文章编号:1671-7872(2005)03-0293-05

Linux下实时流媒体的编程实现

陈小平,王皖陵
(安徽工业大学 计算机学院, 安徽 马鞍山 243002)

摘要:RTP协议是进行实时流媒体传输的标准协议和关键技术,重点分析了在Linux平台上运用RTP协议进行实时流媒体编程的实现方法。本方法可在网络中实现多媒体数据的发送、接收以及对控制信息的设置。文中给出了C++实现的代码。

关键词: RTP;Linux;实时流传输;编程
中图分类号:TP316.8 **文献标识码**:B

Programing of Real-time Streaming of Multimedium in Linux

CHEN Xiao-ping , WANG Wan-ling
(School of Computer Science, Anhui University of Technology, Ma´anshan 243002, China)

Abstract: Protocol of RTP is a standard and key technology of real-time streaming transport. A programing method of Real-time Streaming of Multimedia with RTP in Linux is analyzed and discussed. With this method,the data of multimedium are sent and received in internet and some controls can be set.C++ codes of the method are presented in the paper.

Key words: RTP; Linux; real-time streaming transport; programing

1 流媒体概述

流[1](Streaming)是近年在Internet上出现的新概念,主要是指通过网络传输多媒体数据的技术总称。流媒体包含广义和狭义两种内涵:广义上的流媒体指的是使音频和视频形成稳定和连续的传输流和回放流的一系列技术、方法和协议的总称,即流媒体技术;狭义上的流媒体是相对于传统的下载-回放方式而言的,指的是一种从Internet上获取音频和视频等多媒体数据的新方法,它能够支持多媒体数据流的实时传输和实时播放。通过运用流媒体技术,服务器能够向客户机发送稳定和连续的多媒体数据流,客户机在接收数据的同时以一个稳定的速率回放,而不用等数据全部下载完之后再进行回放。

流媒体技术的发展依赖于网络的传输条件、媒体文件的传输控制、媒体文件的编码压缩效率及客户端的解码等几个重要因素。其中任何一个因素都会影响流媒体技术的发展和应用。而目前实现流媒体传输主要有两种方法:顺序流(progressive streaming)传输和实时流(real-time streaming)传输,它们分别适合于不同的应用场合。

1.1 顺序流传输

顺序流传输采用顺序下载的方式进行传输,在下载的同时用户可以在线回放多媒体数据,但给定时刻只能观看已经下载的部分,不能跳到尚未下载的部分,也不能在传输期间根据网络状况对下载速度进行调整。由于标准的HTTP服务器就可以发送这种形式的流媒体,而不需要其他特殊协议的支持,因此也常常被称作HTTP流式传输。顺序流式传输比较适合于高质量的多媒体片段,如片头、片尾或者广告等。

1.2 实时流传输

实时流传输保证媒体信号带宽能够与当前网络状况相匹配,从而使得流媒体数据总是被实时地传送,因此特别适合于现场事件。实时流传输支持随机访问,即用户可以通过快进或者后退操作来观看前面或者后面

收稿日期:2004-09-07

作者简介:陈小平(1965-),男,安徽当涂人,安徽工业大学计算机学院工程师。

的内容。从理论上讲,实时流媒体一经播放就不会停顿,但事实上仍有可能发生周期性的暂停现象,尤其是在网络状况不太好的情况下更是如此。与顺序流传输不同的是,实时流传输需要用到特定的流媒体服务器,而且还需要特定网络协议的支持。

2 流媒体协议

实时传输协议[2](Real-time Transport Protocol, RTP)是在Internet上处理多媒体数据流的一种网络协议,利用它能够在单播或多播的网络环境中实现流媒体数据的实时传输。RTP通常使用用户数据报协议(User Datagram Protocol,UDP)来进行多媒体数据的传输,但如果需要的话可以使用TCP或者ATM等其它协议,整个RTP协议由两个密切相关的部分组成:RTP数据协议和RTP控制协议(Real-timeTransport Control Protocol, RTCP)。其中RTP数据协议负责对流媒体数据进行封包并实现媒体流的实时传输,而RTCP会采用与RTP相同的分发机制,向会话中的所有成员周期性地发送控制信息,应用程序通过接收这些数据,从中获取会话参与者的相关资料,以及网络状况、分组丢失概率等反馈信息,从而能够对服务质量进行控制或者对网络状况进行诊断。实时流协议(Real Time Streaming Protocol,RTSP),位于RTP和RTCP之上,是一个流媒体表示协议,主要用来控制具有实时特性的数据发送,但它本身并不传输数据,而是必须依赖于下层传输协议所提供的某些服务。RTSP可以对流媒体提供诸如播放、暂停、快进等操作,负责定义具体的控制消息、操作方法、状态码等,此外还描述了与RTP间的交互操作。

