A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service

Huang, T-Y. et al.
Proc. of ACM SIGCOMM '14, 44(4):187-198, Oct. 2014.

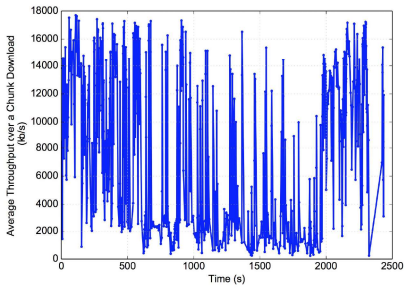
Allison and Chun-Yu

Outlines

- Introduction
- Buffer-based Algorithm
  - BBA-0 (CBR)
  - BBA-1 (VBR)
  - BBA-2 (Startup phase)
  - BBA-Other (Rate oscillation)
- Conclusions and Discussion

Video Streaming Challenges

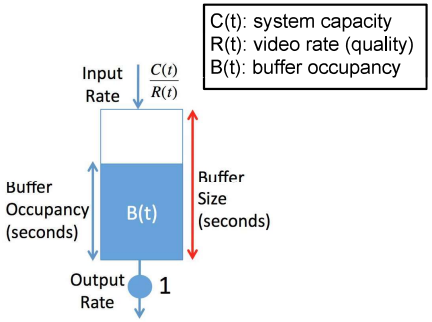
- Throughput while streaming is highly variable
- Objective: maximize video quality, minimize rebuffering events
- Adaptive Bitrate (ABR) algorithms try to estimate network capacity to select video rate



Netflix Sample Trace

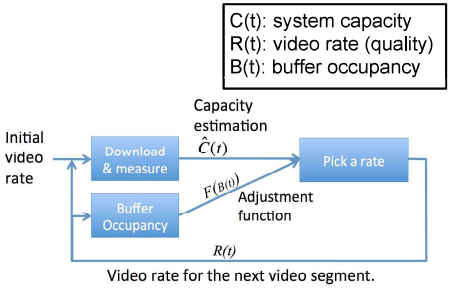
Playback Buffer

- Data is requested in chunks (4 seconds for Netflix)
- Buffer drains at 1 unit/second
- If R(t) is greater than system capacity C(t), new data is added at C(t)/R(t) < 1, depleting the buffer



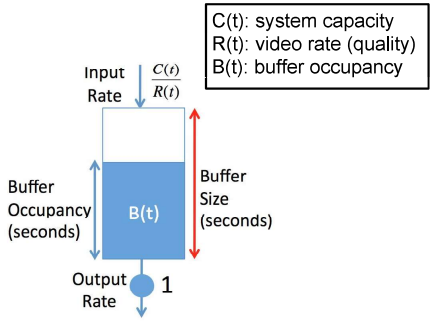
Estimating Capacity

- Previous ABR algorithms have used both capacity history and buffer occupancy
- $R(t) = F(B(t))\hat{C}(t)$
  - For stable throughput, $\hat{C}(t)$ would be most optimal
  - For variable throughput, it can be dangerous not to rely on buffer occupancy



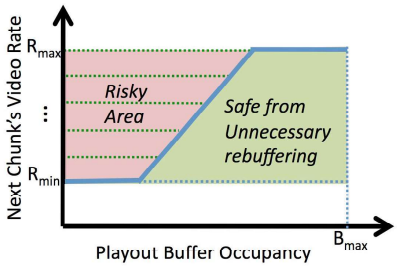
Buffer-Based Approach

- Prioritize buffer occupancy to decrease rebuffering events
- If buffer approaches empty, request R_{min}
- If C(t) increases, increase R(t)



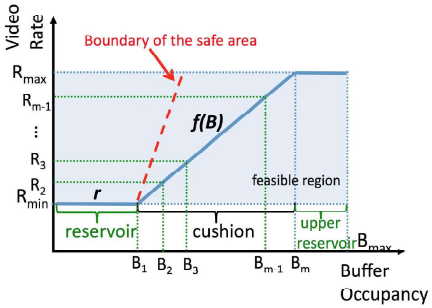
Buffer Based Algorithm: Theory

- Assumptions:
  - Chunk size is infinitesimal
  - Video rates between R_{min} and R_{max} are continuous
  - Videos encoded at a constant bitrate
  - Videos are infinitely long
- As long as $C(t) \geq R_{min}$ and $f(B) \rightarrow R_{min}$ when $B \rightarrow 0$, there won't be unnecessary rebuffering events
- As long as f(B) increases to R_{max}, the average video rate will match average capacity when $R_{min} < C(t) < R_{max}$



