计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(9):81−87 [doi: 10.15888/j.cnki.csa.007059]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

屏幕直播系统延迟优化①

付鹏斌, 任　衡, 杨惠荣

(北京工业大学 信息学部, 北京 100124)
通讯作者: 杨惠荣, E-mail: yanghuirong@bjut.edu.cn

摘　要: 低延迟直播是视频直播中一个颇具挑战性的技术难点. 文章从网络推流、首帧延迟、视频编码延迟三个角度进行延迟优化: 首先依据网络状况自适应调整推流码率; 其次实现关键帧缓存算法, 优化首帧打开时间, 使得首帧打开时间减少到 2~4 秒; 然后改进帧内预测模式快速选择算法, 通过降低候选模式个数减少编码复杂度, 在视频质量基本不变的前提下使编码时间最多可降低约 22%. 最后基于以上三种优化策略设计实现屏幕直播系统并应用于课堂教学, 满足课堂屏幕共享低延迟场景.

关键词: 直播延迟; 自适应推流; 关键帧缓存; 帧内预测; 屏幕直播

引用格式: 付鹏斌,任衡,杨惠荣.屏幕直播系统延迟优化.计算机系统应用,2019,28(9):81–87. http://www.c-s-a.org.cn/1003-3254/7059.html

Delay Optimization of Screen Live System

FU Peng-Bin, REN Heng, YANG Hui-Rong

(Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China)

Abstract: Low delay live is a challenging technical difficulty in video live broadcast. In this study, delay optimization is carried out from three perspectives: network push stream, first frame delay, and video coding delay. The details are as follows. Firstly, according to the network condition, the code rate of live pushing stream is adjusted adaptively. Secondly, the key frame caching algorithm is implemented to reduce the opening time of the first frame to 2~4 seconds. Then, a fast selection algorithm is improved for intra-frame prediction mode, the coding complexity is reduced by cutting down the number of candidate modes and the encoding time can be reduced by up to about 22% under the premise that video quality is basically unchanged. Finally, based on the above three optimization strategies, the screen live system is designed, implemented, and applied to classroom teaching to meet the low delay scenes of classroom screen sharing.

Key words: live delay; adaptive push stream; keyframe cache; intra-frame prediction; screen live

智能手机的出现和移动互联网的崛起为观看清晰的直播提供了基础保障, 然而, 随着直播并发量的骤增、网络传输的不确定性等原因, 在直播时会产生不同程度的画面延迟. 市面上基于 HLS(HTTP Live Streaming)[1]协议的直播, 理论延迟在 10 s~30 s 之间, 基于 RTMP (Real Time Messaging Protocol)[2]协议的直播在多终端播放条件下延迟在 5 s~10 s 之间, 无法满足实时直播的需求. 所以如何降低直播延迟, 增强直播的实时性体验成为一个关键性的技术挑战.

本文通过分析直播的各个环节, 从网络推流、首帧打开时间、视频编码三个角度进行了直播延迟优化. 一是依据网络状况自适应调整直播推流码率, 优化推流策略; 二是设计实现关键帧缓存算法, 降低首帧加载延迟; 三是对 x264 算法帧内预测部分进行改进, 降低编码延迟. 最终实现了一个低延迟的屏幕直播系统, 增强了直播的实时性体验.

① 收稿时间: 2019-02-28; 修改时间: 2019-03-22; 采用时间: 2019-03-29; csa 在线出版时间: 2019-09-05

1 网络自适应推流

网络自适应推流的作用是在网络波动较大的情况下, 根据网络带宽条件调整码流, 使得视频播放保持整体流畅. 视频码率越高, 视频质量越好, 对网络带宽的消耗就会越大. 因此当弱网络带宽不足的情况发生时, 如果不对分辨率码率等参数进行调整, 将会造成推流上行网络的负载加大, 进而造成直播延迟.

1.1 网络质量评价

网络质量由上行网络平均带宽和平均往返时延两个指标确定, 它们共同反映网络的传输状况.

上行网络平均带宽的值越大, 代表网络状况越好. 它的值由 Jpcap[3]获取, 检测过程如下:

(1) 获取网络接口列表: 调用函数库中的 getDeviceList() 方法获取本机网络接口列表.