3 流媒体编程

RTP是目前解决流媒体实时传输问题的最好办法,在Linux平台上进行实时流媒体编程,可使用一些开放源代码的RTP库,如LIBRTP、JRTPLIB等(http://research.edm.luc.ac.be/jori/jrtplib/jrtplib.html)。JRTPLIB是一个面向对象的RTP库,下面就以JRTPLIB为例,介绍在Linux平台上运用RTP协议进行实时流媒体编程的使用方法[1,3]。

3.1 RTP会话初始化

JRTPLIB是一个用C++语言实现的RTP库,可以运行在Windows、Linux、FreeBSD、Solaris、Unix和VxWorks等多种操作系统上[3]。在Linux系统中安装JRTPLIB,在使用JRTPLIB进行实时流媒体数据传输之前,首先应该生成RTPSession类的一个实例来表示此次RTP会话,然后调用Create()方法来对其进行初始化操作。RTPSession类的Create()方法只有一个参数,用来指明此次RTP会话所采用的端口号。下面是一个初始化框架的代码:

```
#include <stdio.h>
#include "rtpsession.h"
int main(void)
{
  RTPSession sess;
  int status;
  char* msg;
  sess.Create(6000);
  msg=RTPGetErrorString(status);
  printf("Error String: %s\\n", msg);
  return 0;
}
```

设置恰当的时戳单元,是RTP会话初始化过程所要进行的另一项重要工作,是通过调用RTPSession类的SetTimestampUnit()方法来实现的,该方法同样也只有一个参数,表示的是以秒为单元的时戳单元。例如,当使用RTP会话传输8 000 Hz采样的音频数据时,由于时戳每秒钟将递增8 000,所以时戳单元相应地应该被设置成1/8 000。如:

```
sess.SetTimestampUnit(1.0/8 000.0);
```

3.2 数据发送

当 RTP 会话成功建立起来之后,接下去就可以开始进行流媒体数据的实时传输了。首先需要设置好数据发送的目标地址,RTP 协议允许同一会话存在多个目标地址,这可以通过调用 RTPSession 类的 AddDestination()、DeleteDestination()和 ClearDestinations()方法来完成。例如,下面的语句表示的是让 RTP 会话将数据发送到本地主机的 7000 端口:

```
unsigned long addr = ntohl(inet_addr("127.0.0.1"));
sess.AddDestination(addr, 7000);
```

目标地址全部指定之后,接着就可以调用 RTPSession 类的 SendPacket()方法,向所有的目标地址发送流媒体数据。SendPacket()是 RTPSession 类提供的一个重载函数,它具有下列多种形式:

```
int SendPacket(void *data,int len)
int SendPacket(void *data,int len,unsigned char pt,bool mark,unsigned long timestampinc)
int SendPacket(void *data,int len,unsigned short hdrextID,void *hdrextdata,int numhdrextwords)
int SendPacket (void *data,int len,unsigned char pt,bool mark,unsigned long timestampinc,unsigned short hdrextID,void *hdrextdata,int numhdrextwords)
```

SendPacket()最典型的用法是类似于下面的语句,其中第一个参数是要被发送的数据,而第二个参数则指明将要发送数据的长度,其后依次是 RTP 负载类型、标识和时戳增量。

```
sess.SendPacket(buffer, 5, 0, false, 10);
```

对于同一个 RTP 会话,负载类型、标识和时戳增量通常都是相同的,JRTPLIB 允许将它们设置为会话的默认参数,这是通过调用 RTPSession 类的 SetDefaultPayloadType ()、SetDefaultMark () 和 SetDefaultTimeStampIncrement()方法来完成的。为 RTP 会话设置这些默认参数的好处是可以简化数据的发送,例如,如果为 RTP 会话设置了默认参数:

```
sess.SetDefaultPayloadType(0);
sess.SetDefaultMark(false);
sess.SetDefaultTimeStampIncrement(10);
```

之后在进行数据发送时只需指明要发送的数据及其长度就可以了。如:

```
sess.SendPacket(buffer, 5);
```

3.3 数据接收

对于流媒体数据的接收端,首先需要调用 RTPSession 类的 PollData()方法来接收发送过来的 RTP 或者 RTCP 数据报。由于同一个 RTP 会话中允许有多个参与者(源),既可以通过调用 RTPSession 类的 GotoFirstSource () 和 GotoNextSource () 方法来遍历所有的源,也可以通过调用 RTPSession 类的 GotoFirstSourceWithData()和 GotoNextSourceWithData()方法来遍历那些携带有数据的源。在从 RTP 会话中检测出有效的数据源之后,接下去就可以调用 RTPSession 类的 GetNextPacket()方法从中抽取 RTP 数据报,当接收到的 RTP 数据报处理完之后,一定要记得及时释放。下面的代码示范了该如何对接收到的 RTP 数据报进行处理:

```
if (sess.GotoFirstSourceWithData()) {
    do {
        RTPPacket *pack;
        pack = sess.GetNextPacket();
        // 处理接收到的数据
        delete pack;
    } while (sess.GotoNextSourceWithData());
}
```

