Journal of Computers Vol. 31 No. 3, 2020, pp. 27-39
doi:10.3966/199115992020063103003

PASA: Towards Fair Rate Adaptation for Http-based Video Streaming Through Server Assistance

Han-Ze Dong[1,2], Zhi-Chuan Guo[1,2], Yi-Qiang Sheng[1], Xiao-Yong Zhu[1*]

[1] National Network New Media Engineering Research Center,
Institute of Acoustics, Chinese Academy of Sciences,
No. 21, N 4th Ring Road West, Haidian District, Beijing, 100190, China
{donghz, guozc, shengyq, zhuxy}@dsp.ac.cn

[2] University of Chinese Academy of Sciences,
No. 19(A), Yuquan Road, Shijingshan District, Beijing, 100049, China

Received 10 September 2018; Revised 23 January 2019; Accepted 6 March 2019

Abstract. HTTP-based adaptive streaming (HAS) has become the de facto standard for video streaming over the Internet. However, when multiple clients compete for available bandwidth at a bottleneck link, traditional rate adaptive algorithms lead to unfair bandwidth allocation. This paper proposes “Probe and Adapt with Server Assistance” (PASA) algorithm based on PANDA. PASA uses a decentralized fairness controller that works with the assistance of server to improve fairness among clients. The main advantage of PASA is that it is still a client-driven, pull-based algorithm and it does not bring challenges as in centralized management. Besides, we use a bitrate switching logic to make a balance between response agility to bandwidth fluctuation and bitrate stability. We use ns-3 simulator to evaluate the performance of PASA. Compared with PANDA and state-of-the-art method, the results show that PASA is able to reduce the unfairness of video bitrate by over 40% and also has excellent stability and convergence rate.

Keywords: convergence rate, DASH, fairness, HTTP-based adaptive streaming, stability

1 Introduction

In the past few years, video streaming has become the majority of Internet traffic [1]. A recent report by Cisco [2] estimates that by 2022 video traffic will account for 82% of the total Internet data traffic. In this context, HTTP-Based Adaptive Streaming (HAS) gradually becomes the de facto standard for video streaming over the Internet.

Dynamic Adaptive Streaming over HTTP (DASH) [3-4] is a typical HAS standard. It has been widely adopted by media content providers such as YouTube and Netflix [5]. In DASH, the video bitrate is decided by the client. The client uses a rate adaptive algorithm to make a proper compromise among various quality of experience (QoE) metrics, such as average video bitrate, switching of video bitrate and duration of playback freezing [6-8].

It is challenging for the rate adaptive algorithm to select the optimal bitrate since the above QoE metrics always conflict with each other. For example, it is expected that the requested bitrate reacts to network changes quickly to maximize the bandwidth utilization and avoid playback freezing. On the other hand, frequent bitrate switches are harmful to user’s experience. So the rate adaptive algorithm should make a balance between video smoothness and responsiveness to bandwidth fluctuation. What’s more, most of the rate adaptive algorithms focus only on improving QoE but neglecting fairness issues. When multiple clients compete for available bandwidth at a bottleneck link, these algorithms easily lead to unfair bandwidth allocation.

[*] Corresponding Author

The majority of rate adaptive algorithms select bitrate based on estimated network throughput (rated-based) or buffer occupancy variation (buffer-based) to maximize QoE. Wang et al. [9] use a multi-step predictive control method that predicts bandwidth and buffer occupancy of next several steps to balance responsiveness and smoothness. Mao et al. [10] propose Pensieve, a system that generates rate adaptive algorithms using reinforcement learning. It can automatically learn algorithms that adapt to various environments and QoE metrics. However, these algorithms don't consider fairness.

Some algorithms try to achieve fairness among multiple clients. PANDA [11] uses an additive-increase-multiplicative-decrease (AIMD) mechanism to probe available bandwidth. Relaying on the underlying TCP, it can converge to fair-share status. Yuan et al. [12] model the problem as a non-cooperative game theory model and prove the existence of Nash Equilibrium, they design a distributed iterative algorithm to maximize the user's QoE with fairness guaranteed. However, the fairness of above algorithms depend on the correct estimation of bandwidth. But the downloading process of video is discrete and the client cannot estimate bandwidth accurately because of the presence of OFF intervals [13].

Seufert et al. [14] find that using some centralized management can effectively enhance the overall video quality and improve fairness among clients. Bentaleb et al. [15] propose an SDN-based HAS system architecture that uses the software defined network (SDN) to dynamically manage and allocate network resources for users. Marai et al. [16] propose a client-server cooperation-based approach to improve efficiency, fairness and stability. However, when the quantity of clients is large, the centralized management will bring great pressure to the system, which is not conducive to large-scale deployment.

Instead of improving the QoE of single client, the goal of this paper is to ensure fairness among multiple clients while maximizing QoE for each client. We proposed a "Probe and Adapt with Server Assistance" (PASA) algorithm aiming to improve fairness and network utilization. It is based on PANDA algorithm and utilizes information at server-side to improve fairness. It keeps the advantages of traditional client-driven, pull-based approach and does not introduce the disadvantages of centralized management.

The main contributions of this paper are summarized as follows.

