面向视频流媒体直播的码率自适应算法研究

金肱羽[1],武霄泳[2],张志龙[1],刘丹谱[1],尹方方[3]

(1.北京邮电大学,先进信息网络北京实验室,网络体系构建与融合北京市重点实验室,北京 100876;
2.上海无线电设备研究所,上海 201109;3.中国传媒大学媒体融合与传播国家重点实验室,北京 100024)

摘要:随着移动互联网技术的发展,视频流媒体直播已成为现今的热门应用。如何改善直播过程中的用户体验质量是学术界和产业界关注的焦点。码率自适应算法通过动态切换视频码率,使业务适应信道的时变特征,可以有效提升用户的体验质量。因此,研究面向流媒体直播场景的码率自适应算法具有重要意义。本文以视频流媒体直播为背景,以提高用户体验质量为目标,充分考虑业务的时延需求和传输环境特征,提出了一种基于PID控制的码率自适应算法。

关键词:流媒体直播;码率自适应算法;用户体验质量;DASH
中图分类号:O422 **文献标识码:**A **文章编号:**1673-4793(2022)01-0008-07

A rate adaptation algorithm for real-time video streaming

JIN Gongyu[1], WU Xiaoyong[2], ZHANG Zhilong[1], LIU Danpu[1], YIN Fangfang[3]

(1.Beijing Laboratory of Advanced Information Network, Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China; 2.Shanghai Radio Equipment Research Institute, Shanghai 201109, China; 3. State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China)

Abstract: With the development of the Internet, real-time video streaming has become one of the most popular applications. An important focus of real-time video is how to improve the quality of experience. A proper rate adaptation scheme can dynamically switch the bitrate of a video to adapt to a time-varying channel, which effectively improves the quality of experience. Therefore, it is important to research on the adaptive bitrate algorithms for live streaming. In this paper, we propose an adaptive bitrate algorithm based on PID controller for live video streaming, with the goal of improving the quality of experience, while taking into full consideration the time delay requirements and transmission environment characteristics.

Key words: real-time video streaming; adaptive bitrate algorithm; quality of experience; DASH

1 引言

直播作为流媒体技术的重要应用形式,近年来得到快速发展。相较于传统的流媒体点播业务,视频直播更具挑战性。直播业务互动性更强,对于实时性要求更高。因此,为提高直播用户的体验质量,不仅需要考虑视频质量、平滑度、卡顿等因素,还需尽量降低端到端时延。

基金项目:媒体融合与传播国家重点实验室(中国传媒大学)开放课题资助(SKLMCC2021KF009);国家自然科学基金(61971069)
第一作者:金肱羽(1999-),女,硕士研究生,主要从事无线多媒体研究。E-mail:gyjin@bupt.edu.cn
通讯作者:尹方方(1986-),女,师资博士后,主要从事未来网络资源管理研究。E-mail:yinff@cuc.edu.cn

码率自适应算法通过动态选择合适的视频码率，使业务适应信道的时变特征，可以有效改善用户的体验质量。现有的码率自适应算法多是针对点播场景设计的，并未考虑端到端的时延因素，直接应用在直播场景很难获得较好的业务性能。

近年来，已有一些针对流媒体直播的码率自适应算法被相继提出。例如，Wang 等人提出了 HCA[1]，通过前馈反馈机制降低时延；Xie 等人提出了动态调整缓冲区阈值的 DTBB[2]避免卡顿；Zhang 等人提出以降低时延为目标的 LAPAS 算法[3]；Peng 等人针对直播场景下，提出通过调整播放速度、增加跳帧等时延控制机制并且选择合适的码率，从而达到降低时延的效果[4]。

然而，很多针对直播的码率自适应算法未能做到在用户体验质量的影响因素之间做到很好的均衡，例如，DTBB 算法只考虑当前客户端的缓冲区容量，而并未考虑到时延机制优化。因此，本文提出了基于 PID 的码率自适应算法。该算法综合考虑网络波动状况以及缓冲区占用情况进行稳健的码率决策，同时对直播场景的时延机制进行优化。仿真实验表明，与基线算法相比，本文所提算法可以有效提高用户体验质量。

