

RDK-C Mobile Application

Introduction:

RDK-C Mobile application will be used to watch live streaming from RPI Camera & Previous recordings from RPI Camera stored in Cloud Servers. In this mobile application we use **WebRTC** protocol for live streaming of video & audio, **HTTP** to play or download the previous recordings from cloud servers.

This mobile app will support android & iOS* platforms. This page is dedicated to bring up the overall picture of the mobile app and how it works.

About WebRTC:

WebRTC, short for Web Real-Time Communication, is both an **API** and a **Protocol**. The WebRTC protocol is a set of rules for two WebRTC agents to negotiate bi-directional secure real-time communication. A similar relationship would be the one between HTTP and the Fetch API. WebRTC the protocol would be HTTP, and WebRTC the API would be the Fetch API.

To establish WebRTC communication one should involve the below 4 steps:

1. Signalling
2. Connecting
3. Securing
4. Communicating

These four steps happen sequentially. The prior step must be 100% successful for the subsequent one to even begin. Each of these steps has dedicated chapters, but it is helpful to understand them at a high level first. Since they depend on each other, it will help when explaining further the purpose of each of these steps.

• Signalling:

When a WebRTC Agent starts it has no idea who it is going to communicate with and what they are going to communicate about. Signalling solves this issue! Signalling is used to bootstrap the call so that two WebRTC agents can start communicating. Signaling uses an existing protocol SDP (Session Description Protocol). SDP is a plain-text protocol. Each SDP message is made up of key/value pairs and contains a list of "media sections".

• Connecting:

The two WebRTC Agents now know enough details to attempt to connect to each other. WebRTC then uses another established technology called ICE. ICE (Interactive Connectivity Establishment) is a protocol that pre-dates WebRTC. ICE allows the establishment of a connection between two Agents. These Agents could be on the same network, or on the other side of the world. ICE is the solution to establishing a direct connection without a central server. The real magic here is 'NAT Traversal' and STUN/TURN Servers. These two concepts are all you need to communicate with an ICE Agent in another subnet. Once ICE successfully connects, WebRTC then moves on to establishing an encrypted transport. This transport is used for audio, video, and data.

• Securing:

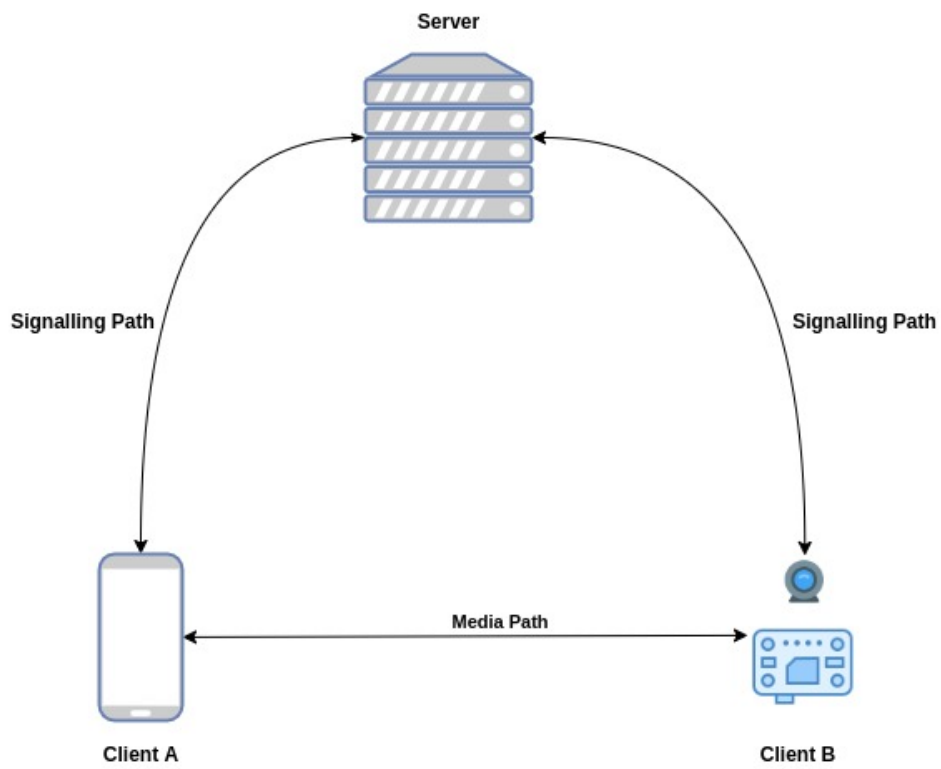
Now that we have bi-directional communication (via ICE) we need to establish secure communication. This is done through two protocols that pre-date **WebRTC**. The first protocol is **DTLS** (Datagram Transport Layer Security) which is just TLS over UDP. TLS is the cryptographic protocol used to secure communication over **HTTPS**. The second protocol is **SRTP** (Secure Real-time Transport Protocol). First, WebRTC connects by doing a DTLS handshake over the connection established by ICE. Unlike HTTPS, WebRTC doesn't use a central authority for certificates. WebRTC then uses a different protocol for audio/video transmission called RTP. We secure our RTP packets using SRTP. We initialize our SRTP session by extracting the keys from the negotiated DTLS session.

