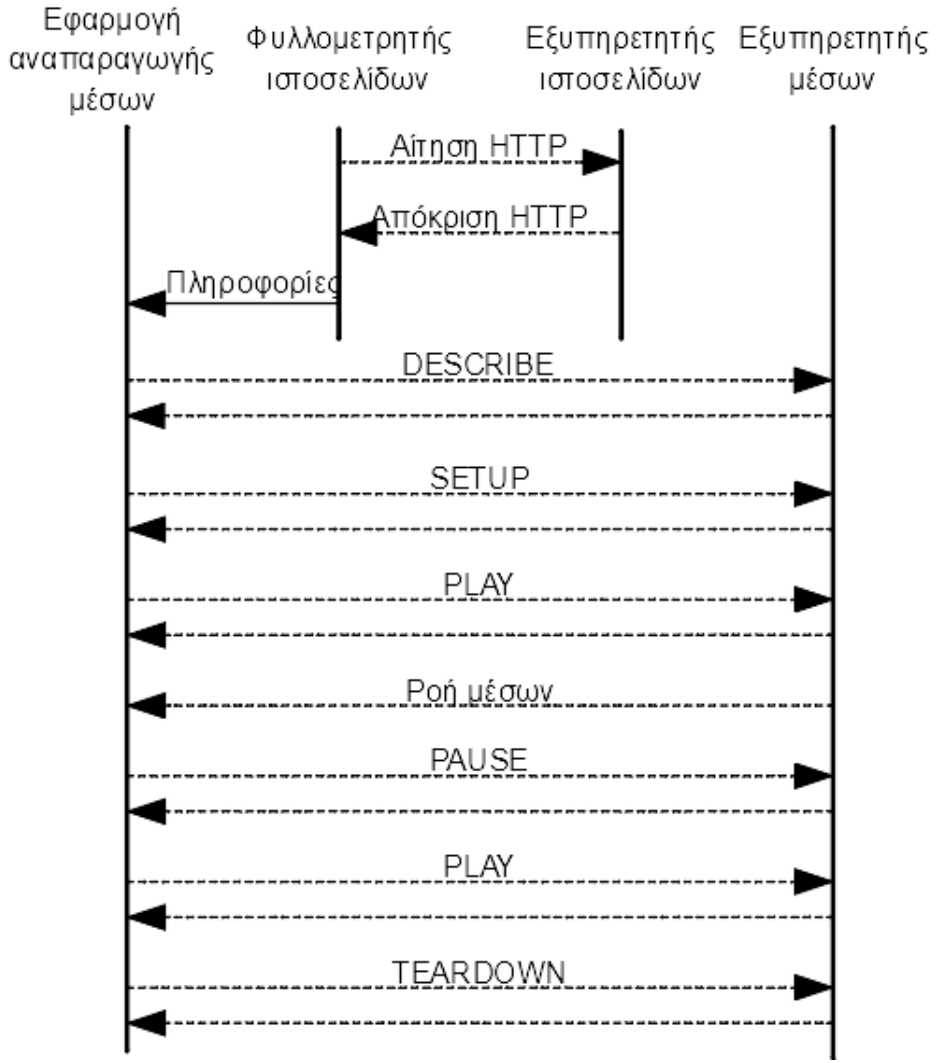
RTSP (1 of 5)

- Interaction during media streaming
  - Pause and resume playback
  - Move backwards or forwards
- The Real Time Streaming Protocol (RTSP)
  - RTSP is a control protocol
    - Used between media player and server
  - Media are streamed separately
    - For example, with RTP over UDP/IP

RTSP (2 of 5)

- We start with media stream info
  - A web page includes media links
  - These are URLs in the form rtsp://w.x.y.z
  - RTSP is then used with this URL
- Media stream control
  - RTSP is transmitted “out of band”
    - Separate port used for RTSP (TCP/UDP 554)
  - Media are transmitted “in band”
    - Usually via RTP/UDP/IP

RTSP (3 of 5)


Εφαρμογή αναπαραγωγής μέσων
Φυλλομετρητής ιστοσελίδων
Εξυπηρετητής ιστοσελίδων
Εξυπηρετητής μέσων
Αίτηση HTTP
Απόκριση HTTP
Πληροφορίες
DESCRIBE
SETUP
PLAY
Ροή μέσων
PAUSE
PLAY
TEARDOWN

RTSP (4 of 5)

- RTSP usage
  - DESCRIBE: Describes media in the stream
    - Response encapsulated in SDP
    - Can include various media and variants
  - SETUP: Session establishment
    - Returns a session ID
    - Messages are numbered within the session
    - The server maintains state for the client

RTSP (5 of 5)

