Connection-Oriented Streaming of Multimedia Content

Mikołaj Leszczuk, Jakub Nawała

# Introduction

Creating multimedia communication systems [[1]](#298zeudmkdqd), including digital video libraries, access to which is available by using a telecommunication network, requires the implementation of mechanisms for streaming video sequences. Streaming is called a synchronous transfer of (multimedia) content from the server to the client using a telecommunication network. In this exercise, you will be testing the operation of streaming video servers.

# Objective of this experiment

The objective is to learn the techniques related to the streaming of video sequences from a server to client machines. Video streaming software will be used for this purpose. During the exercises, we will test an exemplary video streaming server.

# Requirements

To best grasp the content of this exercise, you should be familiar with the basics of streaming. What is more, you need VLC media player to be installed on your machine.

## Knowledge

You should have a general understanding of the following areas:

* Why are the servers used for streaming?
* HTTP-based streaming technologies (mainly HTTP Live Streaming [[28]](https://developer.apple.com/documentation/http_live_streaming)).
* Protocols used in media streaming: IP [[5]](#2qv7kxhn80vt), TCP [[7]](#abfuyr15u4e9), UDP [[6]](#rszcsjsiuifn), RTP [[10]](#n0uc7kbpun5z), RTSP [[11]](#ft6fkcww0tx4).

## Software

Depending on the lab where classes are held, the appropriate software (for sending, receiving, and displaying multimedia streams) may or may not be installed on your computer. Please follow the instructions to install all necessary software components.

### Check if VLC is already installed

We will be using a free and open-source VLC media player. Please check if it is already installed on your machine. [Figure 1](#vhcz3qi15gur) shows its icon.

|  |
| --- |
|  |
| **Fig. 1.** The VLC media player icon |

If the software is missing, please follow the instructions from the next section.

### Download and install VLC

To install the VLC media player, please follow the instructions given on [the project’s main web page](https://www.videolan.org) [[15]](#apt9kgqppsa2).

## IP address

Please check and **write down** the network address (i.e., IP address) of your machine. Depending on the operating system, the method of checking your IP is different. When using Linux/Mac OS, please start the terminal and issue ifconfig. If on Windows, start the command prompt and type ipconfig /all.

# Streaming content using VLC

VideoLAN Client (aka VLC) is a software developed by students of the Ecole Centrale Paris and distributed under the GNU’s General Public License (GPL). It is being developed by more than 20 developers across the globe. VideoLAN is a complete software solution for video streaming. VLC can be used as a server to stream MPEG-1, MPEG-2, MPEG-4, H.264, DVD materials. On top of that, it can also stream a live video (e.g., webcam images) to a unicast or multicast network. VLC may as well serve as a receiver, decoder, and player of MPEG streams. Moreover, it works under multiple operating systems. [Figure 2](#94desrdirfqu) shows how VLC can be used when implementing a multimedia streaming solution.

|  |
| --- |
|  |
| **Fig. 2.** A diagram showing how VLC can be used for streaming solutions |

## Streaming from the network

The stream, instead of being a static local file, can be retrieved from Wide Area Network (WAN). For example, the client computer can pick a stream from e.g., Vimeo or YouTube. For this purpose, on the reception side (i.e., on your computer), when selecting the source, select the “Open Network…”[[1]](#footnote-2) item (see [Figure 3](#9uzb3wrcxz9q)) and enter the appropriate e.g., Vimeo or YouTube address, for example:

|  |
| --- |
|  |
| **Fig. 3.** An exemplary URL of a network stream (shown in the VLC’s “Open Source” dialog”) |

Note 1: The source address specified in Figure 3 is just an example. Feel free to paste the address of your own choice.

Note 2: Sometimes you can experience the following error:” VLC is unable to open the MRL […] check the log for details”. What causes the problem? YouTube transmission protocols change over time, and these changes sometimes break the YouTube module embedded in VLC. Whenever it happens, try to choose another video or video provider.

## HTTP streaming file

VLC can be used as an HTTP streaming server as well.

### On the HTTP server side (computer sending stream)

On the HTTP server side (the computer sending the stream), select the “File” menu, then “Streaming/Exporting Wizard…”[[2]](#footnote-3) and then “Stream on the network”, and “Next”.