• Communicating:

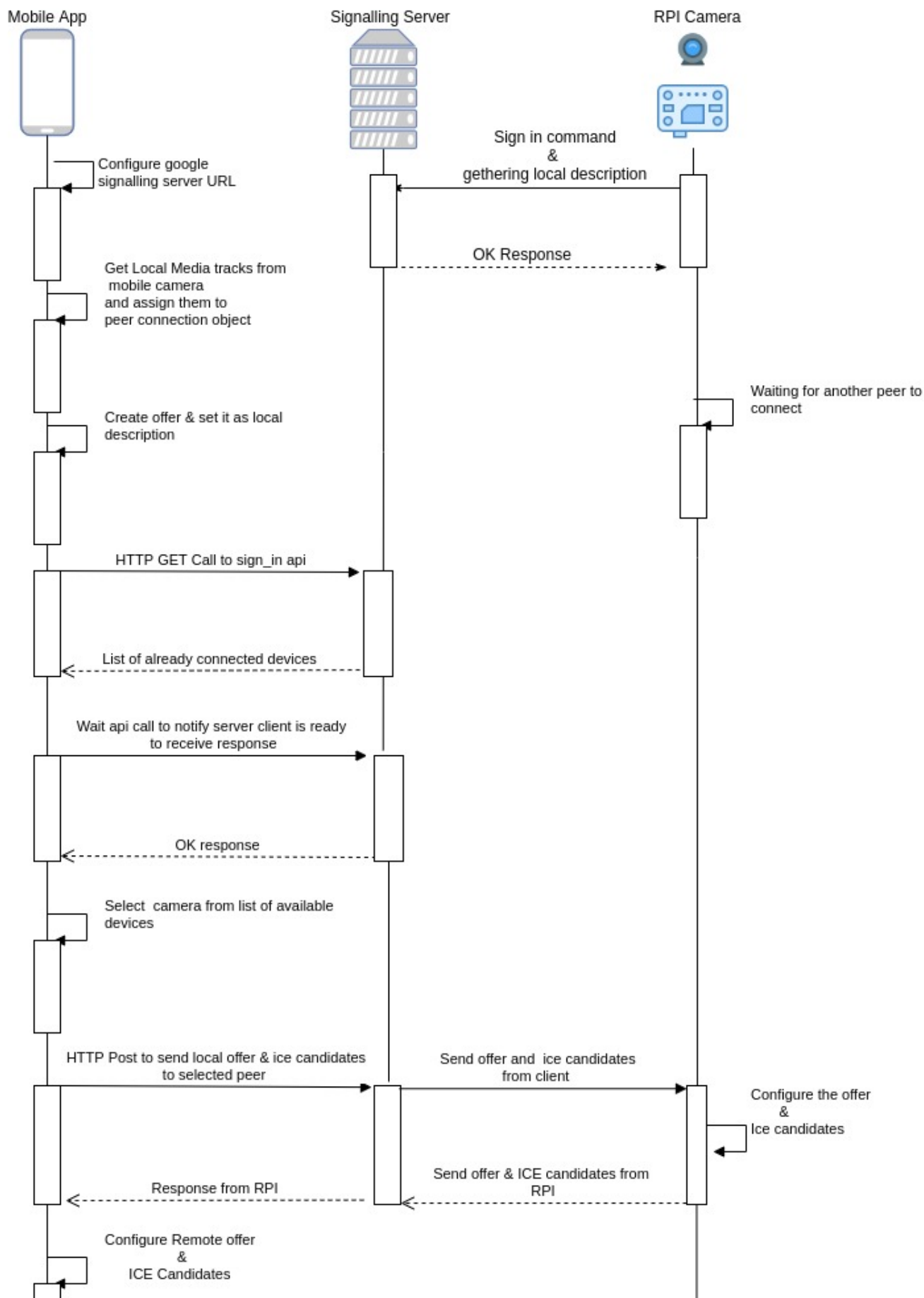
We now have two WebRTC Agents with secure bi-directional communication. we use two pre-existing protocols: RTP (Real-time Transport Protocol), and SCTP (Stream Control Transmission Protocol). Use RTP to exchange media encrypted with SRTP, and use SCTP to send and receive DataChannel messages encrypted with DTLS.

