Doc Version: Drafted Reviewed by:Panduranga,xyz

Requirements for Baby & Nanny Monitoring Camera

Requirements

1. Stream recorded videos from SD card with "AWS webrtc SDK" since the team has already invested a good amount of time on the same. We might have our own webrtc server in future to bring down the cost further.(From Vivek)

2. Each event will be 30 seconds clips duration or configurable. (From Vivek)

3. Clip will be H264 and Opus format for Video & Audio. AWS Webrtc SDK only supports G711 and Opus.

4. Event information and Recorded Clips will not be stored at AWS. Clips should not be forwarded to Kinesis SFU & S3 buckets for storage. We have to get event names and clips from SD cards with websocket APIs or webrtc Data Channel. (From Vivek)

5. Camera and Android APP should work in Office premises having Nated wifi IPs. Both devices can be in different environments or locations. (From Vivek)

6. Should work with T31 and T41 memory constraints of 512MB. (From Vivek)

7. Should work with hardware encoders if available.

8. All the video encoding and AI DNN should be offloaded to GPU. For analytics, luminance with gradient information on ROI will be provided. AI Model should be in json format. Shared objects templates will be shared.

9. If a customer wants to pay for AWS streaming cost then only we will allow the AWS streaming.

Events

Let suppose we have 2 events as follows. 30 Seconds H264 Clip will be recorded as

1. /dev/sd0/MotionEv/Cam1_2023-11-21-07-40-17_780 (Motion Events)
2. /dev/sd0/FacialRecEv/Cam1_2023-11-21-08-40-17_790 (Facial recognition Events)

Question from Natraj for Vivek
Quality of life will be affected when we store events in SD cards only. Everyone might not have SD cards.

Index files for events will be updated on each event. Arvind will share the index files structure by next week.

We have two events with Cam1. Motion Events at time 2023-11-21-07h-40min-17s_780ms and Facial recognition Events at time 2023-11-21-08h-40m-17s_790ms respectively.

Event information and Clips will not be stored at AWS. Only on demand from Android APP, event information will be provided to APP.

API Interface

Webpages & Adappt Android APP call APIs

Rest APIS

1. Post https://example.com/ name of cameras(aws channels) connected eg Cam1, Cam2, BackDoorCam. I assume this rest api already exists. For detail check https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/API_ListSignalingChannels.html
2. Login with Auth token. We assume Apis already exists.

Websocket Apis wss://example.com/Cam1

1. /MotionEv/023-11-21-08 will return list of MotionEvents at hour 8
2. /FacialRecEv/ will return list of dates
3. /Cam1/ will return types of events for example Facial Recognition, Motion Events etc

