基于 Live555 的实时流媒体传输系统①

吕少君, 周渊平

(四川大学 电子信息学院, 成都 610065)

摘 要: 针对 Live555 开源项目进行二次开发, 设计并初步实现了一个实时流媒体传输系统. 所做的主要工作是在 LiveMedia 类库中新增了 WebcamFrameSource 类和 Webcam Ondemand MediaSubsession 类, 实现了实时接收和转发 H.264 码流的流媒体服务器; 同时增加了视频采集和 H.264 编码部分, 实现了实时采集、编码和传输的流媒体传输系统. 所提出的设计方案有助于解决在网络上传输多媒体数据的延时和不同步等问题, 从而可提高系统的健壮性和可扩展性.

关键词: 流媒体服务器; Live555; H.264

Real-Time Streaming Media Transmission System Based on Live555

LV Shao-Jun, ZHOU Yuan-Ping

(College of Electronics and Information, Sichuan University, Chengdu 610065, China)

Abstract: In this paper, secondary development of Live555 open source projects was achieved, designing and preliminary implementing a real-time streaming media transmission system. WebcamFrame Source and WebcamOndemand MediaSubsession classes are added to LiveMedia Library, realizing the real-time receiving and forwarding H.264 stream; at the same time, video captrue and H.264 coding part are also added, achieving video capture, H.264 coding, and transmission in real time. The proposed design helps solving the problems of time delay and out of sync when transferring multimedia data over a network, which can improve robustness and scalability of the system.

Key words: streaming media server; Live555; H.264

随着互联网的不断发展及网络带宽的不断提高, 流媒体数据的实时传输成为近年来数据传输方面研究的热点, 该技术已广泛应用在视频会议、网络直播和在线教育等领域. 目前在网上传输数据只提供尽力而为(Best-effort), 这样往往会造成数据包的延时和丢失, 从而导致媒体数据播放不清晰或者不同步的问题. 本文设计并实现了一种基于 Liv555 的流媒体解决方案, 重点针对 Live555 开源项目进行二次开发, 设计实现了实时流媒体传输系统.

1 流媒体传输系统的相关技术

1.1 流媒体

流媒体是指在 Internet 上以流式传输方式传输音频、视频和多媒体文件的技术. 所谓流式传输方式是指现将编码后的多媒体文件分片成一个一个的压缩包, 然后由服务器向用户端实时、连续地转发. 使用流式传输方式用户只需等待几秒到几十秒的启动延时即可在客户端进行播放, 而不必像采用传统的下载方式那样要等到整个文件全部下载完后才可以正常观看. 除此之外, 流式传输的另一个优势在于可以传输那些事先不知道大小的多媒体数据, 例如网络监控、视频会议、网上直播等.

1.2 RTP/RTCP 协议

实时传输协议 RTP(Real-Time Transport Protocol)是针对在 Internet 上传输多媒体数据流的一种传输协议,主要负责实现流之间同步和提供时间信息. RTP 不能为数据包的顺序传输提供可靠的传送机制, 也不提供拥塞控制和流量控制, 它只能保证媒体数据的实时

① 收稿时间:2014-04-19;收到修改稿时间:2014-06-03

传输.

实时传输控制协议 RTCP(Real-Time Transport Control Protocol)是负责在服务器和用户端之间交换传输质量的信息的一种协议. 在进行 RTP 会话期间, 每个参与者都会周期性地向服务器发送 RTCP 包, 里面包括已发送的 RTP 数据包的数量, 以及丢失的 RTP 数据包的数量等统计信息. 流媒体服务器就可以根据这些信息动态地调整数据的传输速率和有效载荷类型.

1.3 RTSP 协议

实时流传输协议 RTSP(Real-Time Streaming Protocol)是 TCP/IP 协议体系中的应用层协议, 负责控制数据的实时发送. RTSP 本身不传输流媒体文件, 主要用来对多媒体服务器进行网络远程控制.