To start streaming in the window “Choose input”, choose “Select a stream”.

In the open media dialog box, you should indicate the multimedia content stored on a hard disk.

|  |
| --- |
|  |
| **Fig. 4.** A VLC’s “Streaming/Transcoding” wizard |

In the “Streaming” (see [Figure 4](#lfnum9skmizg)) dialog, in the “Streaming method” pane, select “HTTP”. Do not choose the “Destination” address field.

The Wizard sets the default port – 8080, the same port for audio and video. Other options (on the remaining windows) should be left unchanged (simply press “Next” repeatedly).

After selecting “Finish”, the main window will appear, and after clicking the play button, VLC Media Player starts to stream!

If your operating system firewall wants to block the transmission, please unlock it. At this point, we have prepared the player to send the media stream at a given address.

### On the HTTP client side (the machine receiving the stream)

From the “File” drop down the menu, select “Open Network”[[3]](#footnote-4), then enter the “URL” (see [Figure 5](#67tt51sc2fgl)).

|  |
| --- |
|  |
| **Fig. 5.** VLC’s “Open Source” dialog with the address of a streaming server is given |

Select the address to the one belonging to the streaming server. In the port part of the URL (8080 in [Figure 5](#67tt51sc2fgl)), please enter the same port as the one configured at the server side.

# Making use of FFmpeg

FFmpeg serves as a versatile framework allowing for various (transcoding (de), multiplexing, streaming, etc.). For this exercise, we will need the following tools to be installed:

* ffmpeg
* ffprobe

First, search through your computer to see both tools are installed. If not, follow the steps described in [[16]](#w4wvxfuhxzgu) (according to the OS version you are working with).

During the first part, we will prepare a playlist from an MP4 video file (all in accordance with the HTTP Live Streaming protocol, HLS).

1. Download sample video from <http://download.blender.org/peach/bigbuckbunny_movies/>[[17]](#4za0z5nf9rau). Preferably, choose among "H.264" or "MP4" files.
2. From Command Prompt (Win OS), or Terminal (Linux/Mac OS) display the full list of options of ffmpeg and carefully review the main sections: Per-file, Video, Audio, Subtitle. For printing the list, please use one of the following ffmpeg’s parameters:

Getting help:

-h -- print basic options

-h long -- print option

-h full -- print all options (including all format and codec specific options, long

1. Before preparing a playlist (.m3u8 file), let us verify with which codecs the streams in our downloaded file have been compressed. For this purpose, run the following command:

>> **ffprobe *downloaded\_file.mp4***

Now look for 2 lines containing "**Stream: \* [Video|Audio]**" information.

The full command for both displaying and filtering stream related information is as follows:

>> **ffprobe** BigBuckBunny\_320x180.mp4 2>&1 > /dev/null | grep 'Stream. \*[Video|Audio]'

**Stream #0:0(und): Video**: h264 (Constrained Baseline) (avc1 / 0x31637661), yuv420p, 320x180 [SAR 1:1 DAR 16:9], 702 kb/s, 24 fps, 24 tbr, 24 tbn, 48 tbc (default)

**Stream #0:1(und): Audio**: aac (LC) (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 159 kb/s (default)

Now, once we know the codec, we need to make sure our file meets Apple's best practices for HLS Streaming. See here [[18]](#wnwvqrt493h).

1. Finally, we can use ffmpeg to create our playlist file. An exemplary command for achieving this goal is as follows:

>> **ffmpeg** -i ***downloaded\_file.mp4*** -hls\_segment\_filename 'segment%02d.ts' -hls\_allow\_cache 1 -hls\_time 10 -hls\_base\_url '<http://localhost/HLS/>' -strict -2 apple\_playlist.m3u8

**Important:** When using Windows OS, remove single quote characters surrounding the URL and segment name generator (i.e., segment%02d.ts).

The result of the presented command is a new file (apple\_playlist.m3u8) built from a sequence of iteratively growing transport segment (.ts) files.