Buffer-Based Algorithm: Real World

- Add reservoir *r* to pad for finite chunk sizes and some C(t) variation
  - While filling reservoir, request R_{min}
  - Must have $r \geq$ minimum chunk size
- Add upper reservoir to reach R_{max} before B is full
- Above the "safe area" is where the buffer will not deplete into *r* if C(t) suddenly drops (but not lower than R_{min})



BBA-0 Algorithm

- Video rate is a little “sticky”
  - Don’t change unless suggested rate crosses barrier into next lower ($Rate_-$) or higher ($Rate_+$) video rate

Algorithm 1: Video Rate Adaptation Algorithm

Input: $Rate_{prev}$: The previously used video rate
Buf_{now}: The current buffer occupancy
r: The size of reservoir
cu: The size of cushion
Output: $Rate_{next}$: The next video rate

if $Rate_{prev} = R_{max}$ **then**
  $Rate_+ = R_{max}$
else
  $Rate_+ = \min\{R_i : R_i > Rate_{prev}\}$

if $Rate_{prev} = R_{min}$ **then**
  $Rate_- = R_{min}$
else
  $Rate_- = \max\{R_i : R_i < Rate_{prev}\}$

if $Buf_{now} \leq r$ **then**
  $Rate_{next} = R_{min}$
else if $Buf_{now} \geq (r + cu)$ **then**
  $Rate_{next} = R_{max}$
else if $f(Buf_{now}) \geq Rate_+$ **then**
  $Rate_{next} = \max\{R_i : R_i < f(Buf_{now})\}$;
else if $f(Buf_{now}) \leq Rate_-$ **then**
  $Rate_{next} = \min\{R_i : R_i > f(Buf_{now})\}$;
else
  $Rate_{next} = Rate_{prev}$;
return $Rate_{next}$;

BBA-0 Algorithm

- If the buffer is in the lower reservoir, request R_{min}
- If the buffer is in the upper reservoir, request R_{max}

Algorithm 1: Video Rate Adaptation Algorithm

Input: $Rate_{prev}$: The previously used video rate
Buf_{now}: The current buffer occupancy
r: The size of reservoir
cu: The size of cushion
Output: $Rate_{next}$: The next video rate

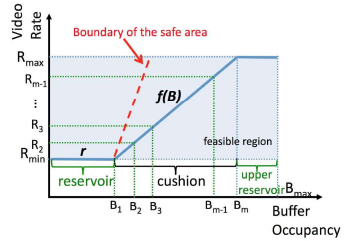
if $Rate_{prev} = R_{max}$ **then**
  $Rate_+ = R_{max}$
else
  $Rate_+ = \min\{R_i : R_i > Rate_{prev}\}$

if $Rate_{prev} = R_{min}$ **then**
  $Rate_- = R_{min}$
else
  $Rate_- = \max\{R_i : R_i < Rate_{prev}\}$

if $Buf_{now} \leq r$ **then**
  $Rate_{next} = R_{min}$
else if $Buf_{now} \geq (r + cu)$ **then**
  $Rate_{next} = R_{max}$
else if $f(Buf_{now}) \geq Rate_+$ **then**
  $Rate_{next} = \max\{R_i : R_i < f(Buf_{now})\}$;
else if $f(Buf_{now}) \leq Rate_-$ **then**
  $Rate_{next} = \min\{R_i : R_i > f(Buf_{now})\}$;
else
  $Rate_{next} = Rate_{prev}$;
return $Rate_{next}$;

BBA-0 Algorithm

- If the curve suggests we should be requesting a higher rate than our upper/lower threshold, select rate nearest the curve



Algorithm 1: Video Rate Adaptation Algorithm

Input: $Rate_{prev}$: The previously used video rate
Buf_{now}: The current buffer occupancy
r: The size of reservoir
cu: The size of cushion
Output: $Rate_{next}$: The next video rate

if $Rate_{prev} = R_{max}$ **then**
  $Rate_+ = R_{max}$
else
  $Rate_+ = \min\{R_i : R_i > Rate_{prev}\}$

if $Rate_{prev} = R_{min}$ **then**
  $Rate_- = R_{min}$
else
  $Rate_- = \max\{R_i : R_i < Rate_{prev}\}$

if $Buf_{now} \leq r$ **then**
  $Rate_{next} = R_{min}$
else if $Buf_{now} \geq (r + cu)$ **then**
  $Rate_{next} = R_{max}$
else if $f(Buf_{now}) \geq Rate_+$ **then**
  $Rate_{next} = \max\{R_i : R_i < f(Buf_{now})\}$;
else if $f(Buf_{now}) \leq Rate_-$ **then**
  $Rate_{next} = \min\{R_i : R_i > f(Buf_{now})\}$;
else
  $Rate_{next} = Rate_{prev}$;
return $Rate_{next}$;