1.4 Live555 开源项目

Live555 是一个开源的、跨平台的流媒体解决方案, 它实现了对 RTSP 、RTP/RTCP 等标准流媒体传输协议的支持. 而且支持多种音视频编码格式的多媒体数据的流化、接收和处理, 包括 H.264、MPEG、JPEG 视频和多种音频编码格式. 由于 Live555 拥有良好的结构化设计, 所以很容易扩展对其他多媒体格式的支持. 正是因为 Live555 有如此强大的功能, 它已经被用于多款播放器的流媒体播放功能中, 如 MPlayer、VLC(VideoLan).

1.5 H.264

H.264 是一种高性能的视频编解码技术标准. H.264 最大的有点是具有很高的数据压缩比, 在同等图像质量的条件下, H.264 的压缩比是 MPEG-4 的 1.5～2 倍, 是 MPEG-2 的 2 倍以上, 而且在高压缩比的同时图像的流畅度也很高, 所以在网络传输中所需的带宽就更少、时间就更短.

2 系统的总体架构

该系统主要由前端视频采集编码、流媒体转发服务器和客户端三部分组成. 其中前端采集编码设备负责采集视频数据, 然后将采集到的数据进行压缩编码后传给流媒体服务器. 流媒体服务器接收到数据后, 把接收到的视频数据进行流化, 然后发送到网络中进行传输. 客户端负责对接收到的视频数据进行解码和播放.

其中前端采集编码包括视频数据采集和 H.264 编码. 流媒体服务器只能接收和传输编码封装好的视频数据, 在本系统的选择的是 H.264 的视频格式, 所以采集好的视频数据要传输给 X264 编码器进行 H.264 压缩编码. 编码完成后的数据就传给流媒体服务器 Live555 Media Server, 流媒体服务器在 RTP 协议的支持下把编码好的数据封装成 RTP 包, 同时将时间戳等信息加入 RTP 包中. 打包好的数据在 RTSP 协议的交互控制下转发出去, 客户端在接收处理 RTP 包的同时, 会在 RTCP 协议的支持下统计出当前 RTP 包的接收情况(如丢包率等), 然后将统计信息发送给流服务器, 服务器则根据该反馈信息相应地改变码流发送速率, 以达到最佳的接收效果. 客户端选用的是 VLC 播放器, 它可以完成 H.264 格式视频的实时接收、解码和播放的任务. 本文主要针对流媒体服务器的设计实现, 而暂不讨论视频采集和编码部分.

3 流媒体服务器

在转发模式下, Live555 流媒体服务器默认是通过 live555MediaServer 来实现音视频文件的转发功能, 而本系统是要实现实时流媒体的直播方式, 所以需要在源码的基础上进行修改, 而实现的方法是通过继承相关类和重写相关方法来实现流媒体的直播. 具体实现方式是通过创建 WebcamFrameSource 对象来获取实时采集编码后的数据源, Webcam Framed Source 类是继承自 FramedSource 类, 并且新建了一个 live-streaming.cpp 文件, 用来搭建 RTSP 服务器和创建 RTP 子会话. RTSP 服务器的搭建流程图如下.

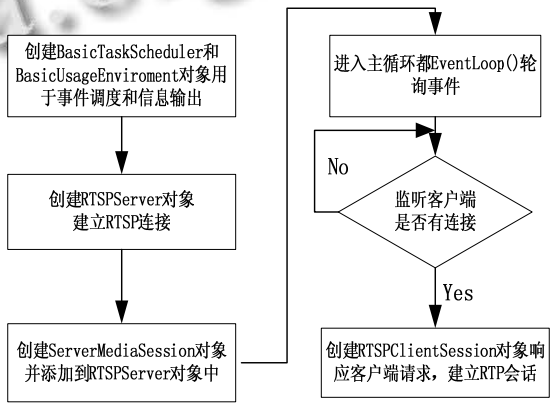

图 1 RTSP 服务器搭建流程图

首先创建 BasicTaskScheduler 和 BasicUsage-Enviroment 对象. TaskScheduler 是整个 Live555 的任务调度中心, 实现事件的异步处理、事件处理函数的