2　流媒体直播架构及系统模型

2.1　直播架构

基于 HTTP 的自适应流媒体 (HTTP Adaptive Streaming, HAS)是目前被广泛采用的流媒体直播技术之一。图 1 展示了使用 HAS 的流媒体直播的典型框架[5]。其中，主播端提供直播内容并上传到 HTTP 服务器，原始视频被切割成相等时长的视频切片，同一个视频切片被编码为不同的版本，每个版本对应一个视频码率。客户端可向服务器请求合适的下一个视频切片，以降低时延和卡顿，从而提高用户体验质量。由内容分发服务器将特定码率的视频切片文件发送到客户端。

2.2　系统模型

系统模型由视频传输模型、缓冲区模型、时延控制模型以及 QoE 模型组成。

a)视频传输模型

主播端从 $t = 0$ 时刻开始采集并上传视频内容到服务器，服务器对视频进行编码和切片处理。在基于 HTTP 的动态自适应流标准 (Dynamic Adaptive Streaming over HTTP, DASH)的视频流媒体架构下，视频切片的时长通常为 2-10 秒，而面向直播场景时，通常使用时长更短的视频切片。在本文中，视频切片的时长用 L 秒进行表示。

假设直播过程中视频内容共分为 N 个视频切片。在服务器端，每个视频切片都有 K 个不同码率和分辨率的版本，并且其大小满足 $V_1 < V_2 < \ldots < V_K$。K 个码率等级对应的码率大小用 $\{R_1, R_2, \cdots, R_K\}$ 表示。用户在观看直播时，按照上传顺序从服务器端下载视频切片。由于视频内容是以视频切片为单位，用户在下载每个视频切片时都可以从 K 个版本中进行选择。

相比于视频点播，直播的主要特点在于其视频内容是实时生成的。为聚焦问题，本文在建立视频传输模型中，不考虑实时视频封装以及视频编码解码过程的耗时。第 n 个视频切片于 $t = nL$ 时刻开始生成，由于客户端无法向服务器请求还未生成的视频内容，所以客户端开始下载和播放第 n 个视频切片时刻 t_n 需满足以下约束条件：

$$t_n \geq nL, \forall n \in N \tag{1}$$

令 τ_n 代表下载第 n 个视频切片所用的时间，可以表示为：

$$\tau_n = \frac{S_n(R_n)}{C_n} \tag{2}$$

其中，$S_n(R_n)$ 表示第 n 个视频切片对应码率为 R_n 的视频大小，C_n 表示下载过程中网络平均吞吐量。当第 n 个视频切片于 $t_n^e = t_n + \tau_n$ 时刻完成下载，而第 $n+1$ 个视频切片还未完全生成时，即 $t_n^e < (n+1)L$ 情况下，用户

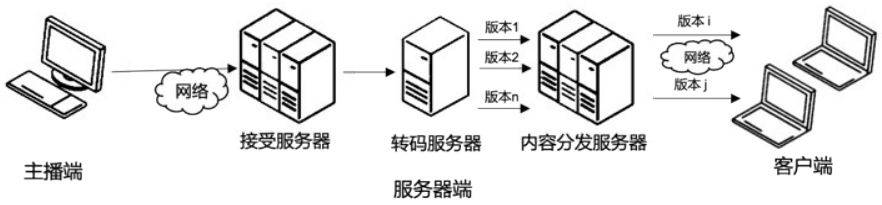

图 1　基于 HAS 的流媒体直播传输系统框架[5]

无法从服务器端获取视频数据,需要等待Δt_n方可进行下一个视频切片的下载;$t_n^e \geq (n+1)L$时,第$n+1$个视频切片需从$t = t_n + \tau_n$时刻开始下载。Δt_n可以表示为:

$$\Delta t_n = \begin{cases} (n+1)L - t_n^e, & t_n^e < (n+1)L \\ 0, & t_n^e \geq (n+1)L \end{cases} \tag{3}$$