(2) 打开网络接口: 调用 openDevice() 方法对网络接口进行监听. 设置每次最大捕获数据包大小为 65 535 Byte、开启混杂模式接收所有类型的数据包、捕获数据包超时时间设置为 50 ms.

(3) 捕获数据包: 定义类 MyPacketReceiver, 它实现了 PacketReceiver 接口, 并实现接口中的 receivePacket 方法, 然后调用 loopPacket() 方法捕获数据包, 获取数据包后会回调 receivePacket() 方法对数据包做处理.

(4) 实时计算上行网络平均带宽: 统计 60 秒内所捕获的源 IP 地址为视频采集端的数据包大小, 计算上行网络平均带宽.

(5) 定时任务每隔 10 min 重复步骤 (3) 和步骤 (4).

平均往返时延的值越小, 代表网络状况越好. 它的值可以通过 java 编程语言调用 cmd 命令“ping ipAddress -n pingTimes -l length”获取, ipAddress 为 Ngnix-rtmp 服务器主机 IP, pingTimes 为向服务器发送的请求个数, length 为每次发送的数据包大小.

使用网络模拟工具 Clumsy 调整网络状况, 利用上行网络平均带宽和平均往返时延两个指标建立了表 1 三种网络质量模型.

表 1 网络质量模型

网络质量	上行网络平均带宽 (MB/s)	平均往返时延 (ms)
网络优秀	>=0.8	<=5
网络良好	0.4~0.8	5~10
网络较差	<=0.4	>20

1.2 自适应推流

每隔一段时间对网络状况进行检测评价. 当网络优秀时, 缓慢提升采集帧率和分辨率; 当网络良好时, 保持原状, 不改变帧率和分辨率; 当网络较差时, 大幅降低采集帧率和视频分辨率, 自适应过程遵循“快降慢升”的原则. 网络自适应推流策略如图 1 所示.

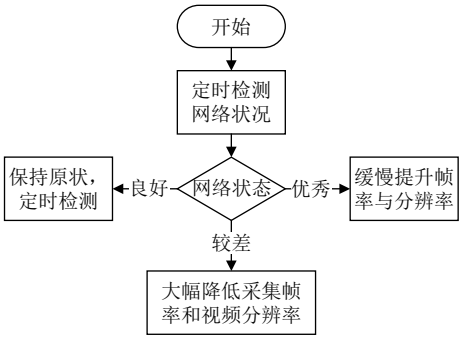


图 1 网络自适应推流策略

如表 2 所示为 6 种推流方案所对应的帧率和分辨率. “快降慢升”的含义即当网络状况较差时, 将当前推流方案减 2. 比如当推流端按照 25 Fps@1280×720 推流时检测到网络较差, 大幅度降低帧率和分辨率, 调整为按照 15 Fps@800×600 方式推流, 减小对网络带宽的压力; 当网络状况优秀时, 若当前推流方案小于 6 时, 则推流方案加 1, 若当前推流方案为 6, 则保持不变. 比如当推流端按照 20 FPS@1024×768 推流时检测到网络优秀, 缓慢增加帧率和分辨率, 按照 25 FPS@1280×720 方式推流.

表 2 不同推流方案对应的帧率和分辨率

推流方案	帧率 (Fps)	分辨率 (Pixel^2)
1	15	800×600
2	20	1024×768
3	25	1280×720
4	30	1366×768
5	35	1600×900
6	40	1920×1080

2 首帧延迟优化

2.1 首帧延迟

直播是实时视频流, 用户以一个随机的时间点接入视频流进行播放, 接入时第一帧帧类型若不是关键帧[4], 此时播放缓冲区为空, 解码器无法解码只能丢帧. 播放器等到下一个关键帧才能播放, 等待时间若大于

5 秒, 会影响用户体验. 如果接入直播流时首帧正好是关键帧, 则可以快速显示首帧画面.

2.2 关键帧缓存算法

为此, 服务器对关键帧进行缓存, 缓存过程如下:

(1) 推流端开启双路推流, 一路视频流推送到服务器, 一路视频流定时推送 flv 小段视频到本地.

(2) 读取本地最新 flv 视频小段并解析.

