National University of Singapore

School of Computing

CS2105 **Assignment 3** Semester 2 AY19/20

Submission Deadline

22nd Apr 2020 (Wednesday) 11:59pm sharp. 2 marks penalty will be imposed on late submission (Late submission refers to submission or re-submission after the deadline). The submission folder will be closed on **29th Apr 2020 (Wednesday) 11:59pm** sharp and no late submission will be accepted afterwards.

Objectives

In this assignment, you will inspect two multimedia streaming applications that run on DASH and WebRTC and discuss some important aspects that characterise these two popular streaming standards. After completing this assignment, you should (1) have good understanding of how DASH and WebRTC work in real-life applications, and (2) know how to use common network inspection tools for simple network debugging tasks.

This assignment is worth **6 marks** in total. There are two exercises and each exercise is worth 3 marks. All the work in this assignment shall be completed **individually**.

Setup

You should use your local machine for all tasks in this assignment. Note that you should **not** need to `ssh` into the `sunfire` server for this assignment. The tools you need (on your local machine) are -

1. Google Chrome or any other modern browser
2. Wireshark

Submission

Please submit a zip file that contains the submission files (details below) and submit it to the `Assignment_3_student_submission` folder of LumiNUS Files. Please name your zip file as **<Student number>.zip**, where **<Student number>** refers to your matric number that starts with letter A. Note that the first and last letters of your student number should be capital letters. One example file name would be **A0112345X.zip**.

Your zip file should only contain these four submission files -

1. `submission-<Matric number>.txt`: A single plain text file to record all your answers to the discussion questions in Exercise 1 and 2.
2. `dash-wireshark-<Matric number>.pcapng`: See Exercise 1, Task 3.

3. `webrtc-browser-<Matric number>.png|jpg`: See Exercise 2, Task 1.
4. `webrtc-wireshark-<Matric number>.png|jpg`: See Exercise 2, Task 1.

Plagiarism Warning

You are free to discuss this assignment with your friends. **However, you should refrain from sharing your answers or network logs.** We highly recommend that you attempt this assignment on your own and figure things out along the way as many resources are available online.

We employ zero-tolerance policy against plagiarism. If a suspicious case is found, student would be asked to explain his/her answers to the evaluator in person. Confirmed breach may result in zero mark for the assignment and further disciplinary action from the school.

Question & Answer

If you have any doubts on this assignment, please post your questions on LumiNUS forum or consult the teaching team. However, the teaching team will NOT debug issues for students. The intention of Q&A is to help clarify misconceptions or give you necessary directions.

Exercise 1 - DASH

In lecture, we learnt about DASH (Dynamic Adaptive Streaming over HTTP), which is a pull-based streaming standard over HTTP. This exercise helps us understand how DASH works in real-life streaming applications. There are three tasks in this exercise and each task is worth 1 mark. **We recommend completing the tasks in order.**

Recall the architecture of a streaming application built on DASH -

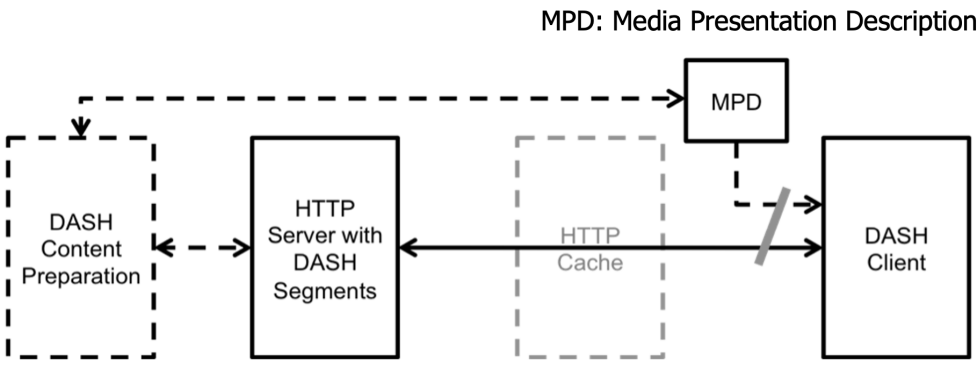

Figure 1: A streaming application built on DASH

The HTTP server provides the playlist and media segments (sometimes referred to as streamlets) for the DASH client to retrieve (i.e., "pull") on-demand.