b)缓冲区模型

缓冲区位于客户端,其状态示意图如图2所示,视频切片从服务器端下载后会先存储到客户端的缓冲区中,用户播放视频时从缓冲区取出视频内容进行观看。

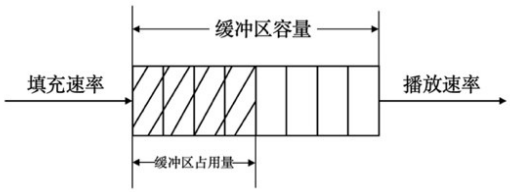

图2 缓冲区状态示意图

缓冲区占用大小由视频填充速率和播放速率共同决定。设$B(t)$表示t时刻缓冲区的占用大小,以视频切片为单位,当下载完成一个视频切片并开始下载下一个切片时,缓冲区大小增加一个视频切片的时长,并且减去下载视频切片过程以及等待时间中客户端播放的视频时长。缓冲区占用量不能小于0,开始下载第$n+1$个视频切片t_{n+1}时缓冲区占用量用B_{n+1}可以表示为:

$$B_{n+1} = \max\left(\max\left(\mathrm{B_n} - \lambda_n\tau_n + \mathrm{L}, 0\right)\lambda_n\Delta t_n, 0\right) \tag{4}$$

其中,λ_n代表着第n个视频的播放速度。在直播场景下,为避免卡顿以及降低时延,引入播放速度控制机制。播放速度会根据需求改变,播放速度的选择会在下一节时延控制模型中进行说明。

在下载过程$t \in \left(nL, nL+\tau_n\right]$中,可以对缓冲区占用量进行求导得到视频填充速率。在下载第n个视频切片过程中,求导过程可以表述为:

$$\frac{dB(t)}{dt} = \frac{L}{\tau_n} - \lambda_n = \frac{R_n L}{R_n \tau_n} - \lambda_n = \frac{C_n}{R_n} - \lambda_n \tag{5}$$

其中,C_n代表下载第n个视频切片过程中的平均吞吐量,R_n代表第n个视频切片对应的码率。

c)时延控制模型

本文所考虑的端到端的时延由两部分构成,一部分为直播用户缓冲区中已经缓冲的视频时长,一部分为存储到服务器端但尚未下载的部分。

$$L_n = B_n + L_n^{\mathrm{server}} \tag{6}$$

时延控制机制由两部分构成:调整视频播放速度以及跳帧机制。调整视频播放速度通过加快或减缓客户端播放视频的速度,达到降低时延及减少卡顿的目的。通过调整播放速度降低时延的能力有限,当时延比较大时,执行跳帧操作,即超过时延阈值时,客户端跳过当前正在进行的下载,直接请求下载下一个关键帧,从而降低时延。

首先,对视频播放速度进行调整。最基本的方法是通过比较实际缓冲区的占用情况和预设的缓冲区阈值来控制播放速度。假设阈值为$\left(B_{\min}, B_{\mathrm{target}}, B_{\max}\right)$,其中$B_{target}$代表目标缓冲区水平,$B_{\min}$和$B_{\max}$分别代表缓冲区阈值的上限和下限。缓冲区阈值需要满足$B_{\min} < B_{\mathrm{target}} < B_{\max}$。缓冲区占用量大于$B_{\max}$,加快视频播放速度,播放速度为正常播放速度的$r_{\mathrm{fast}}$倍;缓冲区占用量小于$B_{\min}$,减缓视频播放速度,播放速度为正常播放速度的$r_{\mathrm{slow}}$倍;其他情况则选择正常的播放速度。

为更好地调整视频播放速度,可以设定两组缓冲区阈值,分别为$\left(B_{\min}^0, B_{\mathrm{target}}^0, B_{\max}^0\right)$以及$\left(B_{\min}^1, B_{\mathrm{target}}^1, B_{\max}^1\right)$,两组缓冲区阈值需满足以下条件:$B_{\min}^0 < B_{\min}^1 < B_{\mathrm{target}}^0 < B_{\mathrm{target}}^1 < B_{\max}^0 < B_{\max}^1$。客户端在设计码率自适应算法时,需要对目标缓冲区水平进行决策,决策变量$target_buffer_n = \{0,1\}$,表明在下载第n个视频分片时所选择的目标缓冲区水平。若$target_buffer_n = 0$,则目标缓冲区水平为B_{target}^0,对应的缓冲区阈值组为$\left(B_{\min}^0, B_{\mathrm{target}}^0, B_{\max}^0\right)$,此时的码率选择更偏向于减小时延,而由于其缓冲区占用较小面临耗尽风险,即视频卡顿风险较大;若$target_buffer_n = 1$,则目标缓冲区水平为B_{target}^1,对应的缓冲区阈值组为$\left(B_{\min}^1, B_{\mathrm{target}}^1, B_{\max}^1\right)$,此时的码率选择更偏向于避免视频卡顿。