(3) 遍历每一个 FLV Tag[5], 获取 FLV Tag header[5] 的第一个字节, 判断值是否为 0x09 即视频类型, 若是则进入第 (4) 步, 否则遍历下一个 FLV Tag.

(4) 获取 Tag header 的第二到第四个字节, 计算视频数据长度 length.

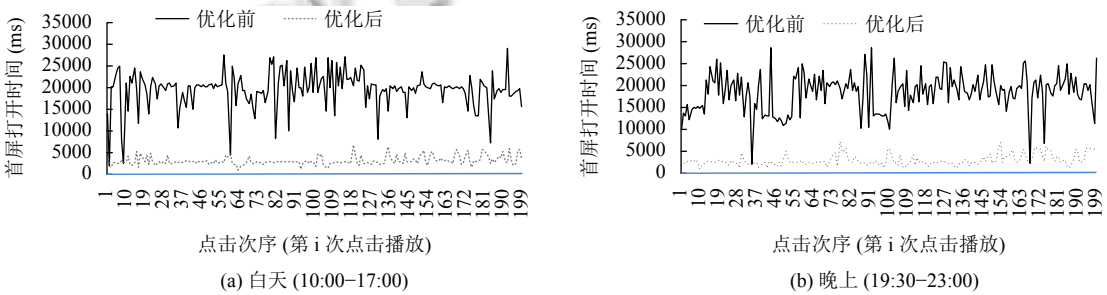
(5) 获取 Tag data 的第一个字节前 4 位, 若值为 0001 则为关键帧, 从第二个字节开始获取长度为 length 的关键帧数据, 若不是关键帧则遍历下一个 FLV Tag.

(6) 查看缓存队列是否含有关键帧, 没有则缓存, 有则更新.

(7) 播放端从服务器内存中先请求关键帧进行播放, 再请求正常的视频流.

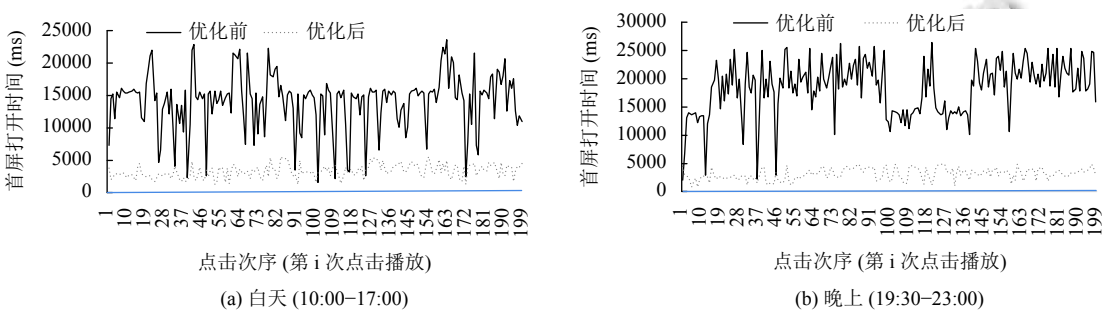
2.3 实验测试及分析

由于校园网白天和晚上的网络流量是不同的, 为了检验算法在不同网络条件下的效果, 实验统计了校园网白天 (10:00~17:00) 和晚上 (19:30~23:00) 两个时间段的三种视频分辨率 (1920*1080、1280*720、800*600) 下的首屏打开时间, 每种视频分辨率下, 共进行 200 次点击播放操作, 记录每一次从点击播放到首屏画面加载出来的时间. 实验结果如图 2、图 3、图 4 所示.



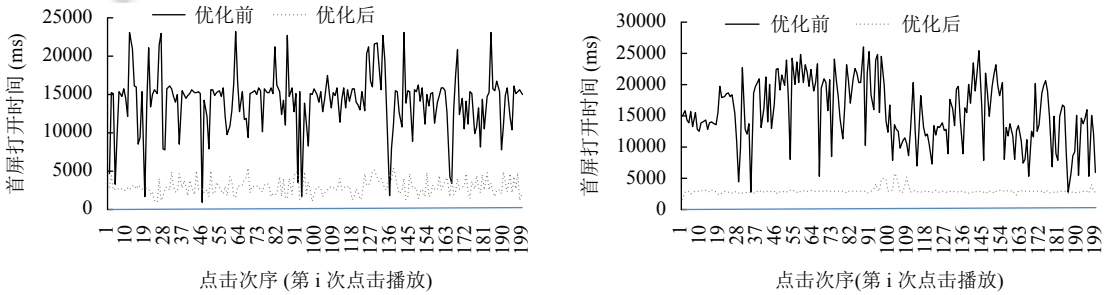
(a) 白天 (10:00−17:00) (b) 晚上 (19:30−23:00)