- We analyze the reason of unfairness among multiple clients and prove that the client cannot estimate network bandwidth accurately because of the presence of OFF intervals. And we use a fairness factor to help estimate the available bandwidth. The fairness factor considers the average bitrate of other clients at the bottleneck link provided by the server. With the server's assistance, the bandwidth for each client can converge to a fair sharing state with fast convergence rate.
- Besides, we design a bitrate switching logic to make a good trade-off between bitrate smoothness and responsiveness to bandwidth fluctuation.
- We use ns-3 simulator to evaluate the performance of PASA and compare it with PANDA and ESTC [16]. The results show that PASA reduces the unfairness by over 40% and it also has excellent stability and convergence rate.

The remainder of this paper is structured as follows: In Section 2, we introduce the problem model and PANDA algorithm. In Section 3, we present the fairness controller, bitrate switching logic and propose PASA algorithm. In Second 4, we provide the experimental settings, results and analyses. Finally, we conclude the paper and talk about the future work.

2 Problem Model and PANDA Algorithm

In this section, we formulate the general client-server interaction process in DASH and introduce PANDA algorithm briefly.

2.1 Problem Model

At the server side, we assume that a video streaming is chopped into segments of τ seconds. Each segment is pre-encoded into a set of L discrete video bitrates $R := \{R_1, \ldots, R_L\}$, $0 < R_1 < \ldots < R_L$. A manifest file called Media Presentation Description (MPD) is generated to describe the information of video segments.

At the client side, there are N competing clients requesting video from the server through a bottleneck link whose bandwidth is W. Each client first downloads the MPD file from the server, and then requests and downloads the video segments sequentially.

According to the number of video segments, the video downloading process of each client can be divided into multiple steps. In the n^{th} step, first, the client determines the delivered bitrate $r[n]$ for n^{th} segment based on estimated bandwidth and current buffer size. Suppose $t_{on}[n]$ denotes the duration of downloading n^{th} segment, the average bandwidth can be estimated as

$$T[n] = \frac{r[n] \cdot \tau}{t_{on}[n]} . \tag{1}$$

We use the average bandwidth when downloading the past few segments as an estimation of current bandwidth. Then, the client requests and downloads the n^{th} video segment of desired bitrate $r[n]$ from the DASH server.

The client stores the downloaded segments in a buffer to absorb temporary mismatch between the video playback rate and video downloading rate. Because video segments of different bitrates occupy different memory size, we cannot use memory size (i.e. KB or MB) to measure the size of buffered video. To tackle the problem, the playback time (i.e. seconds) of buffered video is used to measure the buffer size. We set $B[n]$ as the buffer size at the end of n^{th} step. When downloading a segment, the video player is consuming the buffer at a rate of one video second per real-time second at the same time. Thus, the buffer size can be denoted as:

$$B[n] = \max(0, B[n-1] + \tau - t[n]) , \tag{2}$$

where $t[n]$ is the interval time between two adjacent requests. $t[n]$ includes the downloading time $t_{on}[n]$ of a segment and the waiting time $t_{off}[n]$ to request the next segment (i.e. OFF intervals), namely $t[n] = t_{on}[n] + t_{off}[n]$.

2.2 PANDA Algorithm

The PANDA algorithm takes inspiration from TCP congestion control, sharing the similar mechanism at the application layer. The difference is that PASA operates at a video-segment rather than RTT timescale. The main observation is that bandwidth estimation is accurate only when the link is in perfect subscription (i.e. the total amount of traffic requested by clients perfectly fills the link) or oversubscription (i.e. the total amount of traffic requested is larger than the link bandwidth) and with no OFF intervals. Otherwise, overestimation occurs.

First, PANDA estimates the target bandwidth $x[n]$ by

$$\frac{x[n] - x[n-1]}{t[n-1]} = a \cdot (\omega - \max(0, x[n-1] - T[n-1] + \omega)) , \tag{3}$$

where $a > 0$ is the probing convergence rate, $\omega > 0$ is the additive increase rate. It has an additive-increase-multiplicative-decrease (AIMD) interpretation as TCP congestion control: $a \cdot \omega$ is the additive increase term, and $-a \cdot \omega(0, x[n-1] - T[n-1] + \omega)$ is the multiplicative decrease term. As shown in (3), the client continuously increases the target data rate $x[n]$ by $a \cdot \omega$ per segment to "probe" the available link bandwidth, until the actual throughput $T[n]$ is less than the target data rate, which means congestion happens.

Second, to eliminate the link noise and outliers, an Exponential Weighted Moving Average (EWMA) smoother is used to filter $x[n]$ and produce the smooth version $y[n]$:

$$\frac{y[n] - y[n-1]}{t[n-1]} = -\beta \cdot (y[n-1] - x[n]) , \tag{4}$$

where $\beta > 0$ is the smoothing convergence rate.

Next, quantizing $y[n]$ to the discrete video bitrate $r[n] \in R$ by

$$r[n] = \begin{cases} r_{up}, & r[n-1] \le r_{up} \\ r[n-1], & r_{up} \le r[n-1] \le r_{down}, \\ r_{down} & otherwise \end{cases} \tag{5}$$

where

$$\begin{cases} r_{up} := \max_{r \in R} r \ \ subject\ to\ r \le y[n] - \in \cdot y[n] \\ r_{down} := \max_{r \in R} r \ \ subject\ to\ r \le y[n] \end{cases}, \tag{6}$$

and $0 < \in < 1$ is the coefficient to control the size of [r_{up}, r_{down}]. (5) is designed to avoid frequent bitrate hopping between two adjacent bitrate levels.

Last, scheduling the waiting time to request the next segment:

$$t_{off}[n] = \frac{r[n] \cdot \tau}{y[n]} + \gamma \cdot (B[n-1] - B_{target}), \tag{7}$$

where $r > 0$ is a constant and B_{target} is a predefined buffer size.

