ECE4016 Assignment 2

ABR algorithm – Bitrate adaptation of of Bitmovin player

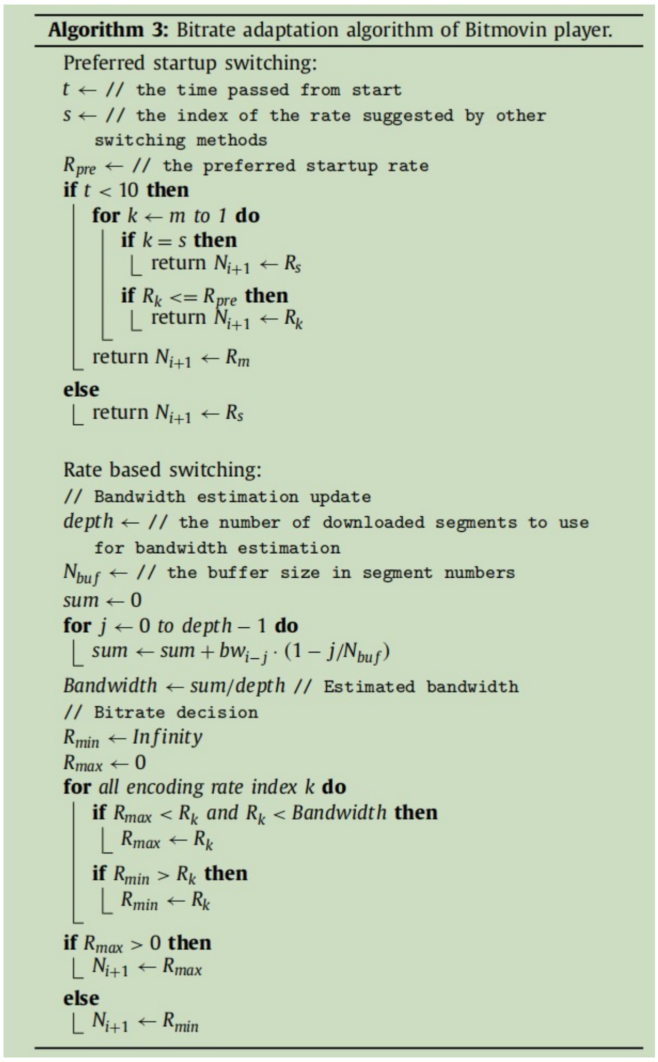
Han Yan 122090897

Oct 27, 2024

Contents

Abstract

In assignment 2, I read the paper ***A Practical Evaluation of Rate Adaptation Algorithms in HTTP-based Adaptive Streaming.*** Specifically, the paper presented the pseudo codes of 3 open source players, Bitrate adaptation algorithm of DASH Industry Forum Reference Client, Bitrate adaptation algorithm of Google's MPEG-DASH Media Source demo and Bitrate adaptation algorithm of Bitmovin player and discussed their differences. I implemented an algorithm called **Bitrate adaptation algorithm of Bitmovin player** from the paper Preferred starting switching and rate-based switching are its two switching strategies.[1] The pseudo code of Bitrate adaptation algorithm of Bitmovin player is included in Figure. 1.



Algorithm 3: Bitrate adaptation algorithm of Bitmovin player.

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

Rate based switching:
// Bandwidth estimation update
$depth \leftarrow$ // the number of downloaded segments to use
    for bandwidth estimation
$N_{buf} \leftarrow$ // the buffer size in segment numbers
$sum \leftarrow 0$
for $j \leftarrow 0$ to $depth - 1$ do
    $sum \leftarrow sum + bw_{i-j} \cdot (1 - j/N_{buf})$
$Bandwidth \leftarrow sum/depth$ // Estimated bandwidth
// Bitrate decision
$R_{min} \leftarrow Infinity$
$R_{max} \leftarrow 0$
for all encoding rate index $k$ do
    if $R_{max} < R_k$ and $R_k < Bandwidth$ then
        $R_{max} \leftarrow R_k$
    if $R_{min} > R_k$ then
        $R_{min} \leftarrow R_k$
if $R_{max} > 0$ then
    $N_{i+1} \leftarrow R_{max}$
else
    $N_{i+1} \leftarrow R_{min}$
```

Figure 1: Bitrate adaptation algorithm of Bitmovin player

1 Analysis of code

The algorithm consists of two switching methods: **preferred startup switching and rate-based switching**. The preferred startup switching overrides the rate suggested by other switching methods with a maximum rate that is equal to or smaller than the preferred startup rate if two conditions are met: the player is in the startup phase and the suggested rate is smaller than the preferred rate. The rate-base switching method selects 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.

The `student_entrypoint` function performs Adaptive Bitrate (ABR) selection with two main strategies: **rate-based switching** and **preferred startup switching**.

1.1 Function Parameters

- **Measured_Bandwidth**: Bandwidth measured for the current chunk.
- **Previous_Throughput**: Bandwidth of the previous chunk.
- **Buffer_Occupancy**: Information about the playback buffer.
- **Available_Bitrates**: Dictionary of available bitrates.
- **Video_Time**: The current playback time.
- **Chunk**: Dictionary tracking the current chunk number.
- **Rebuffering_Time**: Time spent rebuffering. (Not Used)
- **Preferred_Bitrate**: User's preferred bitrate, if provided.

1.1.1 Global State Variables