综上所述,第n个视频切片的播放速度λ_n可以表示为:

$$\lambda_n = \begin{cases} r_{\mathrm{slow}}, & B_n \in \left[0, B_{\min}^{\mathrm{target\_buffer}_n}\right) \\ 1, & B_n \in \left[B_{\min}^{\mathrm{target\_buffer}_n}, B_{\max}^{\mathrm{target\_buffer}_n}\right) \\ r_{\mathrm{fast}}, & B_n \in \left[B_{\max}^{\mathrm{target\_buffer}_n}, \infty\right) \end{cases} \tag{7}$$

其次,对跳帧的阈值进行选择。对于第n个视频切片执行跳帧的时延阈值表示为l_n,即当$L_n > l_n$时,客户端向服务器请求新的关键帧,跳过的帧数表示为S_n。通常情况下将l_n设置为常数,即每一个视频切片触发跳帧的时延阈值都相同。

d)QoE模型

码率自适应算法以提高QoE为目标[6]，为了明确视频流媒体直播QoE与其影响因素之间的定量关系，需要对QoE进行客观评价。可以将QoE映射为数值，为之后设计码率自适应算法并且进行性能优化和比较建立基础。在评价视频点播业务的性能时，文献[7]提出将各个QoE指标进行加权平均，得到的数值作为QoE估计值。面向直播场景时，QoE不仅需要考虑视频质量、码率切换频次、卡顿时长等影响因素以外，还需计算端到端的时延以及跳帧时长对QoE数值的影响。因此，本文建立如下直播QoE模型：

$$QoE_{\text{total}} = \sum_{n=1}^{N}\left(\theta_q f\left(R_n\right) - \theta_r T_n - \theta_d L_n - \theta_s S_n\right) - \sum_{n=1}^{N-1}\theta_a\left|R_{n+1} - R_n\right| \tag{8}$$

其中，N代表该直播视频总共由N个视频切片组成，QoE_{total}表示N个视频切片的QoE数值的总和，R_n表示第n个视频切片的码率大小；$f\left(R_n\right)$为计算视频质量的函数，通常可以计算视频切片中每帧的视频质量并进行求和，表示为：$f\left(R_n\right) = R_n\tau_{\text{frame}}F_n$。其中，$\tau_{\text{frame}}$表示每帧的时长，由视频画面每秒传输的帧数决定，$F_n$代表第$n$个视频切片所包含的帧数，$f\left(R_n\right)$随着码率增加而增大；$T_n$表示由于缓冲区耗尽而造成的卡顿时长；$L_n$代表端到端的时延；$S_n$表示跳帧的时长；$\left|R_{n+1} - R_n\right|$代表码率切换的幅度；$\theta_q$、$\theta_r$、$\theta_d$、$\theta_s$、$\theta_a$分别代表各个指标的权重指数，由于不同的用户对于视频观看有着不同的偏好，可以根据不同的用户需求以及应用场景设置不同的参数。

该模型不仅体现了服务质量，还体现用户对于不同影响因素的偏好，可用于评价视频流媒体直播的QoE。

本文的设计目标为最大化用户在直播会话过程中的QoE，同时尽量减少卡顿以及端到端的时延。

3　基于PID的码率自适应算法

3.1　基于PID控制的码率自适应算法

PID控制即比例(Proportional)、积分(Integral)、微分(Derivative)控制，是一种常见的闭环控制算法，由于其结构简单、鲁棒性强，常用于工业控制中。PID控制结构示意图如图3所示。

