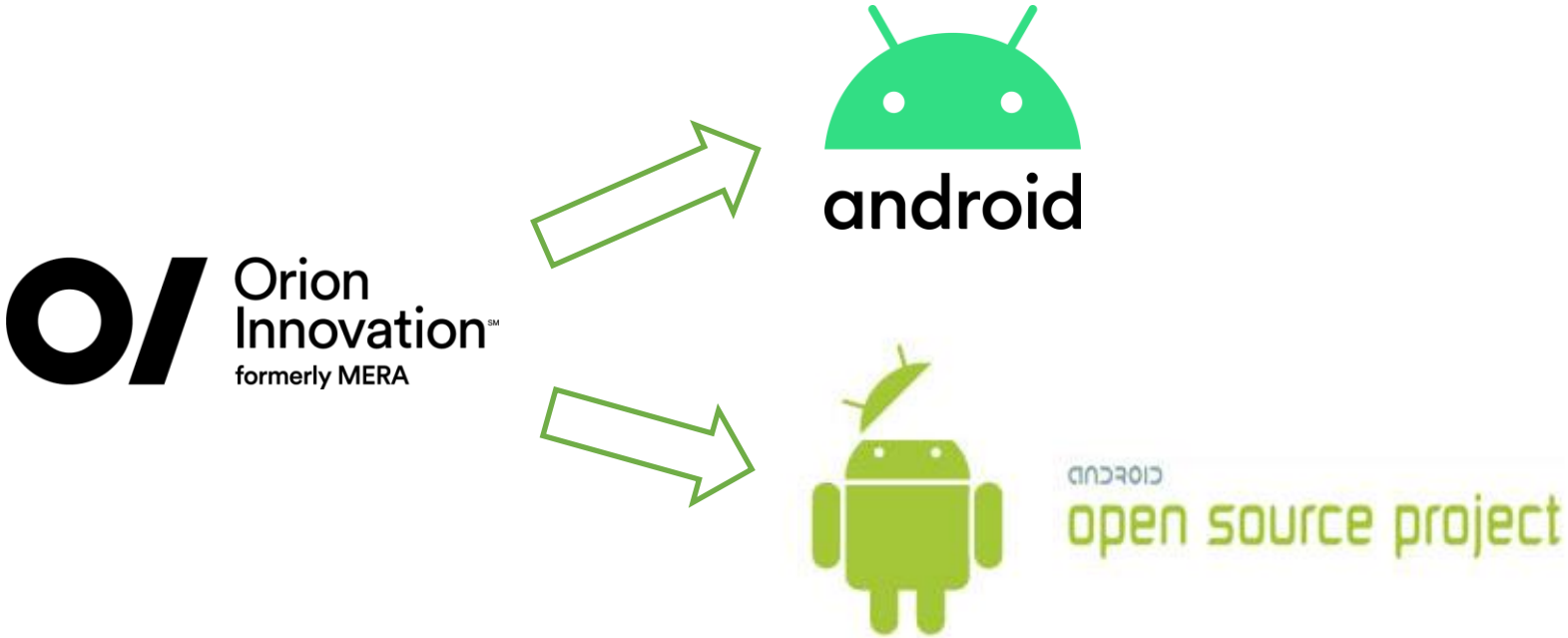
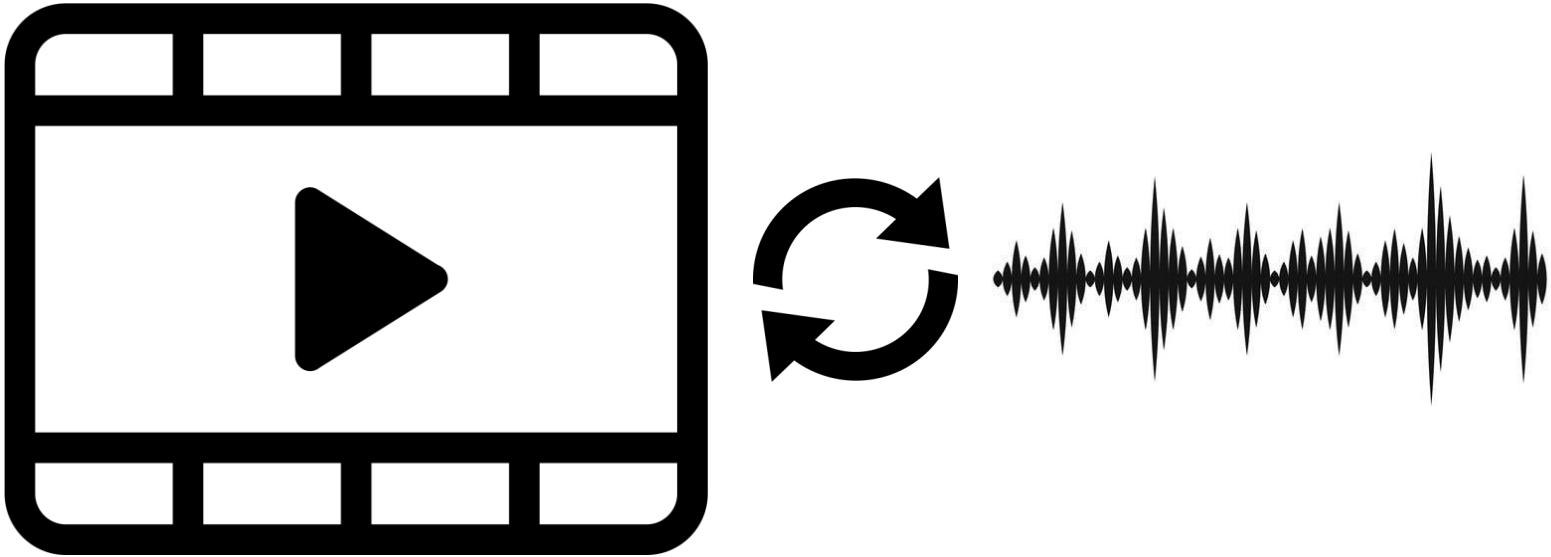
MOBIUS 2021

A/V Sync in Android

April, 2021


Orion
Innovation℠
formerly MERA
android
android
open source project

A/V Sync



Human Sensitivity

Skew	Effect
20ms	Not visible
50ms	Something’s wrong
1000ms	Video is ignored

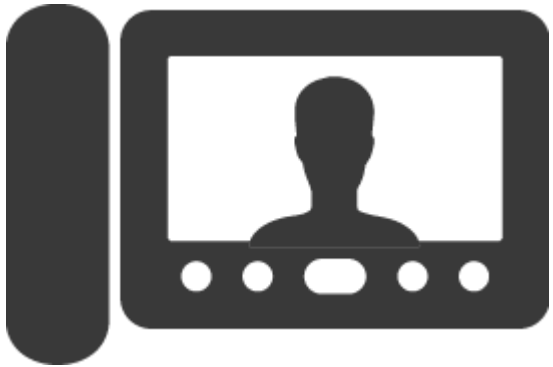
Agenda

- Customer Request
- A/V Sync Theory
- A/V Sync in Android
  - MediaSync
  - ExoPlayer
  - Multimedia Tunneling
  - WebRTC

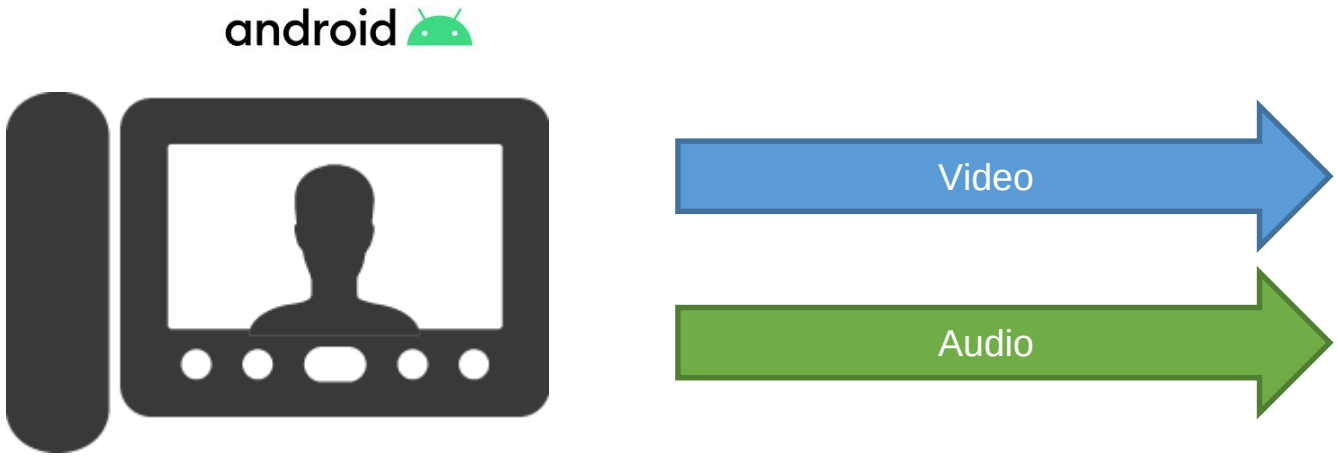
Customer Request

Task: create Android Videophone

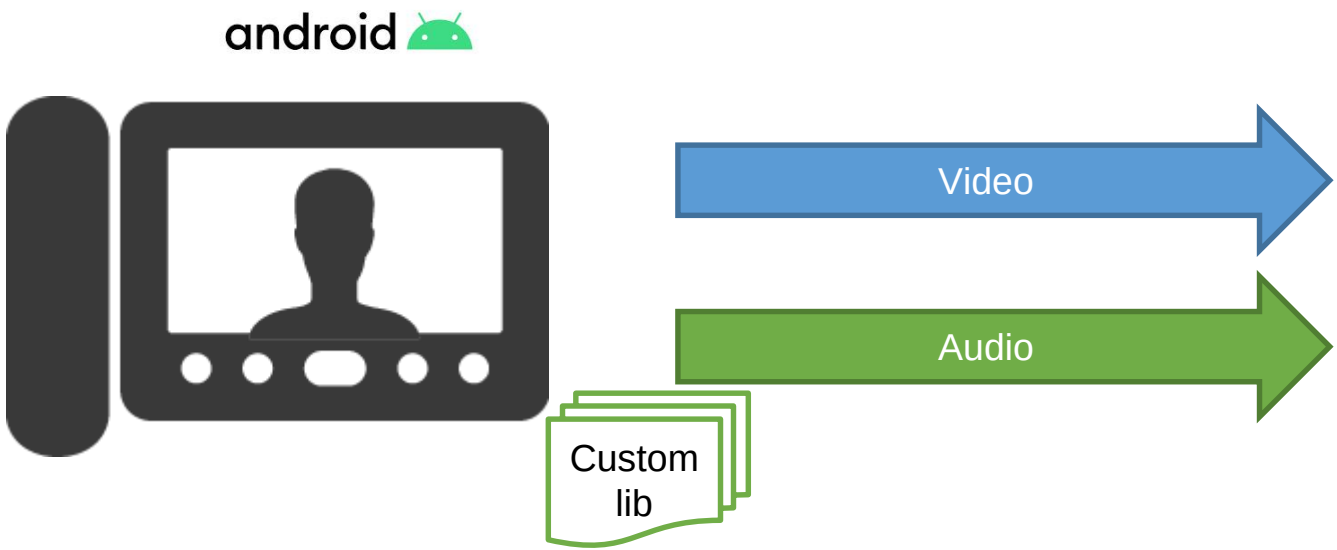
android



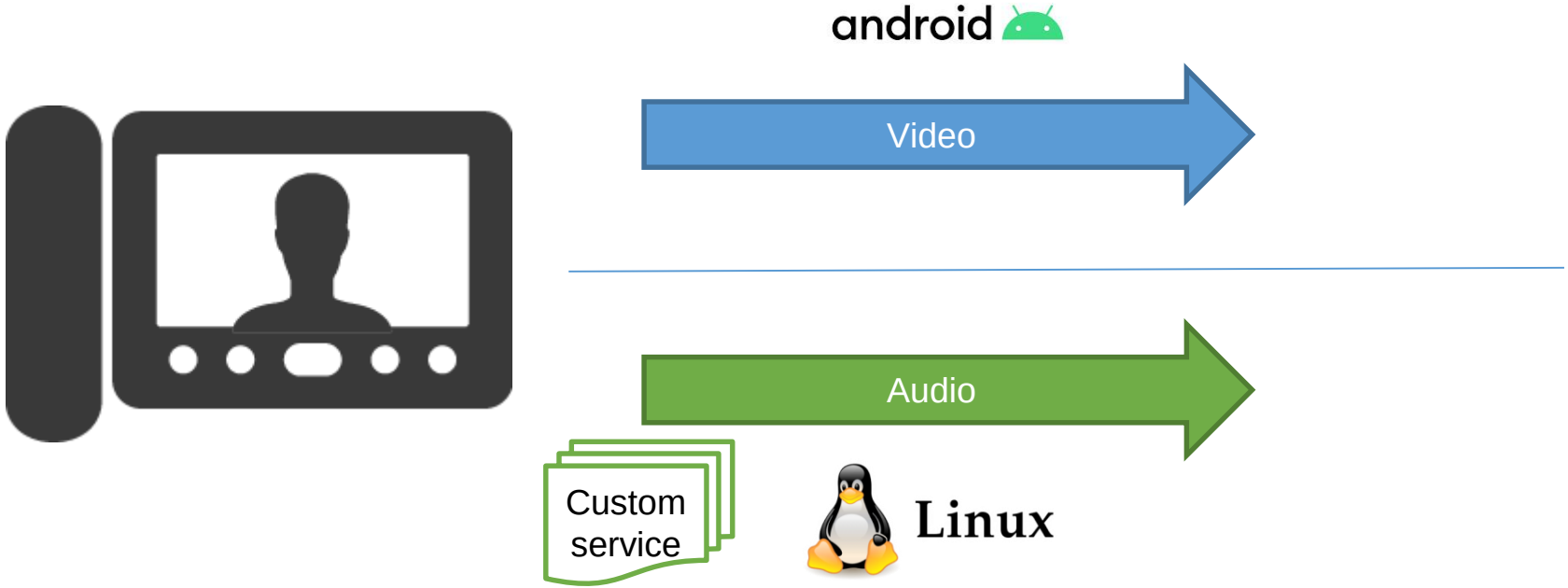
Task: create Android Videophone



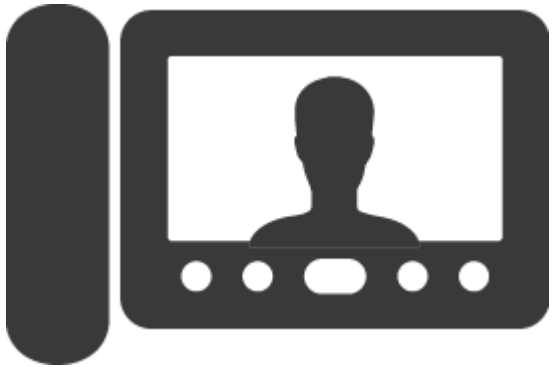
Task: create Android Videophone



Task: create Android Videophone



Task: create Android Videophone



android

Orion Innovation formerly MERA

Video

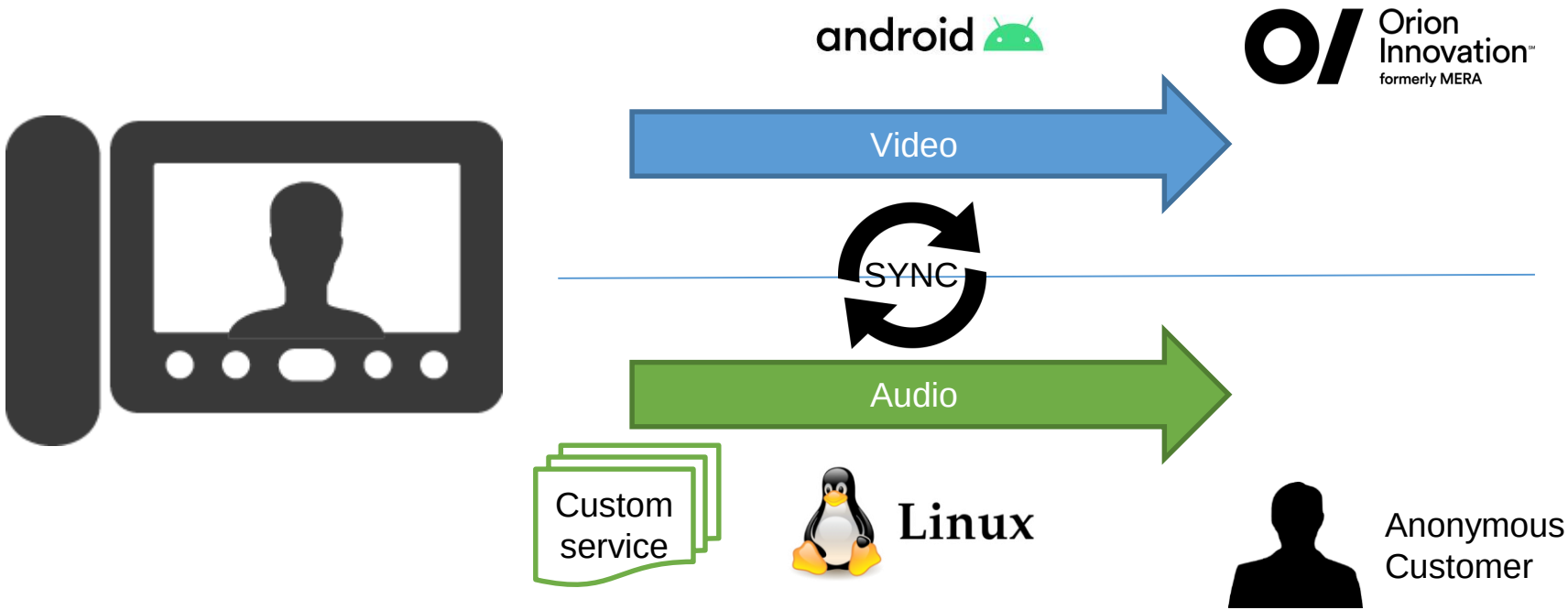
Audio

Custom service

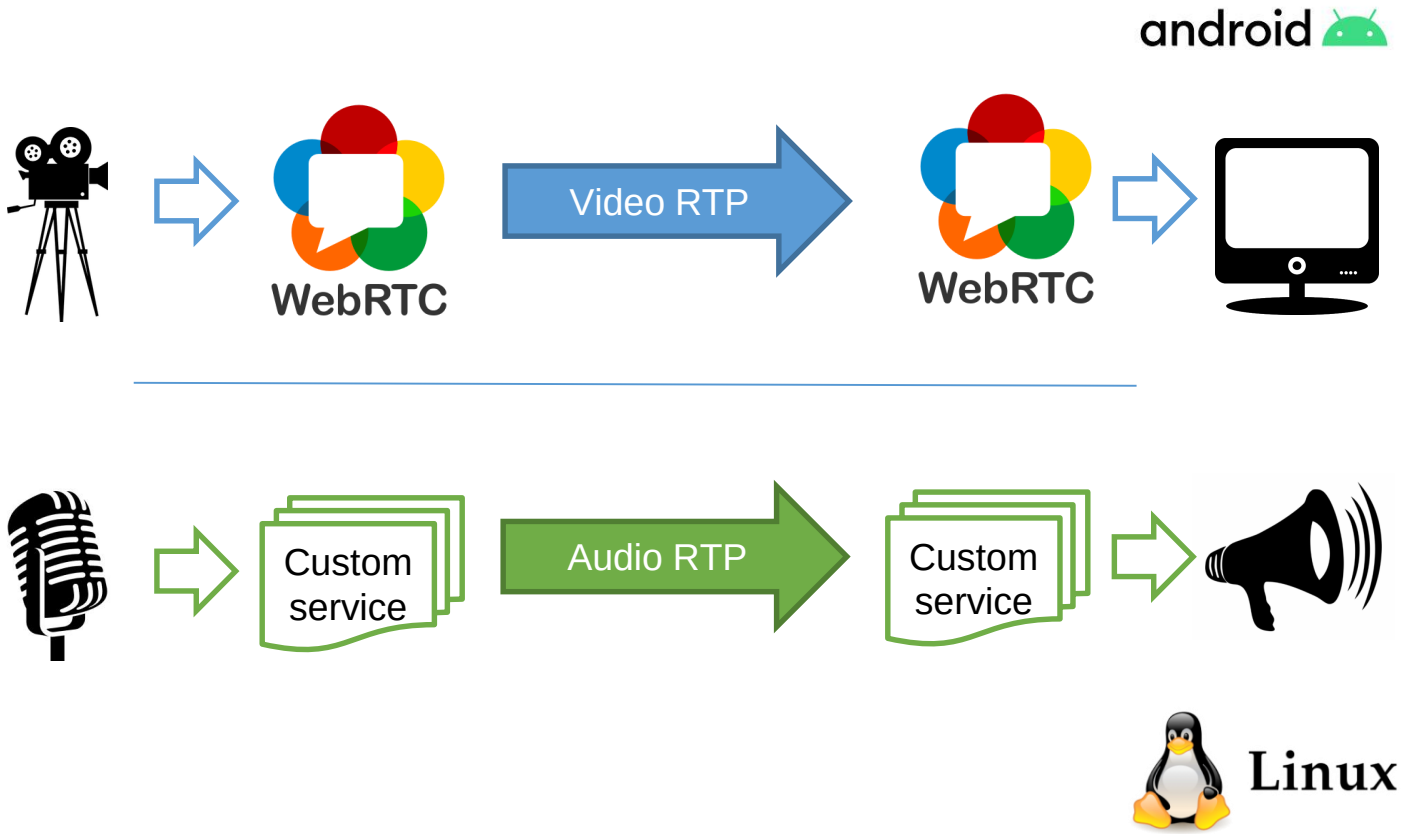
Linux

Anonymous Customer

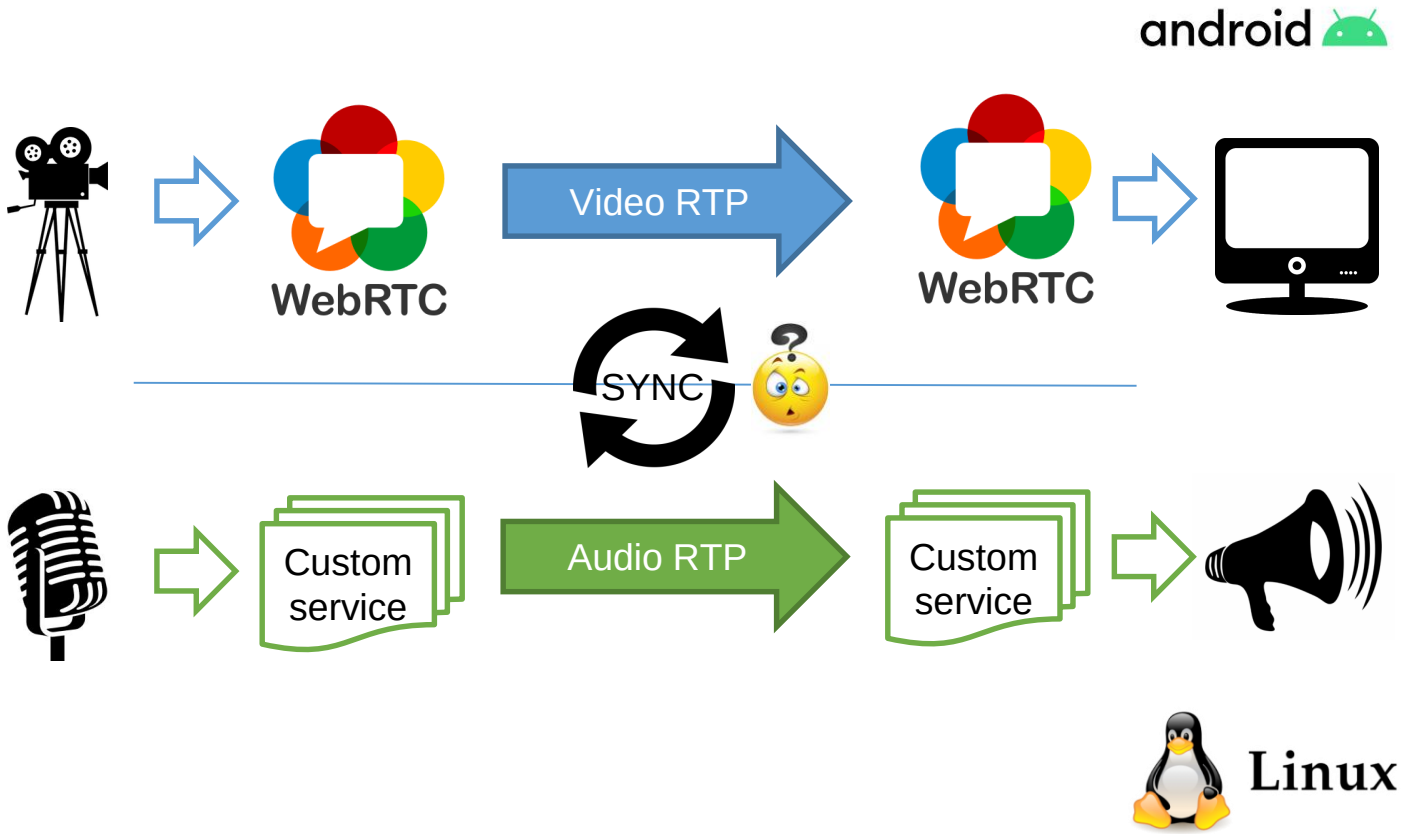
Task: create Android Videophone



Architecture



Architecture



How A/V sync works?

How A/V sync works?

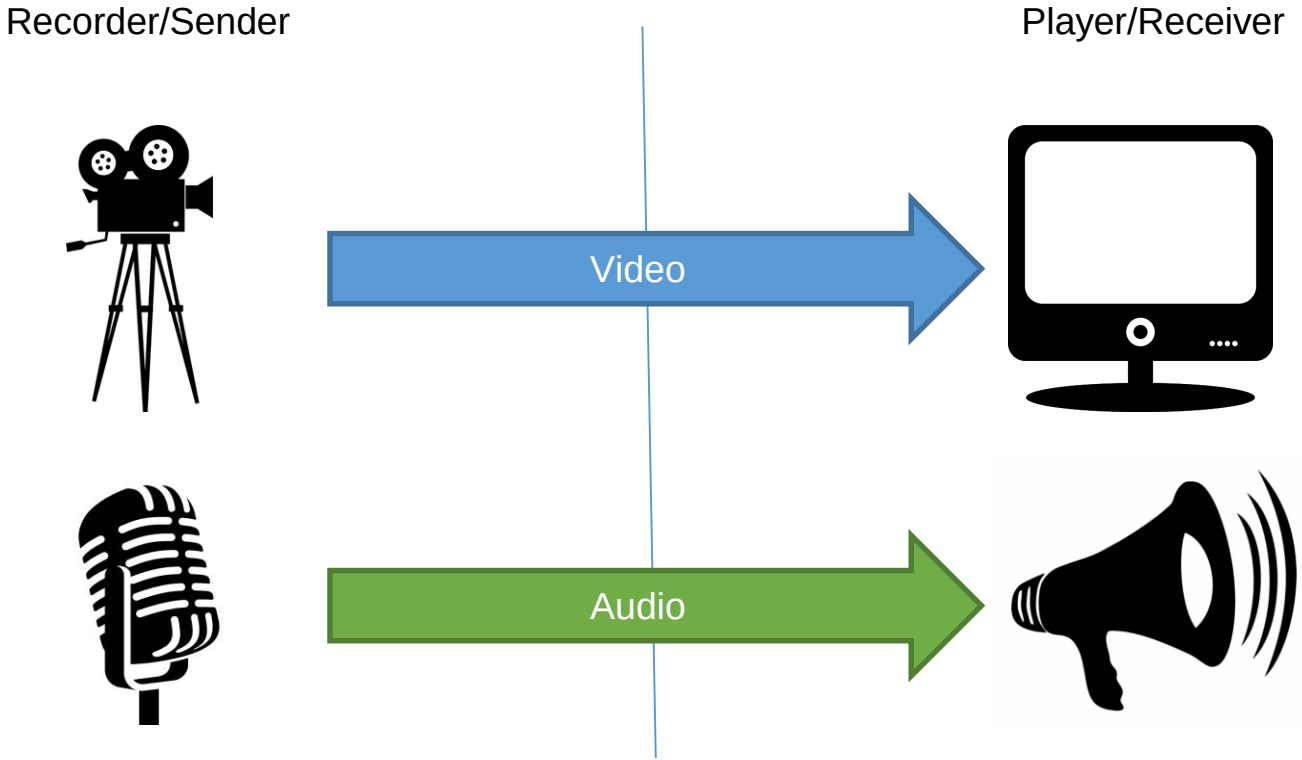
How A/V sync works?

How A/V sync works?

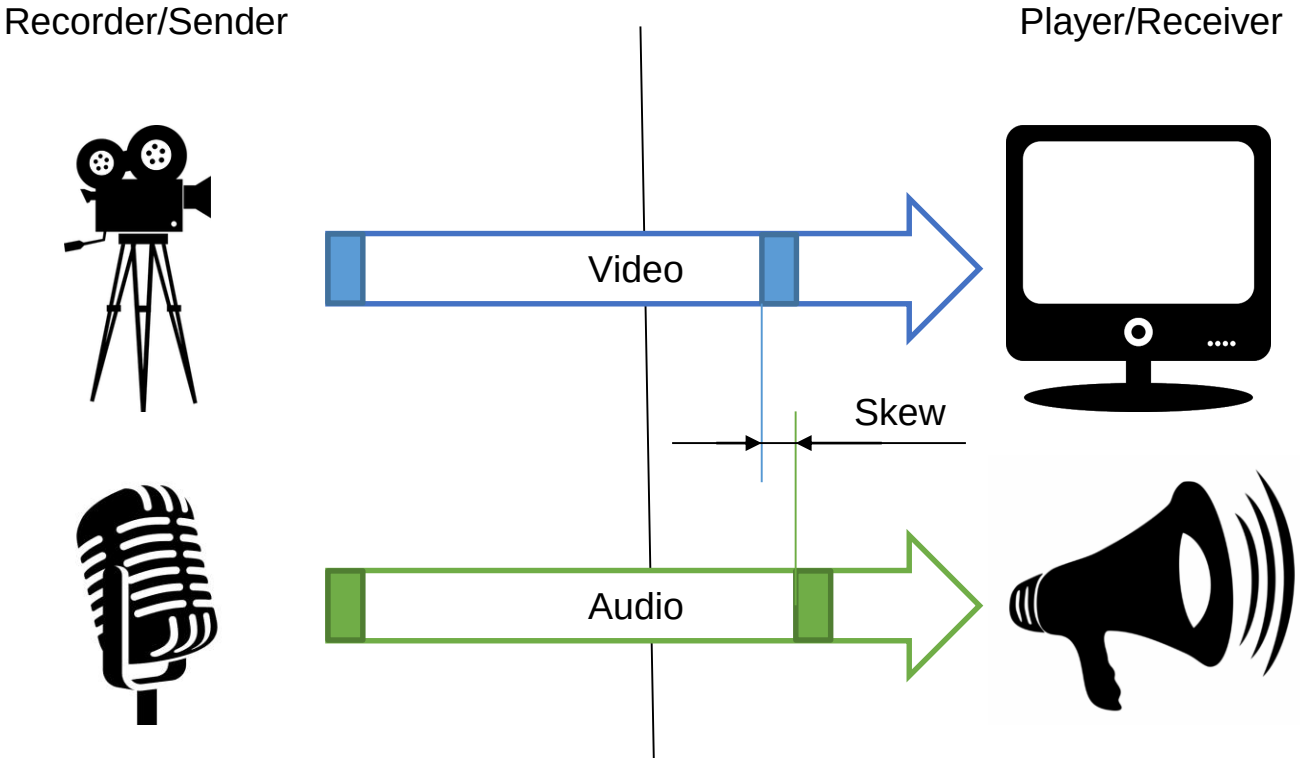
Timestamps

Theory

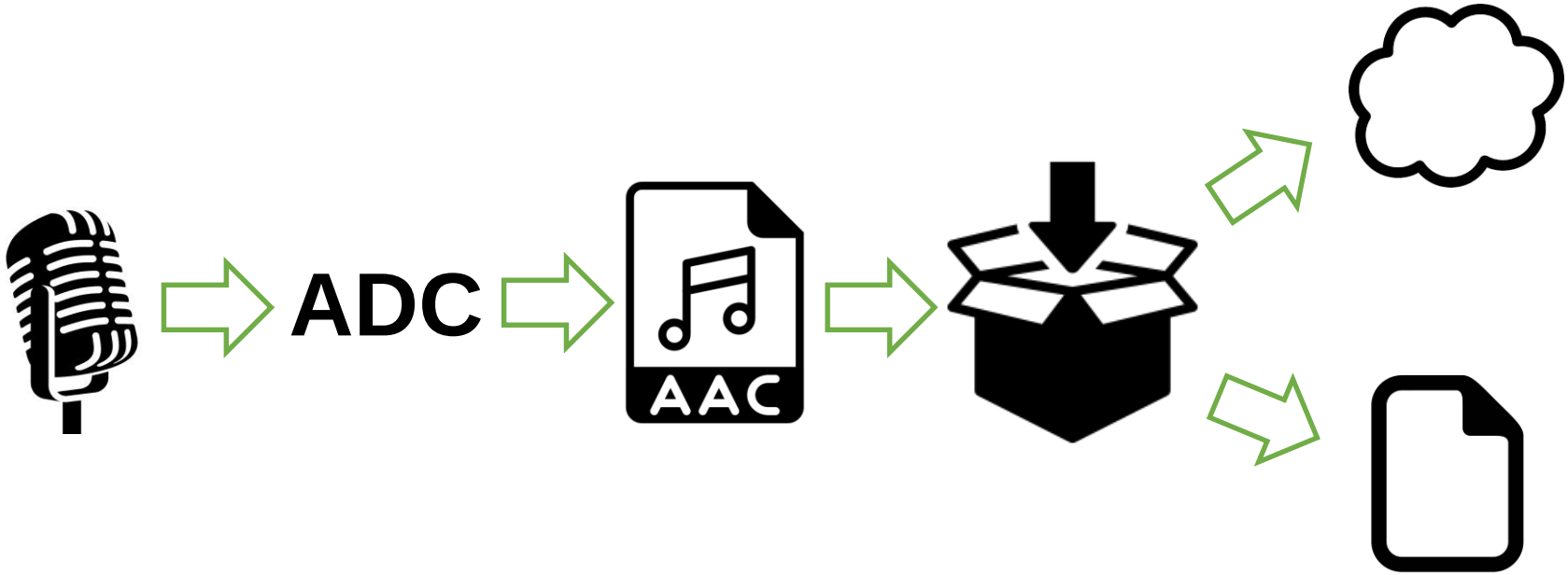
Two Streams



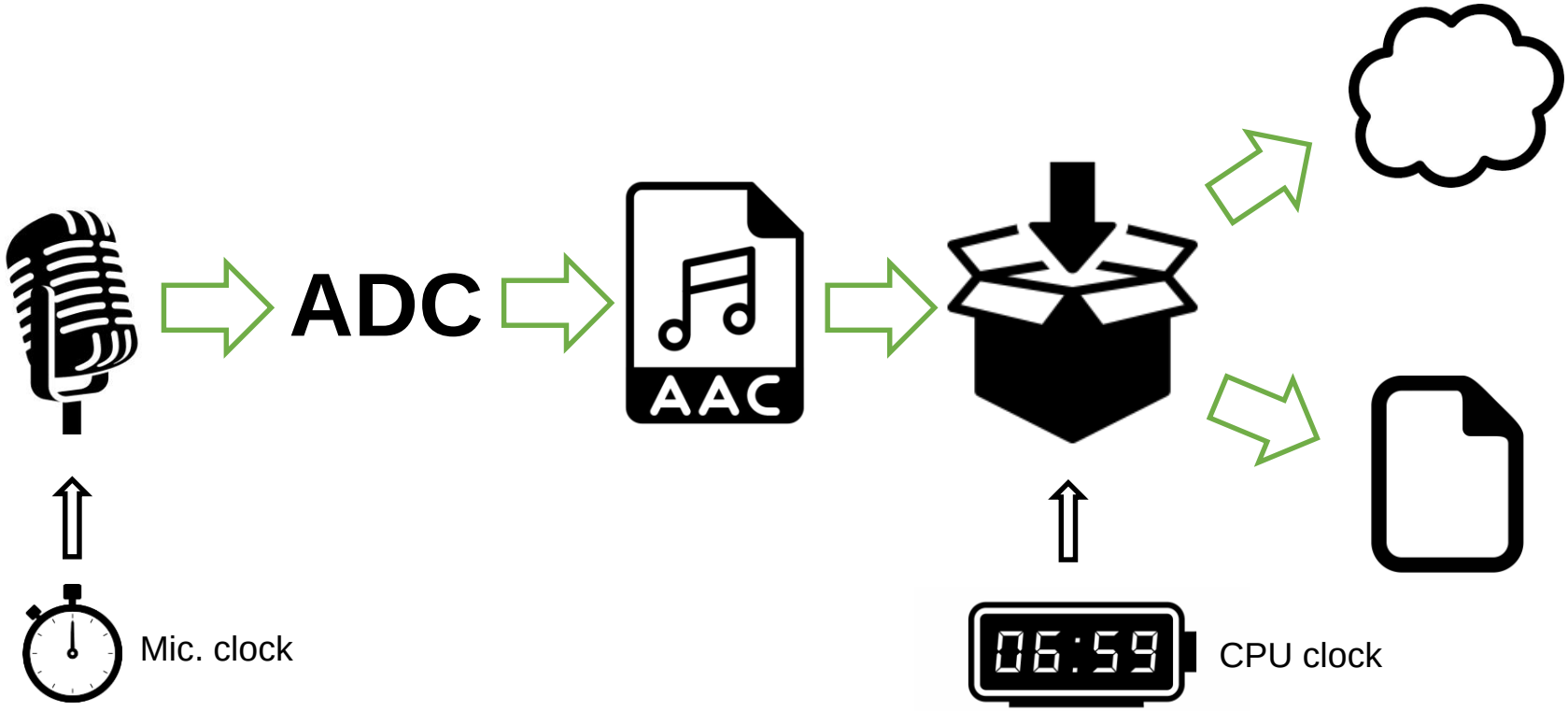
Skew



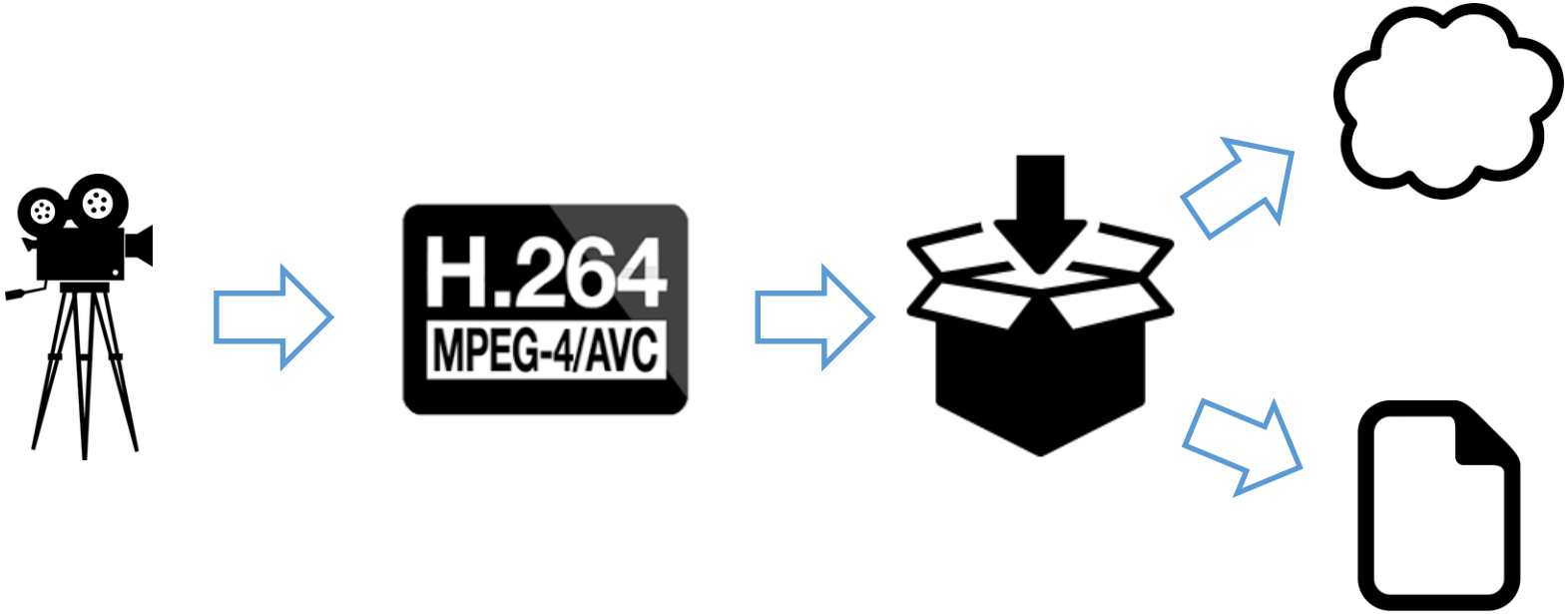
Recorder Delays - Audio



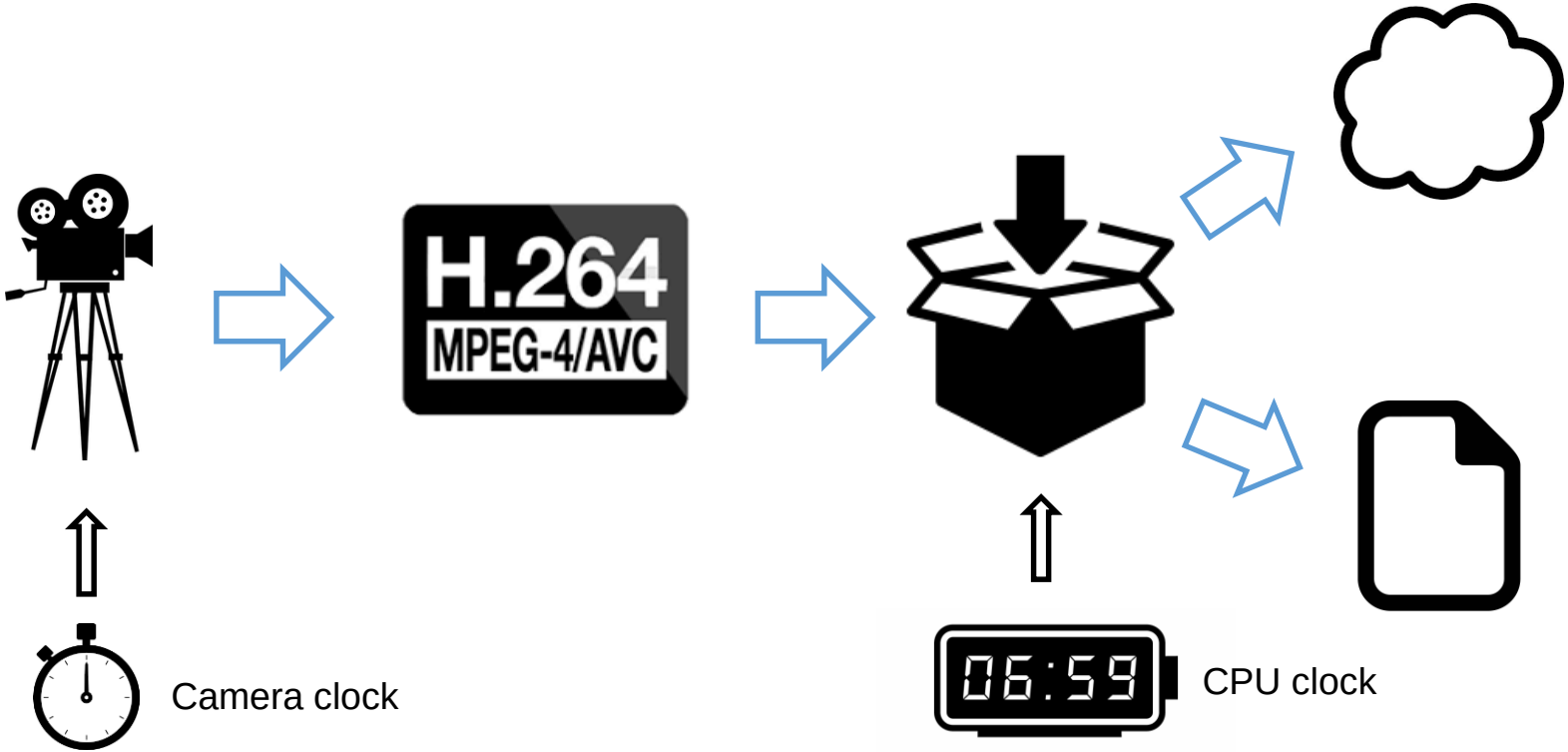
Recorder Delays - Audio



Recorder Delays - Video



Recorder Delays - Video



Master Clock

Video timestamps

Video RTP

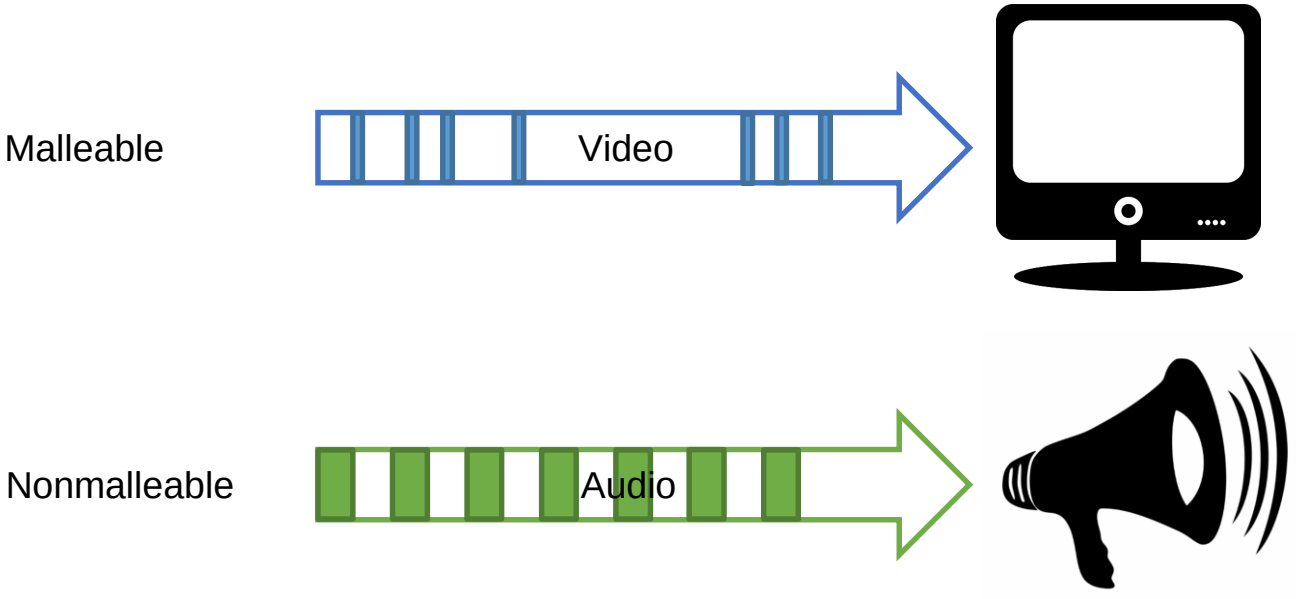
Wall Clock

RTCP

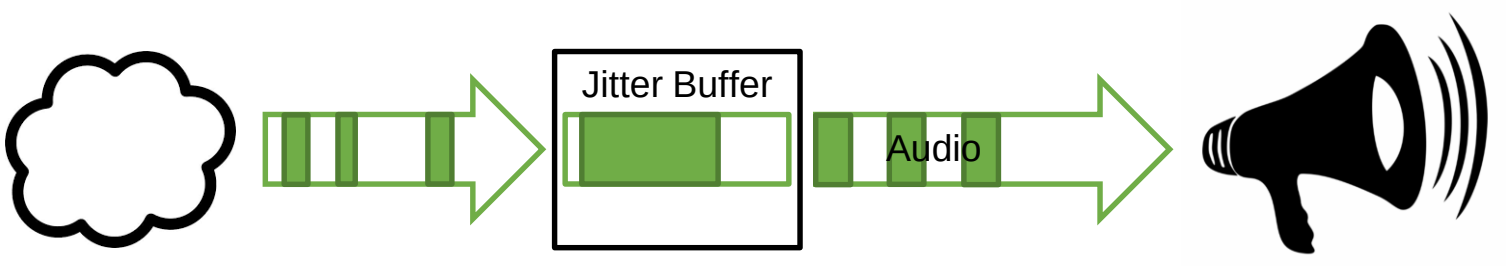
Audio timestamps

Audio RTP

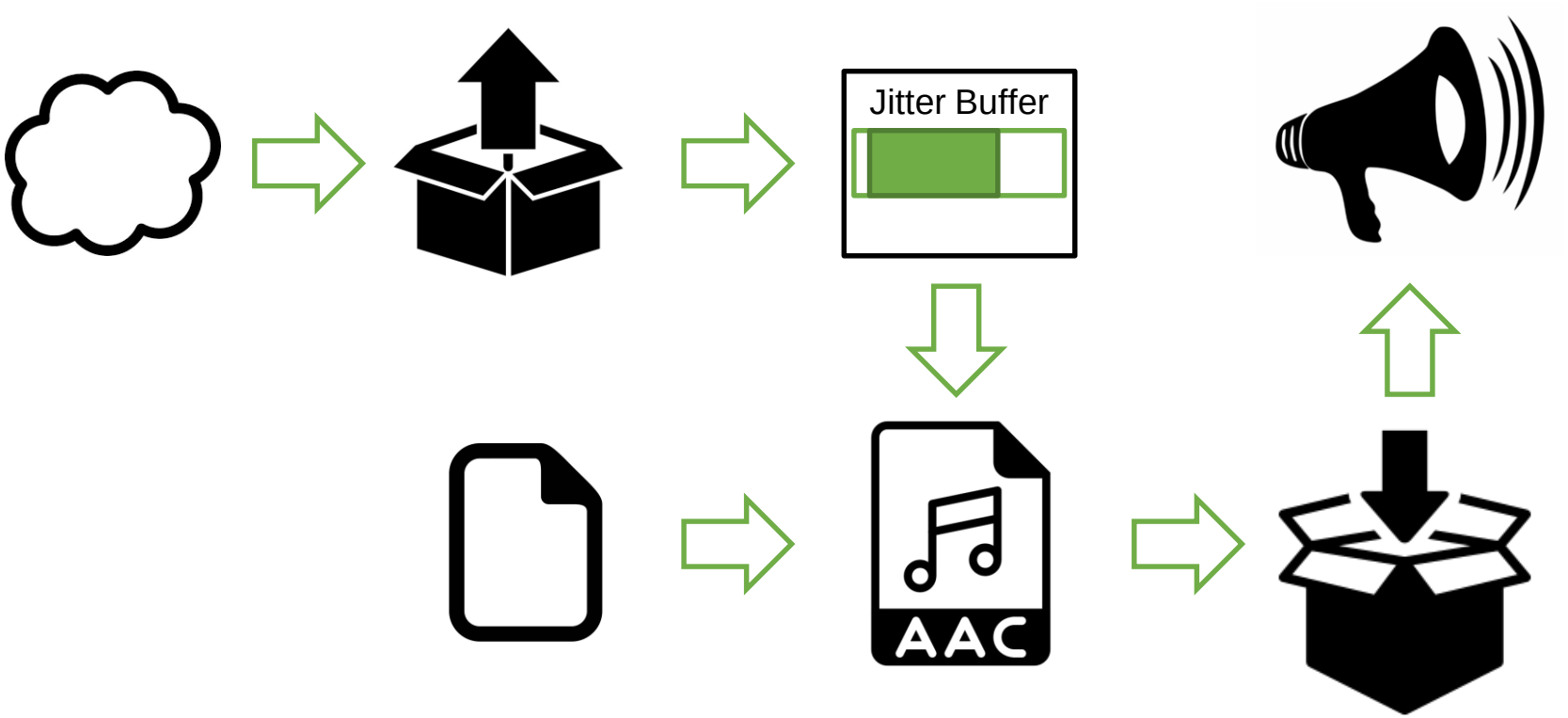
Playback difference



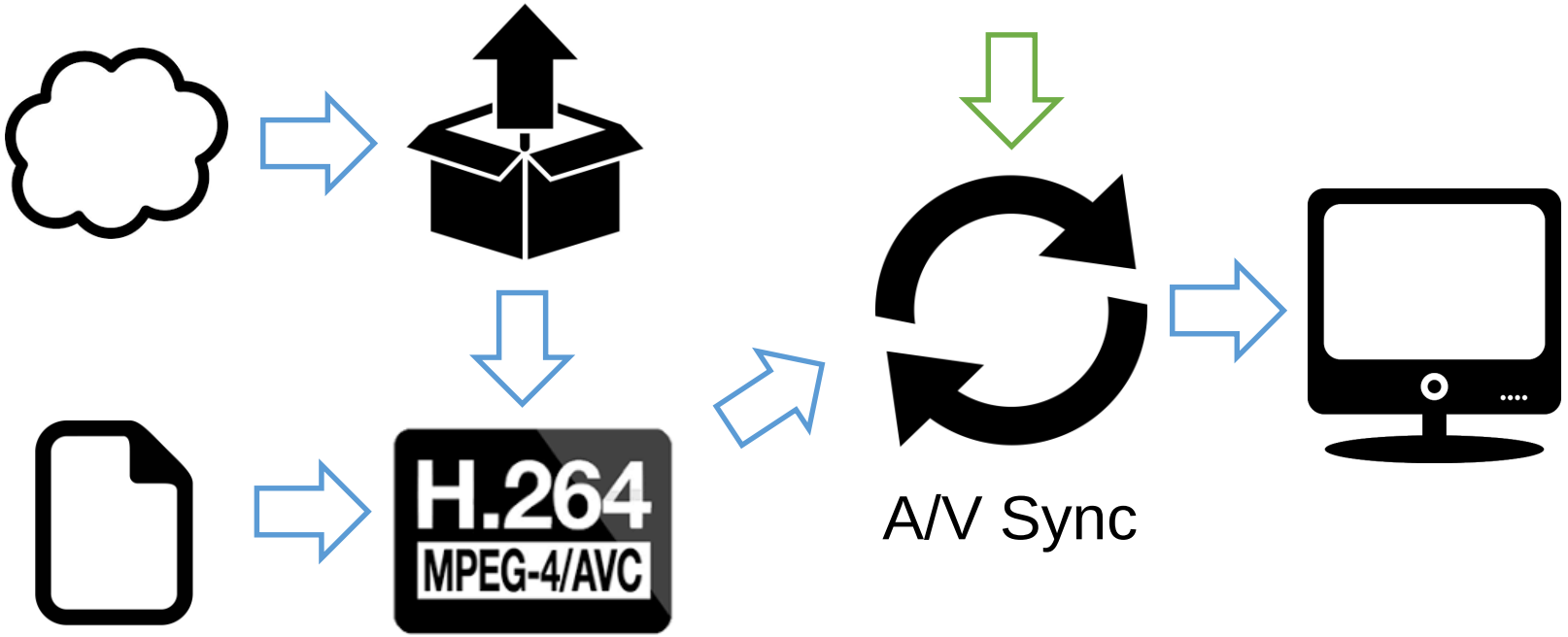
Jitter Buffer



Player Delays - Audio



Player Delays - Video



A/V Sync Rules

1. The only way to ensure sync is to carry timestamps from start to finish

A/V Sync Rules