```
state = {
    bandwidth_history: [],
    preferred_startup_rate: None,
}
```

- `bandwidth_history`: Stores the recent bandwidth measurements.
- `preferred_startup_rate`: Holds the preferred bitrate in the startup phase if listed in the JSON file.

1.2 Logic Breakdown

1.2.1 Initialization of Variables

- Retrieves the current chunk number and buffer size.
- Sorts available bitrates into a list `Rk_list`.
- The preferred startup rate (`R_pre`) is set based on `Preferred_Bitrate` if it is provided, otherwise defaulting to the lowest bitrate.

1.2.2 Updating Bandwidth History

```
if Previous_Throughput > 0:
    bandwidth_measurement = Previous_Throughput
else:
    bandwidth_measurement = Measured_Bandwidth
```

- This part updates the `bandwidth_history` based on the `Previous_Throughput` or, if unavailable, the `Measured_Bandwidth`.

1.2.3 Rate-Based Switching

- **Depth**: The function uses the last 3 or fewer bandwidth measurements for bandwidth estimation and saved in `recent_bandwidths`.

  ```
  depth = min(3, len(state['bandwidth_history']))
  recent_bandwidths = state['bandwidth_history'][-depth:]
  ```

- **Bandwidth Estimation**: Weighted averaging is applied to `recent_bandwidths`, where recent measurements are given more weight. The estimated bandwidth (`Bandwidth`) is calculated as:

$$\text{Bandwidth} = \frac{\sum_{j=0}^{\text{depth}-1} \text{recent\_bandwidths}[-(j+1)] \times \max(1 - \frac{j}{N\_buf}, 0)}{\text{depth}}$$

  where N_buf is calculated as $\frac{\text{buffer\_size}}{\text{chunk\_num}}$.

  ```
  sum_bw = 0
  for j in range(depth):
      sum_bw += recent_bandwidths[-(j+1)] * max((1 - (j / N_buf)), 0)
  Bandwidth = sum_bw / depth  # Estimated bandwidth
  ```

1.2.4 Bitrate Decision

The rate-based switching method selects the maximum rate that is smaller than or equal to the estimated bandwidth. The logic is as follows:

```
R_max = 0
R_min = float('inf')
for Rk in Rk_list:
    if R_max < Rk and Rk < Bandwidth:
        R_max = Rk
    if R_min > Rk:
        R_min = Rk
if R_max > 0: Rs = R_max
else: Rs = R_min
```

- **Loop through** `Rk_list`: Iterates through the sorted bitrates in `Rk_list`.
- Updates `R_max` to the highest bitrate that is still below or equal to the estimated `Bandwidth`. And then update the `R_min` to the current Rk. Then set the Rs ready to be sent to the Preferred Startup Switching.

1.2.5 Preferred Startup Switching

The preferred startup switching method overrides the rate suggested by other switching methods (Rate-Based Switching) with the highest available bitrate that is equal to or smaller than the preferred startup rate, `R_pre`, if the following conditions are met:

- The player is in the startup phase, defined as `Video_Time < 10`.
- The rate suggested by rate-based switching, `Rs`, is lower than the preferred startup rate, `R_pre`.

If both conditions hold, the function selects the highest bitrate that does not exceed `R_pre`, overriding the rate-based suggestion (`Rs`).

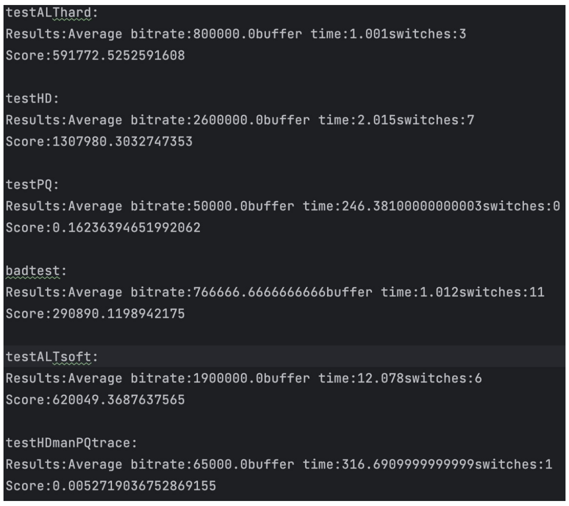
1.2.6 Return Statement

The function returns the selected bitrate based on either **rate-based switching** or **preferred startup switching**, depending on the playback time and available bandwidth.

2 Results evaluation

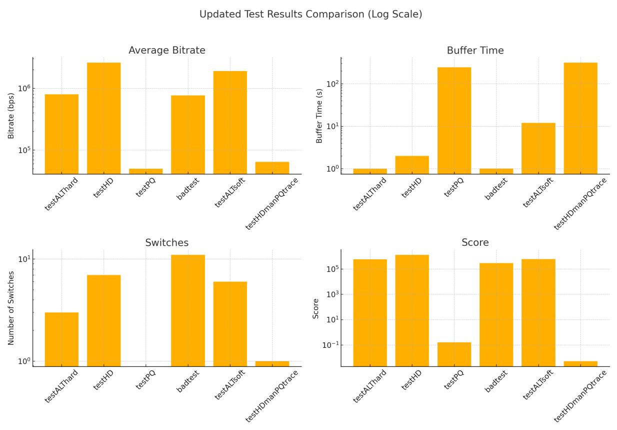
2.1 Overall Performance Summary

The Bitmovin player's adaptive bitrate algorithm demonstrates varying levels of efficiency among different bandwidth conditions, from stable high bandwidth to extreme low bandwidth and fluctuating network conditions. The player's performance and visualization can be seen in the following:



