ajaxhe的专栏

ortp学习笔记

分类： C/C++ 音频编解码 2012-07-16 22:07 2885人阅读 评论(9) 收藏 举报

session video codec parameters header

ortp版本：ortp-0.18.0.tar.gz

操作系统：window 7 32bit

1.windows下编译ortp.lib

直接打开ortp-0.18.0\build\win32native的工程文件即可，VS2008下无需任何修改，即可编译出动态链接库 ortp.lib以及ortp.dll。

2.使用ortp提供的测试程序：ortp-0.18.0\src\tests下的win_receive以及win_sender目录下程序

在windows7下使用win_receiver时会提示下如下错误，这个在错误在windows XP下是不存在的

QOSAddSocketToFlow failed to add a flow with error 87

具体问题暂时还没有深究，这个应该是一个系统兼容性问题，需要系统支持qwave.lib。

在查找了错误出处，对比了前几个ortp的版本后，对ortp0.18的源代码进行了修改

rtp_session_inet.c

```
/* set socket options (but don't change chosen states) */
/*
        rtp_session_set_dscp( session, -1 );
        rtp_session_set_multicast_ttl( session, -1 );
        rtp_session_set_multicast_loopback( session, -1 );
*/
```

3. 发送H.264视频帧

ortp提供的win_receive以及win_sender一次只发送160字节的数据。但我们在发送一帧H.264视频帧，每次需要发送2000-3000字节的数据，关于ortp发送H.264视频帧，可以参考：

ortp编程示例代码

谈谈RTP传输中的负载类型和时间戳

除了上面两篇文章提到需要注意的负载类型和时间戳之外，在ortp中win_receive中还需要显示调用：

```
rtp_session_set_recv_buf_size(rtp_session_mgr .rtp_session , recv_bufsize);
```

这里的recv_bufsize必须要比win_sender中

```
sended_bytes = rtp_session_send_with_ts (rtp_session_mgr .rtp_session ,
                    ( uint8_t *) send_buffer,
                     wrapLen,
                    rtp_session_mgr.cur_timestamp );
```

的wrapLen要大，否则在receive端会出现如下错误：

ortp-warning-Error receiving RTP packet: Error code : 10040, err num [10040],error [-1]

---

4. 一个结合H.264编码，rtp发送，接收，保存mp4文件的小程序

下载地址：ortp测试程序（修改版）

个人资料

ajaxhe

访问： 81960次

积分： 1459分

排名： 第9376名

原创： 54篇 转载： 15篇

译文： 1篇 评论： 98条

文章搜索

文章分类

文章存档

展开

编译环境：vs2008

压缩文件中包括了ffmpeg以及ortp的动态链接库，但你还需要在工程文件中修改它们的路径才能正确链接这两个库.