- RTSP usage
  - PLAY: Starts media flow
    - Indicates which variant we want
    - Indicates starting point
  - PAUSE: Temporary media pause
    - We resume with PLAY
  - TEARDOWN: Session termination
    - Server removes state about client

Adaptive HTTP streaming

Class: Multimedia Technology, **Section # 19:** Streaming

Instructor: George Xylomenos, **Department:** Informatics

Issues with classic streaming

- Disadvantages of RTSP/RTP/RTCP
  - Requires media server in addition to web server
    - And the media server is not stateless
  - Distribution is not easy
    - Firewalls block UDP by default
    - There is no state as in TCP
  - Incompatible with CDNs and Web caching
    - Which are widely used

Issues with HTTP streaming

- Classic HTTP-based streaming
  - We can do progressive downloads
    - Using HTTP GET with byte ranges
  - Compatible with existing servers and CDNs
  - But it is very inflexible
    - TCP can get stuck on a lost packet
  - Cannot work with live content
    - The media file must exist before we start

Adaptive HTTP streaming (1 of 4)

- Solution: Adaptive HTTP streaming
  - Relies on multiple media variants
    - For devices with different resolutions
    - For devices with different network capabilities
  - All variants described in a metafile
  - Dynamically changes media variant
    - To adapt to current network conditions
  - But what if TCP gets stuck?

Adaptive HTTP streaming (2 of 4)

- Media broken in smaller pieces
  - One piece for each variant
  - The pieces have a fixed duration
    - Example: 2 seconds
  - Pieces are named systematically
    - Example: stream0_0, stream0_1, …
  - We ask for the correct next piece
    - We can switch every 2 seconds

Adaptive HTTP streaming (3 of 4)

- Bypasses TCP policies
  - We ask for new pieces all the time
  - If we get stuck, we can create a new connection
    - And ask for the appropriate variant
- But: tons of files and HTTP GETs
  - One file for each 2 second variant
    - The files can be virtual
  - Client sends new GETs every 2 seconds

Adaptive HTTP streaming (4 of 4)

- Operation of adaptive HTTP streaming
  - The client first downloads a metafile
    - Describes the available media variants
    - Includes parameters and naming scheme
  - The it chooses a variant and starts fetching
    - Can start with lowest quality to test the network
    - Adapts the quality depending on conditions
  - The server is not involved at all!
    - Only the client monitors the connection state

Working with CDNs (1 of 2)

- CDN: Content Distribution Network
  - Set of co-operating web servers (or caches)
  - Strategically located around the network
  - Content actively pushed to CDN servers
- CDN goals
  - Reduced latency (use a nearby server)
  - Load balancing (many servers)
  - Traffic reduction (many clients)

Working with CDNs (2 of 2)

- CDN and adaptive streaming
  - Can use different strategies
  - Different pieces can come from different servers
    - Reduces latency visible to user
  - Different variants located in different servers
    - Depending on the network
  - Different content in different servers
    - Depending on language and area

Working with web caches

- Pieces can be cached
  - Each piece is an independent file
    - It has (some) unique name
    - It can be fetched independently
  - A web proxy can easily cache it
    - To serve subsequent requests
    - No need to store all pieces
    - No need to store all variants

Proprietary solutions (1 of 3)

- Microsoft Smooth Streaming (Silverlight)
  - Metafile listing the variants
  - Single file per variant
    - Coded with H.264+AAC
  - Dynamic fragmentation in 2 sec pieces
    - HTTP GET with byte ranges to ask for right piece
    - No need to know different file names
    - Allows live streaming (file gets extended)

Proprietary solutions (2 of 3)

- Adobe HTTP Dynamic Streaming (Flash)
  - One file per variant coded with H.264+AAC
  - Uses MP4 fragment format
    - File broken down into segments
    - Segments broken down into fragments
    - We ask for each fragment separately
    - Fragments can be 2-5 sec
  - Metafile maps fragments to time

Proprietary solutions (3 of 3)

- Apple HTTP Live Streaming (iOS/Android)
  - One file per piece per variant
    - Segments are 10 sec
    - Uses MPEG-2 TS file format
    - Coded with H.264+AAC
  - Metafile describes pieces
    - For live content the metafile is updated
    - Delays mandate the larger piece length

MPEG-DASH

Class: Multimedia Technology, **Section # 19:** Streaming

Instructor: George Xylomenos, **Department:** Informatics

What is MPEG-DASH?