3 PASA Algorithm

In this section, we introduce the fairness controller and the bitrate switching logic. Finally, the PASA algorithm is proposed.

3.1 Fairness Controller

The rate adaptive algorithms can be divided into two families [17]: rate-based and buffer-based. PANDA is a typical rate-based algorithm. Essentially, rate-based algorithms determine the bitrate of current segment based on the estimated throughput which is calculated by downloading duration and video size of previous segment, and video size depends on bitrate. So the request bitrate $r[n]$ is approximate to a function of $r[n-1]$, namely $r[n] \sim f(r[n-1])$. Usually, buffer-based algorithms [12, 18] also need to consider previous bitrate in order to avoid frequent bitrate switching and large switching amplitude. So $r[n]$ also can be seen as a function of $r[n-1]$ in buffer-based algorithms.

The clients converge to a relatively stable state from the initial bitrate gradually. During the convergence process, the clients may have different convergence rates because of various disturbances occurring on the link. The disturbances accumulate so that different client converges to different bitrate eventually. By probing the available bandwidth, PANDA can converge to a fair-share state if the underlying TCP is fair but it may take a long convergence time.

Besides, there is a fundamental limitation in the bandwidth estimation of rate adaptive algorithms. To maintain the buffer in an appropriate size, the client needs to postpone the request of next segment for a while (i.e. OFF intervals) to avoid buffer overflow. However, the OFF intervals lead to inaccurate bandwidth estimation. For example, suppose there are two clients connecting to a server through a bottleneck link whose capacity is *C*. The fair-share bandwidth for each client should be *C*/2. If client A is in OFF interval when client B are downloading video segments, the estimated bandwidth of client B will be larger than *C*/2. So the clients cannot detect the available bandwidth accurately by themselves because of OFF intervals. And it further leads to unfairness.

Note that an accurate decision cannot be achieved with the information of client-side, we propose a server-assisted fairness controller, which takes into account the information collected by the server to estimate bandwidth and improve fairness

For a client N_i, the server collects current bitrates of all other clients on the same link and calculates the average:

$$\tilde{T}_{-i} = \frac{1}{N-1} \cdot \sum_{j=0, j\neq 1}^{N} \hat{r}_j, \tag{8}$$

where $\hat{r}_j$ is the weighted bitrate to distinguish different types of devices. A new field for $\tilde{T}_{-i}$ is added to the packet header of video segment and sent to the client. The client N_i parses the value and marks it as $\tilde{T}$. This value is used to calculate the fairness factor:

$$F_{fair}[n] = \frac{\tilde{T}[n-1] - x[n-1]}{\min(x[n-1], \tilde{T}[n-1])} \cdot x[n-1]. \tag{9}$$

In section 3.3, we will use the fairness factor to estimate the available bandwidth $x[n]$ with the probing factor.

The mechanism of the fairness controller is: The server knows more information about the link state than clients. It offers the information about link state to clients as a factor to revise the estimated bandwidth. The server does not directly control the bitrates of clients. Thus, the fairness controller avoids the troubles bought by the centralized management.

3.2 Bitrate Switching Logic

Frequent bitrate switching has a terrible impact on the user experience [19]. However, excessive pursuit of video smoothness will make the client insensitive to bandwidth fluctuation. When the link bandwidth decreases, if the bitrate cannot switch to a lower version in time, the video buffer may be empty, resulting in video freezing. To make a trade-off between the video smoothness and the responsiveness to bandwidth fluctuation, we design a bitrate switching logic.

Bitrate switching state can be divided into two categories. One is that the bitrate increases/decreases steadily to the equilibrium state, we call it steady state of bitrate switching, which mostly happens in the startup stage of a client. The other one is that bitrate changes irregularly because of link bandwidth fluctuation, we call it fluctuant state. The former is beneficial to QoE while the latter is harmful.

To avoid unnecessary bitrate switching, the client should keep requesting a bitrate $r[n]$ for at least $m[n]$ segments before switching to a different bitrate version. $m[n]$ is the key of our bitrate switching logic, it is calculated by the bitrate switching state of the past s seconds:

$$m[n] = \kappa \cdot \sum_{t_i \in \mathbb{T}} \frac{t_i - (t_{now} - s)}{s} \cdot flag_i, \tag{10}$$

$$flag_i = \begin{cases} 1 \; in\ fluctuant\ state \\ 0 \; in\ steady\ state \end{cases}, , \tag{11}$$

where t_i is the moments when the i^{th} segment has been downloaded, $\mathbb{T}$ is the set of t_i during $[t_{now} - s, t_{now}]$, $\kappa > 0$ is the smoothing coefficient. Typically, we set $s = 20$, $\kappa = 2$.

$flag_i$ represents the bitrate switching state of a segment. If $r[n] \geq r[n-1] \&\& r[n-1] \geq r[n-2]$ (or $r[n] \leq r[n-1] \&\& r[n-1] \leq r[n-2]$), n^{th} segment is in fluctuant state. If $r[n] > r[n-1] \&\& r[n-1] < r[n-2]$ (or $r[n] < r[n-1] \&\& r[n-1] > r[n-2]$), n^{th} segment is in steady state. The more unstable the network bandwidth is, the larger the value of $m[n]$ is, and the client should be more cautious to bitrate switching.

