

视音频数据处理入门：UDP-RTP协议解析

2016年01月31日 21:39:37 阅读数：65360

视音频数据处理入门系列文章：

[视音频数据处理入门：RGB、YUV像素数据处理](#)

[视音频数据处理入门：PCM音频采样数据处理](#)

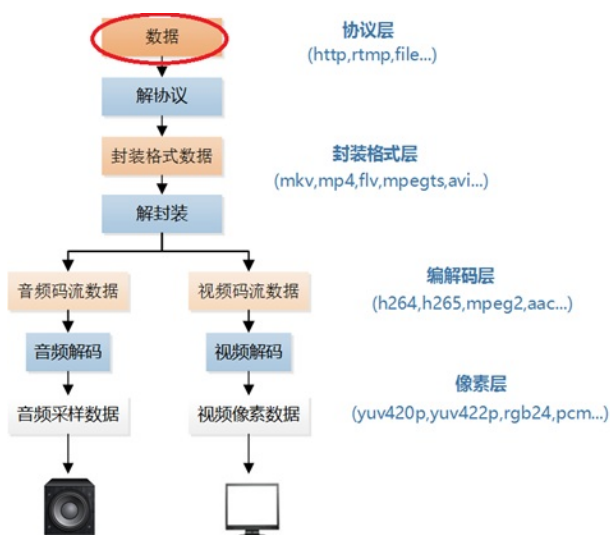
[视音频数据处理入门：H.264视频码流解析](#)

[视音频数据处理入门：AAC音频码流解析](#)

[视音频数据处理入门：FLV封装格式解析](#)

[视音频数据处理入门：UDP-RTP协议解析](#)

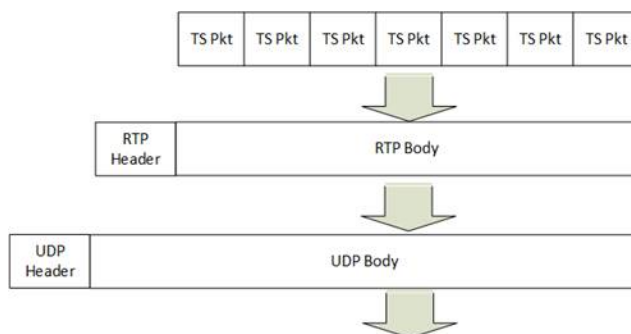
本文介绍网络协议数据的处理程序。网络协议数据在视频播放器中的位置如下所示。



本文中的程序是一个UDP/RTP协议流媒体数据解析器。该程序可以分析UDP协议中的RTP 包头中的内容，以及RTP负载中MPEG-TS封装格式的信息。通过修改该程序可以实现不同的UDP/RTP协议数据处理功能。

原理

MPEG-TS封装格式数据打包为RTP/UDP协议然后发送出去的流程如下图所示。图中首先每7个MPEG-TS Packet打包为一个RTP，然后每个RTP再打包为一个UDP。其中打包RTP的方法就是在MPEG-TS数据前面加上RTP Header，而打包RTP的方法就是在RTP数据前面加上UDP Header。



有关MPEG-TS、RTP、UDP的知识不再详细介绍，可以参考相关的文档了解其中的细节信息。本文记录的程序是一个收取流媒体的程序，因此本文程序的流程和上述发送MPEG-TS的流程正好是相反的。该程序可以通过Socket编程收取UDP包，解析其中的RTP包的信息，然后再解析RTP包中MPEG-TS Packet的信息。