```
testALThard:
Results:Average bitrate:800000.0buffer time:1.001switches:3
Score:591772.5252591608

testHD:
Results:Average bitrate:2600000.0buffer time:2.015switches:7
Score:1307980.3032747353

testPQ:
Results:Average bitrate:50000.0buffer time:246.38100000000003switches:0
Score:0.16236394651992062

badtest:
Results:Average bitrate:766666.6666666666buffer time:1.012switches:11
Score:290890.1198942175

testALTsoft:
Results:Average bitrate:1900000.0buffer time:12.078switches:6
Score:620049.3687637565

testHDmanPQtrace:
Results:Average bitrate:65000.0buffer time:316.6909999999999switches:1
Score:0.0052719036752869155
```

(a) Generated Results



(b) Visualization in log scale

Figure 2: Two images displayed side by side

2.1.1 testALThard

In the stable, moderate bandwidth conditions of `testALThard`, the player maintains a balanced adaptation with high average bitrate and low rebuffering time, showing minimal bitrate switches. The algorithm successfully stabilizes at a rate close to the available bandwidth.

2.1.2 testHD

Under stable, high-bandwidth conditions in `testHD`, the player maximizes video quality by selecting the highest available bitrate with minimal switching and buffering. The algorithm adapts well, maintaining playback stability due to the consistent and abundant network resources.

2.1.3 testPQ

In the extremely low-bandwidth scenario of `testPQ`, the player struggles to maintain playback as the lowest available bitrate still exceeds the network capacity. This results in significant rebuffering time and minimal bitrate switching, leading to a poor viewing experience. This case reveals the limitations of the current algorithm in handling limited bandwidth environments. The player can only use the lowest bitrate.

2.1.4 badtest

In `badtest`, where bandwidth fluctuates significantly, the player adapts by frequently switching bitrates, leading to increased rebuffering and a moderate average bitrate. While the algorithm responds quickly to changes, the high frequency of switches and resulting playback instability indicate that it doesn't fit such conditions very well.

2.1.5 testALTsoft

In `testALTsoft`, where bandwidth fluctuates between high and low phases, the player achieves a relatively high average bitrate but at the cost of increased buffering and more frequent bitrate switches. The high rebuffering time suggests that while the player tries to maintain higher quality during high bandwidth phases, it struggles to adapt quickly during sudden drops.

2.1.6 testHDmanPQtrace

In the `testHDmanPQtrace` scenario, with extremely low bandwidth and a small buffer size, the player experiences extensive rebuffering despite adapting to the lowest available bitrate. The combination of limited bandwidth and small buffer severely limits playback stability.

3 Conclusion

In summary, the Bitmovin player algorithm performs effectively in stable network conditions. However, under unstable conditions like the badtest, where bandwidth fluctuates significantly, the algorithm would switch most, which shows that the algorithm isn't suitable for unstable condition and is more suitable for stable connection so that it can provide higher bitrate. The results coincide with the results shown in the paper.

References

[1] I. Y. K. E. . H. S. Ayad, I., "A practical evaluation of rate adaptation algorithms in http-based adaptive streaming.," *Computer Networks*, vol. 133, no. 8, pp. 90–103, 2018.