基于 DSS 的直播系统研究与实现*

汪夏荣 赵海武 王国中 范 涛
(上海大学通信与信息工程学院 上海 200444)

摘 要：本文设计并实现了一个基于达尔文流媒体服务器(darwin streaming server,DSS)的直播系统,首先对 DSS 服务器的软件核心架构进行了研究分析,针对直播系统中 UDP 传输导致的网络拥塞现象和丢包问题,采用了基于 RTCP 反馈的拥塞控制方案,经过系统测试表明直播时的平均延时约为 100 ms,画面质量总体清晰流畅,有效减轻了网络拥塞、降低了丢包;且通过 CPU 占用率测试表明随着用户的增多 CPU 占用率并未呈现明显增加,单用户情况下 CPU 占用率稳定在 1%～3%,20 个用户情况下 CPU 占用率稳定在 6%～9%,说明系统并发处理性能较好。系统总体具有较好的性能,能够满足实际应用需求。

关键词：直播;拥塞控制;RTCP;FFmpeg;VLC
中图分类号：TN943 **文献标识码**：A **国家标准学科分类代码**：510.5025

Research and implementation of living video system based on DSS

Wang Xiarong Zhao Haiwu Wang Guozhong Fan Tao
(School of Communication & Information Engineering, Shanghai University, Shanghai 200444, China)

Abstract: This paper designs and implements a living video system based on DSS. Firstly, the software core architecture of DSS server is analyzed. In view of the network congestion and packet loss caused by UDP transmission in living video system, the congestion control scheme based on RTCP feedback is adopted, After the system test shows that the average delay when living is about 100 ms, and the overall test picture is smooth, effectively reduce the network congestion and packet loss; and CPU occupancy rate test shows that with the increase in user CPU occupancy rate is not presented obviously increase, single-user CPU occupancy rate stabilized between 1% to 3%, 20 users CPU occupancy rate stabilized between 6% to 9%, indicating that the system concurrent processing performance is better. The overall system has a good performance, to meet the practical application needs.

Keywords: live; congestion control; RTCP; FFmpeg; VLC

0 引 言

近年来,随着现代通信设备性能和国内网络环境的不断提升,以及多媒体压缩编码技术的发展,网络对数据的负载能力大幅加强,人类所能接受的讯息从最开始的文字、图像发展成现在的音频、视频信息,因此流媒体技术[1]得到了普遍的应用,整个流媒体市场正在以极高的速率向前发展。另外,随着手机的智能化、平板电脑的出现,流媒体技术已不再局限于个人 PC,视频播放随处可见的时代已经到来[2]。在这种流媒体快速发展、全世界互联互通的大环境下,视频直播需求应运而生,各视频服务提供商纷纷开始了自己的直播业务;而在搭建直播业务平台的期间,如何选择合适的平台成为一个非常重要的问题。当前最受市场青睐的直播流媒体平台主要有 RealNetwork 公司的 RealMedia、Microsoft 公司的 Windows Media 以及 Apple 公司的 QuickTime[3-4]。

Apple 公司的 QuickTime 除了可作为媒体播放器之外,还能提供各类完备的流式媒体技术架构,与此同时,Apple 公司也在 QuickTime 基础之上发行了开源的达尔文流媒体服务器(darwin streaming server, DSS)[5],不仅能够很好的进行二次开发,而且能够支持 Windows、Linux、Android、IOS 等多种系统和平台,它实现了 4 种 IETF 指定的国际工业标准,分别是 RTSP、RTP、RTCP 与 SDP[6,7],并支持 3GPP、MP4、MOV 等媒体格式,可以为用户提供较好的实时直播服务。针对互联网直播巨大的市场需求,本文在 DSS 基础架构进行二次开发,设计并实现了一个直播系统,具有良好的现实应用价值。