代码

整个程序位于simplest_udp_parser()函数中，如下所示。

```
[cpp]  
1.  /**
2.   * 最简单的视音频数据处理示例
3.   * Simplest MediaData Test
4.   *
5.   * 雷霄骅 Lei Xiaohua
6.   * leixiaohua1020@126.com
7.   * 中国传媒大学/数字电视技术
8.   * Communication University of China / Digital TV Technology
9.   * http://blog.csdn.net/leixiaohua1020
10.  *
11.  * 本项目包含如下几种视音频测试示例：
12.  * (1) 像素数据处理程序。包含RGB和YUV像素格式处理的函数。
13.  * (2) 音频采样数据处理程序。包含PCM音频采样格式处理的函数。
14.  * (3) H.264码流分析程序。可以分离并解析NALU。
15.  * (4) AAC码流分析程序。可以分离并解析ADTS帧。
16.  * (5) FLV封装格式分析程序。可以将FLV中的MP3音频码流分离出来。
17.  * (6) UDP-RTP协议分析程序。可以将分析UDP/RTP/MPEG-TS数据包。
18.  *
19.  * This project contains following samples to handling multimedia data:
20.  * (1) Video pixel data handling program. It contains several examples to handle RGB and YUV data.
21.  * (2) Audio sample data handling program. It contains several examples to handle PCM data.
22.  * (3) H.264 stream analysis program. It can parse H.264 bitstream and analysis NALU of stream.
23.  * (4) AAC stream analysis program. It can parse AAC bitstream and analysis ADTS frame of stream.
24.  * (5) FLV format analysis program. It can analysis FLV file and extract MP3 audio stream.
25.  * (6) UDP-RTP protocol analysis program. It can analysis UDP/RTP/MPEG-TS Packet.
26.  *
27.  */
28.  #include <stdio.h>
29.  #include <winsock2.h>
30.
31.  #pragma comment(lib, "ws2_32.lib")
32.
33.  #pragma pack(1)
34.
35.  /**
36.   * [memo] Ffmpeg stream Command:
37.   * ffmpeg -re -i sintel.ts -f mpegts udp://127.0.0.1:8880
38.   * ffmpeg -re -i sintel.ts -f rtp_mpegts udp://127.0.0.1:8880
39.   */
40.
41.  typedef struct RTP_FIXED_HEADER{
42.      /* byte 0 */
43.      unsigned char csrc_len:4;      /* expect 0 */
44.      unsigned char extension:1;     /* expect 1 */
45.      unsigned char padding:1;       /* expect 0 */
46.      unsigned char version:2;       /* expect 2 */
47.      /* byte 1 */
48.      unsigned char payload:7;
49.      unsigned char marker:1;        /* expect 1 */
50.      /* bytes 2, 3 */
51.      unsigned short seq_no;
52.      /* bytes 4-7 */
53.      unsigned long timestamp;
54.      /* bytes 8-11 */
55.      unsigned long ssrc;            /* stream number is used here. */
56.  } RTP_FIXED_HEADER;
57.
58.  typedef struct MPEGTS_FIXED_HEADER {
59.      unsigned sync_byte: 8;
60.      unsigned transport_error_indicator: 1;
61.      unsigned payload_unit_start_indicator: 1;
62.      unsigned transport_priority: 1;
63.      unsigned PID: 13;
64.      unsigned scrambling_control: 2;
65.      unsigned adaptation_field_exist: 2;
66.      unsigned continuity_counter: 4;
67.  } MPEGTS_FIXED_HEADER;
68.
69.
70.
71.  int simplest_udp_parser(int port)
72.  {
73.      WSADATA wsaData;
74.      WORD sockVersion = MAKEWORD(2,2);
75.      int cnt=0;
76.
77.      //FILE *myout=fopen("output_log.txt", "wb+");
78.      FILE *myout=stdout;
79.
80.      FILE *fp1=fopen("output_dump.ts", "wb+");
81.
82.      if(WSAStartup(sockVersion, &wsaData) != 0){
83.          return 0;
84.      }
85.
86.      SOCKET serSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
87.      if(serSocket == INVALID_SOCKET){
```

```

88.     printf("socket error !");
89.     return 0;
90. }
91.
92. sockaddr_in serAddr;
93. serAddr.sin_family = AF_INET;