JRTPLIB 为 RTP 数据报定义了三种接收模式:RECEIVEMODE_ALL、RECEIVEMODE_IGNORESOME、

RECEIVEMODE_ACCEPTSOME。每种接收模式都具体规定了哪些到达的RTP数据报将会被接受,而哪些到达的RTP数据报将会被拒绝。通过调用RTPSession类的SetReceiveMode()方法可以设置这些接收模式。

3.4 控制信息

JRTPLIB是一个高度封装后的RTP库,编程时不用关心RTCP数据报是如何被发送和接收的,这些都可以由JRTPLIB来自动完成。只要PollData()或者SendPacket()方法被成功调用,JRTPLIB就能够自动对到达的RTCP数据报进行处理,并且还会在需要的时候发送RTCP数据报,从而能够确保整个RTP会话过程的正确性。

而另一方面,通过调用RTPSession类提供的SetLocalName (),SetLocalEMail (),SetLocalLocation(),SetLocalPhone(),SetLocalTool()和SetLocalNote()方法,JRTPLIB又允许程序员对RTP会话的控制信息进行设置。所有这些方法在调用时都带有两个参数,其中第一个参数是一个char型的指针,指向将要被设置的数据;而第二个参数则是一个int型的数值,表明该数据中的前面多少个字符将会被使用。例如下面的语句可以被用来设置控制信息中的电子邮件地址:

```
sess.SetLocalEMail("soony@263.net",13);
```

在RTP会话过程中,不是所有的控制信息都需要被发送,通过调用RTPSession类提供的EnableSendName (),EnableSendEMail (),EnableSendLocation (),EnableSendPhone (),EnableSendTool () 和EnableSendNote()方法,可以为当前RTP会话选择将被发送的控制信息。

4 应用实例

下面通过一个简单的流媒体发送-接收实例,介绍怎样利用JRTPLIB来进行实时流媒体的编程。受篇幅限制,文中只给出数据发送端部分代码,它负责向用户指定的IP地址和端口,不断地发送RTP数据包。数据接收端的代码(它负责从指定的端口不断地读取RTP数据包)略。程序代码在Linux+C++环境中调试通过。本实例在局域网多媒体教室中作了简单应用,无论音频还是视频的传输信息都甚为流畅,无需下载即可收听收看媒体信息。

```
#include <stdio.h>
#include <string.h>
#include "rtpsession.h"
void checkerror(int err) // 错误处理函数
{
  if (err < 0) {
      char* errstr = RTPGetErrorString(err);
      printf("Error:%s\\n", errstr);
      exit(-1);
  }
}
int main(int argc, char ** argv)
{
  RTPSession sess;
  unsigned long destip;
  int destport;
  int portbase = 6000;
  int status, index;
  char buffer[128];
  if(argc! =3){
     printf("Usage: ./sender destip destport\\n");
```

```
        return -1;
    }
    destip = inet_addr(argv[1]);  // 获得接收端的 IP 地址和端口号
    if (destip == INADDR_NONE) {
        printf("Bad IP address specified.\\n");
        return -1;
    }
    destip = ntohl(destip);
    destport = atoi(argv[2]);
    status = sess.Create(portbase);  // 创建 RTP 会话
    checkerror(status);
    status = sess.AddDestination(destip, destport);  // 指定 RTP 数据接收端
    checkerror(status);
    sess.SetDefaultPayloadType(0);  // 设置 RTP 会话默认参数
    sess.SetDefaultMark(false);
    sess.SetDefaultTimeStampIncrement(10);
    index = 1;  // 发送流媒体数据
    do {
        sprintf(buffer, "%d: RTP packet", index ++);
        sess.SendPacket(buffer, strlen(buffer));
        printf("Send packet ! \\n");
    } while(1);
    return 0;
    }
```

5 小　　结

随着多媒体数据在 Internet 上所承担的作用越来越重要,需要实时传输音频和视频等多媒体数据的场合也将变得越来越多,如 IP 电话、视频点播、在线会议、远程监控等。RTP 是用来在 Internet 上进行实时流媒体传输的一种协议,目前已经被广泛地应用在各种场合,JRTPLIB 是一个面向对象的 RTP 封装库,利用它可以很方便地完成 Linux 平台上的实时流媒体编程。

参考文献:

[1]肖磊,陈卓,郑重,谷守斌. 流媒体技术与应用完全手册[M]. 北京:清华大学出版社,2003.

[2]黄永峰. IP 网络多媒体通信技术[M].北京:人民邮电出版社,2003.

[3]徐延明,林立志,王罡. Linux 编程指南与实例[M]. 北京:人民邮电出版社, 2000.

[4] Eckel, Bruce(美). C++ 编程思想[M]. 刘宗田,邢大红译. 北京:机械工业出版社, 2000.