因为这个小程序是一个项目的一部分，源代码中还有些冗余代码，并没有来的及删掉，大家在看的时候需要注意下

receiver

[cpp]

```cpp
#include <string.h>
#include "ortp/ortp.h"

extern "C"{
#include <libavformat/avformat.h>
#include <libswscale/swscale.h>
};

bool m_bExit = FALSE;

struct RtpSessionMgr
{
    RtpSession *rtp_session;
    int timestamp;
};


RtpSessionMgr rtp_session_mgr;
const int timestamp_inc = 3600; // 90000/25

const char recv_ip[] = "127.0.0.1";
const int recv_port = 8008;
const int recv_bufsize = 10240;
unsigned char *recv_buf;

/** 帧包头的标识长度 */
#define CMD_HEADER_LEN 10

/** 帧包头的定义 */
static uint8_t CMD_HEADER_STR[CMD_HEADER_LEN] = { 0xAA,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7

/** 帧的包头信息 */
typedef struct _sFrameHeader
{
    /** 命令名称标识 */
    unsigned char cmdHeader[CMD_HEADER_LEN];

    /** 采集的通道号 0~7*/
    unsigned char chId;

    /** 数据类型，音频 或者 视频*/
    unsigned char dataType;

    /** 缓冲区的数据长度 */
    uint32_t len;

    /** 时间戳 */
    uint32_t timestamp;

}FrameHeader;

///////ffmpeg//////////////////////////////////////////////////////////////////
AVOutputFormat *fmt;
AVFormatContext *oc;
AVStream *video_st;
AVCodecContext *codecContext;

const int image_width = 704;
const int image_height = 576;
const int frame_rate = 25;
static int frame_count;

BOOL ctrlHandlerFunction(DWORD fdwCtrlType)
{
    switch (fdwCtrlType)
    {
        // Handle the CTRL+C signal.
        // CTRL+CLOSE: confirm that the user wants to exit.
    case CTRL_C_EVENT:
```

```
    case CTRL_CLOSE_EVENT:
    case CTRL_BREAK_EVENT:
    case CTRL_LOGOFF_EVENT:
    case CTRL_SHUTDOWN_EVENT:
        m_bExit = TRUE;
        return TRUE;

    default:
        return FALSE;
    }
}

void rtpInit()
{
    int ret;
    WSADATA wsaData;

    /** 初始化winsocket */
    if ( WSAStartup(MAKEWORD(2,2), &wsaData) != 0)
    {
        fprintf(stderr, "WAStartup failed!\n");
        return ;
    }

    ortp_init();
    ortp_scheduler_init();

    rtp_session_mgr.rtp_session = rtp_session_new(RTP_SESSION_RECVONLY);

    rtp_session_set_scheduling_mode(rtp_session_mgr.rtp_session, 1);
    rtp_session_set_blocking_mode(rtp_session_mgr.rtp_session, 1);
    rtp_session_set_local_addr(rtp_session_mgr.rtp_session, recv_ip, recv_port);

    rtp_session_enable_adaptive_jitter_compensation(rtp_session_mgr.rtp_session, TRUE);
    rtp_session_set_jitter_compensation(rtp_session_mgr.rtp_session, 40);

    rtp_session_set_payload_type(rtp_session_mgr.rtp_session, 34);
    rtp_session_set_recv_buf_size(rtp_session_mgr.rtp_session, recv_bufsize);

    rtp_session_mgr.timestamp = timestamp_inc;
}

int rtp2disk()
{
    int err;
    int havemore = 1;

    while (havemore)
    {
        err = rtp_session_recv_with_ts(rtp_session_mgr.rtp_session,
                                       (uint8_t *)recv_buf, recv_bufsize,
                                       rtp_session_mgr.timestamp, &havemore);

        if (havemore)
            printf("==> Warning: havemore=1!\n");

        if (err > 0)
        {
            FrameHeader *frameHeader;

            printf("receive data is %d\n", err);

            frameHeader = (FrameHeader *)recv_buf;
            printf("frame_len = %d\n", frameHeader->len);

            AVPacket pkt;
            av_init_packet(&pkt);
            pkt.stream_index= video_st->index;
            pkt.data= recv_buf + sizeof(FrameHeader);
            pkt.size = frameHeader->len; // not the video_outbuf_size, note!
            // write the compressed frame in the media file
            err = av_write_frame(oc, &pkt);

            if (err != 0)
            {
                printf("av_write_frame failed\n");
            }
        }
    }
```

```
149.
150.     return 0;
151. }
152.
153. AVCodecContext* createCodecContext(AVFormatContext *oc)
154. {
155.     AVCodecContext *video_cc = avcodec_alloc_context();
156.
157.     video_cc =  avcodec_alloc_context();
158.     if (!video_cc)
159.     {
160.         fprintf(stderr, "alloc avcodec context failed\n");
161.         exit(1);
162.     }
163.
164.     video_cc->codec_id = (CodecID)CODEC_ID_H264;
165.     video_cc->codec_type = AVMEDIA_TYPE_VIDEO;
166.
167.     video_cc->me_range = 16;
168.     video_cc->max_qdiff = 4;
169.     video_cc->qmin = 10;
170.     video_cc->qmax = 51;
171.     video_cc->qcompress = 0.6f;
172.
173.     /* put sample parameters */
174.     video_cc->bit_rate = 400000;
175.
176.     /* resolution must be a multiple of two */
177.     video_cc->width = image_width;
178.     video_cc->height = image_height;