Algorithm 1. bitrate switching logic

Inputs: $r[n]$, $r[n-1]$, $m[n]$

Output: $r[n]$

```
1. if $r[n] > r[n-1]$ then
2.    $Count$ ++;
3.    if $Count \geq m[n]$ then
```

```
4.        Count = 0;
5.    else
6.        $r[n] = r[n-1]$;
7.    end if
8. else if $r[n] = r[n-1]$ then
9.    Count ++;
10. else
11.   Count = 0;
12. end if
```

The bitrate switching logic is shown in Algorithm 1. When requesting the n^{th} segment, if the target bitrate $r[n]$ is smaller than the previous bitrate $r[n-1]$, the client should switch to $r[n]$ immediately to avoid buffer underflow. If the target bitrate is larger than the previous, the client switches to $r[n]$ only when it has requested $r[n-1]$ for at least m[n] segments, otherwise the previous bitrate $r[n-1]$ should be maintained.

3.3 PASA Algorithm

PASA algorithm is based on PANDA and incorporates the proposed fairness controller and bitrate switching logic. Its detail is shown in Algorithm 2. The requesting and downloading process of n^{th} segment is divided into 5 steps.

Algorithm 2. PASA algorithm

At the beginning of each step *n*:

(1) Estimate the bandwidth share $x[n]$ by

$$x[n] = F_{probe}[n] + \delta \cdot F_{fair}[n]. \quad \textbf{(12)}$$

(2) Smooth out $x[n]$ to get filtered version $y[n]$ by

$$\frac{y[n] - y[n-1]}{\hat{t}[n-1]} = -\beta \cdot (y[n-1] - x[n]). \quad \textbf{(13)}$$

(3) Quantize $y[n]$ to the discrete target bitrate $r[n] \in R$ by (5).

(4) Control bitrate switching with Algorithm 1.

(5) Schedule the waiting time to request next segment by

$$t_{off}[n] = \begin{cases} 0 & if\ B[n-1] < \Delta \\ \gamma \cdot (B[n-1] - \Delta) & otherwise \end{cases}. \quad \textbf{(14)}$$

At the first step, the available bandwidth is estimated by a probe factor and a fairness factor. The probing factor reflects the probing mechanism of PANDA. It is derived from (2):

$$F_{probe}[n] = a \cdot (\omega - \max(0, x[n-1] - T[n-1] + \omega)) \cdot \hat{t}[n-1] + x[n-1].. \quad \textbf{(15)}$$

It should be noted that we use $\hat{t}[n]$ to control the convergence rate rather than $t[n]$ as in PANDA. $t[n]$ is the actual interval time between two requests. It is mainly decided by the downloading time of the segment, which changes a lot when network bandwidth fluctuates strongly. In the experiment, we found that if the bandwidth is high, the value of $t[n]$ is small, which makes it take a long time to converge to the equilibrium state. If network congestion occurs, the value of $t[n]$ is large, even resulting in $1 - \beta \cdot t < 0$. So we limit the scope of $t[n]$:

$$\hat{t}[n] = \begin{cases} t_{up}, & if\ t[n] \geq t_{up} \\ t[n], & if\ t_{down} < t[n] < t_{up}, \\ t_{down}, & otherwise \end{cases} \quad \textbf{(16)}$$

where t_{up} is a constant that guarantees $1-\beta \cdot t_{up} > 0$, and t_{down} guarantees the additive increase rate $a \cdot \omega \cdot t_{down}$ is large enough. $0 < \delta < 1$ is the weight of fairness factor.

The second step uses the EWMA smoother with $\hat{t}[n]$ to generate smoothed bandwidth $y[n]$. The third step uses the same quantizer as PANDA to generate the target bitrate $r[n]$, which is filter by the bitrate switching logic in fourth step.

The fifth step schedules the waiting time to request next segment. If the client keeps requesting without interval, buffer overflow may happen when the available bandwidth is high. However, if the client requests with a periodical interval, Jiang et al. [21] shows the initial conditions of clients can lead to unfairness in bandwidth allocation. So we use a randomized scheduler. Δ is a random number chosen from the range $(B_{target} - \chi, B_{target} + \chi]$.

3.4 Stability Analysis for PASA Algorithm

It is assumed that the TCP throughput is equal to the link capacity W and the N clients are identical. When close to equilibrium state, each client's throughput satisfies $T = C/N$ and the interval time between two requests matches the playback duration of a video segment, namely $\hat{t}[n-1] = \tau$. So (12) can be expressed as:

$$\begin{aligned} x[n] &= -a\tau \cdot (x[n-1] - \frac{N}{W}) + \delta \cdot \frac{\frac{N}{W} - x[n-1]}{\frac{N}{W}} + x[n-1] \\ &= (1 - a\tau - \frac{\delta N}{W}) \cdot x[n-1] + a\tau \cdot \frac{\delta W}{N} + \delta. \end{aligned} \quad \textbf{(17)}$$

The sequence converges if and only if $|1 - a\tau - \frac{\delta N}{W}| < 1$, so we can conclude the proposed algorithm is stable if the following stability condition is satisfied:

$$0 < a\tau + \frac{\delta N}{W} < 2. \quad \textbf{(18)}$$

4 Experimental Results

Our experiment is implemented by using the ns-3 simulator [20]. The network topology used in the experience is shown in Fig. 1.