BBA-0 Algorithm

- Otherwise, keep the current video rate

Algorithm 1: Video Rate Adaptation Algorithm

Input: $Rate_{prev}$: The previously used video rate
Buf_{now}: The current buffer occupancy
r: The size of reservoir
cu: The size of cushion
Output: $Rate_{next}$: The next video rate

if $Rate_{prev} = R_{max}$ **then**
  $Rate_+ = R_{max}$
else
  $Rate_+ = \min\{R_i : R_i > Rate_{prev}\}$

if $Rate_{prev} = R_{min}$ **then**
  $Rate_- = R_{min}$
else
  $Rate_- = \max\{R_i : R_i < Rate_{prev}\}$

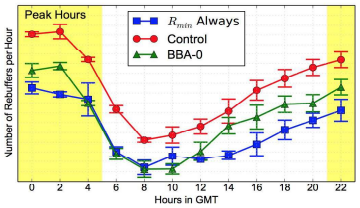
if $Buf_{now} \leq r$ **then**
  $Rate_{next} = R_{min}$
else if $Buf_{now} \geq (r + cu)$ **then**
  $Rate_{next} = R_{max}$
else if $f(Buf_{now}) \geq Rate_+$ **then**
  $Rate_{next} = \max\{R_i : R_i < f(Buf_{now})\}$;
else if $f(Buf_{now}) \leq Rate_-$ **then**
  $Rate_{next} = \min\{R_i : R_i > f(Buf_{now})\}$;
else
  $Rate_{next} = Rate_{prev}$;
return $Rate_{next}$;

BBA Experiments

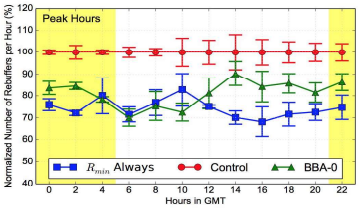
- Samples of real Netflix users around the world
- 3 groups:
  - **R_{min} Always**: Stream at R_{min} for entire experiment (☹)
  - **Control**: Netflix's current (2014) ABR Algorithm
  - **BBA-0**: Buffer-based rate selection

BBA-0 Results

- *R_{min} Always* estimates fewest number of rebuffers possible
- *BBA-0* consistently has fewer rebuffers than *Control*, and performs about the same as *R_{min} Always* after peak hours



(a) Number of rebuffers per playhour during the day.



(b) Normalized number of rebuffers per playhour, normalized to the average rebuffer rate of *Control* in each two hour period.

BBA-0 Results

- Consistently worse video rate
  - Variable Bitrate (VBR) encoding
  - 90s reservoir in a 240s buffer
- But less video rate switching!
  - $Rate_+$ and $Rate_-$ padding helped

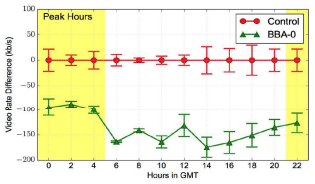


Figure 8: Comparison of video rate between *Control* and BBA-0. The error bars represent the variance of video rates from different days in the same two-hour period. The Y-axis shows the difference in the delivered video rate between *Control* and BBA-0.

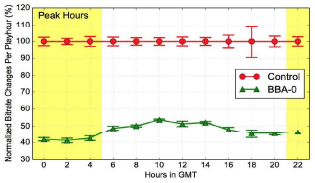


Figure 9: Average video switching rate per two hour window for the *Control* and *BBA-0* algorithms. The numbers are normalized to the average switching rate of the *Control* for each window.

Handling Variable Bitrate

- Further improve rebuffer rate and video rate by taking VBR into consideration

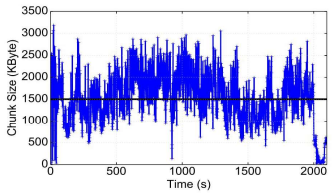


Figure 10: The size of 4-second chunks of a video encoded at an average rate of 3Mb/s. Note the average chunk size is 1.5MB (4s times 3Mb/s).