179.
180.     /* time base: this is the fundamental unit of time (in seconds) in terms
181.     of which frame timestamps are represented. for fixed-fps content,
182.     timebase should be 1/framerate and timestamp increments should be
183.     identically 1. */
184.     video_cc->time_base.den = frame_rate;
185.     video_cc->time_base.num = 1;
186.
187.     video_cc->gop_size = 12; /* emit one intra frame every twelve frames at most */
188.     video_cc->pix_fmt = PIX_FMT_YUV420P;
189.
190.     // some formats want stream headers to be separate
191.     if(!strcmp(oc->oformat->name, "mp4") || !strcmp(oc->oformat-
>name, "mov") || !strcmp(oc->oformat->name, "3gp"))
192.         video_cc->flags |= CODEC_FLAG_GLOBAL_HEADER;
193.
194.     return video_cc;
195. }
196.
197. void openVideo(AVFormatContext *oc)
198. {
199.     AVCodec *codec;
200.
201.     /* find the video encoder */
202.     codec = avcodec_find_encoder(codecContext->codec_id);
203.     if (!codec) {
204.         fprintf(stderr, "codec not found\n");
205.         exit(1);
206.     }
207.
208.     /* open the codec */
209.     if (avcodec_open(codecContext, codec) < 0) {
210.         fprintf(stderr, "could not open video codec\n");
211.         exit(1);
212.     }
213. }
214.
215. void ffmpegEncodeInit()
216. {
217.     // initialize libavcodec, and register all codecs and formats
218.     av_register_all();
219.
220.     char filename[] = "test.mp4";
221.     fmt = av_guess_format(NULL, filename, NULL);
222.
223.     oc = avformat_alloc_context();
224.     oc->oformat = fmt;
225.
226.     fmt->video_codec = (CodecID) CODEC_ID_H264;
```

```
227.     _snprintf(oc->filename, sizeof(oc->filename), "%s", filename);
228. 
229.     // add the video streams using the default format codecs and initialize the codecs
230.     video_st = NULL;
231.     if (fmt->video_codec != CODEC_ID_NONE) {
232.         video_st = av_new_stream(oc, 0);
233.         if (!video_st) {
234.             fprintf(stderr, "Could not alloc stream\n");
235.             exit(1);
236.         }
237.     }
238. 
239.     // alloc codecContext
240.     codecContext = createCodecContext(oc);
241.     video_st->codec = codecContext;
242. 
243.     // set the output parameters (must be done even if no parameters).
244.     if (av_set_parameters(oc, NULL) < 0) {
245.         fprintf(stderr, "Invalid output format parameters\n");
246.         exit(1);
247.     }
248. 
249.     dump_format(oc, 0, filename, 1);
250. 
251.     /* now that all the parameters are set, we can open the audio and
252.     video codecs and allocate the necessary encode buffers */
253.     if (codecContext)
254.         openVideo(oc);
255. 
256.     // open the output file, if needed
257.     if (!(fmt->flags & AVFMT_NOFILE)) {
258.         if (url_fopen(&oc->pb, filename, URL_WRONLY) < 0) {
259.             fprintf(stderr, "Could not open '%s'\n", filename);
260.             exit(1);
261.         }
262.     }
263. 
264.     // write the stream header, if any
265.     av_write_header(oc);
266. }
267. 
268. void ffmpegEncodeClose()
269. {
270.     int i;
271. 
272.     /* close each codec */
273.     if (video_st)
274.         avcodec_close(video_st->codec);
275. 
276.     // write the trailer, if any
277.     av_write_trailer(oc);
278. 
279.     /* free the streams */
280.     for(i = 0; i < oc->nb_streams; i++) {
281.         av_freep(&oc->streams[i]->codec);
282.         av_freep(&oc->streams[i]);
283.     }
284. 
285.     if (!(fmt->flags & AVFMT_NOFILE)) {
286.         /* close the output file */
287.         url_fclose(oc->pb);
288.     }
289. 
290.     /* free the stream */
291.     av_free(oc);
292. }
293. 
294. int main()
295. {
296.     recv_buf = (uint8_t *)malloc(recv_bufsize);
297. 
298.     rtpInit();
299.     ffmpegEncodeInit();
300. 
301.     // =============== INSTALL THE CONTROL HANDLER ===============
302.     if (SetConsoleCtrlHandler( (PHANDLER_ROUTINE) ctrlHandlerFunction, TRUE) == 0)
303.     {
304.         printf("==> Cannot handle the CTRL-C...\n");
305.     }
```

```
306.
307.     printf("==> RTP Receiver started\n");
308.
309.     while (m_bExit == FALSE)
310.     {
311.         rtp2disk();
312.
313.         rtp_session_mgr.timestamp += timestamp_inc;
314.     }
315.
316.     printf("==> Exiting\n");
317.
318.     free(recv_buf);
319.
320.     ffmpegEncodeClose();
321.
322.     rtp_session_destroy(rtp_session_mgr.rtp_session);
323.     ortp_exit();
324. }
```