注册, 主要负责任务的调度和执行; UsageEnviroment 代表整个系统的运行环境, 负责消息的输入输出和用户交互功能. 他们共同构建了整个 Live555 框架的灵魂. 然后创建 RTSPServer 对象. 创建 RTSPServer 对象时, 先通过 setUPSocket()调用 setStreamSocket()函数来创建一个 Socket 连接, 同时监听该端口; 然后把接连处理函数句柄 incomingConnectHandler 和 Socket 句柄 serverSocket 传给任务调度器进行关联; 最后调用 select()函数进行阻塞, 等待客户端的连接. 接着创建 ServerMediaSession 对象, ServerMediaSession 代表的就是一次会话, 同时创建一个 WebcamOndemand-Media Subsession 子会话, 它负责将 Webcam Framed Source 中获取的采集编码后数据源进行 H.264 格式的 RTP 封包. 然后将 ServerMediaSession 对象添加到 RTSPServer 对象中. 最后进入主循环 doEventLoop(). doEventLoop()负责 Live555 中整个任务调度. 它是一个消息循环, 一旦进入该循环就会不断地查找延时事件. 当有延时事件发生时, 任务调度器就会获取延时事件的句柄, 然后调用与之相关联的处理函数去处理相应的任务, 主要是负责响应 RTSP 请求和 RTP 包的定时发送.

当客户端向服务端发送 OPTIONS 请求报文后, 服务端会进行响应, 并提供的其可用方法. 接着客户端会向服务器端发送 DESCRIBE 报文请求, 服务器在收到请求报文后就会去查找对应的流媒体信息, 如果存在该流媒体信息的话, 就将获取的 SDP (Session Description Protocol)描述信息发送给客户端, 如果没有的话就告诉客户端不存在该流媒体信息. 接着客户端会发送 SETUP 请求报文, 用来提醒流媒体服务器建立会话和确定 RTP 包的传输模式, 在本系统中默认的是使用 UDP 协议传输方式. 服务器收到 SETUP 请求后会去查看有没有创建过响应的 ServerMediaSession, 如果有的话就不用再创建, 如果没有的话要重新创建一个 ServerMediaSession 对象、Source 对象和 RTPSink, 而本系统在服务器进程刚开始的时候就已经创建完成了, 所以这里就不用重新创建了. 最后客户端端向服务器发送 PLAY 请求报文, 服务器收到 PLAY 请求后, 就会启动相应的流. 下面是 RTP 打包发送的流程图.

首先调用 MediaSink 中的流启动函数 start-Playing(), RTP 的打包与发送就是从这里开始的. 接着会调用 continuePlaying()函数, 而该函数是定义在