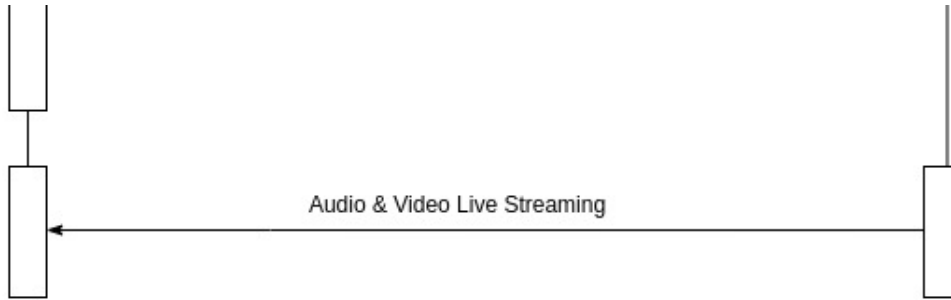
Now we are done! You now have bi-directional and secure communication. If you have a stable connection between your WebRTC Agents, this is all the complexity you may need.

Application Architecture (Live streaming)



Application Flow (Live streaming)





How to Connect:

To view live streaming video from rpi-camera in your mobile you need to follow the below steps

Step 1: Please download the [app-release.apk](#) and install in your android device

Step 2: Please follow the steps from this [URL](#) to start R-PI camera client and peer connection server

Step 3: After successful completion of step 2, Navigate to **Open_WebRTC/openwebrtc-m72/out/Default** and **./peerconnection_server** hit Enter to start the server

Step 4: SSH into R-PI and enter "**./peerconnection_client**" which asks for the server IP & Enter Server IP (Your machine IP where peerconnection_server is running) and port number as **8888**.

Console

```
root@raspberrypi3-rdk-camera:~# ./peerconnection_client
Inside constructor CustomSocketServer().....
Initialized thread...
"Registering PeerConnectionClient::RegisterObserver()"
Enter server IP Address
```

Step 5: Open your mobile app and click on scan in live-feed page to get the list of cameras in your network.

Step 6: Select the camera you want to watch the live feed.

Note: Make sure your R-PI board and mobile app should be in the same network.

Mobile App Features

Following are the mobile application features.

Sign-up: To register new user

Sign-in: To enter into the mobile app with valid user credentials.

Dashboard: In this page we are having three sections

1. **Live Feed** - To get the live footage from the list of selected cameras.
2. **Recordings** - To get the list of previously recorded footage from cloud.
 - a. Today - List of current date recordings
 - b. Last week - List of recordings up to 1 week
 - c. Custom - List of recordings in b/w selected date range
3. **Settings**
 - a. Camera - Resolution, IP, Storage
 - b. User - Add camera, Permissions, Details
 - c. Other - Terms & Conditions, Privacy Policy, About, Rating, Logout

Mobile App Screens

Sign-In: User should enter valid email and password to sign in into the application. Click register to register new user.