94. serAddr.sin_port = htons(port);
95. serAddr.sin_addr.S_un.S_addr = INADDR_ANY;
96. if(bind(serSocket, (sockaddr *)&serAddr, sizeof(serAddr)) == SOCKET_ERROR){
97.     printf("bind error !");
98.     closesocket(serSocket);
99.     return 0;
100. }
101.
102. sockaddr_in remoteAddr;
103. int nAddrLen = sizeof(remoteAddr);
104.
105. //How to parse?
106. int parse_rtp=1;
107. int parse_mpegts=1;
108.
109. printf("Listening on port %d\n",port);
110.
111. char recvData[10000];
112. while (1){
113.
114.     int pktsize = recvfrom(serSocket, recvData, 10000, 0, (sockaddr *)&remoteAddr, &nAddrLen);
115.     if (pktsize > 0){
116.         //printf("Addr:%s\r\n",inet_ntoa(remoteAddr.sin_addr));
117.         //printf("packet size:%d\r\n",pktsize);
118.         //Parse RTP
119.         //
120.         if(parse_rtp!=0){
121.             char payload_str[10]={0};
122.             RTP_FIXED_HEADER rtp_header;
123.             int rtp_header_size=sizeof(RTP_FIXED_HEADER);
124.             //RTP Header
125.             memcpy((void *)&rtp_header, recvData, rtp_header_size);
126.
127.             //RFC3551
128.             char payload=rtp_header.payload;
129.             switch(payload){
130.                 case 0:
131.                 case 1:
132.                 case 2:
133.                 case 3:
134.                 case 4:
135.                 case 5:
136.                 case 6:
137.                 case 7:
138.                 case 8:
139.                 case 9:
140.                 case 10:
141.                 case 11:
142.                 case 12:
143.                 case 13:
144.                 case 14:
145.                 case 15:
146.                 case 16:
147.                 case 17:
148.                 case 18: sprintf(payload_str,"Audio");break;
149.                 case 31: sprintf(payload_str,"H.261");break;
150.                 case 32: sprintf(payload_str,"MPV");break;
151.                 case 33: sprintf(payload_str,"MP2T");break;
152.                 case 34: sprintf(payload_str,"H.263");break;
153.                 case 96: sprintf(payload_str,"H.264");break;
154.                 default:sprintf(payload_str,"other");break;
155.             }
156.
157.             unsigned int timestamp=ntohl(rtp_header.timestamp);
158.             unsigned int seq_no=ntohs(rtp_header.seq_no);
159.
160.             fprintf(myout, "[RTP Pkt] %5d| %5s| %10u| %5d| %5d|\n", cnt, payload_str, timestamp, seq_no, pktsize);
161.
162.             //RTP Data
163.             char *rtp_data=recvData+rtp_header_size;
164.             int rtp_data_size=pktsize-rtp_header_size;
165.             fwrite(rtp_data, rtp_data_size, 1, fp1);
166.
167.             //Parse MPEGTS
168.             if(parse_mpegts!=0&&payload==33){
169.                 MPEGTS_FIXED_HEADER mpegts_header;
170.                 for(int i=0;i<rtp_data_size;i=i+188){
171.                     if(rtp_data[i]!=0x47)
172.                         break;
173.                     //MPEGTS Header
174.                     //memcpy((void *)&mpegts_header, rtp_data+i, sizeof(MPEGTS_FIXED_HEADER));
175.                     fprintf(myout, " [MPEGTS Pkt]\n");
176.                 }
177.             }
178.
179.