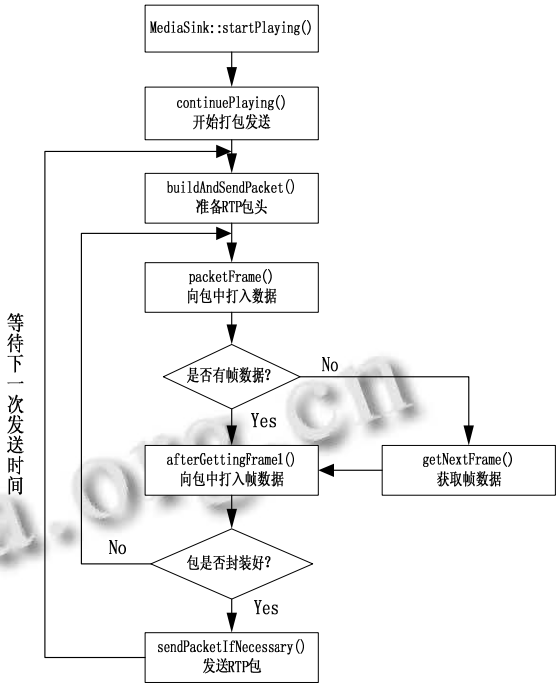

图 2　RTP 打包发送流程图

MediaSink 类中纯虚函数, 必须要在 Sink 子类中实现. 本系统中传输的是 H.264 格式的视频流, 所以会调用 H.264 的 Sink 子类 H264VideoRTPSink 中的 continue-Playing()函数, 该函数会创建一个辅助类 H264FUA-Fragmenter, 这个类负责 H.264 负载的 RTP 封包. 接着会调用 buildAndSendPacket()函数来完成 RTP 包头的准备工作, 如序列号、同步源和负载类型等信息. 接着调用 PacketFrame()函数, 此时分为两种情况:

(1) 如果在发送缓冲区 buffer 中还有未发送的数据, 可以调用 afterGettingFrame1()函数将帧数据打入到包中.

(2) 如果 buffer 中不存在数据, 就会调用 getNextFrame()函数来获取新的帧数据.

当调用 afterGettingFrame1()函数时, 该函数会处理帧的分片, 当一帧数据大于 UDP 的有效载荷时就必须进行分片, 同时也会调用 doSpecialFrameHandling()函数, 该函数负责对前面的 RTP 包头的预留空间进行填充, 包括标志位和时间戳.

随后会调用 sendPacketIfNecessary()函数来完成 RTP 包发送前的准备工作. 如果 Packet(封装的 RTP 包)中存在帧数据, 便会调用 sendPacket()函数将 packet 发送出去, 同时调用 taskScheduler().scheduleDelayer-Task()函数进行延时处理, 并且将下一次需要发送的

RTP 包添加到任务调度器中. 然后调用 sendNext()函数进行下一次 RTP 包发送. 这样服务器就可以不断地向客户端发送 RTP 包了.

4 系统测试

本系统的前端采集编码和服务器端测试是在带有摄像头的 Linux 系统下完成的, 客户端是 Windows 系统下的 VLC 播放器. 在 Linux 系统下正常运行系统后会显示下图所示的运行结果.

然后在 VLC 中打开网络串流并输入 rtsp://192.168.0.146/webcam, 完成后点击播放, 就可以观看到来自摄像头实时采集的视频. 观察可以发现流媒体传输比一般的监控系统画面跟清晰, 时延也更小.

5 结语

本文提出了一种基于 Live555 的实时流媒体传输系统, 在Live555 开源系统基础上进行改进, 并加入了视频采集和 H.264 视频编码, 从而实现了 H.264 码流的实时编码、接收和转发. 本系统能较好的解决在网络上传输多媒体数据的延时和不同步等问题, 而且系统有着很好的健壮性和可扩展性, 可广泛应用于各种网络音视频传输场合. 由于本系统还处在测试阶段, 所以功能有限, 后续将添加更实用的功能, 提高用户体验.

参考文献

1 章闽融,徐亚锋,等.RTP/RTCP 协议在视频监控系统中的实现.计算机应用与软件,2006,(1):79–81.
2 茅炎菲.基于 RTSP 协议和 H.264 编码的网络视频监控系统的设计与实现[硕士学位论文].杭州:浙江大学,2011.
3 陈锋锋.基于 RTSP 的流媒体传输系统的应用开发[硕士学位论文].南京:南京邮电大学,2013.
4 刘畅棂,包杰,王宁国.基于 Live555 的网络视频监控技术与实现.现代电信科技,2012,12(12):38–42.
5 梅大成,杨大千,赵娜.基于 Linux 的嵌入式网络摄像机设计.微计算机信息,2007,23(8):45–47.
6 李校林,刘海波.RTP/RTCP,RTSP 在无线视频监控系统的设计与实现.电视技术,2011,35(19):89–92.
7 毕厚杰.新一代视频压缩标准 H.264/AVC.北京:人民邮电出版社,2005.