One can review the file’s content using the following command:

>> **cat** apple\_playlist.m3u8

#EXTM3U

#EXT-X-VERSION:3

#EXT-X-ALLOW-CACHE: YES

#EXT-X-TARGETDURATION:11

#EXT-X-MEDIA-SEQUENCE:55

#EXTINF:10.416667,

<http://localhost/HLS/segment55.ts>

#EXTINF:10.416667,

…

Please open the HTTP Live Streaming specification (available under [[19]](#6krqxvq3r6a)) and find the role of each "#EXT\*" tag present in your playlist file.

**Important:** Be sure that you have the “HLS” directory at your local HTTP server. On Windows OS go to C:/Apache24/htdocs/ and create the “HLS” folder.

Having the playlist file, move it to the C:/Apache24/htdocs/HLS folder, along with all transport segments.

1. Once you have familiarized yourself with various possible tags describing HLS protocol, re-run the command:

>> **ffmpeg** -i ***downloaded\_file.mp4*** -hls\_segment\_filename 'segment%02d.ts' -hls\_allow\_cache 1 -hls\_time 10 -hls\_base\_url '<http://localhost/HLS/>' -strict -2 apple\_playlist.m3u8

BUT this time add '-hls\_list\_size 0' parameter. Rename the resulting (.m3u8) file to be able to compare both (apple\_playlist.m3u8 and apple\_playlist2.m3u8). What are the differences?

1. Now try to playback both files using VLC.

Select: **'File'**  -> '**Open File...'** -> choose your .m3u8 file -> '**Open**'

# Tracing HTTP streaming through the browser

Aim of the second part is to track HTTP request/response flow during a playback of HLS Streaming. For this purpose, we will need:

* Google Chrome or Mozilla Firefox (regardless of OS on your machine)
* Browser add-on allowing to track HTTP requests (i.e., Firebug for Firefox)
* [Optionally] Terminal (Linux/Mac)

In the browser, please enter <https://www.hlstester.com/> [[30]](#jcp833utqemd) or <https://hlsanalyzer.com>. This page contains a web player capable to understand and properly playback and HLS streaming (VOD and Live)

1. Before starting a playback, please change the URL of the stream to: <https://bitdash-a.akamaihd.net/content/MI201109210084_1/m3u8s/f08e80da-bf1d-4e3d-8899-f0f6155f6efa.m3u8>
2. Now, please activate the relevant add-ons:
   1. if working with Mozilla Firefox, run Firebug. It’s usually available through: Firefox -> View -> 'Firebug’ (or simply press F12)
   2. if working with Google Chrome, select View -> Developer -> 'Developer Tools' (or try to press F12)

In both cases, switch to the 'Network' tab. It will display all HTTP requests sent by your browser.

1. Start playback of the test stream (press 'Play') and make note of the requests sent from the browser.

To better understand the order of requests, please use the curl command (available in Linux’s Terminals) to see how the playlist file looks like (replace the URL below with the one you have used):

>> **curl** <http://osmfhls.kutu.ru/static/vod/sl_vod.m3u8>

Does the displayed \*.m3u8 file look different from the one you have created before for BigBuckBunny?

1. Within the file, you should be able to detect some new HLS tags. For example:

**#EXT-X-MEDIA: URI="sl\_reverb.m3u8",TYPE=AUDIO,LANGUAGE="en",NAME="Reverb"**

Once again, investigate HLS specifications [[19]](#6krqxvq3r6a) and read about their features.

1. Come back to your browser and try to force subtitles (Closed Captioning, CC)

Do you see any differences in HTTP requests? Are there any new files that were not displayed before?

1. [Optional] if you can upload your playlist file .m3u8 (as well as all required segments .ts) to any webserver, you can try to play it from the browser and check if it behaves the same way as it did for VLC.

# Report

In the report (if required: by default, reports are required, the teacher can, however, decide otherwise), one should consider the significant findings and observations made in the class.

