ECE4016
COMPUTER NETWORK
Assignment 2

117010437
HAJUN LEE

Introduction

In this assignment, I have implemented ABR algorithm after reading the paper "A practical Evaluation of Rate Adaptation Algorithms in HTTP=based Adaptive Streaming-Published". First I will tell what is the ABR algorithm.
ABR is Adaptive Bitrate Streaming which is for improving streaming over HTTP networks. In ABR, most important part is about "Bitrate". Bitrate is the speed when data goes to network and also, it can be use it as saying about internet connection speed.
In this paper, it said about the video is divided into time intervals of a few seconds (typically 2-10 seconds) and encoded at different bitrates. So, let's go to the algorithm implement part.

Algorithm Implement /Analysis

In the paper, I chose algorithm3 which is Bitmovin Player. The reason why I chose Bitmovin Player is because, since, in Fig. 4, the highest of Bandwidth, delay and packet loss is Bitmovin.

Algorithm 3: Bitrate adaptation algorithm of Bitmovin player.

```
Preferred startup switching:
t ← // the time passed from start
s ← // the index of the rate suggested by other
   switching methods
R_pre ← // the preferred startup rate
if t < 10 then
    for k ← m to 1 do
        if k = s then
            return N_{i+1} ← R_s
        if R_k <= R_pre then
            return N_{i+1} ← R_k
    return N_{i+1} ← R_m
else
    return N_{i+1} ← R_s

Rate based switching:
// Bandwidth estimation update
depth ← // the number of downloaded segments to use
   for bandwidth estimation
N_buf ← // the buffer size in segment numbers
sum ← 0
for j ← 0 to depth − 1 do
    sum ← sum + bw_{i−j} · (1 − j/N_buf)
Bandwidth ← sum/depth // Estimated bandwidth
// Bitrate decision
R_min ← Infinity
R_max ← 0
for all encoding rate index k do
    if R_max < R_k and R_k < Bandwidth then
        R_max ← R_k
    if R_min > R_k then
        R_min ← R_k
if R_max > 0 then
    N_{i+1} ← R_max
else
    N_{i+1} ← R_min
```

It has two switching method which is preferred startup switching and rate-based switching. Preferred startup switching is when following two conditions are met. First one is when the rate suggested by other switching methods with maximum rate that equal or smaller than

preferred startup rate. Second one is when the play is in the startup phase and the suggested rate is smaller than the preferred rate.
The other one which is the rate-based switching method is selecting the maximum rate that is smaller than the estimated bandwidth. The bandwidth is estimated by averaging the measured bandwidths of recently received segments with higher weights on more recent segments.

So, first I made a bitrate to function of algorithm3

```
bitrate = algorithm3(time = Video_Time,
                     switch_rate = bitrate,
                     pref_rate = Preferred_Bitrate,
                     R_i = R_i,
                     current_buffer = Buffer_Occupancy['current'])
```

And for the function of algorithm3, I just directly followed the pseudocode of algorithm3.

```
def algorithm3(time, switch_rate, pref_rate, R_i, current_buffer):
    time = time #the time passed from start
    total_bitrate_video = len(R_i)  #total bitrate of video, R_i is array of bitrate of video
                                    #key = bitrate, value = byte size of chunk
    previous_rate = prevmatch(switch_rate, R_i) #Suggest rate from the previous
    prefer_rate = prevmatch(pref_rate, R_i)   #Get prefer rate
    #switch base bitrate
    min_R = ['float("inf")', float("inf")] #Highest
    max_R = ['0', 0]                       #Lowest
    #threshold of video time
    if time < 10:
        for i in range(total_bitrate_video):
            #if suggested rate > prefer rate
            if R_i[i] == previous_rate:
                decision_rate = previous_rate[0]
                return decision_rate

            if R_i[i][1] <= prefer_rate[1]:
                decision_rate = R_i[i][0]
                return decision_rate
        #else (or nothing matches in previous, return the high bitrate video)
        decision_rate = R_i[0][0]
        return decision_rate

    else:
        #From lowest bitrate --> To highest bitrate
        for i in range(total_bitrate_video - 1, -1, -1):
            #Lowest bitrate
            if min_R[1] > R_i[i][1]:
                min_R = R_i[i]

            #Highest bitrate
            if max_R[1] < R_i[i][1] and R_i[i][1] < current_buffer:
                max_R = R_i[i]


        #if max_R[1] is not 0, decision_rate
        if max_R[1] > 0:
            decision_rate = max_R[0]
        #rest, set to lowest
        else:
            decision_rate = min_R[0]
        return decision_rate
```

Evaluation

As I compare with the example, first experiment has been equal score. However, after the first experiment, given example has better performance in rest of experiments. So, after I finish making an algorithm, I figured out that, there has huge differences between high bandwidth and low bandwidth.