sender

```
[cpp]
01. #include <ortp/ortp.h>
02. #include <string.h>
03.
04. extern "C"{
05. #include <libavformat/avformat.h>
06. #include <libswscale/swscale.h>
07. };
08.
09. struct RtpSessionMgr
10. {
11.     RtpSession *rtp_session;
12.     uint32_t timestamp_inc;
13.     uint32_t cur_timestamp;
14. };
15.
16. RtpSessionMgr rtp_session_mgr;
17.
18. const char g_ip[] = "127.0.0.1";
19. const int g_port = 8008;
20. const uint32_t timestamp_inc = 3600; // 90000 / 25
21.
22. const int image_width = 704;
23. const int image_height = 576;
24. const int frame_rate = 25;
25. static int frame_count, wrap_size;
26.
27. AVCodecContext *video_cc;
28.
29. AVFrame *picture;
30.
31. /** 帧包头的标识长度 */
32. #define CMD_HEADER_LEN 10
33.
34. /** 帧包头的定义 */
35. static uint8_t CMD_HEADER_STR[CMD_HEADER_LEN] = { 0xAA,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7
36.
37. /** 帧的包头信息 */
38. typedef struct _sFrameHeader
39. {
40.     /** 命令名称标识 */
41.     unsigned char cmdHeader[CMD_HEADER_LEN];
42.
43.     /** 采集的通道号 0~7*/
44.     unsigned char chId;
45.
46.     /** 数据类型, 音频 或者 视频*/
47.     unsigned char dataType;
48.
49.     /** 缓冲区的数据长度 */
50.     uint32_t len;
51.
52.     /** 时间戳 */
```

```
    uint32_t timestamp;

}FrameHeader;

// set frame header
FrameHeader frameHeader;

void rtpInit()
{
    char *m_SSRC;

    ortp_init();
    ortp_scheduler_init();
    printf("Scheduler initialized\n");

    rtp_session_mgr.rtp_session = rtp_session_new(RTP_SESSION_SENDONLY);

    rtp_session_set_scheduling_mode(rtp_session_mgr.rtp_session, 1);
    rtp_session_set_blocking_mode(rtp_session_mgr.rtp_session, 1);
    rtp_session_set_remote_addr(rtp_session_mgr.rtp_session, g_ip, g_port);
    rtp_session_set_send_payload_type(rtp_session_mgr.rtp_session, 34); // 34 is for H.26

    m_SSRC  = getenv("SSRC");
    if (m_SSRC != NULL)
    {
        rtp_session_set_ssrc(rtp_session_mgr.rtp_session, atoi(m_SSRC));
    }

    rtp_session_mgr.cur_timestamp = 0;
    rtp_session_mgr.timestamp_inc = timestamp_inc;

    printf("rtp init success!\n");
}

int rtpSend(unsigned char *send_buffer, int frame_len)
{
    FrameHeader *fHeader = (FrameHeader *)send_buffer;
    fHeader->chId      = 0;
    fHeader->dataType  = 0; // SESSION_TYPE_VIDEO
    fHeader->len       = frame_len;
    fHeader->timestamp = 0;

    printf("frame header len = %d\n", fHeader->len);

    int wrapLen;
    wrapLen = frame_len + sizeof(FrameHeader);

    int sended_bytes;
    sended_bytes = rtp_session_send_with_ts(rtp_session_mgr.rtp_session,
                                            (uint8_t *)send_buffer,
                                            wrapLen,
                                            rtp_session_mgr.cur_timestamp);

    rtp_session_mgr.cur_timestamp += rtp_session_mgr.timestamp_inc;

    return sended_bytes;
}

void createCodecContext()
{
    video_cc =  avcodec_alloc_context();
    if (!video_cc)
    {
        fprintf(stderr, "alloc avcodec context failed\n");
        exit(1);
    }

    video_cc->codec_id = (CodecID)CODEC_ID_H264;
    video_cc->codec_type = AVMEDIA_TYPE_VIDEO;

    video_cc->me_range = 16;
    video_cc->max_qdiff = 4;
    video_cc->qmin = 10;
    video_cc->qmax = 51;
    video_cc->qcompress = 0.6f;

    /* put sample parameters */
    video_cc->bit_rate = 400000;