# Bibliography

1. **Hulicki, Zbigniew.** Multimedia communication systems. Krakow. Lecture
2. **RealNetworks Inc.** Helix-Server Home Page. [Online] <https://helix-server.helixcommunity.org/>.
3. **Apple Computer, Inc.** Darwin Streaming Server Administrator’s Guide.
4. **YoungZSoft.** CCProxy – Proxy Server Software. [Online] <http://www.youngzsoft.net/ccproxy/index.html>.
5. **Postel, J.** *Internet Protocol.* The Internet Engineering Task Force. 1981. P. 45, RFC. An updated specification of IP. Replaces RFC 760
6. *User Datagram Protocol.* The Internet Engineering Task Force. 1980s. RFCs. The specification of the UDP.
7. *Transmission Control Protocol.* The Internet Engineering Task Force., 1981. P. 85, RFC. The specification of TCP. Replaces RFCs 761 and 675.
8. **The MPlayer Project.** MPlayer – The Movie Player. [Online] <http://www.mplayerhq.hu/design7/news.html>.
9. **Apple.** Open Source – Server – Streaming Server. [Online] <http://developer.apple.com/opensource/server/streaming/index.htm>.
10. **Schulzrinne, Henning, and others.** *RTP: A Transport Protocol for Real-Time Applications.* The Internet Engineering Task Force., 2003. P. 104, RFC. This memorandum Describes RTP, the real-time transport protocol.
11. **Schulzrinne, Henning, Rao, Asha and Lanphier, R.** *Real Time Streaming Protocol (RTSP).* Internet Engineering Task Force. , 1998. P. 92, RFC. The Real Time Streaming Protocol.
12. **~~Apple Computer, Inc.~~** ~~Apple – Public Source – Darwin Streaming Server. [Online]~~ [~~http://developer.apple.com/darwin/projects/streaming/~~](http://developer.apple.com/darwin/projects/streaming/)~~.~~
13. **Apple Inc.** Apple – QuickTime – QuickTime Player. *Apple.* [Online] February 23, 2008. [Quoted February 23, 2008.] <http://www.apple.com/quicktime/player/>.
14. **RealNetworks, Inc.** New RealPlayer – [www.real.com](http://www.real.com). *Video –* [*www.real.com*](http://www.real.com)*.* [Online] February 23, 2008. [quoted: 23 February 2008.] <http://uk.real.com/player/more>.
15. **The VideoLAN Project.** VideoLAN Media Player. [Online] <http://www.videolan.org/>.
16. **FFMpeg**, <https://www.ffmpeg.org/download.html>
17. **Big Buck Bunny**, <http://download.blender.org/peach/bigbuckbunny_movies/>
18. **Apple Inc.**, HTTP Live Streaming Overview, <https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/FrequentlyAskedQuestions/FrequentlyAskedQuestions.html>
19. Pantos, R., & May, W. (2017). *HTTP live streaming* (No. RFC 8216).   
    [Online] <https://tools.ietf.org/html/rfc8216>
20. **OSMF HLS plugin**, <http://osmfhls.kutu.ru/>
21. “RTP: A Transport Protocol for Real-Time Applications”  
    <http://www.rfc-editor.org/rfc/rfc1889.txt>.
22. Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, USC/Information Sciences Institute, October 1994.
23. Eastlake, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, DEC, Cybercash, MIT, December 1994.
24. “The ATM Forum” <http://www.atmforum.com/>
25. “Jitter – Wikipedia” <http://en.wikipedia.org/wiki/Jitter>
26. “Quality of Service – Wikipedia” <http://en2.wikipedia.org/wiki/Quality_of_Service>
27. “Asynchronous Transfer Mode – Wikipedia”, <http://en.wikipedia.org/wiki/ATM>
28. Apple Inc. HTTP Live Streaming [Online] <https://developer.apple.com/documentation/http_live_streaming>
29. <https://softron.zendesk.com/entries/22255163-HOW-TO-View-an-HLS-Stream-in-QuickTime-or-VLC>
30. HLSTESTER, [online] <https://www.hlstester.com>, 2019

1. ^N (on Linux and Windows)/⌘N (on Mac OS) [↑](#footnote-ref-2)
2. ^W (on Linux and Windows)/⇧⌘W (on Mac OS) [↑](#footnote-ref-3)
3. ^N (on Linux and Windows)/⌘N (on Mac OS) [↑](#footnote-ref-4)