收稿日期:2017-03
* 基金项目:国家“863”计划项目(2015AA015903)资助

1 系统总体框架设计

文中所设计的系统总体框架如图 1 所示,主要由 3 大核心部分组成:分别是发送端、服务器、接收端。

发送端的主要工作是通过各类音视频采集设备(USB、枪式、球形摄像头及麦克风等)将采集到的音视频通过 H.264、AAC 压缩编码后传输给服务器;服务器接收到数据后,封装成 RTP 包,发送到网络中进行传输,本文所设计系统的服务器在 DSS 基础架构上,针对直播系统中 UDP 传输导致的网络拥塞现象和丢包问题,设计并实现了基于 RTCP 反馈的拥塞控制方案,最终结果表示所用方案效果显著;接收端首先分析接收的 RTP 数据包并进行重新组合,接着送入缓冲区等待解码播放。

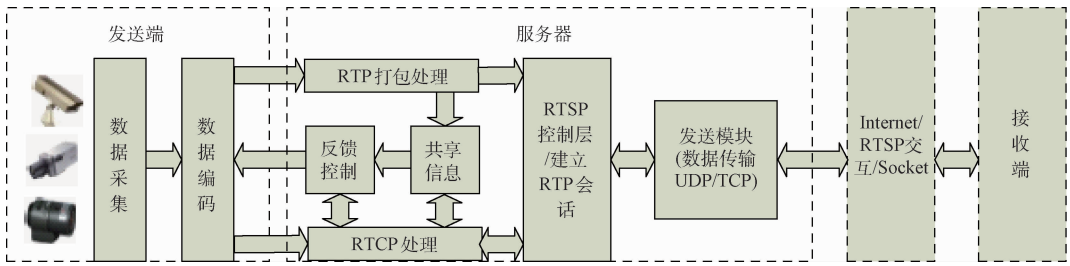

图 1 系统基础框架

2 DSS 服务器软件架构

本系统中所涉及的 DSS 服务器整体核心结构如图 2 所示,其中主要包含 3 种类型的要素,分别是线程、Task 队列或堆、被侦听的事件。

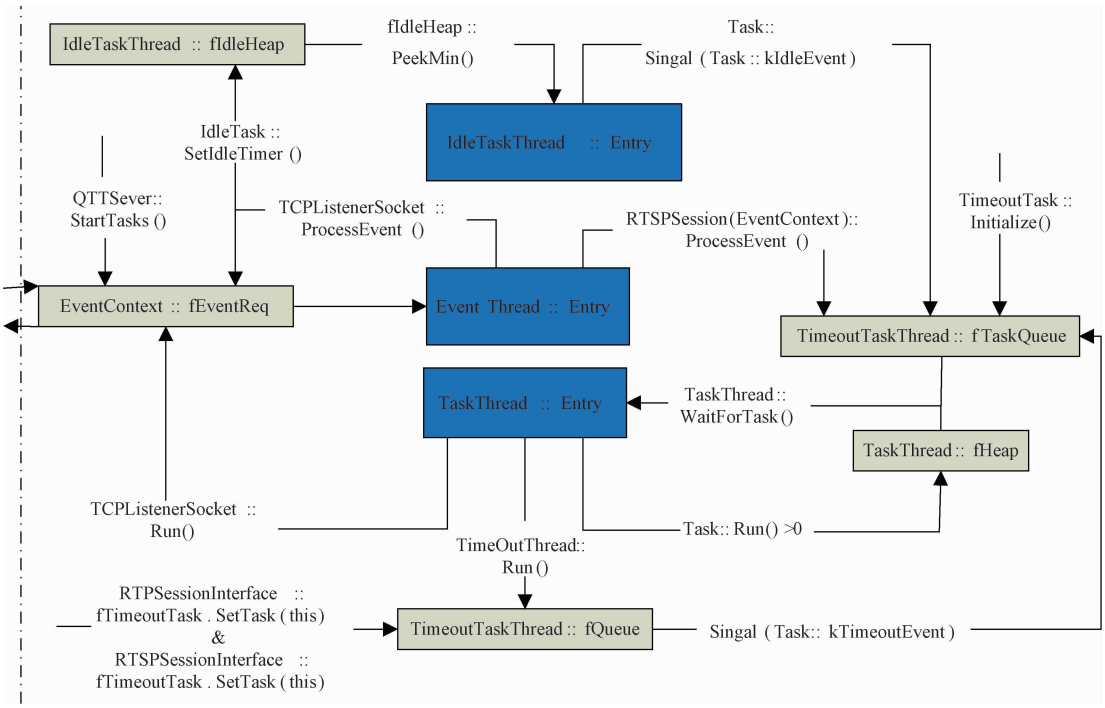

图 2 服务器核心架构

图中 Task 类包含 Signal 和 Run 方法。Signal 方法将 Task 添加至任务线程(TaskThread)的 Task 队列中,Run 方法主要负责处理任务。基于 Task 类,定义了 3 种类型的 Task:IdleTask、TimeoutTask、及普通的 Task。

