CS5248: Systems Support for Continuous Media
Fall 2015

Project Assignment

Due: Report Draft – 11 Nov 2015 (11:59:59 pm)
Due: Code & Final Report – 17 Nov 2015 (11:59:59 pm)
Due: Presentation – 17/18 Nov 2015 (6:30 to 8:30 pm)

THIS IS A TEAM-OF-3 HOMEWORK

REMEMBER TO CHECK FOR ANNOUNCEMENTS ON THE CLASS WEB SITE:
http://www.comp.nus.edu.sg/~cs5248/

TA: Aditya Kulkarni email.aditya.kulkarni@gmail.com
Lecturer: Roger Zimmermann rogerz@comp.nus.edu.sg

Project Assignment Description

In this project, your task is to build a small DASH-compliant (Dynamic Adaptive Streaming over HTTP – that is, compatible with BOTH Apple's HTTP Live Streaming (HLS) and MPEG DASH standard) client-server-based live video streaming and hosting service on top of a LAMP stack (Linux, Apache, MySQL, PHP).

The DASH approach of streaming is becoming very popular. It is basically a replacement for the RTSP/RTP/RTCP based approach to streaming. With DASH, the server is a simple HTTP web server (e.g., Apache). The media (video) is divided into small individually playable video segments which are, for example, 10 seconds long. These segments are also called *streamlets*. The client media player retrieves the streamlets from the web server, one at a time, and plays them without interruption. For the client to know the streamlet files that belong to a complete video, the server provides a *playlist* file. The playlist file has a special format; it is basically an ***XML*** file for the MPEG DASH standard and ***M3U8*** format file for HLS. To start streaming, the client player loads the playlist file and then starts to download the streamlets that are listed in this file. Your task is to generate such a playlist file onto the server dynamically as per the new streamlets are available from one mobile device and support live playback of those streamlets onto another client device.

You will be given a number of utilities that will help you to get the project done. Additional information will be given during the lecture on 2 September 2015 and can be found in the slides for that lecture.

The project work is divided into three tasks: (1) create a mobile application that captures a live video and upload its streamlets to a server as they become available, (2) create server-side utilities that prepare incoming streamlets serviceable via DASH playlist, and (3) create another mobile application that plays a live/stored DASH playlist, while adaptively switching and playing video streamlets.

Task 1 (Mobile Video Capture and Uploader): The mobile application, running on an ASUS Transformer or Samsung tablet computer (running the Android 4.2 *Jelly Bean* OS or the Android 4.4 *KitKat* OS), is required to provide the following sub-tasks:

1) Capture a live video feed of 720p resolution from the device camera. You can use MediaCodec API from android to encode frames with h.264 encoding with 3Mbps bitrate.

2) Upload the captured frames to a web server reliably on-the-fly.

    a. Segment the original live feed into a number of self-contained 3-second-long MP4 segments before uploading on-the-fly. Segmentation can either be done first, before the upload, or in parallel, together with the upload.

    b. Use the HTTP POST method to deliver the segmented MP4 video feed to the server.

For you to understand the internal structure of the MP4 format, we intentionally ask you to segment the video at the mobile client (i.e., at the tablet computer), not at the server. To segment the video, you may use third-party libraries such as MP4Parser.

To upload the segmented video chunks reliably, you are required to design a simple protocol on top of HTTP, such as checking the current upload status or providing segments with a sequence number.

The following are additional functionalities that the mobile app can provide, which will receive extra credits:

3) Provide a resumed upload when the network connection is interrupted.

4) Retrieve the list of the uploaded videos available from the web server.

5) Videos should be playable live onto the client device during a session as well as stored on the server for VoD playback.

Task 2 (Server-side DASH Preparation): The server-side utilities may be written in the PHP language, Python based framework such as Django and there is no restriction on the use of any other third-party software packages. The minimum requirements for the server functionality are:

1) Receive segmented MP4 videos from a mobile client via the HTTP POST method and store them in a video repository location (i.e., directory) which you define. To keep track of the uploading status, you may use MySQL.

    a. Support resumed uploading.

    b. Support dynamic generation of a playlist and deliver it to a client during a live DASH streaming session.

2) Transcode every received MP4 video segment into the following three encoded media versions:

    a. 720x480 3 Mbps H.264/AVC and AAC audio (High)

    b. 480x320 768 kbps H.264/AVC and AAC audio (Medium)

    c. 240x160 200 kbps H.264/AVC and AAC audio (Low)

3) Convert every version into an MPEG-2 Transport Stream.

4) Provide two versions of DASH playlists (M3U8 for HLS and MPD XML for MPEG DASH).

5) The M3U8 playlist should be playable from iOS web browsers.

6) The MPD XML playlist should be playable from Android MPEG-DASH player (see Task 3 below). Note that the file format for the MPD playlist does *not* have to be a MPEG-2 Transport Stream.

Task 3 (Mobile MPEG-DASH Player): The mobile application should retrieve the list of the uploaded videos available from your web server in the form of XML-formatted MPEG DASH playlist files. There should be two options for a video playback. One option should be of a live video playback from a client device and second option should be of a VoD playback. It should support adaptive stream switching ─ that means it adaptively chooses the most suitable next video streamlet based on your bandwidth estimation algorithm and prepares it before its playback time, while receiving and playing the current streamlet. The minimum subtasks for this task are:

1) Retrieve a list of MPEG-DASH XML-formatted playlist files.

2) Read the playlist files and schedule the retrievals of individual streamlets on-the-fly.

3) Switches the video rendering of one video streamlet to another.

Note that this player does not have to support HLS, since it has already been a part of Android. The suggested format of video streamlets is MP4.

The following additions are given extra credits.

1) Implement video switching and rendering seamlessly.

