快直播 LEB MediaTransport SDK 1.0.2 接入文档

1. SDK 简介

快直播（Live Event Broadingcasting）是腾讯云基于 WebRTC 技术的超低延时直播，通过 MediaTransport SDK 可以方便地接入基于 Webrtc 的多媒体传输协议，直接获取音视频 RTP 数据流。

2. SDK 功能及组成

2.1 功能

- 支持加密的开和关
- 支持 AAC 和 OPUS 拉流
- 支持音频 FlexFec
- 支持 UDP 和 HTTP 信令

2.2 数据输出

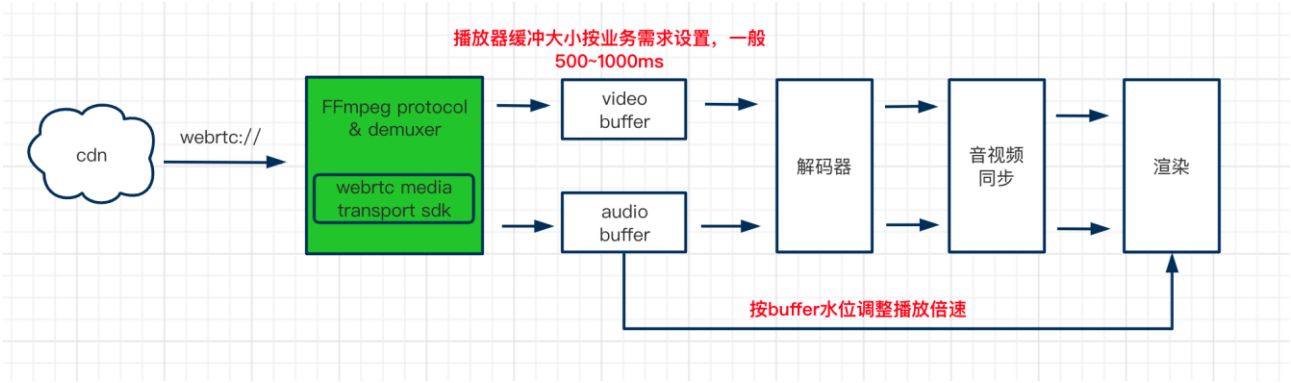
- 视频，RTP 输出并附带 pts，接入侧只需将 RTP 数据组成一帧即可送给视频解码器
- 音频，RTP 输出并附带 pts，每个 RTP 包是一帧音频数据(aac 格式为 adts)，可直接送给音频解码器

2.3 组成

- include: sdk 头文件
- libs: 包括 android 动态库，ios 静态库
- docs: 接入文档，包括 HTTP 信令接入文档(UDP 信令 SDK 内部处理)和 SDK 接入文档
- example: lebnetwork 封装示例和 demo