For our DASH client in this exercise, we will use a browser-based video player provided by the DASH Industry Forum at -

`https://reference.dashif.org/dash.js/v3.0.3/samples/dash-if-reference-player/index.html`

Please access the player using any modern browser (we recommend using Google Chrome).

Task 1: MPD Playlist (1 mark)

The player provides several MPD playlists for users to choose from (see the list under "Stream"). In this assignment, we will use our playlist at -

`https://nustreaming.github.io/streaming/bbb.mpd`

Download the playlist by pasting the MPD URL in your browser's address bar. Inspect the file with your favorite text editor and answer the following discussion questions -

1. How many different video quality levels are provided by this playlist?
2. What are the highest and lowest quality levels?
3. Why do we need different quality levels? Please give an example where the player would retrieve the video at different quality levels.

Record your answers clearly in `submission-<Matric number>.txt`.

Task 2: Video Segments (1 mark)

Next, play the video provided by our playlist by replacing the MPD URL in the DASH client and click "Load". Now let's inspect the video segments retrieved by the client. On your browser, open up the network inspection tool [1] to view the browser's network log.

While the player is playing (i.e., streaming) the video, you should see the network log being updated dynamically. This log contains the list of HTTP responses received by the client, including the video segments retrieved.

Pick one video segment from the list, download it and answer the following discussion questions -

1. What is the URL of your selected video segment?
2. What is the file format of this video segment?
3. What is the duration of this video segment? (Note that all video segments from the same playlist should have the same duration.)
4. Different applications may use different video segment durations (typically between 2 and 10 seconds). From a networking perspective, describe one advan-

[1] For Google Chrome, right-click "Inspect" and click on the "Network" tab. For more details, refer to https://developers.google.com/web/tools/chrome-devtools/network.

tage and one disadvantage of using longer video segments (e.g., 10 seconds per segment).

Record your answers clearly in `submission-<Matric number>.txt`.

Task 3: HTTP-based Transfer (1 mark)

From Task 1 and 2, we can see that for DASH, the MPD playlist and video segments are retrieved over HTTP (i.e., the HTTP URLs). Now let's inspect the HTTP request and response in greater detail using Wireshark.

Open Wireshark on the machine running the browser. Using the **same MPD URL** as in Task 1, try retrieving the MPD file again on your browser and observe the capture log being updated on Wireshark.

Save Wireshark's capture log for this HTTP request and response in the default `pcapng` file format. Use Wireshark capture filters[2] **to capture and save only the relevant packets**.

Save your file as `dash-wireshark-<Matric number>.pcapng`.

Exercise 2 - WebRTC

Note: We recommend completing Exercise 1 before working on this exercise.

In lecture, we also learnt about WebRTC (Web browsers with Real-Time Communications), another popular multimedia streaming standard.

Task 1: Inspecting a Simple WebRTC Application (1 mark)

To understand how WebRTC works in real-life applications, let's run a simple experiment (similar to Exercise 1).

For our WebRTC client in this exercise, we will use a browser-based video chat application provided by the WebRTC project authors at -

`https://appr.tc/`

Please access the application using any modern browser (we recommend using Google Chrome) and create a chat room for your own use. As this is a two-way communication application, we will need another device to join the same chat room created. We recommend using your mobile phone or work with another student to get the video chat running.

Once the video chat is running, open and view the browser's network log and Wireshark's capture log as done in Exercise 1. (Note that we can only view very limited information about the DASH and WebRTC streams in Wireshark because they run on additional security protocols that encrypt most of the stream information.)

[2] See https://wiki.wireshark.org/CaptureFilters

Take two screenshots - one of the browser's network log and one of the Wireshark's capture log to show completion of the experiment. The screenshots do not have to contain any particular information as long as we can reasonably deduce the logs belong to the WebRTC stream.

Save your files as `webrtc-browser-<Matric number>.png|jpg` and `webrtc-wireshark-<Matric number>.png|jpg`.

Task 2: Discussion on WebRTC and DASH (2 marks)

After completing the above experiment, please answer the following discussion questions -

1. You should notice that the browser's network log here behaves differently from DASH's case in Exercise 1. Specifically, it does not update dynamically with the video segment responses. Why do you think this is so?
2. Which transport protocol(s) do WebRTC and DASH use (as observed in their Wireshark capture logs)?
3. Do WebRTC and DASH use the same or different transport protocols? Give two reasons why.
4. From an application and/or architecture perspective, discuss two other differences between WebRTC and DASH.

Record your answers clearly in `submission-<Matric number>.txt`.