图 2 首屏打开耗时对比 (视频分辨率 1920×1080)



(a) 白天 (10:00−17:00) (b) 晚上 (19:30−23:00)

图 3 首屏打开耗时对比 (视频分辨率 1280×720)



(a) 白天 (10:00−17:00) (b) 晚上 (19:30−23:00)

图 4 首屏打开耗时对比 (视频分辨率 800×600)

对实验结果进行分析, 当视频分辨率为 1920×1080 时, 如图 2 所示, 白天和晚上, 优化前首屏打开时间均在 15~25 秒左右, 优化后首屏打开平均时间, 白天为 3133 ms, 晚上为 3010 ms; 视频分辨率为 1280×720 时, 如图 3 所示, 优化前首屏打开时间在 10~25 秒左右, 优化后首屏打开平均时间, 白天为 3241 ms, 晚上为 3150 ms; 当视频分辨率为 800×600 时, 如图 4 所示, 白天或晚上均在 10 s 以上, 优化后白天平均首屏打开时间为 2864 ms, 晚上为 2915 ms. 首屏打开时间明显降低, 提高了接入直播画面的速度.

此外, 观察折线图的走势, 如图 2(a) 中优化前有 3 次首屏打开时间在 5000 ms 以下, 图 2(b) 中优化前有 2 次首屏打开时间也在 5000 ms 以下, 图 3~图 4 中也出现类似的情况. 这是因为用户以一个随机的时间点接入直播流, 若首帧正好是关键帧, 则可以快速加载播放. 未优化前, 进行 200 次点击播放操作, 仅有 10 次以下能做到快速打开, 具有随机性, 进一步说明了利用关键帧缓存算法对首帧打开时间进行优化的必要性, 该算法保证了首帧正好是关键帧, 避免了首帧类型不确定所导致的首屏长时间等待.

3 编码延迟优化

3.1 H.264 帧内预测及模式选择

H.264 中 4×4 亮度块有 9 种预测模式, 16×16 亮度块和 8×8 色度块都各有 4 种预测模式[6,7]. 每一种宏块的帧内预测模式选择都是遍历所有的预测模式, 找到率失真代价值最小的模式作为最优帧内预测模式进行编码, 计算复杂度较大.

3.2 x264 帧内预测分析

x264 是开源且大规模商用的视频编码器[8,9], 因此本文选择 x264 作为屏幕直播系统的实时编码器.

H.264 标准中帧内预测模式的选择采用全搜索算法, 计算复杂度较大. x264 针对预测模式全搜索算法做出了优化: x264 中在对 4×4 块 9 种预测模式进行循环遍历时采取提前终止策略, 提前退出循环终止计算. 以下对 x264 算法的优化即在这个基础上通过减少候选模式的数量进一步减少计算复杂度.

3.3 帧内预测模式快速选择算法

对于 4×4 宏块, 基于图像的纹理方向, 降低候选模式的数量. 具体的算法流程和代码如下:

(1) 定义 4 个视频纹理方向 0º, 45º, 90º, 135º, 用变量 *Angle*0、*Angle*45、*Angle*90、*Angle*135 表示[10].