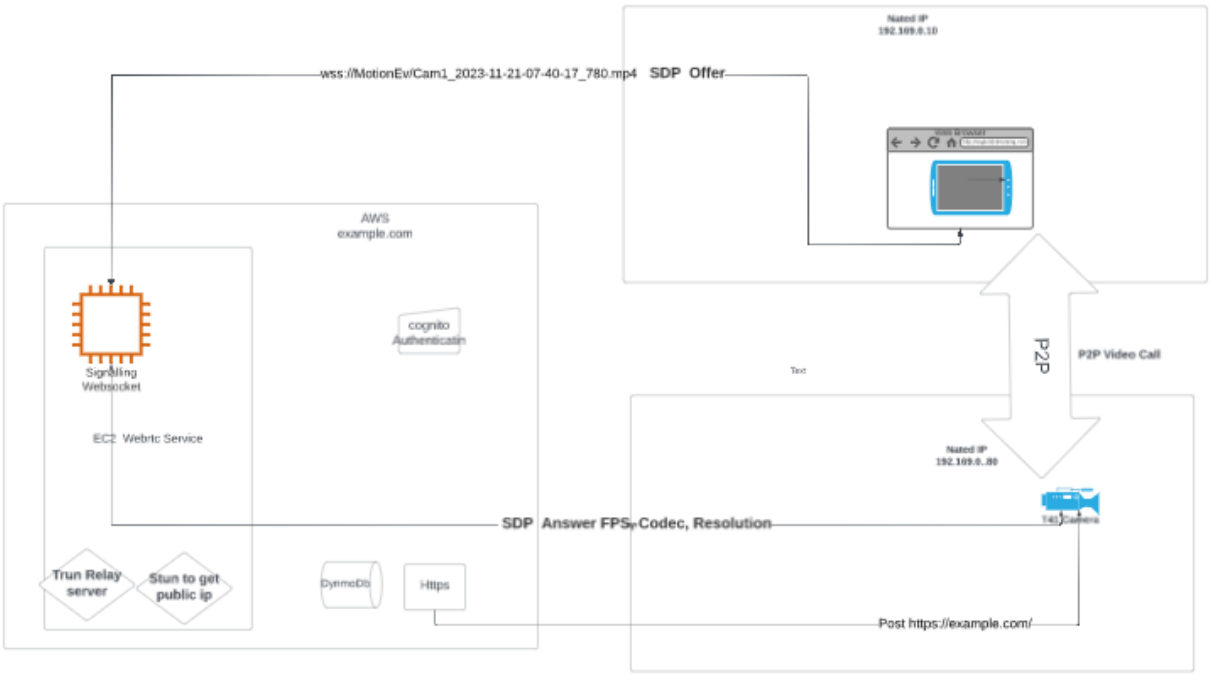
Once we have a list of Camera and Events from Rest API. Android App will Create SDP Offer to Cam1. Cam1 will Answer back SDP to APP.

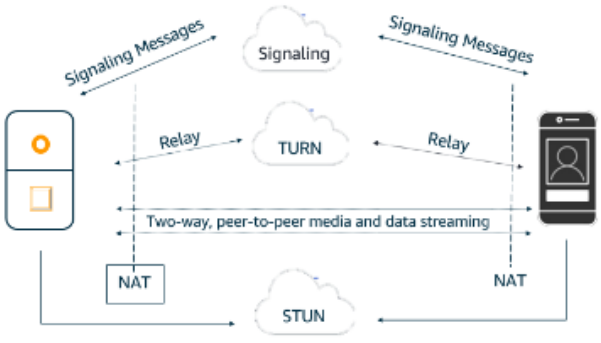
SDP Offer from APP with wss://MotionEv/Cam1_2023-11-21-07-40-17_780 to Cam1.

Cam1 Answer SDP back to APP with resolution, frame rate and codec information

After Webrtc Candidate Exchange with SDP, Pear to Pear connection established for streaming video
CAM1 < P2P conection> APP (video/audio and Data channel)

Diagram





https://lucid.app/lucidchart/cb56b841-13bf-40d2-b6dd-ceb3ab6f210d/edit?beaconFlowId=4D23BF0444D6D435&invitationId=inv_0b445b9f-b720-4ba4-85af-a7b39f278bbc&page=0_0#

Goals

1. Using Ingenic T41 chip-set brings down the cost of sensors, surveillance and monitoring devices.
2. Record video & Audio on SD Card and remove the dependency on AWS Kinesis. This will reduce the cost to the company.
3. Create development and testing environment with software stacks, which helps in rapid development of AI solutions:
    a) Counting objects like cars and detecting human faces with a T41 chip-set.
    b) Thin AI solution based on Gradient or Laplacian with T41 AI Engine
4. Replace factory uboot Boot Loader of T41 MIPS with https://github.com/OpenIPC/u-boot-t41. Change File system of SD card.

Use Cases

We are planning to generate revenue by selling customized AI solutions to Adappt Clients and to potential new customers. For example

a) Camera for counting Car with Number Plate. Each camera would cost in the range 4-8 thousands INR.

b) Camera for Attendance, storing Entry and Exit time of employee etc.

c) Camera working as sensor where video streaming is not allowed from office premises.

f) Monitory Nanny and baby from workplace and giving directions to Nanny from phone.

