**Event: H264 encoding begins**

1. RTSP server is initialized with sps and pps parameters generated after encoding.
2. An array of connections is initialized with capacity 10
3. A TCP socket object named ‘listener’ is created and the address is set to INADDR\_ANY to bind it to all the interfaces.
4. The above socket acts as a listener for incoming connections. The port is set to 554
5. A new run loop source is created on the socket and is attached to the main run loop

**Event: A new connection is accepted**

1. RTSPClientConnection object is initialized and is added to the connections array.
2. A CFSocket is created using already created listener socket, a run loop source is attached to it and the source is added to the main run loop.
3. Callback type for this socket is kSocketDataCallback and the callback function is named onSocket
4. Multiple messages are received on this socket from the corresponding RTSP client. The messages in order are as follows:
5. **Request(Options)**

OPTIONS rtsp://192.168.43.81/ RTSP/1.0

CSeq: 2

User-Agent: LibVLC/2.2.4 (LIVE555 Streaming Media v2016.02.22)

**Response**:

RTSP/1.0 200 OK

CSeq: 2

Server: AVEncoderDemo/1.0

Public: DESCRIBE, SETUP, TEARDOWN, PLAY, OPTIONS

1. **Request(Describe)**

DESCRIBE rtsp://192.168.43.81/ RTSP/1.0

CSeq: 3

User-Agent: LibVLC/2.2.4 (LIVE555 Streaming Media v2016.02.22)

Accept: application/sdp

**Response(SDP format)**

RTSP/1.0 200 OK

CSeq: 3

Content-base: rtsp://192.168.43.81/

Date: December 15, 2017 at 11:03:31 AM MST

Content-Type: application/sdp

Content-Length: 421

v=0

o=- 1804289383 1804289383 IN IP4 192.168.43.81

s=Live stream from iOS

c=IN IP4 0.0.0.0

t=0 0

a=control:\*

m=video 0 RTP/AVP 96

b=TIAS:4512

a=maxprate:1.0000

a=control:streamid=1

a=rtpmap:96 H264/90000

a=mimetype:string;"video/H264"

a=framesize:96 720-480

a=Width:integer;720

a=Height:integer;480

a=fmtp:96 packetization-mode=1;profile-level-id=64001e;sprop-parameter-sets=J2QAHqxWwLQ9pqAgICBA,KO48sA==

1. **Request(Setup)**

SETUP rtsp://192.168.43.81/streamid=1 RTSP/1.0

CSeq: 4

User-Agent: LibVLC/2.2.4 (LIVE555 Streaming Media v2016.02.22)

Transport: RTP/AVP;unicast;**client\_port=58176-58177**

\*\*Port numbers for RTP and RTCP sockets at the client end are extracted from this\*\*

\*\*This phase is further explained below\*\*

**Response**

RTSP/1.0 200 OK

CSeq: 4

Session: 1714636915

Transport: RTP/AVP;unicast;client\_port=58176-58177;**server\_port=6970-6971**

**\*\***Server ports are mentioned in this response\*\*

1. **Request(Play)**

PLAY rtsp://192.168.43.81/ RTSP/1.0

CSeq: 5

User-Agent: LibVLC/2.2.4 (LIVE555 Streaming Media v2016.02.22)

Session: 1714636915

Range: npt=0.000-

**Response**

RTSP/1.0 200 OK

CSeq: 5

Session: 1714636915

**Setup phase**

1. In the setup phase, the client sends two ports numbers, one for RTP and one for RTCP respectively.
2. A new RTP session is created as follows:

* Two sockets are created, one for each RTP and RTCP. The corresponding port numbers are assigned
* The client address is copied from the peer address given by the already connected RTSP socket
* Another socket object is created for receiving RTCP messages and a run loop source is attached to it
* Session id is given a random value and SSRC is also set to a random value

**Play phase**

1. The maximum packet size is 1200 bytes
2. If the NAL unit size is less that 1200 bytes, it attaches the RTP header and sends it.
3. The header that header that it attaches is described below:

* V: 2
* P: 0
* X: 0
* CC: 0
* M: 1 if this packet is the last packet of 1 large frame; else 0
* Payload type: 96 (for H264)
* Sequence number: Packet number(starts from 0)
* Timestamp
* SSRC: Synchronization source identifier randomly generated during session creation

1. If the NAL unit size is more than 1200 bytes, then fragmentation is used. The size of the fragmented packets in this implementation is 1186 bytes

* A fragmented unit type called FU type and header called FU header is attached to each fragmented unit and the packet is sent.
* Details about fragmentation can be found in the following link: [https://tools.ietf.org/html/rfc3984 - section-5.8](https://tools.ietf.org/html/rfc3984#section-5.8)

**\*\*I did a small experiment to find out if the delay is caused by fragmentation or not. I increased the maximum single packet size to 60000 bytes so that no fragmentation takes place. But, the delay was same and it resulted into corruption of video on the receiving end.\*\***