(2) 将 4×4 亮度块平均分成 4 个 2×2 的子块, 它们的像素按照行优先的顺序分别表示为 *S1*、*S2*、*S3*、*S4*, 每个像素值由其包含的 4 个像素求均值得到[10].

```
int16_t S1=(*(p_src_by)+*(p_src_by+1)+*(p_src_by+16)+*(p_src_by+17))>>2;
int16_t S2=(*(p_src_by+2)+*(p_src_by+3)+*(p_src_by+18)+*(p_src_by+19))>>2;
int16_t S3=(*(p_src_by+32)+*(p_src_by+33)+*(p_src_by+48)+*(p_src_by+49))>>2;
int16_t S4=(*(p_src_by+34)+*(p_src_by+35)+*(p_src_by+50)+*(p_src_by+51))>>2;
```

其中, p_src_by=p_src+block_idx_xy_fenc[idx], p_src 为编码帧, idx 为 16 个 4×4 块的序号, 经过计算得出宏块所包含的第一个 4×4 块的左上角元素首地址即像素 A 的地址 p_src_by.

(3) 计算纹理系数 *Angle*0、*Angle*45、*Angle*90、*Angle*135.

```
int16_t Angle0=abs(S2–S1)+abs(S4–S3);
int16_t Angle45=abs(S1–S4)+abs(S4–S1);
int16_t Angle90=abs(S1–S3)+abs(S2–S4);
int16_t Angle135=abs(S2–S3)+abs(S3–S2);
```

(4) 计算八种预测模式的纹理方向值, 对应关系如表 3 所示.

表 3 八种预测模式对应的纹理方向值

预测模式	纹理方向值
模式 0	*Angle*90
模式 1	*Angle*0
模式 3	*Angle*135
模式 4	*Angle*45
模式 5	(*Angle*90+*Angle*45)>>1
模式 6	(*Angle*0+*Angle*45)>>1
模式 7	(*Angle*90+*Angle*135)>>1
模式 8	(*Angle*0+*Angle*135)>>1

(5) 对纹理方向值进行排序, 选取纹理方向值最小的两种模式作为候选模式. 此外, 如表 4 所示, 四种视频序列在实际编码中使用频率最高的是模式 0~模式 2, 故将这三种模式也作为候选模式. 考虑到纹理方向值最小的两种模式可能已包含模式 0 或者模式 1, 故最终的候选模式最少 3 种, 最多 5 种.

(6) 对 3~5 种候选模式进行率失真代价值计算, 选取值最小的预测模式为最优的预测模式.

表 4 4×4 亮度块 9 种预测模式编码使用比例 (%)

视频序列	模式 0	模式 1	模式 2	模式 3	模式 4	模式 5	模式 6	模式 7	模式 8
news	25	25	13	2	5	7	8	8	6
ppt	29	26	17	4	5	6	5	4	5
highway	7	43	25	2	5	5	3	5	4
silent	19	22	44	3	9	8	8	8	9

3.4 实验测试及分析

由于文章已实现了网络自适应推流, 使得输出码率大小与网络相适应, 视频编码算法的优化主要关注编码速度和视频质量. 编码速度由编码帧率和编码时间来衡量, 编码帧率越大、编码时间越短则算法效果越好. 图像客观质量 psnr[11]值越大表示图像质量越好.

3.4.1 编码参数确定

屏幕直播系统编码参数设置如表 5 所示.

表 5 屏幕直播系统编码参数

编码参数	含义
profile=main	主要档次
tune=zerolatency	零延迟
aq-mode=2	弹性量化模式
bframes=1	在 I 帧和 P 帧间可插入 B 帧的数量
colormatrix=smpte170 m	设置从 RGB 转换时亮度和色度的矩阵系数
deblock=0:0	减少编码过程中产生的色块
direct=auto	direct 动态向量预测模式
ipratio=1.41	i 帧量化值相比 P 帧量化值的目标平均增量
keyint=240	关键帧之间的间隔
level=3.1	级别
me=hex	全像素运动估计算法
merange=16	控制运动估计的最大范围
min-keyint=24	最小关键帧间隔
partitions=i4x4, p8x8, b8x8	块划分方式
psy-rd=0.5:0.0	psy-rdo 和 psy-Trellis 强度值
qcomp=0.6	量化值曲线压缩系数
qpmax=51	最大的量化值
qp=26	量化值设置为默认值
rc-lookahead=30	为 mb-tree ratecontrol 和 vbv-lookahead 设置可用的帧数量
ref=1	参考帧个数为 1
scenecut=40	设定 I/IDR 帧位置的阈值
subme=7	设定子像素 (subpixel) 估算复杂度
threads=0	不配置并行单元数, 由程序根据当前 cpu 性能决定 N 值
trellis=2	在所有模式决策上启用 Trellis quantization

3.4.2 改进的算法实验对比

在屏幕直播系统中录制 500 帧 YUV(4:2:0) 格式的 ppt 教学视频, 视频特点是视频变化较为平缓, 局部纹理细节丰富, 能代表屏幕直播系统中大部分应用场景的视频特点. 此外, 选择与该视频特点相似的两个官方测试序列 news 和 highway 进行测试.