Login

Email

Password

SIGN IN

New user

SIGN UP

Sign-up: User should enter valid details then click sign up to register into the application.



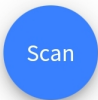
Register

SIGN UP

Live Feed: Populates list of available cameras in your network which are connected to server and ready to stream the content

Note: To view the list of available cameras in the network user should configure the Server IP and Port in **Settings tab > Server IP option**

Livefeed



Click to scan & view the list of available cameras in your network



IP Settings

Server IP *

192.16

Invalid Format

Port

8888

SAVE

LiveFeed: Lists the available camera connections in your network. Click on any camera name to view the live streaming



Available Connections

1. user@raspberrypi3-rdk-camera



Live Feed

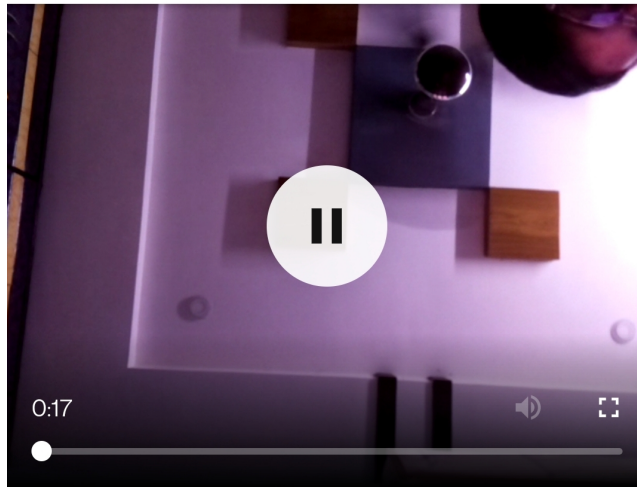


Recordings



Settings

Live Streaming: Playing live streaming video from rpi camera in the mobile app



Recordings: In this user can select the specific option such as today, last week and custom range to view the list of recordings stored in the cloud under logged in user id or user name. For custom user should suppose to select start and end date to get the list of recordings. In the list of recordings user can able to see the recording name, recording date and duration of recording.

Recordings

TODAY

LAST WEEK

CUSTOM



Recording_7234
21-02-2020

22h 29m



Recording_4026
21-02-2020

3h 34m



Recording_3494
21-02-2020

8h 7m



Recording_5110
21-02-2020

9h 10m



Recording_4660
21-02-2020

5h 26m



Recording_4195
21-02-2020

24h 49m



Recording_3366
21-02-2020

15h 33m



Recording_2262
21-02-2020

14h 39m



Live Feed



Recordings



Settings

Recordings

TODAY

LAST WEEK


CUSTOM

Start Date

17 Jun 2020

End Date


17 Jun 2020



Recording_7234

21-02-2020


22h 29m



Recording_4026

21-02-2020


3h 34m



Recording_3494

21-02-2020


8h 7m



Recording_5110

21-02-2020


9h 10m



Recording_4660

21-02-2020


5h 26m



Recording_4195

21-02-2020


24h 49m




Recording_3366

21-02-2020


15h 33m



Live Feed



Recordings




Settings

Settings: From settings user can able to edit :


1. camera settings such as video resolution for live feed and recordings and IP address, Storage path which is changing the internal storage path for downloading recordings.
2. User settings such as camera access permissions, Adding new camera to the account, editing user details like password etc.
3. Other settings which are app related settings like terms and conditions, Privacy Policy, About application, App rating, Logout.

Settings

Camera Settings


 Resolution

 IP

 Storage


User Settings

 Permissions

 Add Camera

 Details


Other

 Terms & Conditions

 Privacy Policy

 About


Live Feed


Recordings


Settings

Settings



IP



Storage

User Settings



Permissions



Add Camera



Details

Other



Terms & Conditions



Privacy Policy



About



Rate the App



Logout



Live Feed



Recordings



Settings