- DASH: Direct Adaptive Streaming over HTTP
  - Created by the MPEG consortium
  - Adopted by 3GPP for mobiles
  - Supported by the DASH Industry Forum
  - Attempt to standardize an existing practice
    - Use a single encoding instead of three
    - Prevent incompatibilities
    - Combine existing approaches

Capabilities (1 of 3)

- Two file formats allowed
  - MPEG-4 (ISO): Compatible with Adobe/MS
  - MPEG-2 TS: Compatible Apple
  - Allows reusing existing media
    - Just needs a new metafile!
- Many encodings possible
  - Encoding with H.264+AAC for compatibility
  - Adds MPEG-2, MPEG-4, even H.265

Capabilities (2 of 3)

- Pieces can be 1-20 sec
  - Interleaved audio/video
  - Independent audio/video
    - Allows different adaptation strategies
    - Works better with multilingual content
- Two metafile types
  - Description file for all variants
  - Initialization file for each variant

Capabilities (3 of 3)

- Many ways to provide content
  - Different file formats and codecs
  - Different number and type of variants
  - Different piece durations
  - Different media multiplexing options
- Client has significant flexibility
  - Policy for requesting pieces
  - Policy for changing variants

Profiles (1 of 3)

- Profiles in MPEG DASH
  - Similar concept to MPEG profiles
    - Which capabilities are allowed
    - Each profile is a compatibility point
    - Different apps need different profiles
  - The media player supports some profiles
  - Content explicitly lists its profile
    - The same content can be available in multiple profiles

Profiles (2 of 3)

- ISO Base media file format On Demand
  - Stored (not live) content
  - MPEG-4 (ISO) file format with two options
  - Single file per variant
    - Relies on HTTP GET with byte ranges
  - Independent files per piece
    - Each file has its own name
  - Metafile maps pieces to time

Profiles (3 of 3)

- ISO Base media file format Live
  - Dynamic creation of pieces for live content
  - Metafile describes a naming scheme
    - How pieces are named
    - How long pieces last
- MPEG-2 main, simple
  - Similar to ISO but with MPEG-2 TS
- More profiles exist

Hierarchical structure (1 of 3)

- Each metafile describes one “presentation”
  - MPD type (Media Presentation Description)
  - Can have different MPDs for the same content
    - Different piece durations
    - Different profiles
- A presentation is divided into periods
  - Parts of the program and/or advertisements
  - Allows dynamic insertion (for advertisements)

Hierarchical structure (2 of 3)

- Each period has adaptation sets
  - Adaptation sets group related media
  - Example: video, English audio, Greek audio
  - Each set has a single encoding
  - Video: H.264 or H.265, Audio: 2 or 5 channels
- The adaptation set is chosen by the user
  - Example: Greek audio, 2 channels

Hierarchical structure (3 of 3)

- Each adaptation set has representations
  - Different variants of the same content
    - Video in different bit rates
  - The client switches between representations
    - Depending on network conditions
- Each representation consists of pieces
  - Initialization part at the beginning
  - Media pieces (lots!)

MPD files (1 of 4)

- MPDs describe the presentation
  - XML scheme used for description
  - Can be updated for live content
    - The new MPD must extend (not change) the old one
- MPD: presentation attributes
  - Profile used
  - Presentation duration
  - Minimum buffer size required

MPD files (2 of 4)

- BaseURL: main content prefix
- Period: when this period starts
- AdaptationSet: properties of set
  - Can have different sets per period
  - Shows what we can choose from
  - Interleaved or independent media streams
  - ContentComponent: many components in the set

MPD files (3 of 4)

- Representation: one variant of the set
  - id: identifier
  - codecs: encoders used (many for interleaved)
  - mimeType: media type according to MIME
  - bandwidth: bitrate expected
  - Type-specific properties
    - width and height: video resolution
    - numChannels: audio channels

MPD files (4 of 4)

- Representation: one variant of the set
  - SegmentBase: initialization data
    - Independent files or pieces of a large file
  - SegmentList: first the duration
    - Then an explicit list of names
  - SegmentUrl: single file name
  - SegmentTemplate: template for names

What MPEG-DASH is not (1 of 3)

- It does not constrain the provider
  - The provider chooses the parameters
    - Can have one quality per resolution
    - Or many qualities per resolution
  - The provider chooses the file format
    - Can have one file per variant
    - Can have one file per piece
  - The server is a regular web server

What MPEG-DASH is not (2 of 3)

- It does not constrain the client
  - The client chooses what and when to ask
    - Estimates bandwidth
    - Switches resolution, quality or both
  - The client can be anything
    - Independent media player
    - JavaScript code in the browser
  - All state resides at the client

What MPEG-DASH is not (3 of 3)

- No changes to the web server
  - The server just stores files
  - Only need to support HTTP methods
  - There are no “MPEG-DASH servers”
- No changes to the CDNs
  - Provider chooses how to push content
  - DNS used to redirect the client
    - Often by using small TTL
    - Which puts a lot of load on DNS

End of Section # 19

Class: Multimedia Technology, **Section # 19:** Streaming

Instructor: George Xylomenos, **Department:** Informatics