```

```

179.         reset
180.         fprintf(myout, "[UDP Pkt] %5d| %5d|\n", cnt, pktsize);
181.         fwrite(recvData, pktsize, 1, fp1);
182.     }
183.
184.     cnt++;
185. }
186. }
187. closesocket(serSocket);
188. WSACleanup();
189. fclose(fp1);
190.
191. return 0;
192. }

```

上文中的函数调用方法如下所示。

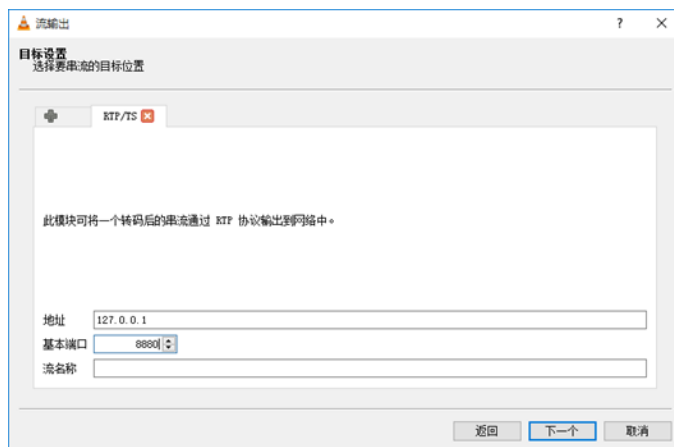
```

1. simplest_udp_parser(8880);

```

结果

本程序输入为本机的一个端口号，输出为UDP/RTP/MPEG-TS的解析结果。程序开始运行后，可以使用推流软件向本机的udp://127.0.0.1:8880地址进行推流。例如可以使用VLC Media Player的“打开媒体”对话框中的“串流”功能（位于“播放”按钮旁边的小三角按钮的菜单中）。在该功能的对话框中添加一个“RTP / MPEG Transport Stream”的新目标。



也可以使用FFmpeg对本机的8880端口进行推流。下面的命令可以推流UDP封装的MPEG-TS。

```

1. ffmpeg -re -i sintel.ts -f mpegts udp://127.0.0.1:8880

```

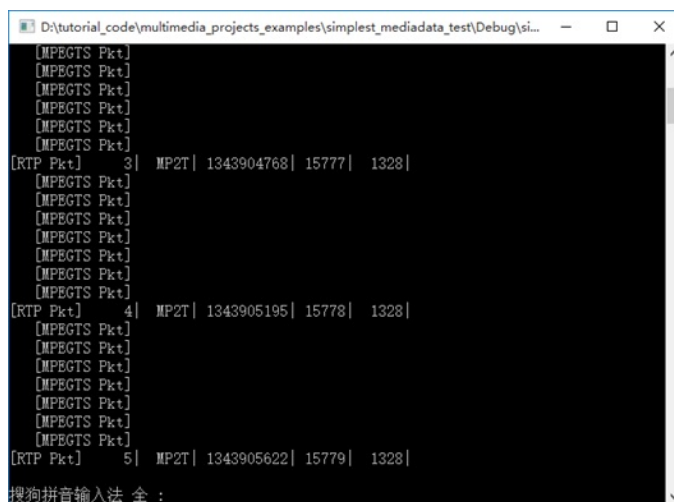
下面的命令可以推流首先经过RTP封装，然后经过UDP封装的MPEG-TS。

```

1. ffmpeg -re -i sintel.ts -f rtp_mpegts udp://127.0.0.1:8880

```

推流之后，本文的程序会通过Socket接收到UDP包并且解析其中的数据。解析的结果如下图所示。



下载

Simplest mediadata test

项目主页

SourceForge：<https://sourceforge.net/projects/simplest-mediadata-test/>

Github：https://github.com/leixiaohua1020/simplest_mediadata_test

开源中国：http://git.oschina.net/leixiaohua1020/simplest_mediadata_test

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/9422409>

本项目包含如下几种视音频数据解析示例：

- (1)像素数据处理程序。包含RGB和YUV像素格式处理的函数。
- (2)音频采样数据处理程序。包含PCM音频采样格式处理的函数。
- (3)H.264码流分析程序。可以分离并解析NALU。
- (4)AAC码流分析程序。可以分离并解析ADTS帧。
- (5)FLV封装格式分析程序。可以将FLV中的MP3音频码流分离出来。
- (6)UDP-RTP协议分析程序。可以将分析UDP/RTP/MPEG-TS数据包。

雷霄骅 (Lei Xiaohua)

leixiaohua1020@126.com

<http://blog.csdn.net/leixiaohua1020>

版权声明：本文为博主原创文章，未经博主允许不得转载。<https://blog.csdn.net/leixiaohua1020/article/details/50535230>

文章标签：[udp](#) [rtp](#) [mpegts](#) [流媒体](#)

个人分类：[我的开源项目](#)

此PDF由[spyyg](#)生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com