对三种视频序列的编码时间变化率 (△time)、编码帧率变化率 (△fps)、图像客观质量变化 (△psnr) 进行比较, 计算方法如下:

$$\Delta time = \frac{Avg\left(\sum_{j=1}^{100} T_j\right) - Avg\left(\sum_{i=1}^{100} T_i\right)}{Avg\left(\sum_{i=1}^{100} T_i\right)} \times 100\% \quad (1)$$

式 (1) 中, $Avg\left(\sum_{i=1}^{100} T_i\right)$和$Avg\left(\sum_{j=1}^{100} T_j\right)$分别表示算法改进前后对视频序列 100 次编码的平均时间.

$$\Delta fps = \frac{Avg\left(\sum_{j=1}^{100} F_j\right) - Avg\left(\sum_{i=1}^{100} F_i\right)}{Avg\left(\sum_{i=1}^{100} F_i\right)} \times 100\% \quad (2)$$

式 (2) 中, $Avg\left(\sum_{i=1}^{100} F_i\right)$和$Avg\left(\sum_{j=1}^{100} F_j\right)$分别表示算法改进前后对视频序列 100 次编码的平均帧率.

$$\Delta psnr = Avg\left(\sum_{j=1}^{100} P_j\right) - Avg\left(\sum_{i=1}^{100} P_i\right) \quad (3)$$

式 (3) 中, $Avg\left(\sum_{i=1}^{100} P_i\right)$和$Avg\left(\sum_{j=1}^{100} P_j\right)$分别表示算法改进前后对视频序列 100 次编码的平均图像客观质量.

将 3 种测试序列实验结果按照式 (1)~(3) 进行计算, 最终 3 种序列的测试结果如表 6 所示.

表 6 改进 x264 算法与未改进 x264 算法比较

视频序列	△time(%)	△fps(%)	△psnr(dB)
ppt	−8.26	+8.79	−0.004
news	−3.74	+4.16	−0.003
highway	−21.94	+27.9	−0.012

从实验结果可以看出, 改进后的 x264 算法压缩屏幕直播系统中的 ppt 视频序列, 编码时间下降了 8.26%, 编码帧率提高了 8.79%. 对于官方测试序列, news 序列编码时间下降 3.74%, 编码帧率提高 4.16%; highway 序列编码时间下降了 21.94%, 编码帧率提高了 27.9%. 3 种序列视频质量基本没有损失. 改进后的 x264 编码算法应用到屏幕直播系统中, 有效的降低了编码时间, 提高了编码的效率.

4 系统延迟测试

基于以上研究, 设计并实现屏幕直播系统, 系统总体架构分为推流端、流媒体服务器、播放端、业务服务器四部分. 运行屏幕直播系统, 使用 13 台平板电脑 (内存 2 GB) 进行播放, 统计系统在 800×600、1024×768、1280×720、1920×1080 四种分辨率下的直播延迟情况. 网络路由器为华为企业级路由器 AR111-S, AP 为华为无线 AP 3010DN-V2. 推流端和服务器为笔记本电脑, CPU 配置信息为 Intel(R) Core(TM)i5-3230 M, 双核 2.6 Ghz, 内存为 8 GB.

每隔 100 帧视频统计一下直播延迟, 一共测试 50 次共计 5000 帧, 将直播延迟数据实时写入 SD 卡的文本文件中, 最终的实验结果如图 5 所示.