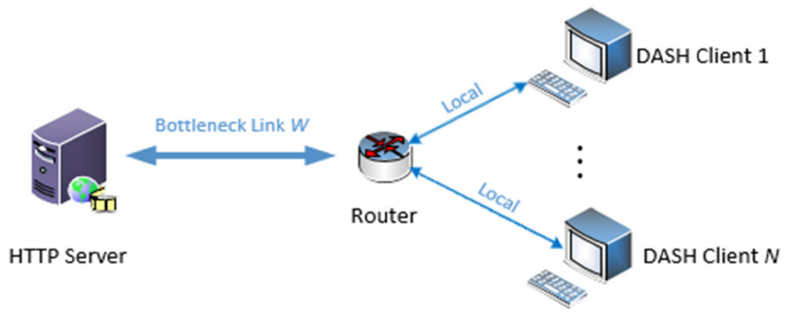


Fig. 1. The network topology used in the experiment

At the sever side, the length of video segment is set as $\tau = 2s$ and the video is pre-encoded into 20 bitrates: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1.0, 1.2, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0 Mbps.

At the client side, there are N identical clients connected to the server via a bottleneck link. Meanwhile, the initial request bitrate is set as 0.1Mbps. And $a = 0.2, \delta = 0.08,$ which satisfy the constraint of stability condition (18). Other parameters are set as follows: $\omega = 0.3,\ \beta = 0.2,\ \gamma = 0.8,\ \in = 0.15,\ B_{target} = 30$ and $\chi = 3$.

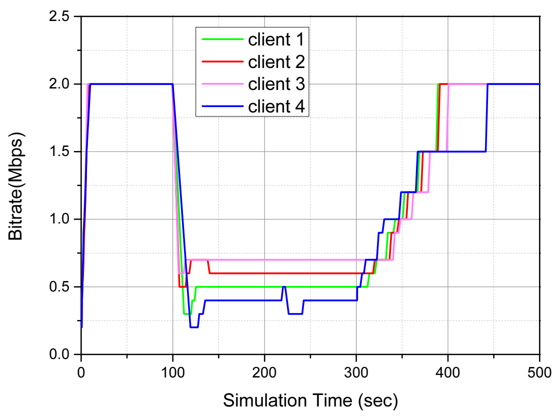
To evaluate the performance of PASA, we choose PANDA and ESTC for reference in the following three cases: four clients over a link with long-periodic throughput changes, four clients over a link with short-periodic throughput changes, and three clients over a Wi-Fi network with background TCP traffic.

We evaluate fairness and stability based on metrics defined in [21]:

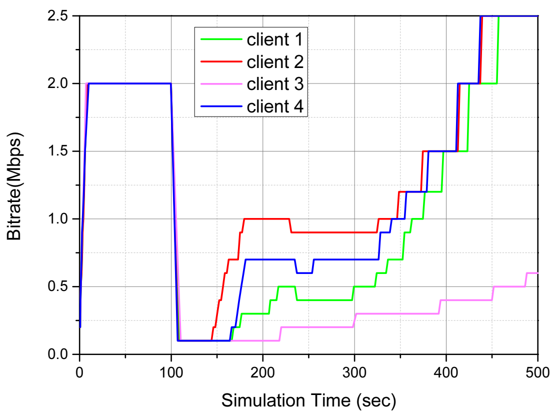
- Unfairness: The unfairness at time t is $\sqrt{1 - JainFair_i}$, where $JainFair_i$ is the Jain fairness index of bitrate $r_{i,t}$ at time t over all clients.
- Instability: The instability for client i at time t is $\dfrac{\sum_{d=0}^{k-1} | r_{i,t-d} - r_{i,t-d-1} | \cdot \omega(d)}{\sum_{d=0}^{k-1} r_{i,t-d} \cdot \omega(d)}$, where $\omega(d) = k - d$. We set $k = 20s$.

4.1 Four Clients over a Link with Long-Periodic Throughput Changes

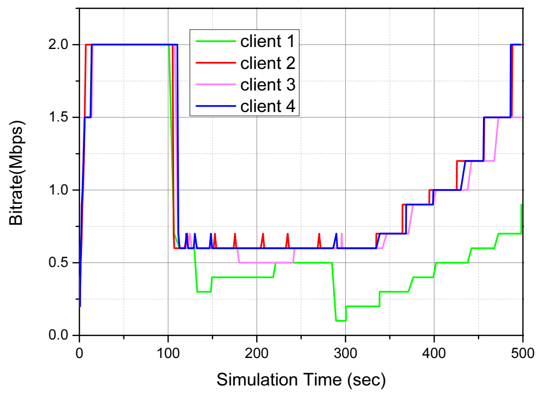
The link bandwidth begins with 10Mbps, drops to 2.5Mbps at 100 seconds and returns to 10Mbps at 300 seconds. Note that we use the same fast startup approach for the three algorithms in this experiment. Fig. 2 depicts the bitrate changes of four clients.



(a) PASA



(b) PANDA



(c) ESTC

Fig. 2. The allocated bitrates for clients in link with long-periodic throughput changes

When the link bandwidth drops to 2.5Mbps, PASA quickly switches to the appropriate lower bitrate and remains stable during 100s to 300s. As for PANDA, it switches to the lowest bitrate and takes a long time to converge to the normal bitrate. It is because link congestion leads to $1-\beta \cdot t<0$ (as analysis in Section 3.3). In ESTC, there are three clients decreasing to the same bitrate and remaining relatively stable, but one client's bitrate is significantly lower than the average. It is because ESTC uses a simple fairness strategy: If a client's bitrate is higher than the average bitrate of all the clients, the server forces it to reduce its bitrate. But the server does nothing to the client whose bitrate is lower than the average.

When the link bandwidth increases to 10M in 300s, PASA converges to the fair bandwidth sharing state within the shortest time. PANDA spends 150 seconds and ESTC spends 200 seconds, respectively. Each of them has a client whose bitrate is significantly lower than other clients, leading to serious unfairness.

Fig. 3 shows the unfairness and instability metrics of the three algorithms. It can be seen that the fairness of PASA is much better than PANDA because of the fairness controller. However, ESTC shows better stability than PASA, because it uses tough smoothing strategy: estimating the available bandwidth with recent 20 segments and maintaining a bitrate level for at least 5 segments before switching to a new bitrate. The cost is that ESTC is not sensitive to bandwidth fluctuation and has the slowest convergence rate.

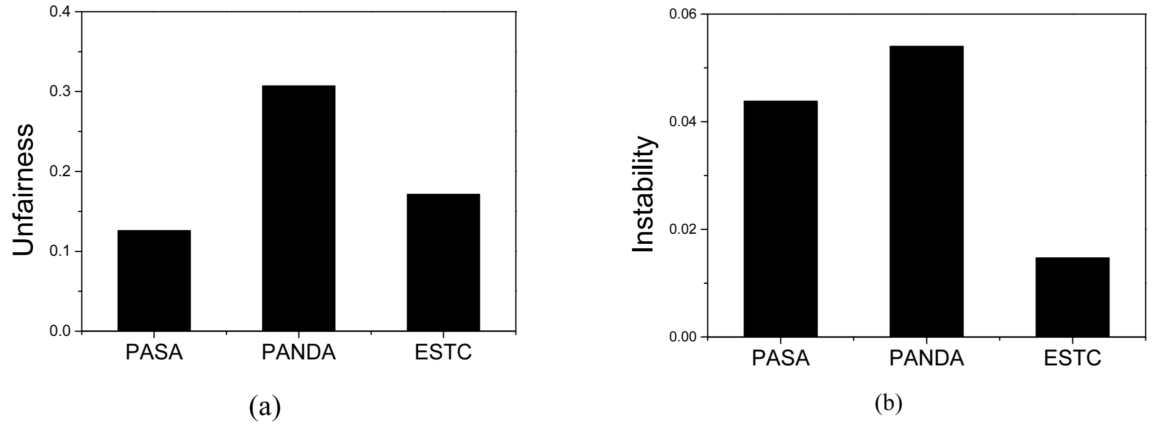
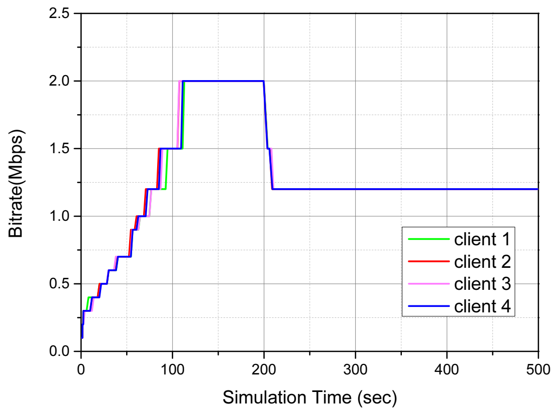


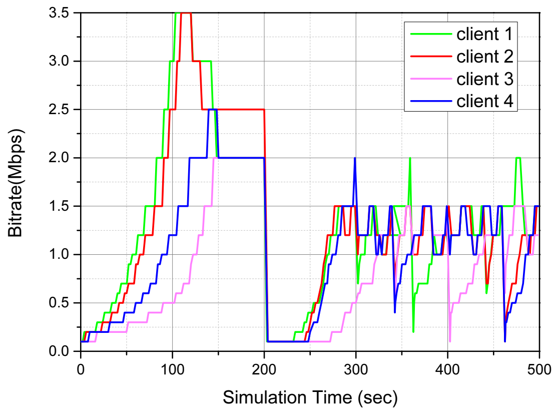
Fig. 3. The comparison of unfairness and instability in link with long-periodic throughput changes

4.2 Four Clients over a Link with Short-Periodic Throughput Changes

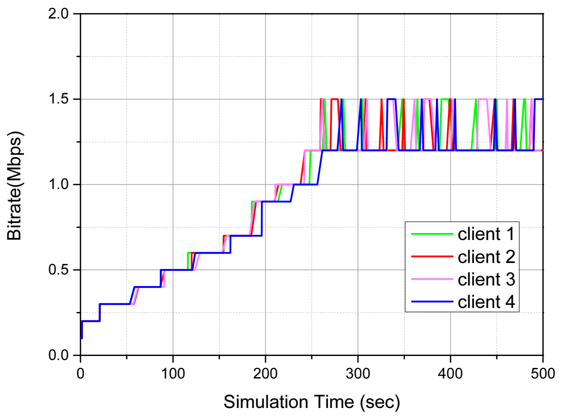
The link bandwidth starts to fluctuate between 10Mbps and 5Mbps from 200 seconds. As shown in Fig. 4(a), PASA presents amazing stability and fairness. The reasons of performance improvement are as follows. First, the revision to the value of inter-request time $t[n]$ guarantees that the estimated bandwidth $x[n]$ and smoothed bitrate $y[n]$ do not deviate from actual bandwidth even if the network bandwidth changes a lot. So, its stability is significantly better than PANDA. Second, PASA uses two smoothing strategies: the EWMA smoother and the bitrate switching logic. They work together to ensure PASA has both good stability and good responsiveness to bandwidth fluctuation.



(a) PASA



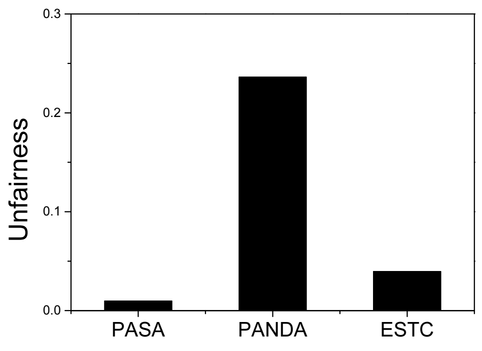
(b) PANDA



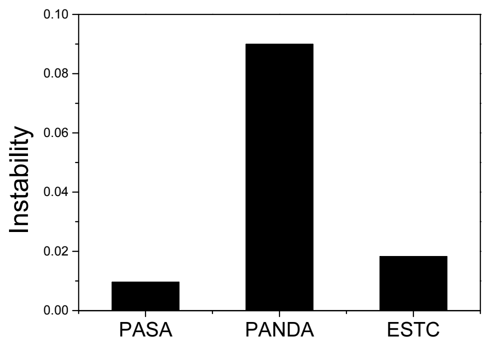
(c) ESTC

Fig. 4. The allocated bitrates for clients in link with short-periodic throughput changes

Fig. 5 shows PASA has the best fairness and stability. Compared to experiment 4.1, here PASA shows better stability than ESTC. Because when the network bandwidth fluctuates drastically, the performance of the proposed bitrate switching logic is much better than general stability strategies.



(a)

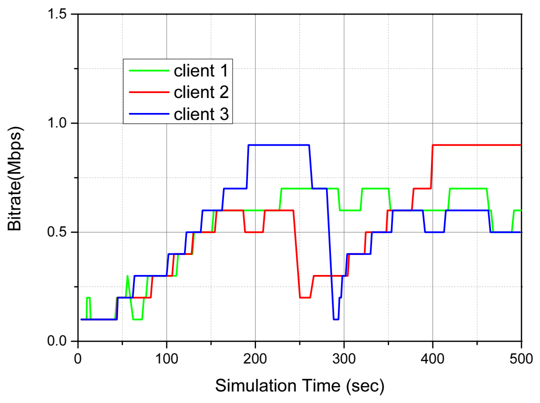


(b)

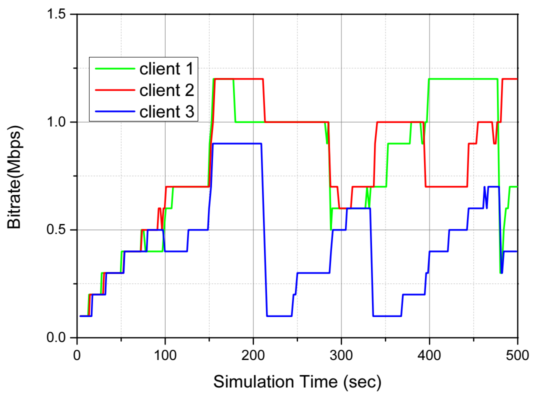
Fig. 5. The comparison of unfairness and instability in link with short- periodic throughput changes

4.3 Three Clients over a Wi-Fi Network with Background TCP Traffic

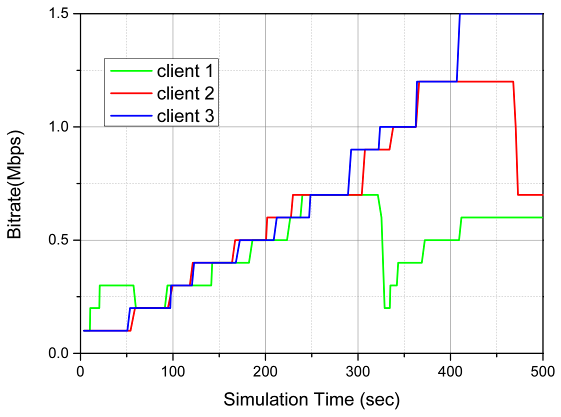
Here we evaluate PASA's performance in Wi-Fi network with interference of four background TCP flows. Fig. 6 and Fig. 7 shows that PASA has the best fairness and much better stability than PANDA even with channel interference. Although ESTC shows best stability, it has the longest convergence time and converges to quite unfair bandwidth sharing state at last.



(a) PASA

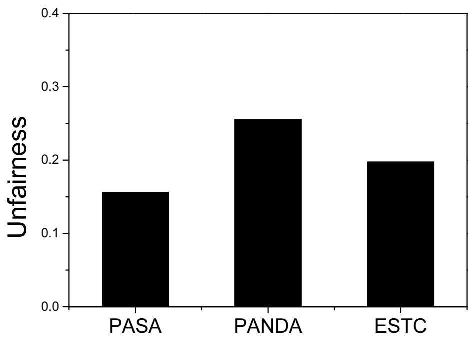


(b) PANDA

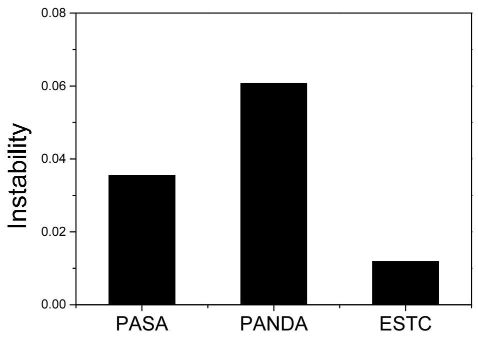


(c) ESTC

Fig. 6. The allocated bitrates for clients in Wi-Fi network with background TCP traffic