Max-to-Average Ratio ~ 2

In terms of chunk in time

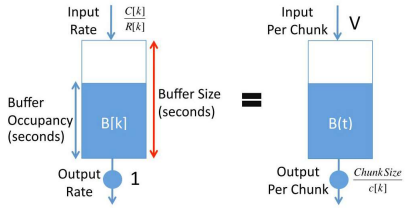


Figure 11: Two equivalent models of the streaming playback buffer.

Chunk[r][k] = V x R[k]

Reservoir Calculation

- Design reservoir based on the instantaneous encoding bitrate of the stream being delivered
  => By looking into the buffer and see the following chunk sizes
- X = 480 sec (twice as buffer size in time)

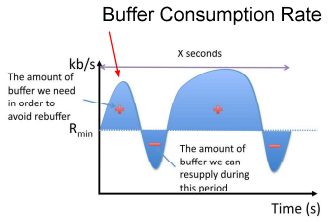
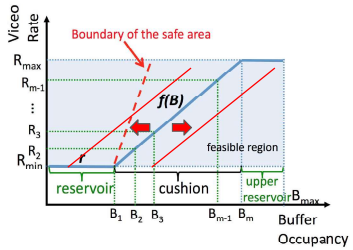


Figure 12: Reservoir calculation: We calculate the size of the reservoir from the chunk size variation.



Chunk Map

- From Buffer size => Rate
  To Buffer size => Chunksize
  (Even though two chunk sizes are the same, the video rates may differ. A further mapping from chunksize to rate is needed.)
- Only switched rate when passing the next higher/lower level threshold

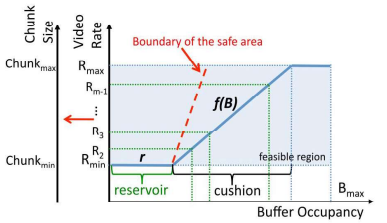
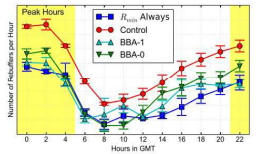


Figure 13: Handling VBR with chunk maps. To consider variable chunk size, we generalize the concept of rate maps to chunk maps by transforming the Y-axis from video rates to chunk sizes.

BBA-1 Results

- Rebuffer rate: Close to R_{min} always
  (20-28% improvement compared to Control)
- Video rate: Better than BBA-0, but still not catching up with Control
  => Because of the startup phase



(a) Number of rebuffers per playhour throughout the day.

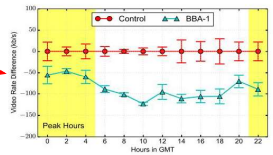
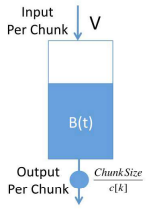


Figure 15: The BBA-1 algorithm improved video rate by 40–70 kb/s compare to BBA-0, but still 50–120 kb/s away from the *Control*.

The Startup Phases

- BBA-2
  - Be aggressive in startup phase
    Aggressively decrease step up threshold linearly
    => Download-to-consume ratio from 8 to 2
- At startup, step up from R_i to R_{i+1} if
  - dB = (V - ChunkSize/c[k])
    <= (V - ChunkSize/R_{i+1}) to safely step up when CBR
    <= (V - ChunkSize/eR_{i+1}) ~ 0.875V when VBR,
    where e is max-to-avg chunk size ratio
    Typo in Section 6: $R_{i+1}/R_i \sim 2$
- Go back to BBA-1 when
  - Buffer is decreasing
  - Chunk map suggests a higher rate



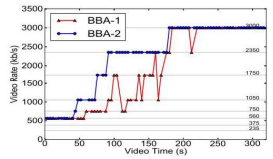


Figure 16: Typical time series of video rates for BBA-1 (red) and BBA-2 (blue). BBA-1 follows the chunk map and ramps slowly. BBA-2 ramps faster and reaches the steady-state rate sooner.

BBA-2 Results

- Video rate of BBA-2 is comparable to Control
- If exclude the first two minutes => BBA-2 has higher video rate
- Maybe Control is more aggressive during startup phase (the first two minutes)?
- BBA-2 has higher rebuffer rate than BBA-1, but lower than Control

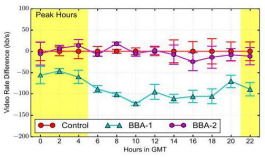


Figure 17: BBA-2 achieved a similar video rates to the *Control* algorithm overall.