```

```
    /* resolution must be a multiple of two */
    video_cc->width = image_width;
    video_cc->height = image_height;

    /* time base: this is the fundamental unit of time (in seconds) in terms
    of which frame timestamps are represented. for fixed-fps content,
    timebase should be 1/framerate and timestamp increments should be
    identically 1. */
    video_cc->time_base.den = frame_rate;
    video_cc->time_base.num = 1;

    video_cc->gop_size = 12; /* emit one intra frame every twelve frames at most */
    video_cc->pix_fmt = PIX_FMT_YUV420P;
}

AVFrame *allocPicture(int pix_fmt, int width, int height)
{
    AVFrame *picture;
    uint8_t *picture_buf;
    int size;

    picture = avcodec_alloc_frame();
    if (!picture)
        return NULL;

    size = avpicture_get_size((PixelFormat)pix_fmt, width, height);
    picture_buf = (uint8_t *)av_malloc(size);
    if (!picture_buf) {
        av_free(picture);
        return NULL;
    }
    avpicture_fill((AVPicture *)picture, picture_buf, (PixelFormat)pix_fmt, width, height
    return picture;
}

void openVideo()
{
    AVCodec *video_codec;

    /* find the video encoder */
    video_codec = avcodec_find_encoder(video_cc->codec_id);
    if (!video_codec) {
        fprintf(stderr, "codec not found\n");
        exit(1);
    }

    /* open the codec */
    if (avcodec_open(video_cc, video_codec) < 0) {
        fprintf(stderr, "could not open video codec\n");
        exit(1);
    }

    /* allocate the encoded raw picture */
    picture = allocPicture(video_cc->pix_fmt, video_cc->width, video_cc->height);
    if (!picture) {
        fprintf(stderr, "Could not allocate picture\n");
        exit(1);
    }
}

/* prepare a dummy image */
void fill_yuv_image(AVFrame *pict, int frame_index, int width, int height)
{
    int x, y, i;

    i = frame_index;

    /* Y */
    for(y=0;y<height;y++) {
        for(x=0;x<width;x++) {
            pict->data[0][y * pict->linesize[0] + x] = x + y + i * 3;
        }
    }

    /* Cb and Cr */
    for(y=0;y<height/2;y++) {
        for(x=0;x<width/2;x++) {
            pict->data[1][y * pict->linesize[1] + x] = 128 + y + i * 2;
            pict->data[2][y * pict->linesize[2] + x] = 64 + x + i * 5;
```

```
        }
    }
}

void ffmpegInit()
{
    // initialize libavcodec, and register all codecs and formats
    av_register_all();

    // create a codec context
    createCodecContext();

    // open H.264 codec
    openVideo();
}

void getEncodedFrame(unsigned char *buffer, int& len)
{
    int out_size;

    fill_yuv_image(picture, frame_count, video_cc->width, video_cc->height);

    // encode the frame
    out_size = avcodec_encode_video(video_cc, buffer, wrap_size-
sizeof(FrameHeader), picture);

    len = out_size;
    frame_count++;
}

int main()
{
    unsigned char *send_outbuf;
    unsigned char *video_part;

    frame_count = 0;
    wrap_size = 20000;
    send_outbuf = (unsigned char *)malloc(wrap_size);

    // copy cmdHeader to frameInfo
    memcpy(frameHeader.cmdHeader,CMD_HEADER_STR,CMD_HEADER_LEN);

    memcpy(send_outbuf, &frameHeader, sizeof(FrameHeader));
    video_part = send_outbuf + sizeof(FrameHeader);

    ffmpegInit();
    rtpInit();

    while (1)
    {
        int frame_len;
        // get encode frame
        getEncodedFrame(video_part, frame_len);

        printf("encodecFrame length is : %d\n", frame_len);

        if (frame_len > 0)
        {
            rtpSend(send_outbuf, frame_len);
        }
    }

    rtp_session_destroy(rtp_session_mgr.rtp_session);

    free(send_outbuf);

    // Give us some time
    Sleep(250);

    ortp_exit();
}
```

