Scott Barrett, 7881005
COMP 4300 Final Project
DASH Protocol

Summary

I chose my project to be an implementation of the DASH Protocol discussed in the Application Layer unit. I chose this because I do a lot of work with videos and I've used many services that use the DASH Protocol. The main purpose of my implementation was to show why the DASH protocol is useful by testing a client with and without dynamic bitrate selection under a variety of network conditions.

Presentation/Demo link:
https://drive.google.com/file/d/1ipDddzCQe6W0mF1nI57y9eTTUHyXOPg9/view?usp=sharing
I recommend reading this over first, as I gloss over a few details while explaining in the video.

Github repository link:
https://github.com/ScottLBarrett/4300FinalProject

DASH Protocol Overview

The DASH protocol begins with the client requesting a manifest file from the server hosting the videos. The manifest file will include links to the same video at multiple bitrates.
Based on the client's current network conditions, they request chunks of multiple seconds of video from the proper bitrate chosen from the manifest file.
Doing this allows the client to have smooth playback of the video even if their network conditions change over time.

Technical Components

My implementation is a simplification of overall HTTP streaming. I simplified the server and how the client interacts with it in order to test how the DASH protocol improves video streaming. I believed playing actual video would be outside the scope of my project to test the effectiveness of the DASH protocol, so my server and client only deal with random bytes of data rather than bytes from an actual video file.

Server

My server is not an actual HTTP server, it has two types of requests it services.

1. A "MANIFEST" request, where it sends the manifest file in response.
    - The manifest file has 5 qualities:
        - lowest = 200 KB/s
        - low = 512 KB/s
        - medium = 1 MB/s
        - high = 2 MB/s

    - highest = 4 MB/s

2. The second is a request for video at a quality from the manifest file, it's response is 3 seconds of video worth of bytes at the requested quality. It does not send an actual video file, just random bytes.

Client

My client is more advanced than the server as this is where the important part of the DASH protocol is implemented.
I had the client request a total of 5 minutes worth of video from the server

It has three threads running while it requests video from the server.

- A requesting thread, which sends the video requests to the server. It also measures the average network speed every couple of seconds then selects which bitrate it should be requesting based on the calculated average.
- A playing thread, which checks if there's atleast 5 seconds worth of video in the buffer and sleeps for that amount of seconds if there is. The thread sleeping is imitating the video playing for the client.
- A network speed thread, which changes the network speed to a random value between the min & max speeds provided when starting the client. This happens every 3 seconds.

When the client finishes running, it opens up two graphs.

- One graph shows how much data is in the buffer over time, with markers showing when the video being watched had to be buffered because of slow network conditions.
- The second graph shows the network speed over time and shows which bitrate was selected over time.

Running My Code

To run my server, run the python file `server.py` with a command-line argument of which port it should be listening on.

- For example, `python server.py 8999` will start the server on port 8999

The run my client, run the file `client.py` with these command-line arguments: Server listening port, minimum network speed (in MB/s), maximum network speed (in MB/s), and optionally the word 'static' to disable dynamic bitrate selection.

- for example, `python client.py 8999 1 2` will start the client connecting to the server on port 8999 with minimum network speed of 1 MB/s and max network speed of 2 MB/s.
- and `python client.py 8999 1 2 static` will start the client with the same network conditions, but it will not dynamically select the bitrate selected

Testing Outcome

All of my tests here I did in my video presentation/demo.

My first test was showing the improvement between static and dynamic bitrate selection with networks speeds from 0.125 MB/s to 2.3 MB/s

- The static bitrate test took about 8 minutes, which is 3 minutes longer than the video itself, as the bitrate it was requesting was high which is 2 MB/s. The average network speed was below that, so the video buffered frequently.
- The dynamic bitrate selection video went much better. It took only 12 seconds more than 5 minutes, which is taken up by intializing the connection and filling the buffer with enough video to start playing a few seconds. The video did not have to buffer during playback at all once it started playing.

My second test was showing how even faster networks can benefit from the dynamic bitrate selection. I set the network speeds to 2 MB/s to 5 MB/s.

- The static bitrate test went smoothly and did not buffer during playback. But it was stuck with the "high" bitrate option.
- The dynamic bitrate test also went smoothly and did not buffer. It also was able to take advantage of it's faster network speeds to choose a higher bitrate option occasionally to improve the video quality.

My last test was to show that even dynamically selecting bitrate cannot help buffering if the internet speed it just too slow. For this I chose speeds from 0.1 MB/s to 0.18 MB/s, which is slower than the lowest bitrate of video available.

- I only tested this one with dynamic bitrate selection. It had to buffer many times during the video and had an extra 90 seconds of overhead.
- I did not test with static bitrate selection, but I could estimate how long it might have taken.
    - The video length is 300 seconds and the client requests the video in 3 second chunks, the client would have had to request 100 chunks from the server.
    - The static bitrate would only request "high" bitrate chunks from the server. And the first chunk requested from the dynamic test was "high" and took over 40 seconds.
    - So 100 chunks at 40 seconds per chunk would take 4000 seconds, or just over an hour.
- So even though the test with dynamic bitrate selection had to buffer many times, that is much better than being forced to wait over an hour to watch a 5 minute video without dynamic bitrate selection.