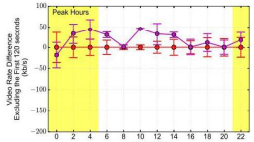
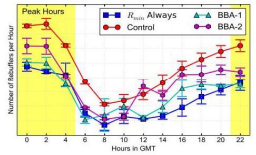
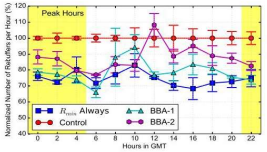


Figure 18: BBA-2 achieved better video rate at the steady state. The steady state is approximated as the period after the first two minutes in each session.



(a) Number of rebuffers per playhour throughout the day.

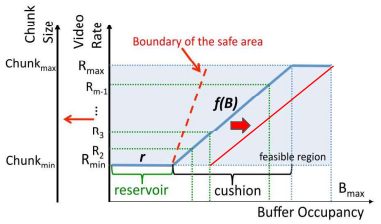


(b) Normalized number of rebuffers per playhour, the number is normalized to the average rebuffer rate of the *Control* in each two hour period.

Figure 19: BBA-2 has a slightly higher rebuffer rate compared to BBA-1, but still achieved 10–20% improvement compared to the *Control* algorithm during peak hours.

Handling Temporary Network Outage

- Moving chunk map to the right
  => increasing reservoir
- For BBA-1, 400ms for each chunk downloaded when the buffer is increasing and still less than 75% full
- For BBA-2, typically 20-40 sec in steady state and bounded at 80 sec



Smoothing Video Switch Rate

- No fixed mapping between buffer level and video rate
  => Even if buffer level stays the same, the video rate may change
  - Solution:
    Looking into the following chunks to decide whether to step up video rate => only limited to stepping up
    Ex. If a small chunk followed by some large chunks, we will not step up to next level to prevent video rate from going back
- Reservoir shrinks and expands depending on the coming up chunks
  - Solution: Reservoir only expands but never shrinks

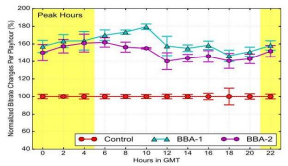


Figure 20: After switching from using a rate map to using a chunk map, the video switching rate of BBA-1 and BBA-2 is much higher than the *Control* algorithm.

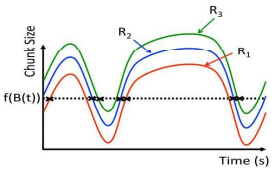


Figure 21: A reason using chunk map increases video switching rate. When using a chunk map, even if the buffer level and the mapping function remains constant, the variation of chunk sizes in VBR streams can make a buffer-based algorithm switch between rates. The lines in the figure represent the chunk size over time from three video rates, R_1, R_2, and R_3. The crosses represent the points where the mapping function will suggest a rate change.

BBA-Other Results

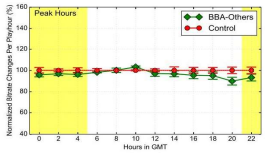


Figure 22: BBA-Others smoothes the frequency of changes to the video rate, making it similar to the *Control* algorithm.

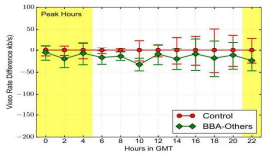
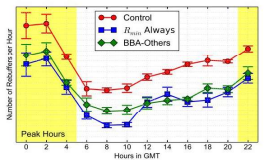
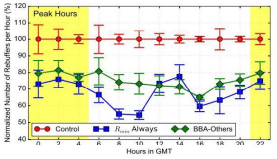


Figure 23: BBA-Others achieves a similar video rate during the peak hours but reduces the video rate by 20–30kb/s during the off-peak. Really?



(a) Number of rebuffers per playhour throughout the day.



(b) Normalized number of rebuffers per playhour, the number is normalized to the average rebuffer rate of the *Control* in each two hour period.

Figure 24: BBA-Others reduces rebuffer rate by 20–30% compared to the *Control* algorithm.

Conclusions and Discussion

- A series buffer-based rate selection algorithms are proposed
  - Start up state => Network capacity
  - Steady state => Buffer occupancy

- Discussion
  - What is Control?
  - Many details are remain hidden
  - Startup state is not compared with Control.