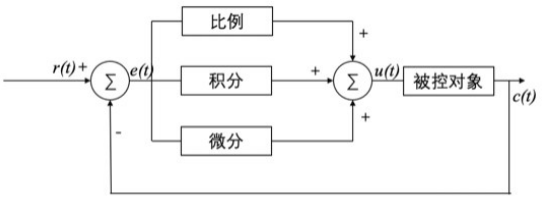

图3 PID控制结构示意图

PID控制器利用实际输出值$c(t)$作为反馈信号，将根据所给定的目标值$r(t)$以及$c(t)$所构成偏差值$e(t) = r(t) - c(t)$,将偏差值进行比例、积分、微分运算，并将计算结果通过线性组合构成控制量$u(t)$对受控对象进行控制。$u(t)$可以表示为：

$$u(t) = K_p e(t) + K_i\int_0^t e(t)dt + K_d\frac{de(t)}{dt} \tag{9}$$

其中，$u(t)$表示PID控制器的输出值，其中K_p、K_i、K_d分别为比例、积分、微分系数。

PID控制的各部分作用如下：

(1)比例部分：比例部分直接通过偏差值对被控对象进行控制；

(2)积分部分：单纯使用比例部分进行控制时，由于存在静态误差，通过积分对过去偏差值的累积即可消除静态误差；

(3)微分部分：微分控制的主要作用是减少超调量，避免因为实际输出值变化过快而超过目标值，从而增加系统的稳定性。

在观看视频直播的过程中，当缓冲区占用量过大时，说明当前所选择的视频切片码率较小，支持选择更高码率的版本，且端到端的时延也会增加；而当缓冲区占用量较小时，说明当前所选择的视频切片码率较大，缓冲区即将耗尽，面临卡顿的风险。因此，可通过缓冲区占用情况作为反馈信号，选择合适的目标缓冲区阈值作为目标给定值，结合PID控制器，并且利用预估网络吞吐量一起对视频切片的码率进行选择。

基于PID控制的码率自适应算法原理框图如图4所示，该算法由以下几部分组成：

(1)目标缓冲区选择

根据2.2节中提到的时延控制模型，提供两种目标缓冲区水平供以选择，决策集合为$target_buffer_n = \{0,1\}$，所对应的目标缓冲区水平分别为$B_{\text{target}}^0$以及$B_{\text{target}}^1$。当$target_buffer_n = 0$时，偏向于减小时延，但有卡顿风险；$target_buffer_n = 1$时，偏向于避免卡顿，同时会增加一定

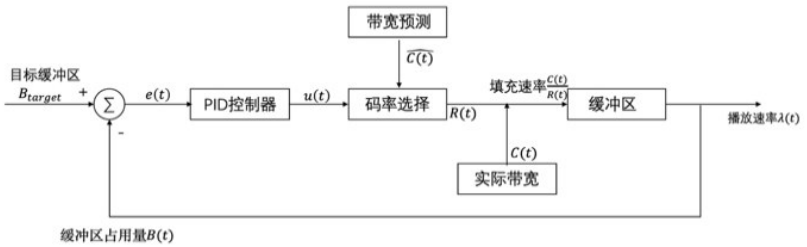

图4 基于PID的码率自适应算法的结构框图

时延。因此,需要在卡顿和时延之间进行权衡,选择合适的目标缓冲区水平从而提高整体用户体验质量。目标缓冲区的选择使用文献[4]提出的目标缓冲区选择方法,第n个视频切片的目标缓冲区水平可利用以下公式进行决策:

$$target\_buffer_n=\begin{cases}1, & B_{n-1}\in\left[B^0_{\min},B^0_{\max}\right)\\ 0, & \text{else}\end{cases} \tag{10}$$

(2)PID控制器

PID控制器的输入分别为:目标缓冲区B_{target}以及现有缓冲区占用大小$B(t)$,则偏差值$e(t)$可以表示为:

$$e(t)=B_{\text{target}}-B(t) \tag{11}$$

根据PID控制模型,PID控制器的输出量$u(t)$如下:

$$u(t)=K_p e(t)+K_i\int_0^t e(\tau)d\tau+K_d\frac{de(t)}{dt} \tag{12}$$

由于在实际情况中,视频播放过程或以视频帧为单位或以视频切片为单位进行播放,$e(t)$实际上是离散变量。本文中以视频切片为单位,并且考虑过去五个视频切片的偏差值,将$u(t)$离散化表示为:

$$\begin{aligned}u(n)=K_p e(n)+K_i\sum_{i=1}^{5}e(n-i)\\ +K_d\left[e(n)-e(n-1)\right]\end{aligned} \tag{13}$$