更多 0

上一篇 Introduction to Sound Programming with ALSA(使用alsa API接口编程)

下一篇 编程之美--2.20(程序理解和时间分析)

查看评论

5楼 mengfanxinqq 2013-08-29 09:27发表

你好，这个QOSAddSocketToFlow failed to add a flow with error 87 错误您是怎么解决的？我现在也遇到了这个错误。。

4楼 xi52qian 2013-08-07 11:09发表

34是H263，怎么可能会发送成功？网上还多都写34.真不知道发贴的人是否测试过？oRtp里定义了H264的PayLoad但是没有添加到ProFile里，要自己在外部添加，而34在ORtp库中已经定义是34了！

3楼 简单的逻辑 2013-06-07 11:42发表

在 VS2008 下编译报错呢

```
[cpp]
01. 错误  1   fatal error C1083: 无法打开包括文
    件:“inttypes.h”: No such file or directory d:\ortp_test\ffmpeg-0.8-win32-
    dev\include\libavutil\common.h    31  video_recver
```

在网上下载了 inttypes.h 加入后，又有其他的问题，怎么解决呢？

Re: ajaxhe 2013-06-08 10:48发表

回复baiynui1983：谢谢反馈，打包时在ortp_test\ffmpeg-0.8-win32-dev\include目录中忘记添加inttypes.h和stdint.h头文件。已经修改，并重新上传了代码。

Re: 简单的逻辑 2013-06-14 14:05发表

回复ajaxhe：是这个问题，我在网上下载这两个文件，也解决了问题。楼主真认真！

Re: 简单的逻辑 2013-06-14 14:06发表

回复baiynui1983：我最近在搞移动视频监控这方面的项目，并且以后也想网这方面发展，希望能向你学习，我的 QQ：99951468

Re: ajaxhe 2013-06-14 14:57发表

回复baiynui1983：多交流，共同进步

2楼 kangear 2013-04-03 11:34发表

牛，学习学习。

1楼 江南烟雨 2012-09-07 08:53发表

顶一顶~嘿嘿~~

发表评论

用 户 名： wzw0114

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Java VPN Android iOS ERP IE10 Eclipse CRM JavaScript Ubuntu NFC WAP jQuery 数据库 BI HTML5 Spring Apache Hadoop .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP Spark HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap