04-SRS流媒体服务器-RTMP拉流框架分析

核心类

SrsServer SRS流媒体服务入口
SrsBufferListener 监听器，主要是TCP的监听
SrsTcpListener TCP监听器
SrsRtmpConn RTMP连接，里面对应了SrsStSocket和SrsCoroutine
SrsRtmpServer 提供与客户端之间的RTMP-命令-协议-消息的交互服务，使用SrsRtmpConn 提供的socket读写数据
SrsSource 描述一路播放源，包括推流和拉流的描述
SrsConsumer 拉流消费者，每一路拉流客户端对应一个SrsConsumer
SrsStSocket 经过封装的socket接口

SrsRecvThread 负责接收数据，但是要注意的是他这里并不是从IO里面读取数据

- 从SrsRtmpServer类拉取数据，然后推送到SrsPublishRecvThread（推流用），或者SrsQueueRecvThread（拉流用）

SrsQueueRecvThread 主要用于拉流，对应的是客户端-服务器的控制消息，和音视频消息没有关系。客户端读取数据还是从consumer的queue里面去读取。
SrsPublishRecvThread 主要用于推流

测试客户端

在客户端进行推流验证
ffmpeg -re -i rtmp_test_hd.flv -vcodec copy -acodec copy -f flv -y rtmp://111.229.231.225/live/livestream

在客户端拉流验证
ffplay rtmp://111.229.231.225/live/livestream

重点难点

不同协程的意义

打断点

客户端和服务器直接的交互，非音视频数据
断点：b SrsRtmpConn::**process_play_control_msg**(SrsConsumer*, SrsCommonMessage*)

打印：print *msg

$3 = {_vptr.SrsCommonMessage = 0x6b79a8 <vtable for SrsCommonMessage+16>, header = {
 _vptr.SrsMessageHeader = 0x6b79d8 <vtable for SrsMessageHeader+16>, timestamp_delta = 1, payload_length = 10,
 message_type = 4 '\004', stream_id = 0, timestamp = 1, perfer_cid = 2}, size = 10, payload = 0xa3aa80 ""}

$4 = {_vptr.SrsCommonMessage = 0x6b79a8 <vtable for SrsCommonMessage+16>, header = {
 _vptr.SrsMessageHeader = 0x6b79d8 <vtable for SrsMessageHeader+16>, timestamp_delta = 9130, payload_length = 4,
 message_type = 3 '\003', stream_id = 0, timestamp = 9131, perfer_cid = 2}, size = 4, payload = 0xa74580 ""}

$5 = {_vptr.SrsCommonMessage = 0x6b79a8 <vtable for SrsCommonMessage+16>, header = {
 _vptr.SrsMessageHeader = 0x6b79d8 <vtable for SrsMessageHeader+16>, timestamp_delta = 9280, payload_length = 4,
 message_type = 3 '\003', stream_id = 0, timestamp = 30731, perfer_cid = 2}, size = 4, payload = 0x10325f0 ""}

以ffmpeg为例

```
/**
 * known RTMP packet types
 */
typedef enum RTMPPacketType {
    RTMP_PT_CHUNK_SIZE   =  1,  ///< chunk size change
    RTMP_PT_BYTES_READ   =  3,  ///< 3 number of bytes read
    RTMP_PT_USER_CONTROL,       ///< 4 user control
    RTMP_PT_WINDOW_ACK_SIZE,    ///< window acknowledgement size
    RTMP_PT_SET_PEER_BW,        ///< peer bandwidth
    RTMP_PT_AUDIO        =  8,  ///< audio packet
    RTMP_PT_VIDEO,              ///< video packet
    RTMP_PT_FLEX_STREAM  = 15,  ///< Flex shared stream
    RTMP_PT_FLEX_OBJECT,        ///< Flex shared object
    RTMP_PT_FLEX_MESSAGE,       ///< Flex shared message
    RTMP_PT_NOTIFY,             ///< some notification
    RTMP_PT_SHARED_OBJ,         ///< shared object
    RTMP_PT_INVOKE,             ///< invoke some stream action
    RTMP_PT_METADATA     = 22,  ///< FLV metadata
} RTMPPacketType;
```

客户端读取的包大于> receive_report_size时，回复**RTMP_PT_BYTES_READ**

receive_report_size 来自 **RTMP_PT_WINDOW_ACK_SIZE** 消息ID

```c
rt->bytes_read += ret;
if (rt->bytes_read - rt->last_bytes_read > rt->receive_report_size) {
    av_log(s, AV_LOG_DEBUG, "Sending bytes read report\n");
    if ((ret = gen_bytes_read(s, rt, rpkt.timestamp + 1)) < 0) {
        ff_rtmp_packet_destroy(&rpkt);
        return ret;
    }
    rt->last_bytes_read = rt->bytes_read;
}
```

ffmpeg对于rtmp这块的注释还算详细。