$e(n)$表示为:

$$e(n)=B^n_{\text{target}}-B_n \tag{14}$$

其中,$e(n)$代表第n个视频切片开始下载时刻的偏差值,B^n_{target}表示第n个视频切片的目标缓冲区水平,B_n代表第n个视频切片开始下载时刻的缓冲区占用量。

(3)吞吐量预测

利用平均值法即通过计算前5个视频切片的平均网络吞吐量对未来网络吞吐量进行预测。预测的第n个视频切片的平均网络吞吐量可表示为:

$$\widehat{C_n}=\frac{\sum_{i=1}^{5}C_{n-i}}{5} \tag{15}$$

其中,$\widehat{C_n}$代表预测第n个视频切片的平均网络吞吐量,C_{n-i}代表过去第$n-i$个视频切片下载过程中的平均网络吞吐量。

(4)码率选择

码率选择是PID模型中的控制进程部分。引入PID控制器的目标是为了通过控制码率的选择使缓冲区的占用水平接近目标缓冲区,避免因缓冲区过大或过小造成的QoE下降。开始下载第n个视频切片时,对视频切片n的码率进行决策,目的是使$B_{n+1}=B_{\text{target}}$,此时缓冲区的占用量为$B_n$。

在$t=nL$时刻,偏差值为$e(n)=B_{\text{target}}-B_n$。在本文中,PID控制器的输出值表示为$u(n)$。

下载时间过长会导致下载时间内缓冲区耗尽,引起卡顿。为便于分析,令$\tau_n=L$,根据第2节缓冲区模型中公式(5)对缓冲区占用量进行求导,可以表示为:

$$\frac{B_{n+1}-B_n}{L}=\frac{\widehat{C_n}}{R_n}-\lambda_n \tag{16}$$

$B_{n+1}-B_n$即偏差量u_n,目标码率可以表示为:

$$R_{\text{target}}=\frac{\widehat{C_n}}{\frac{u_n}{L}+\lambda_n} \tag{17}$$

第n个视频切片的码率选择不超过R_{target}的最大码率,表示为:

$$R_n=\max\left\{R_i|R_i\le R_{\text{target}}\right\},i\in[1,K] \tag{18}$$

同时,需要考虑到跳帧机制对用户体验的影响。一方面,跳帧可以减少端到端时延,对QoE产生积极影响;另一方面,跳帧会导致用户跳过部分画面,视频观看的连续性下降,对QoE产生消极影响。为提高QoE,本文采用文献[4]所提方法,自适应性调整触发跳帧的延迟阈值。具体的,在客户端下载第n个视频切片时,如果端到端的时延高于l_n,则触发跳帧且跳帧数量为S_n。用QoE_{sp}和QoE_{sn}分别表示QoE的积极和消极影响,并采用如下模型:

$$QoE_{\text{sp}}=\theta_d\eta l_n S_n \tag{19}$$

$$QoE_{\text{sn}}=\theta_q R_n\tau_{\text{frame}}S_n+\theta_s\tau_{\text{frame}}S_n \tag{20}$$

其中,ηl_n用于估计不触发跳帧情况时的平均时延,τ_{frame}表示视频帧的时长,当$QoE_{\text{sp}}>QoE_{\text{sn}}$触发跳帧。$l_n$可以设

置为：

$$l_n = \frac{\theta_q R_n \tau_{\text{frame}} S_n + \theta_s \tau_{\text{frame}} S_n}{\theta_d \eta l_n S_n} \tag{21}$$

码率自适应过程的算法如表1所示：

表1 码率自适应算法

码率自适应算法
输入 B_n ，$e[1\cdots n-1]$
输出 $R_n, target\_buffer_n, l_n, e[1\cdots n]$
1. 估计吞吐量 $\widehat{C_n}$
2. 设定参数 K_p，K_i，K_d
3. 根据公式(10)得到目标缓冲区决策变量 $target\_buffer_n$ ，并根据决策变量确定第 n 个视频切片的目标缓冲区阈值 B^n_{target}
4. 由公式(7)得到第 n 个视频切片的播放速度 λ_n
5. 更新开始下载第 n 个视频切片时缓冲区占用量的偏差值，即 $e[n] \leftarrow B^n_{\text{target}} - B_n$
6. 计算该视频切片开始下载时PID控制器的输出量 u_n，即 $u_n \leftarrow K_p e[n] + K_i \sum_{i=1}^{5} e[n-i] + K_d\{e[n]-e[n-1]\}$
7. 计算第 n 个视频切片的目标码率，即 $R_{\text{target}} \leftarrow \frac{\widehat{C_n}}{\frac{u_n}{L}+\lambda_n}$
8. 第 n 个视频切片的码率 R_n 选择不超过 R_{target} 的最大码率，即 $R_n \leftarrow \max\{R_i \mid R_i \leqslant R_{\text{target}}\}, i \in [1,K]$
9. 由公式(21)计算跳帧阈值，即 $l_n \leftarrow \frac{\theta_q R_n \tau_{\text{frame}} S_n + \theta_s \tau_{\text{frame}} S_n}{\theta_d \eta l_n S_n}$