Deliverables(70 Man Days on T41 , 55 Man days on Pi)

Prototype and screenshots are available. Demo on every Sprint Cycle.

1. Very high performing Streamer. Which support webrtc on Arm64 and MIPs64 .
2. High number of webrtc streaming from Camera to browser.
3. H264 & Opus Recording on SD card and playback
4. Authentication, Auth token, Timeout of login. No storage of password.
6. Rest API and Json for integration with python.

Work Area

1. Pi and T41 Camera side

1. Webrtc Client and websocket client.
2. Storing of 30 Secs Clips in SD
3. Index files for Events(Clip file name) . Hour and Date index files.
4.Retention configuration of Clips.
5. Unit Demo of Event Display and Clip play on Browser with local or Natted ips.

6. Webpage to configure Storage Retention and Clip Size. This feature is missing in XIaomi Cam.

2. Android APP

1.Webrtc client and websocket client. POST Rest api to know the list of camera connected
2. Websocket or DataChannel call to get the list of events as described above
3.Unit Demo of Events and Clip Play on Android APP. It should be same as the demo provided by Arvind

3. AWS Services

1. Websocket Server and Lambda services.
2.Routing the websocket call and to the right camera.
3. Web Page with company domain,
    To display a list of cameras connected.
    Health of Camera
    Configuration of Camera.
4. Unit Demo of web pages.

4. Pi Image (Hardware team)

1.New Pi image/SD image for Pi and T41 with webrtc and signaling binaries and /etc/systemd/system services.
2. Two binaries webrtc and signaling will be provided by Arvind. Also service files.
3. Automatic update of Pi Images when we release new binaries.
3.Unit Demo with flashing new image.

Task

1. Compare Xiaomi's performance with our deliverables.

1. T41 based webcam, dashcam and IP camera are available in the market. Compare Network layer performance with TCP and UDP streaming with wifi.
2. IO performance of SD card
3. After replacing Open Source Boot loader https://github.com/OpenIPC/u-boot-t41., find if performance degraded.
4. Test performance of T41 with encoding video and audio.
5. Test performance of T41 AI with Sample 8x8 Matrix multiplication.

2. Compare T41 with T31 with performance vs cost

3. Development Environment setup (7 Man days)

1. Like aarch64 pi board, we will first enable ssh login at T41. Do uname -a to find T41 arch. is 32 LSB MIPS64. Do binwalk/file to find the filesystem of root and SD card.

2. Copy any test.264 at /mnt/ folder of T41 and test recorded video stream to AWS Kinesis with Gstreamer pipeline.

3. At the console terminal or screen gst-launch-1.0 v4l2src do-timestamp=TRUE device=/tmp/test.264 ! h264parse ! video/x-h264,stream-format=avc,alignment=au ! kvssink stream-name="plugin" storage-size=512 access-key="YourAccessKey" secret-key="YourSecretKey" aws-region="YourAWSRegion"

4. Install nodejs and npm. This will help in bench-marking camera and network performance.

5. Create a project and repository at github or bitbucket. Name could be SD_Live_AI_Streamer, camAnalytics, anything which makes sense.

6. Compile helloworld c/c++ sample with gcc/g++ with 32 LSB MIPS64 tool chain on T41 and linux Host. Run ldd, file, hexdump on binary, md5sum, gdb, strace etc

7. Create an emulator of T41 with Qemu. Install Qemu with Image of 32 LSB MIPS64 on linux ubuntu laptop.

8. VPN setup to work from home.

9. IDE and workspace with gcc/g++

3. Test Environment setup (25 Man days * 3)

1. Check if https://github.com/cgrrty/Ingenic-SDK-T31-1.1.1-20200508/tree/main/opensource works for T41

Check the CPU performance with SIMD128 (Vectorization) enabled. gcc -fno-tree-vectorize with and without.

Run Sample codes SDK to verify single-instruction multiple-data instructions operating on hardware vectors which are 128 bit wide 2x f64/u64/i64, 4x f32/u32/i32, 8x u16/i16 & 16x u8/i8

2. Test 64KB + 128KB L1/L2 Cache

Create struct with #pragma pack of 64KB and 128kb and find the cache Hit & Miss. This will help in improving performance of Streamer.

3. Video Encoder Test. Maximum FPS and Resolution supported with 2592*1920@30fps

Audio Encoder Test. With 16K/24K/32/44.1K/48K sampling

Check Maximum number of Multiple Streams possible to encode . Check if chip-Set is heating up and its behavior with outdoor 45 degree Celsius.

4. OS performance test

Linux should boot from flash and the root file system should be mounted on /dev/xxxram0. Where u-boot, the Linux kernel, and the compressed RAMdisk are loaded on the flash memory.

Check for Address Content something like
0x0200 0000 – 0x0204 FFFF u-boot and u-boot parameters (327 KB)
0x0205 0000 – 0x0211 9223 uImage – Linux kernel (823KB)
0x0211 9224 – 0x0234BCA3 Compressed RAM disk (2.1MB)
0x0234BCA4 – 0x02FF FFFF Unused (12.70MB)

Login as root.

Create test Scripts for

<A HREF="/cgi-bin/memory">Memory usage</A>
<A HREF="/cgi-bin/log">Kernel boot log</A>
<A HREF="/cgi-bin/config">Kernel Config options</A>

5. Test Samples

T41 Features	Test https://github.com/cgrrty/Ingenic-SDK-T31-1.1.1-20200508/tree/main/samples
Signal Processor	3A library (auto exposure, auto white balance, auto focus) Dark current correction, lens shadow correction, color space conversion, dead pixel correction, gamma correction , defogging algorithm Adaptive dynamic range compression 2 frames wide dynamic range (digital overlap / frame) Edge Adaptive Demosaic Local contrast enhancement adaptive Multi-level NR (2DNR/3DNR) and sharpening X+Y lens distortion correction

Memory	DDR2/DDR3/DDR3L/LPDDR2/LPDDR3 SIP 512Mb/1Gb/2Gb or external memory SPI Nor Flash/SPI Nand Flash MSC/SD card interface eMMC interface
Security/Crypto	Encryption algorithms: AES, DES, RSA and HASH Digital True Random Number Generator
AI algorithm	deep learning algorithm with high precise and good flexibility Human detection, Facial detection/recognition Cry detection, Vehicle detection, Pets detection
IO	FLASH and SD
Fast Boot	Fast Boot Time - Dual boot Time - Fast AE / AWB - ~200ms stable video output
Wide extension	Support 4-channel digital MIC array Support IoT-WIFI / BT / 4 Support SLCD Display Support UVC / UAC
Audio	Integrated Audio Codec Support Rate 8K/12K/16K/24K/32/44.1K/48K/96K Support I2S Interface Echo cancellation

Future Development plans

Right now we have Script which reads from a USB camera and streams to Kinesis with the Gstreamer Pipeline. But to make the Pi board and T41 similar to an IP camera, we have to provide the following software stacks. One Binary for webrtc, mpegDash, RTSP and onvif which work on both PI and T41.

1. RTSP streaming with h264, h265, G7xx) 7 Man Days
2. Onvif 14 Man Days
3. HTTPS(Moving JPG, Rest API for camera settings, Resolution settings, FPS settings, webserver, ssl certificates) 4 Man Days
4. webrtc streaming with Opus/G7xx and h264) 7 Man Days
5. AI analytics(DNN, segmentation, Key feature detection) 7 Man Days
6. SD card storage(H264 with retention configuration) 7 Man Days
7. Archiving Stream in H264 format to SD card (5 ManDays) 7 Man Days
8. Moving JPG streaming. 2 Man Days
9. Streaming from SD card for video and audio. 7 Man Days
10. H264 clips based on Motion events only. Also full recording support if customer wishes.