DSS 核心主要包含以下几类线程:主线程(MainThread)、任务线程(TaskThread),事件线程(EventThread)及空闲任务线程(IdleTaskThread),详细介绍如下:

1)MainThread,负责跟踪服务器是否有关闭需要,并记录状态信息或打印统计信息。

2)TaskThread,负责接收 EventThread 中 RTSP 和 RTP 请求,并将请求转发至相应的服务器模块进行处理,把响应

后的数据包返回给接收端；并通过调用 Task：Run 方法来处理 Task，比如 RTSPSession 和 RTPSession；另外 TaskThread 的数量默认与 CPU 的数量相同，可通过配置文件更改。等待被 TaskThread 调用的 Task 放在队列或堆中。

3）EventThread，主要负责监听 socket 事件，例如收到 RTSP 请求和 RTP 数据包后把事件传递给 TaskThread。在 DSS 中，当 RTSP 请求事件到达时，除会被监听之外，EventThread 把对应 RTSPSession 类型的 Task 加到 TaskThread 队列中，等待 RTSP 请求被处理；而当建立 RTSP 连接请求事件发生时，启动对应的 socket 进行监听的同时，EventThread 还会创建一个 RTSPSession。

4）IdleTaskThread，主要负责管理一个周期性的任务队列，该队列中有超时和 socket 两种类型的任务。其中，管理空闲任务类型对象的队列，依据预先设置的定时器发出消息，触发空闲任务的管理和调度；除此之外，若派生自空闲任务的 TCPListenerSocket 类的并发连接数达到预先规定好的最大值，则将派生自 TCPListenerSocket 的 RTSPListenerSocket 加到 IdleTaskThread 管理的空闲任务队列中，并暂时关闭对 RTSP 端口的监听直到预先设定好的定时器触发[8]。

3　直播系统实现

3.1　发送端

发送端是一个提供直播流的应用程序，本系统中发送端采用 FFmpeg 进行数据采集及编码。FFmpeg 是一套开源程序，能够用来采集音视频，并可将其编码为流，其中包含了领先的音/视频编码库 libavcodec 等[9]。

发送端调用摄像头及麦克风对音视频数据进行采集，最初会得到 yuv422 格式的视频数据，然后通过 FFmpeg 中的编码库 libavcodec 编码压缩成 H.264 格式的直播视频码流，再将压缩编码后的直播视频码流经通信网络发布到服务器相应的应用程序上[10]。

3.2　服务器 RTCP 反馈拥塞控制的实现

服务器是存储（或接收）媒体流并等待接收端请求连接的应用，本系统在 Apple 公司的开源实时流媒体播放服务器程序（darwin streaming server，DSS）的基础架构之上进行了二次开发，作为本文直播系统的服务器。针对直播系统中 UDP 传输导致的网络拥塞现象和丢包问题，本文设计了基于 RTCP 反馈的拥塞控制方案，并在 DSS 架构中进行了实现。

拥塞现象通常发生在通信网络环境中并发或分组数量过多的情况下，此时若网络满足不了应用的需求，即该部分网络已来不及处理，最终将会引起这部分乃至整个网络性能下降。直播场景中经常发生网络拥塞的现象，很大程度上影响了接收端直播流的质量，而拥塞控制作为一种可以有效减轻网络拥塞的方式，在实际研究中意义重大。由于本文所设计系统是采用 UDP 方式进行数据传输的，因而本文中所设计的基于 RTCP 反馈的拥塞控制机制是针对 UDP 方式设计的。

目前关于 UDP 的拥塞控制机制主要有两类：第一类是模仿 TCP 和式增加积式减少的 AIMD（additive increase multiplicative decrease）拥塞控制，第二类则是 TCP 友好拥塞控制机制 TFRC（TCP-friendly rate control）。与 AIMD 机制中速率的讯速变化所造成的抖动因素相比，TFRC 机制的平稳性使得它更适合在实时直播系统中使用[11]。接收端将数据包的接收情况周期性地反馈给服务器，发送端则根据 TCP 稳态流量公式计算出新的速率，并与当前发送速率进行比较，依据调整机制动态地调整当前发送速率。

上述 TFRC 机制采用如下公式计算吞吐量（padhye 模型）[12] 如下：

$$S=\frac{Q_{\text{size}}}{t_{\text{RTT}}\sqrt{\frac{2bp}{3}}+t_{\text{RTO}}\left[3\cdot\sqrt{\frac{3bp}{8}}\cdot p(1+32\,p^2)\right]} \tag{1}$$

式中：S 为平均吞吐率，单位为 B/s；Q_{size} 为数据包的大小，单位为 Byte；t_{RTT} 为链路回环时间 RTT（round trip time），单位为 s；t_{RTO} 为 TCP 的重传超时时间，单位为 s；p 为丢包事件率，范围 0～1.0；b 为每个 TCP 应答所确认的接收报文数，默认为 1。

本文所设计系统采用了 RTCP 反馈控制的方式，实现了 TFRC。利用 RTCP 协议，接收端定期向服务器返回接收者报告 RR（receiver report）包，发送端根据 RR 包中的相关信息，计算出 RTT 和 p，然后代入式（1），得出目前传输最佳的吞吐率，从而对发送端的发送数据速率进行相应的控制和调整。因此只要得到 t_{RTT}、t_{RTO} 以及 p 这 3 个必要的参数，就能根据式（1）计算出平均吞吐率。接下来说明参数 R、t_{RTO}、p 的具体获取方法及发送速率的调整机制。

3.2.1　参数的获取

1）回环时间 t_{RTT}：若送端发出 SR（sender report）包时的开始时间记为 t_1，接收端收到 RR 包时的结束时间记为 t_2，则回环时间为 $t=t_2-t_1$。每个 RTCP 周期都有自己的相关采样 t_{RTT}，其计算公式如式（2）所示：

$$t_{\text{RTT}}=t_{\text{NOW}}-t_{\text{LSR}}-t_{\text{DLSR}} \tag{2}$$

式中：t_{NOW} 为发送端收到 RR 包的当前时间，t_{LSR} 为接收端最近收到的 SR 包中 NTP 时间戳的中间 32 位值，t_{DLSR} 为接收端最近收到 SR 包到发送当前 RR 包的时延。t_{LSR} 和 t_{DLSR} 都存放在发送端收到的 RR 包中。

2）重传超时时间 t_{RTO}：发送端开始发送数据的时候会预先设定 1 个计时器，如果在规定的时间内未接收到正确的返回消息，则计时器将发生超时的消息告诉发送端，发送端将重新发送该数据。由于 UDP 传输时不存在超时重传问题。在系统实现时，通常令 $t_{\text{RTO}}=4t_{\text{RTT}}$。

3）丢包事件发生率 p：根据指数加权移动平均算法公式[13-14]：

$$p_{i+1}=(1-\lambda)p_i+\lambda p_{\text{FL}} \tag{3}$$

其中 p_i 代表第 i 次 RTCP 报告间隔内的包丢失事件发生率，p_{FL} 代表第 $i+1$ 次 RTCP 报告间隔内的包丢失比率(fraction lost,FL)，在实际应用中，$\lambda = 0.3$。

3.2.2 发送端速率的调整

基于 RTCP 反馈的拥塞控制机制模仿了 TCP 的慢启动方式来探测网络的可用带宽，在丢包现象发生之前，在每个 RTCP 间隔内成倍地增大发送速率。当收到 FL 阈值非零的 RR 包时，计算出上述分析过的各个参数，代入式(1)，计算出当前时刻对 TCP 最佳的发送速率 S，再与当前发送速率 $S_{now}X_{Sample}$ 进行对比：

1)若 $S_{now} < S$，说明当前网络状况良好，存在富余的带宽。因而发送端在接下来的 RTCP 间隔内，令 $S_{now} = S_{now} + 1/R$。

2)若 $S_{now} > S$，说明当前网络已达饱和状态，在下一个 RTCP 间隔内，将 X 作为当前数据发送速率，即令 $S_{now} = S$。

每当发送端收到 RR 包时，由式(1)计算出对 TCP 最佳的发送速率，与当前发送速率进行比较来相应地调整数据的发送速率，从而能够减少直播系统中网络拥塞现象的发生。

3.3 接收端

接收端是播放视频流的应用程序。本系统接收端播放器采用了 VLC Media player 作为基础框架原型，VLC Media player 是一个开源多媒体播放的项目，本文所设计系统的接收端在 VLC 基础之上进行了简单的二次开发，作为本系统的接收端直播播放器。

主要任务是将从服务器上请求过来的音视频流暂时存储在本地数据缓存区，并将相应拥塞控制所需要的信息反馈给服务器，最后对缓冲区中的数据进行解码、播放[15]。

4 系统性能测试

4.1 CPU 占用率测试

衡量直播系统性能的指标主要体现在系统对于并发处理场景下 CPU 的占用率。测试结果为：分别在单个用户及 20 个用户的情况下，每隔 5 min 取 1 次使用数据，共取 20 次，最终性 CPU 占用率在 1%～3%；20 个用户的情况下 CPU 占用率在 6～9%，如图 3 所示。

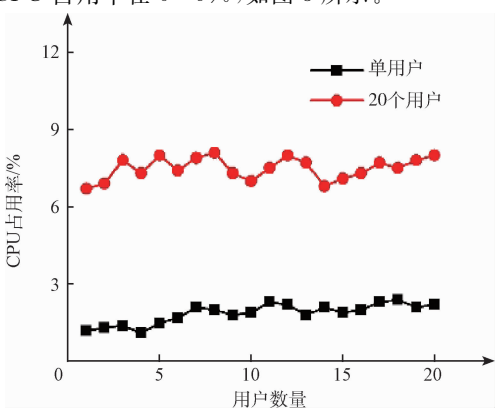

图 3 系统服务器的 CPU 占用率

4.2 基于 RTCP 反馈拥塞控制的效果测试

网络拥塞主要体现在视频延时及画面质量上，本文在网络使用高峰段的时候，在 20 个用户的并发情况下分别对采用反馈控制方案前后各自进行了测试。其中延时测试的具体方法为：发送数据包的同时记录系统时间 T_1，并将其放入要传输的数据包，在接收端接收到该数据包时，从中提取出时间戳，然后再次获取当前系统时间 T_2，则可以得到精确延时($T_2 - T_1$)[16]。

表 1 为基于 RTCP 反馈拥塞控制的效果测试的结果汇总。结合表 1，以及通过图 4 及图 5 进行对比，未采用本文拥塞控制方案时延时约 2 s，且画面会出现跳帧和马赛克现象；而采用后延时约 100 ms，画面比较清晰流畅，基本不会出现掉帧和画面模糊的现象。表明 UDP 方式下加入本文所提基于 RTCP 反馈的拥塞处理机制达到了预期效果，直播流传输的实时性较好、延时小；系统并发处理性能得到提升，画面质量较为清晰流畅，减轻了网络拥塞、降低了丢包，能够满足音视频直播对于实时性及画面质量的要求。

表 1 基于 RTCP 反馈拥塞控制的效果

	延时	画面质量
未采用方案	约 2 s	画面出现跳帧和马赛克现象，有丢包和网络拥塞情况发生。
采用方案后	约 100 ms	画面较为清晰流畅，基本不会出现掉帧和画面模糊的现象。