1. The only way to ensure sync is to carry timestamps from start to finish
2. Play audio at constant rate and adjust video to it

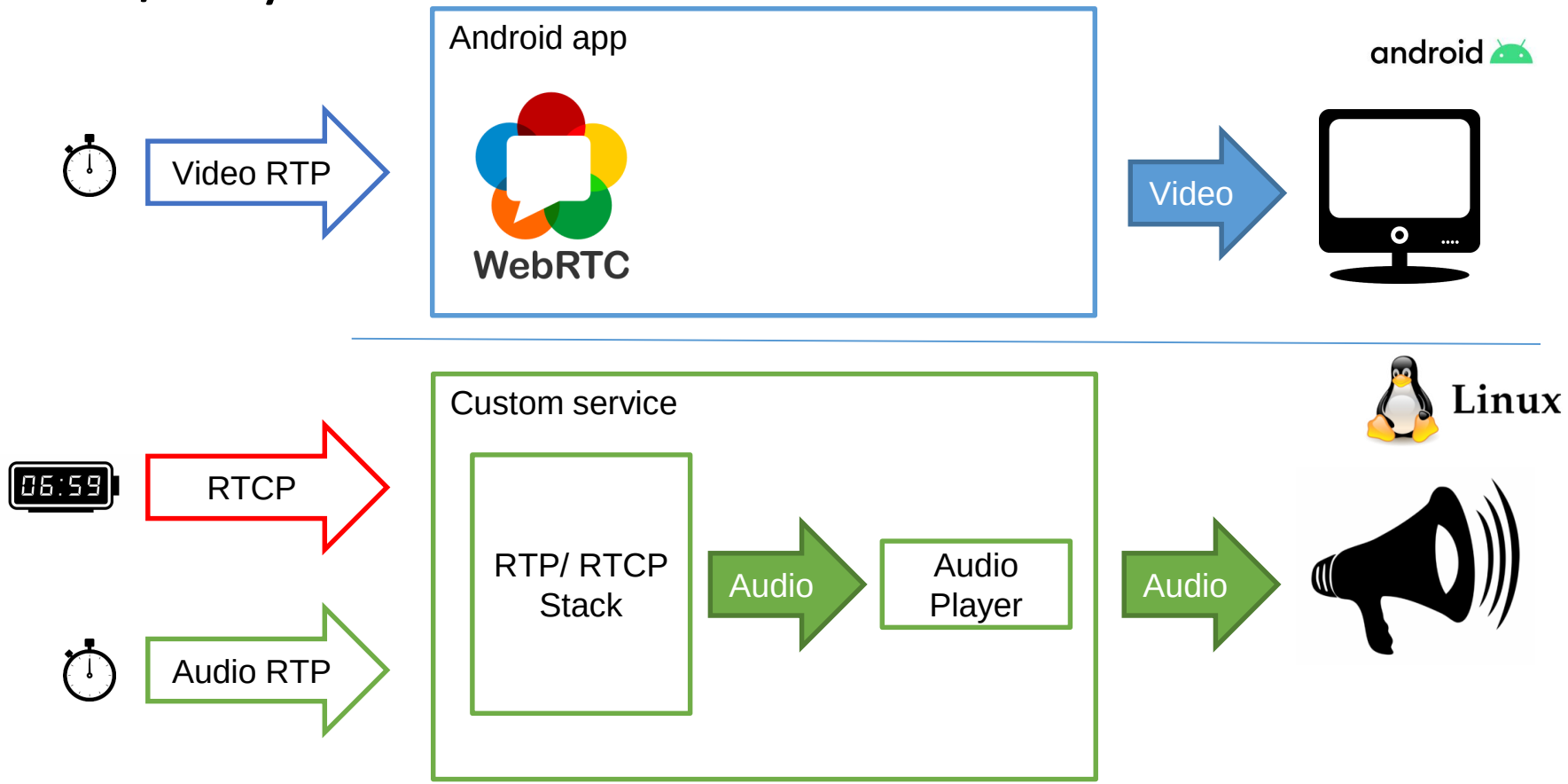
A/V Sync Rules

1. The only way to ensure sync is to carry timestamps from start to finish
2. Play audio at constant rate and adjust video to it
3. If audio packets come before video packets the whole audio stream needs to be delayed