Other desirable features
11. Door Bell
12. Audio Call

Bottleneck

1. Some customer request Cloud media storage Analysis with AI/ML services which might be bit tricky with webrtc
   https://drive.google.com/file/d/1D6RDWEPfJHJNayETLpM7BuBJmsOAOntr/view?usp=sharing

2. Some companies block webrtc protocols, in that case we need to have a fall back mechanism to hls, mpeg dash or rtsp.

AWS Pricing

It only costs if you are streaming to AWS. Most of the time we do not need to stream to AWS.

Full HD_on opening mobile app only_2 or more users simultaneously $.17

ICE candidates connection cost

1. For P2P. Device to Device connection $0.03
Almost zero cost with P2P Streaming, even if streaming happens round the clock. Mostly customer, does not need Cloud Storage and Cloud Analytics,

Host candidates. Almost Zero Cost. Around $0.03 per month for an active signaling channel(Camera)
Srflx candidates. Almost Zero Cost.. Around $0.03 per month for an active signaling channel(Camera)
Prflx candidates. Almost Zero Cost.. Around $0.03 per month for an active signaling channel(Camera)

2. For Peer To Relay(Turn Server) $0.12
Flat streaming cost when doing Relay Streaming. Some companies block all the ports except https and Proxy ports. In that case we have to pay for the Turn server. Many APPs give "Cost to Customer" features in GUI, when Relay Streaming is happening.

Relay candidates to AWS (tcp, udp). Around **$0.12** per thousand TURN streaming minutes.

Camera Cost Est.

Basic Pi Board	Rs 2000 - 4000
T.41	Rs 800
Pi camera	Rs 250-400
Ingenic camera	$1 not very sure
Microphone cost?	
Speaker Cost	?
Plastic Chassis	?
Red and Green LED	Rs 3x2

Webrtc Streaming cost Est.

Device to Android APP	. 03$ per month flat on AWS.
Device to AWS with storage(optional)	?
EC2 Cost	?
Network consumption	?
IAM	?
Channel info and Device details in DB	?
SD Image & Binary hosted at FTP	?
Analytics on AWS (optional)	?

Camera commissioning

Xiaomi way

Xiaomi camera gets cameraId & wifi password by scanning QR code from each other. First the Android APP scans the QR code of Camera and gets cameraId, model and network ids. Then the Mi Camera scans the QR code generated by Android APP to get the wifi and password details.

Adappt ways

a) Hotspot creation with predefined Name & Password at Android device. Camera is configured with wifi setting as follows

cat /etc/wpa_supplicant/wpa_supplicant.conf

```
network={
 ssid="adappt"
 psk=d8bac9438ea8c07a92ee2510a18593ada5f28cf20c8601fba85a50eb36f9f0d9
}
```

b) By creating QR code with CameraID and IPs at Camera side for predefined networks. Then store same information in database.

Otherways

Bluetooth Pairing. Change the wifi config file in the camera from APP.

Why Webrtc

	streaming	webrtc
Cost		P2P flat, almost zero
Cloud media storage Analysis with AI/ML services	yes	no
Typical latency for live playback	1-4 seconds	20-60 millisec
Number of simultaneous playback sessions	100 sessions Up to	about Up to 10 sessions
Bidirectional streaming	No	yes

SDK for camera devices Producer SDK WebRTC SDK

Annex

ROI Rectangle of Interest. Only relevant rectangles with motion will be processed at AI. Number of Bits allocation for ROI will be higher than the background.

SDP. Session description Protocol

Offer. Session Description offer from Android Device to Camera

Answer Session Description answer from Camera to Android Device