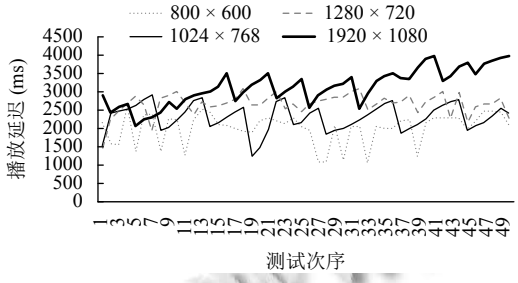


图 5 屏幕直播系统在四种分辨率下的延迟测试结果

对实验结果进行分析, 当推流分辨率为 800×600、1024×768、1280×720 时, 系统延迟稳定在 1 s~3 s 之间. 与优化前 5 s~10 s 的延迟相比, 经过优化后的屏幕直播系统在 1280×720 甚至更低分辨率下延迟较低, 能够满足实际教学中课堂屏幕共享需求. 对于 1920×1080 超清分辨率的屏幕直播, 系统的延迟较高且随着播放帧数增多产生累积延迟, 出现延迟上升的趋势, 后续工作需要进一步研究超清视频序列的特点, 有效地降低超清视频直播场景下的系统延迟.

5 结论与展望

如何在现有直播架构下进行直播延迟优化对增强用户实时性体验有很大的现实意义. 文章从网络推流、首帧延迟、编码延迟三个角度对直播延迟问题进行分析和优化, 实现了低延迟的屏幕直播系统, 并将该系统应用于课堂屏幕共享教学中, 能高效实时的进行课件展示、应用软件教学等操作, 提高了课堂效率, 具有很高的应用价值.

参考文献

1 Li XR, Wang L, Cui JC, *et al*. A new fragmentation strategy for video of HTTP live streaming. Proceedings of 2016 International Conference on Mobile Ad-Hoc & Sensor Networks. Hefei, China. 2016. 86–89. [doi: 10.1109/MSN.2016.022]

2 刘率. 基于 rtmp 的远程教学直播系统的设计与实现[硕士学位论文]. 呼和浩特: 内蒙古大学, 2018.

3 Kumar GD, Rao CVG, Singh MK, *et al*. Using jpcap API to monitor, analyze, and report network traffic for DDoS attacks. Proceedings of 2014 International Conference on Computational Science and Its Applications. Guimaraes, Portugal. 2014. 35–39. [doi: 10.1109/ICCSA.2014.18]

4 蔡家楣, 陈洋, 陈铁明, 等. 结合互信息量与模糊聚类的关键帧提取方法. 计算机系统应用, 2010, 19(4): 73–76. [doi: 10.3969/j.issn.1003-3254.2010.04.018]

5 夏俊锋. HTML5 下的视频会议系统中基于 RTMP 的直播解决方案的设计与实现[硕士学位论文]. 广州: 华南理工大学, 2017.

6 Meuller J, Hu X, Bunthof C, *et al*. Identification of an aspartic acid residue in the β subunit which is essential for catalysis and proton pumping by transhydrogenase from Escherichia coli. Biochimica et Biophysica Acta (BBA) - Bioenergetics, 1996, 1273(3): 191–194. [doi: 10.1016/0005-2728(95)00154-9]

7 王林, 贠境孺. 基于宏块预判的快速帧内预测模式选择算法. 计算机系统应用, 2018, 27(4): 109–116. [doi: 10.15888/j.cnki.csa.006347]

8 赵太飞, 孙孝彬, 娄俊鹏. 基于 H.264 的多传输模式 IP 网络视频监控系统. 计算机系统应用, 2015, 24(11): 81–87. [doi: 10.3969/j.issn.1003-3254.2015.11.013]

9 Javadtalab A, Omidyeganeh M, Shirmohammadi S, *et al*. A bitrate-conservative fast-adjusting rate controller for video conferencing. Proceedings of 2017 IEEE International

Symposium on Multimedia. Taichung, China. 2017. 338–341. [doi: 10.1109/ISM.2017.62]

10 古辉, 陈强. H.264 帧内预测模式选择改进算法. 浙江工业大学学报, 2014, 42(2): 204–209. [doi: 10.3969/j.issn.1006-4303.2014.02.017]

11 杨嘉琛, 侯春萍, 沈丽丽, 等. 基于 PSNR 立体图像质量客观评价方法. 天津大学学报, 2008, 41(12): 1448–1452. [doi: 10.3969/j.issn.0493-2137.2008.12.011]