4　仿真结果

本文采用的仿真环境源于2019年ACM Multimedia的为提高用户体验质量而进行的流媒体直播竞赛的仿真环境[8]。在仿真过程中，选择不同直播场景以及网络环境的数据集，其中每种网络都有20个数据文件，对应20种网络条件。根据公式(8)计算用户QoE，平均后最后得到该网络下用户的平均QoE。

为了验证所提算法的性能，将所提算法与HYSA[4]、HCA[1]进行性能对比。

HYSA算法引入播放速度控制机制，选择合适的目标缓冲区，控制视频播放速度；估计跳帧的影响，调整跳帧阈值；估计下一个视频切片的时延，选择使时延最小的码率。

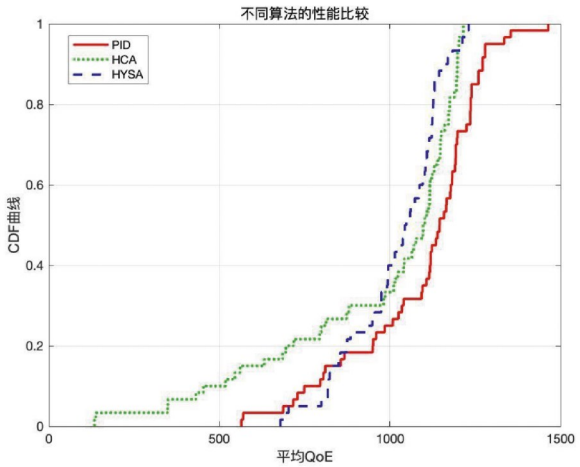

图5 三种算法的平均QoE的CDF曲线

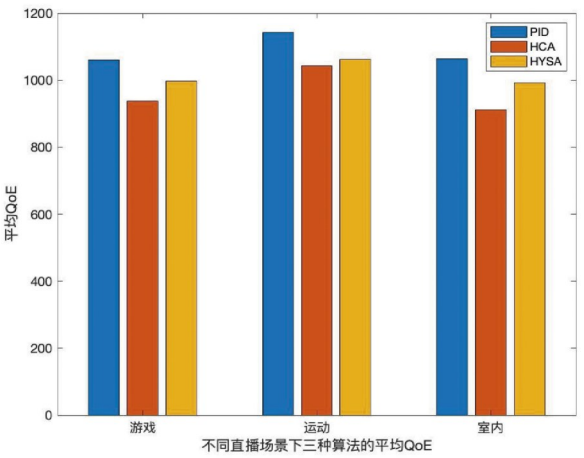

图6 不同直播场景下三种算法的平均QoE

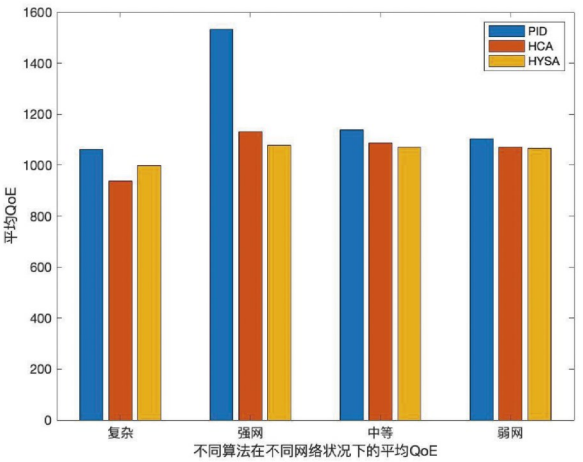

图7 不同网络环境下三种算法的平均QoE

表2 不同算法各项指标比较

算法	平均QoE	视频质量(kbps)	卡顿时长(s)	平均时延(s)
PID	1063.2681	533.0993	96.1667	0.6304
HCA	955.4248	521.6578	79.4288	0.7662
HYSA	998.0177	501.7717	54.4761	0.8176