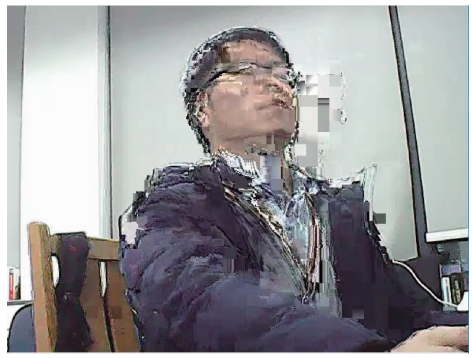

图 4 采用方案前

图 5 采用方案后

5 结　论

本文设计并实现了一个基于 DSS 的直播系统，针对直播系统中 UDP 传输导致的网络拥塞现象和丢包问题，采用了基于 RTCP 反馈的拥塞控制方案，经过系统测试表明直播时的平均延时约为 100 ms，且测试画面总体流畅，显著减少了网络拥塞、降低丢包现象的发生，说明了该方案的可行性；且通过 CPU 占用率测试表明随着用户的增加 CPU 占用率并未呈现线性增加，单个用户情况下 CPU 占用率稳定在 1%～3%，20 个用户情况下 CPU 占用率稳定在 6%～9%。同时可以为设计与实现更为复杂的基于 DSS 直播系统提供一个参考。

参考文献

[1] 陈珏奇，刘峰. 流媒体传输技术[J]. 计算机技术与发展，2012(3)：6-10.

[2] 雷霄骅，姜秀华，王彩虹. 基于 RTMP 协议的流媒体技术的原理与应用[J]. 中国传媒大学学报：自然科学版，2013(6)：59-64.

[3] YANG H，YU J，WANG Y，et al. An implementation of designing media streaming system for live broadcast [J]. WSEAS transactions on communications，2010，9(3)：226-235.

[4] ZHEN P F，KANG B. Analysis and implementation of streaming media system based on RTP and MPEG-4 [C]. 2015 4th International Conference on. Computer Science and Network Technology (ICCSNT)，IEEE，2015：1286-1289.

[5] URBANO F，CHANCHI G，GABRIEL E，et al. Testing environment for video streaming support using open source tools[J]. Ingenieria y Desarrollo，2016，34(2)：333-353.

[6] ALVESTRAND H. Overview：Real time protocols for browser-based applications[J]. 2016.

[7] SCHULZRINNE H，RAO A，LANPHIER R，et al. Real Time Streaming Protocol 2.0 (RTSP)：draft-ietfmmusic-rfc2326bis-27 [J]. MMUSIC Working Group of the Internet Engineering Task Force (IETF)，2011，296.

[8] 彭国刊. 一个基于 RTSP 协议的自适应流媒体传输系统[D]. 广州：华南理工大学，2014.

[9] 郭玉霞. FFMPEG SDK 结构分析[J]. 计算机与网络，2013 (11)：48-50.

[10] 辛长春，娄小平，吕乃光. 基于 FFmpeg 的远程视频监控系统编解码[J]. 电子技术，2013，(1)：3-5.

[11] 陈锋锋. 基于 RTSP 的流媒体传输系统的应用开发[D]. 南京：南京邮电大学，2013.

[12] 范浩，严军，郑新元，等. 移动网络下流媒体传输拥塞控制机制的研究[J]. 电子测量技术，2015，38(12)：125-128，132.

[13] ZHAOX，YANG Y，MAO W. TCP-like congestion control algorithm for stream media transmission[J]. Microelectronics & Computer，2014(4)：27.

[14] 尹洪，洪玫，曾明. 基于 RTCP 的实时流式传输拥塞控制算法[J]. 云南大学学报：自然科学版，2008 (S2)：127-132，139.

[15] 余义雄. 在视频直播系统中 QoS 与拥塞控制的应用研究[D]. 成都：成都理工大学，2016.

[16] 高沫. Adobe Flash 平台若干新技术在富媒体课程直播系统中的应用[D]. 上海：上海交通大学，2011.

作者简介

汪夏荣(通讯作者)，硕士研究生，主要研究方向为数字音视频编解码技术及应用软件开发。

E-mail：shuwangxr@163.com

赵海武，副教授，博士，主要研究方向为数字音视频编解码技术、编解码技术标准等。

E-mail：zhaohaiwu@shu.edu.cn

王国中，教授，博士，主要研究方向为中国音视频编解码标准 AVS、三网融合、基于 AVS 和 DTMB 的双国标数字地面电视系统和 3DTV 等。

E-mail：wanggz@shu.edu.cn

范涛，博士，主要究方向为视频编码标准、3DTV、新媒体技术、三网融合技术等。

E-mail：fantao19830627@163.com