2) Display the current network bandwidth estimation results visually.

All the above functionalities are required to run automatically, meaning that there is no manual work other than a user's upload request of a recorded video via the mobile app.

Finally, the project report should include a survey of existing streaming switching techniques (Apple's Live Streaming, Microsoft Smooth Streaming, and Adobe's HTTP Dynamic Streaming) and their comparison in terms of server software requirements, overall end-to-end delay, and network efficiency.

Reference and Software Information

All the students taking the CS5248 module are able to get an user account on our server `pilatus.d1.comp.nus.edu.sg`. The machine is running CentOS 6.7 Linux (http://www.centos.org). You will need to use `ssh` to connect to the machine. You will also write PHP or Python code in your `pilatus` *your_account*`/public_html` directory.

When segmenting the video with the mobile application, you may use the `MP4Parser` package. When developing server utilities, you are allowed to use any useful MPEG-4 parsing and transcoding utilities such as `MP4Box` or `ffmpeg`. However, server-side segmentation is not allowed — segmentation must be done at the client side. For your convenience, you can use `ffmpeg`, `MP4Box`, and `mp4info`, etc., commands from /usr/local/bin on the pilatus server.

Additional information and links:

- Android software packages (http://developer.android.com/sdk/installing/index.html)
  - Java http://www.java.com/en/download/
  - Android Studio IDE https://developer.android.com/sdk/index.html
  - Android SDK http://developer.android.com/sdk/index.html
  - Google USB Driver http://developer.android.com/sdk/win-usb.html
- Apple Live Streaming Internet Draft (http://tools.ietf.org/html/draft-pantos-http-live-streaming-06)
- Microsoft Smooth Streaming (http://www.iis.net/downloads/microsoft/smooth-streaming)
- Sample instruction how to build iPhone HTTP Streaming (http://www.ioncannon.net/programming/452/iphone-http-streaming-with-ffmpeg-and-an-open-source-segmenter/)
- MP4Parser http://code.google.com/p/mp4parser/
- FFmpeg http://ffmpeg.org
- Bento4 package http://www.bento4.com/
- MP4Box http://www.videohelp.com/tools/mp4box
- Python Django https://www.djangoproject.com/
- DASH-JS http://www-itec.uni-klu.ac.at/dash/?page_id=746

If you have any questions, email either the TA or the professor!

Submission Guidelines

1. Materials which you need to submit

(1) Your Android app source code and installable package (.apk file); your PHP, Python or shell script code. Create a tarball (i.e., a tar file) or ZIP file of all your **sources** (please exclude sample videos, object files, etc.).

(2) A detailed README.txt file, that includes
a) Your name(s), matriculation number(s), and your username(s) on the host pilatus.
b) A brief description of each file and how it works and is related to your project.

Please create a tarball (i.e., an archive created with the `tar` utility or the `zip` utility) that collects all your files into a package with your matriculation number. Then compress it (with `gzip`, in case you are using `tar`) before you submit the file. **Make sure that your code compiles without ANY errors!**

(3) A project report. Your write-up should be similar to a standard research paper (see details below). Please submit a PDF file.

(4) Submit to your report and compressed package into the course module's **IVLE Workbin** (https://ivle.nus.edu.sg/workbin/workbin.aspx?workbinid=f5c1f929-2c17-4aee-8353-7c4560ee81fe) to the folder called 'Student Submission' by 23:59 on 17/11/2015 (final project report) and 23:59 on 17/11/2015 (software source code).

(5) Late submission policy for report.
- late within 24 hours: 30% reduction in marks;
- late within 3 days: 50% reduction in marks;
- late within a week: 70% reduction in marks;
- after one week: zero marks.

Note that the project demonstrations will be held on 17 and 18 November 2015 during in-class time (6:30 to 8:30 pm) in the Reading Week and cannot be done later.

Grading Policy (50 full marks)

1. **Project Demonstration (30 marks)**. You will present your project in class in November 2015. You will have approximately 20 minutes to demo your project. The demo should include (1) recording a video from your mobile app **(5 marks)**, (2) segmenting the video on-the-fly **(5 marks)**, (3) uploading the segmented video from your mobile app **(3 marks)**, (4) resuming a failed upload from the same point where the network error occured **(2 marks)**, (5) playing the playlist of your uploaded video from any mobile browser (or app) or desktop browser **(5 marks)**, and (6) playing the playlist from your mobile DASH player **(10 marks)**.

2. **Project Source Code (5 marks)**. We will have a look at your code. Criteria are for example, how well it is documented, how well it is understandable, how robust it is (i.e., can it process all relevant input files), etc.
We will also check whether your MPD can pass the MPEG-DASH MPD validator (see http://www-itec.uni-klu.ac.at/dash/?page_id=605).

3. **Project Report (15 marks)**. Your write-up should be similar to a standard research survey paper. Please include (at least) the following sections:
    a) Abstract
    b) Introduction and motivation for your project work (your specific design and implementation choices)
    c) Summary of existing streaming switching techniques (i.e., related work)
    d) Description of your implementation, i.e., details on what you did, any difficulties encountered, any special features or functionalities that you implemented, etc.
    e) Results, comparison and discussion, i.e., what should we learn from this and what could possibly be further improved
    f) Other research directions regarding lowering end-to-end delays for MPEG-DASH streaming
    g) Conclusions
    h) References

    Most of the marks will be given for sections b), d), and e).

Additional Notes

No plagiarism is tolerated. While you can talk to other teams and you are encouraged to used the IVLE Forum, you cannot copy source code and/or text in your report. If you are not sure whether something is permissible, either talk to the instructor or consult the university policy at http://www.usp.nus.edu.sg/acad_matters/guidelines/acad_code.html.