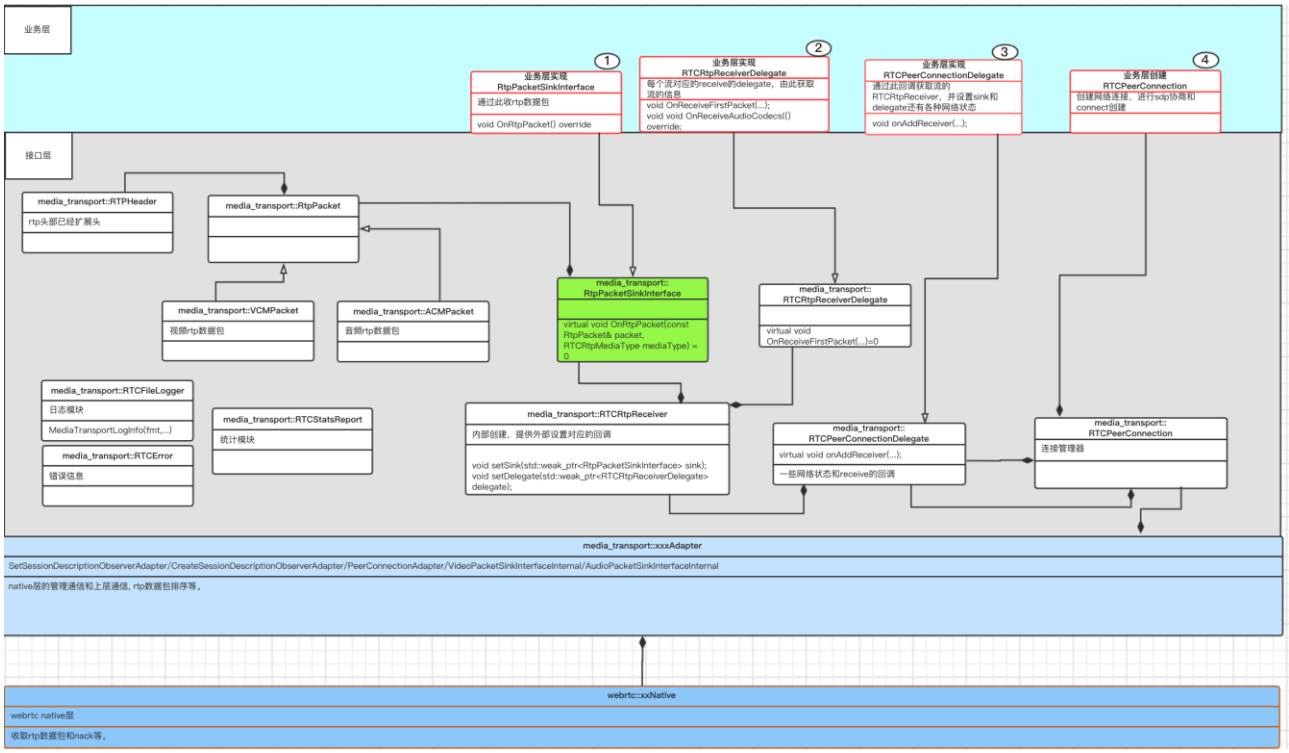
3. SDK 接入

下图为普通播放器接入 MediaTransport SDK 的示意图，如播放器不依赖 ffmpeg，可直接实现为 demuxer。具体请配合 example lebnetwork 封装示例了解 sdk 接入实现



3.1 实现接口

- 接口框架



- RTP 收包回调接口，见 include/rtp_packet_sink_interface.h，demo 中 dump 音视频 RTP 数据进行验证的代码示例

```
class RtpPacketSinkInterface {
```

```
 public:
  virtual ~RtpPacketSinkInterface() = default;
   /** RTP 收包回调 */
  virtual void OnRtpPacket(const RtpPacket& packet, RTCRtpMediaType mediaType) = 0;
};
```

- RTP Receiver 回调接口，见 include/rtc_receiver.h

```
class RTCRtpReceiverDelegate {
 public:
  virtual ~RTCRtpReceiverDelegate() = default;
 /** 收到音视频第一个 RTP 包回调 */
  virtual void OnReceiveFirstPacket(RTCRtpMediaType mediaType) = 0;
 /** setRemoteSdp 后音频解码器信息的回调   */
  virtual void OnReceiveAudioCodecs(const std::map<int,
                                   media_transport::SdpAudioFormat>& codecs) = 0;
 /** setRemoteSdp 后视频解码器信息的回调 */
  virtual void OnReceiveVideoCodecs(std::map<uint8_t, media_transport::VideoCodecType>
&payload_type, std::map<uint8_t, std::map<std::string, std::string> > &pt_codec_params) = 0;
};
```

- Peer Connection 回调接口，见 include/peer_connection.h

```
class RTCPeerConnectionDelegate {
public:
  /** Called when the SignalingState changed. */
  virtual void onChangeSignalingState(RTCPeerConnection * peerConnection,
                                      RTCSignalingState stateChanged){}

  /** Called any time the IceConnectionState changes. */
  virtual void onChangeIceConnectionState(RTCPeerConnection * peerConnection,
                                          RTCIceConnectionState stateChanged){}

  /** Called any time the IceGatheringState changes. */
  virtual void onChangeIceGatheringState(RTCPeerConnection * peerConnection,
                                         RTCIceGatheringState stateChanged){}

  /** Called any time the PeerConnectionState changes. */
  virtual void onChangeConnectionState(RTCPeerConnection * peerConnection,
                                       RTCPeerConnectionState stateChanged){}

  /** Called when a receiver and its media streams are created. */
  virtual void onAddReceiver(RTCPeerConnection * peerConnection,
                             std::shared_ptr<RTCRtpReceiver> receiver) = 0;

  /** Called when getStats() required */
  virtual void onStatsReport(RTCStatsReport &stats) = 0;

  /** Called when error happend */
  virtual void onError(RTCPeerConnection * peerConnection,
                       const RTCError *error) = 0;
```

```
  virtual ~RTCPeerConnectionDelegate() = default;
};
```

具体请配合 example lebnetwork 封装示例了解 sdk 封装实现

4. lebnetwork demo 流程

example/leb_network_test/leb_connection_main.cc

```
//拉流地址
const char* streamUrl = "webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1";
int main(int argc, char* argv[]) {
  printf("hello lebconnection demo!!, %d\n", argc);
  // signal(SIGSEGV, handler);
  if (argc != 2) {
    printf("usage: ./demo_bin streamurl, use default url\n");
  } else {
    streamUrl = argv[1];
    printf("input streamUrl %s\n", streamUrl);
  }
  /**
   * 1. 创建 lebconnection
   */
  std::unique_ptr<liteav::lebnetwork::LEBConnectionTest> leb_connection_test;
  leb_connection_test.reset(new liteav::lebnetwork::LEBConnectionTest("/sdcard/"));

  /**
   * 2. 配置拉流选项
   */
  liteav::lebnetwork::LEBConnection::Configuration config;
  config.stream_url = streamUrl;
  config.receive_audio = true;
  config.receive_video = true;
  config.enable_encryption = false;
  config.enable_aac = true;
  config.enable_flex_fec = true;

  /**
   * 3. 开始拉流
   */
  leb_connection_test->Start(config);

  getchar();
  leb_connection_test->GetStats();
  getchar();

  /**
   * 4. 关闭 lebconnection
```

```
   */
  leb_connection_test->Stop();
  leb_connection_test.reset();

  printf("bye lebconnection demo!!");
  return 0;
}
```