(a)



(b)

Fig. 7. The comparison of unfairness and instability in Wi-Fi network

In summary, the unfairness of PASA is 65% and 40% lower than PANDA and ESTC respectively. Because PASA utilizes the average bitrate of the link, making it estimate the throughput correctly. By combing probe factor and fairness factor, PASA can converge to fair-share state with the shortest time when the bandwidth changes. Moreover, PASA's bitrate switching logic shows good performance especially when the bandwidth changes frequently. And the PASA client outperforms the PANDA client by 50% reduction in instability.

5 Conclusions

This paper proposes PASA, a rate adaptive algorithm aiming to improve fairness and stability. We prove that purely client-driven method cannot realize fairness among clients because of the existence of OFF intervals. So PASA utilizes other clients' average bitrate which is offered by the server to help converge to the fair bandwidth sharing state. In PASA, a bitrate switching logic is also designed to make a good balance between smoothness and responsiveness to the bandwidth fluctuation. The algorithm is implemented in ns-3 simulator and compared with existing solutions. The experiment results show PASA outperforms state-of-the-art algorithms by 40% reduction in unfairness and has excellent stability and convergence rate.

In the future, we will consider the situation of multiple servers and test PASA in the realistic network environment.

Acknowledgements

This work is partially supported by the Special Fund for National High-tech R&D Program under Grant NO. 2017ZX03001019. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] Y. Zhang, W. Xie, 5G Mobile and Broadcast TV Convergence Network, Journal of Network New Media 7(5)(2018) 6-12.

[2] Cisco, Cisco Visual Networking Index: Forecast and Trends, 2017-2022. <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html>, 2018 (accessed 20.01.18).

[3] Standard I.S.O. Dynamic adaptive streaming over HTTP (DASH)— part 1: media presentation description and segment formats, ISO/IEC (2014) 23009-1.

[4] T. Stockhammer, Dynamic adaptive streaming over HTTP: standards and design principles, in: Proc. the Second Annual ACM Conference on Multimedia Systems, 2011.

[5] B. Rainer, D. Posch, H. Hellwagner, Investigating the performance of pull-based dynamic adaptive streaming in NDN, IEEE Journal on Selected Areas in Communications 34(8)(2016) 2130-2140.

[6] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, Z. Wang, A quality-of-experience index for streaming video, IEEE Journal of Selected Topics in Signal Processing 11(1)(2017) 154-166.

[7] D.Z. Rodríguez, Z. Wang, R.L. Rosa, G. Bressan, The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over HTTP, EURASIP Journal on Wireless Communications and Networking 2014(1)(2014) 216.

[8] S. Mahapatra, Quality of Experience Driven Rate Adaptation for Adaptive HTTP Streaming, IEEE Transactions on Broadcasting 64(2)(2018) 602-620.

[9] B. Wang, F. Ren, Towards forward-looking online bitrate adaptation for dash, in: Proc. the 2017 ACM on Multimedia Conference, 2017.

[10] H. Mao, R. Netravali, M. Alizadeh, Neural adaptive video streaming with pensieve, in: Proc. the Conference of the ACM Special Interest Group on Data Communication, 2017.

[11] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A.C. Begen, D. Oran, Probe and adapt: Rate adaptation for HTTP video streaming at scale, IEEE Journal on Selected Areas in Communications 32(4)(2014) 719-733.

[12] H. Yuan, H. Fu, J. Liu, J. Hou, S. Kwong, Non-cooperative game theory based rate adaptation for dynamic video streaming over HTTP, IEEE Transactions on Mobile Computing 17(10)(2018) 2334-2348.

[13] S. Akhshabi, L. Anantakrishnan, A.C. Begen, C. Dovrolis, What happens when HTTP adaptive streaming players compete for bandwidth? In: Proc. the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, 2012.

[14] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, P. Tran-Gia, A survey on quality of experience of HTTP adaptive streaming, IEEE Communications Surveys & Tutorials 17(1)(2015) 469-492.

[15] A. Bentaleb, A.C. Begen, R. Zimmermann, SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking, in: Proc the 2016 ACM on Multimedia Conference. ACM, 2016.

[16] O. El Marai, T. Taleb, M. Menacer, M. Koudil, On improving video streaming efficiency, fairness, stability, and convergence time through client-server cooperation, IEEE Transactions on Broadcasting 64(1)(2018) 11-25.

[17] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, D. Rossi, Dynamic adaptive video streaming: towards a systematic comparison of ICN and TCP/IP, IEEE Transactions on Multimedia 19(10)(2017) 2166-2181.

[18] K. Spiteri, R. Urgaonkar, R.K. Sitaraman, BOLA: Near-optimal bitrate adaptation for online videos, in: INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, 2016.

[19] R.K. Mok, X. Luo, E.W. Chan, R.K. Chang, QDASH: a QoE-aware DASH system, in: Proc. the 3rd Multimedia Systems Conference, 2012.

[20] Network Simulator 3. <https://www.nsnam.org>, (accessed 19.01.20).

[21] J. Jiang, V. Sekar, H. Zhang, Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive, IEEE/ACM Transactions on Networking 22(1)(2014) 326-340.