HCA算法同时采用前馈和反馈机制,通过混合控制使码率决策更精确。其中,前馈是指利用预测的网络吞吐量来优化QoE,以适应网络变化,同时利用缓冲区占用情况作为反馈信号。HCA算法通过前馈控制对网络吞吐量变化做出快速反应,同时借助反馈机制抵抗错误预测的影响。

图5所示为三种算法的平均QoE的CDF曲线图,从图中可以看出所提算法相比于HYSA、HCA能达到更高的QoE,总体性能比较好。

图6展示了三种算法在不同场景下的平均QoE。通过游戏直播、体育直播以及室内直播三种直播场景的视频序列进行比较,所提算法的平均QoE在三种场景下都达到最高。

图7展示了三种算法在四种不同网络环境的平均QoE,通过比较可以看出所提算法在四种网络环境(复杂、强网、中网、弱网)下,与HYSA、HCA算法比较,所提算法的平均QoE均达到最高。

为验证算法在QoE细分指标方面的表现,选择视频质量、卡顿时长以及平均时延3个指标进行仿真和分析。

表2展示了三种算法在以上三个方面的表现。其中视频质量用平均码率表示。总体QoE比HCA算法提高13.1%,比HYSA算法提高6.5%。从细分指标上看,在视频质量方面,所提算法的平均码率最高,视频质量最高,比HYSA算法提高5%。卡顿方面,HYSA、HCA算法偏向于选择较小的码率,所提算法卡顿时长有所增加。在端到端的时延方面,三种算法的时延都比较低,体现针对直播场景实时性要求而设计的特点,所提算法三种算法平均时延最小。从各项指标可以看出,每种算法都有其优势所在,若想达到平均QoE最佳,需要算法在进行码率决策时对QoE的影响因素进行均衡,以HYSA为例,在码率决策时关注的重心在于降低时延,视频质量较低,而所提算法兼顾了视频质量与平均时延,总体的QoE比以上两种算法更高。

5 结论

本文以提高用户体验质量为目标,引入PID控制模型,通过权衡直播场景下影响用户体验质量的关键指标,选择合适的切片码率从而提高用户体验质量。仿真结果表明,所提算法与基线算法相比,整体性能更好,能够有效提高用户的体验质量。

参考文献(References):

[1] B Wang, F Ren and C Zhou. Hybrid Control-Based ABR: Towards Low-Delay Live Streaming[C]//2019 IEEE International Conference on Multimedia and Expo (ICME), 2019.

[2] Lan X, Chao Z, Zhang X, et al. Dynamic threshold based rate adaptation for HTTP live streaming[C]// IEEE International Symposium on Circuits & Systems. IEEE, 2017: 1-4.

[3] Zhang G, Lee J. LAPAS: Latency-Aware Playback-Adaptive Streaming[C]// 2019 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2019.

[4] Peng H, Zhang Y, Yang Y, et al. A Hybrid Control Scheme for Adaptive Live Streaming[C]// the 27th ACM International Conference on Multimedia. ACM, 2019: 2627-2631.

[5] 宋靳锞,张远,王博.HTTP自适应流媒体直播系统中的用户体验质量优化[J].中兴通讯技术,2021,27(01):48-53.

[6] Dobrian F, Sekar V, Awan A, et al. Understanding the Impact of Video Quality on User Engagement[C]// Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM, 2011.

[7] Mao H, Netravali R, Alizadeh M. Neural Adaptive Video Streaming with Pensieve[C]// the Conference of the ACM Special Interest Group on Data Communication. ACM, 2017: 197-210.

[8] Yi G, Yang D, Bentaleb A, et al. The ACM Multimedia 2019 Live Video Streaming Grand Challenge[C]// the 27th ACM International Conference on Multimedia. ACM, 2019: 2622-2626.

编辑:龙学锋,李树锋