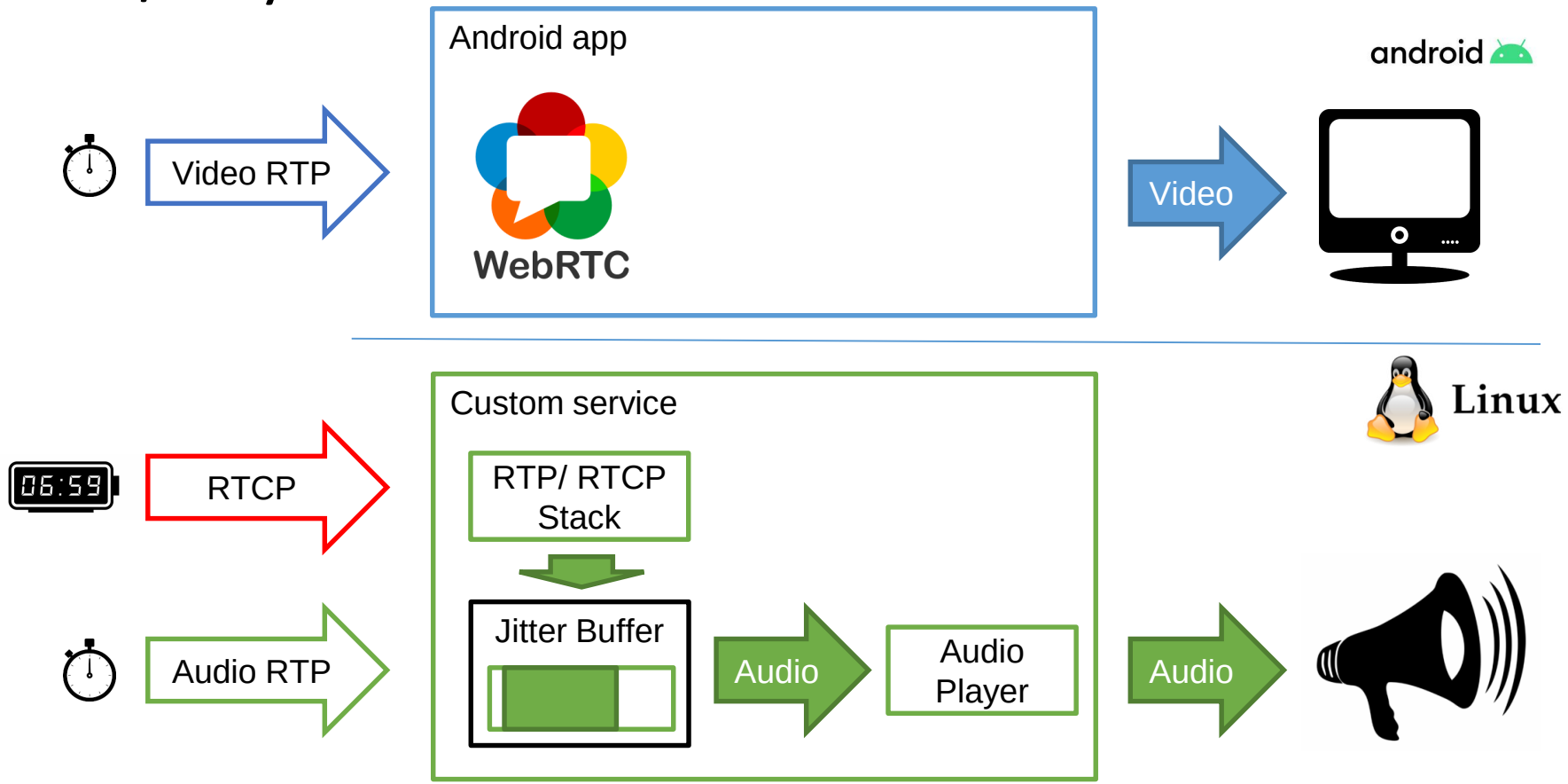
A/V Sync Rules

1. The only way to ensure sync is to carry timestamps from start to finish
2. Play audio at constant rate and adjust video to it
3. If audio packets come before video packets the whole audio stream needs to be delayed
4. Make sure that timestamps for Audio and Video streams are coming from the same clock

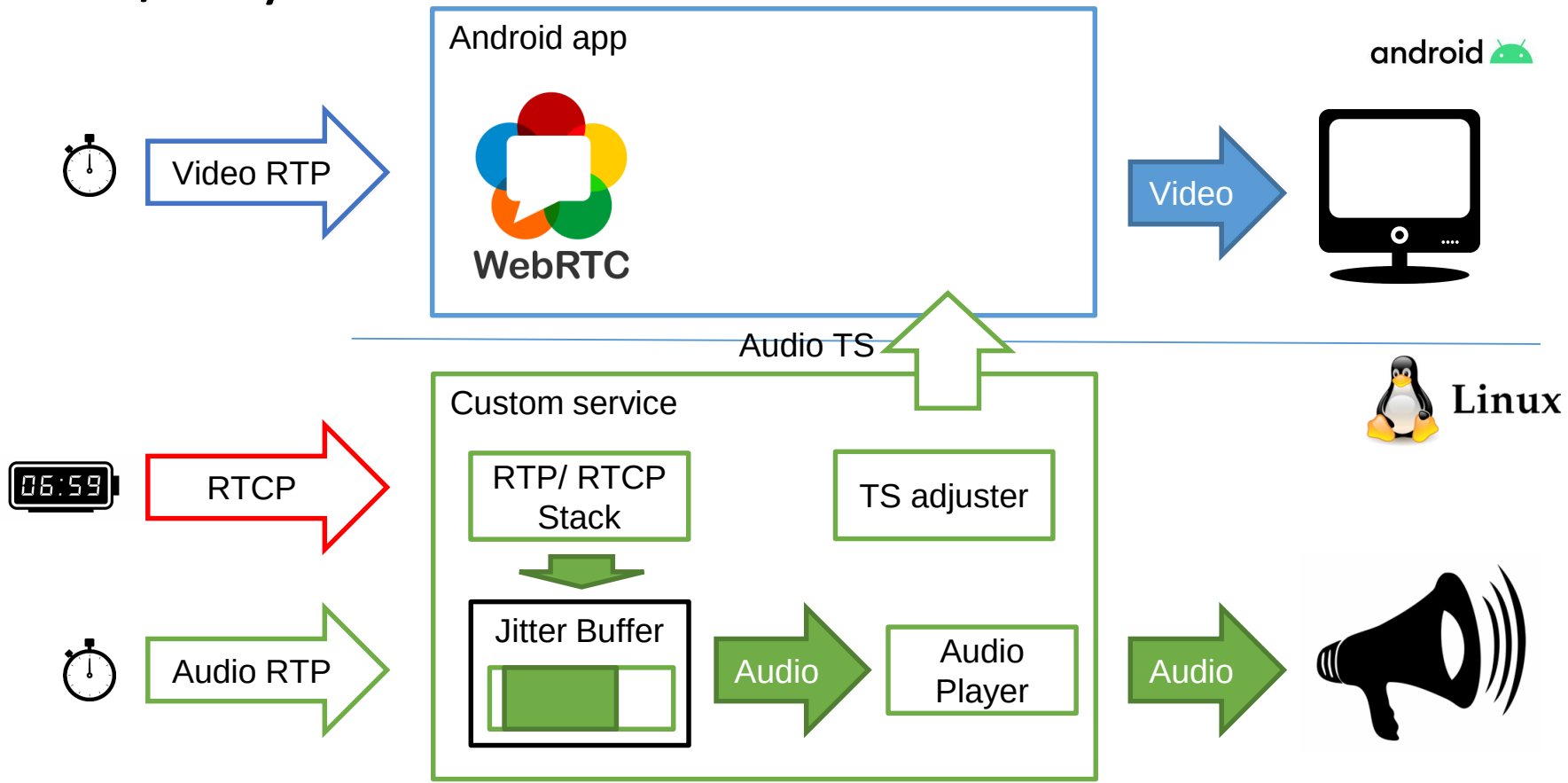
A/V Sync Architecture



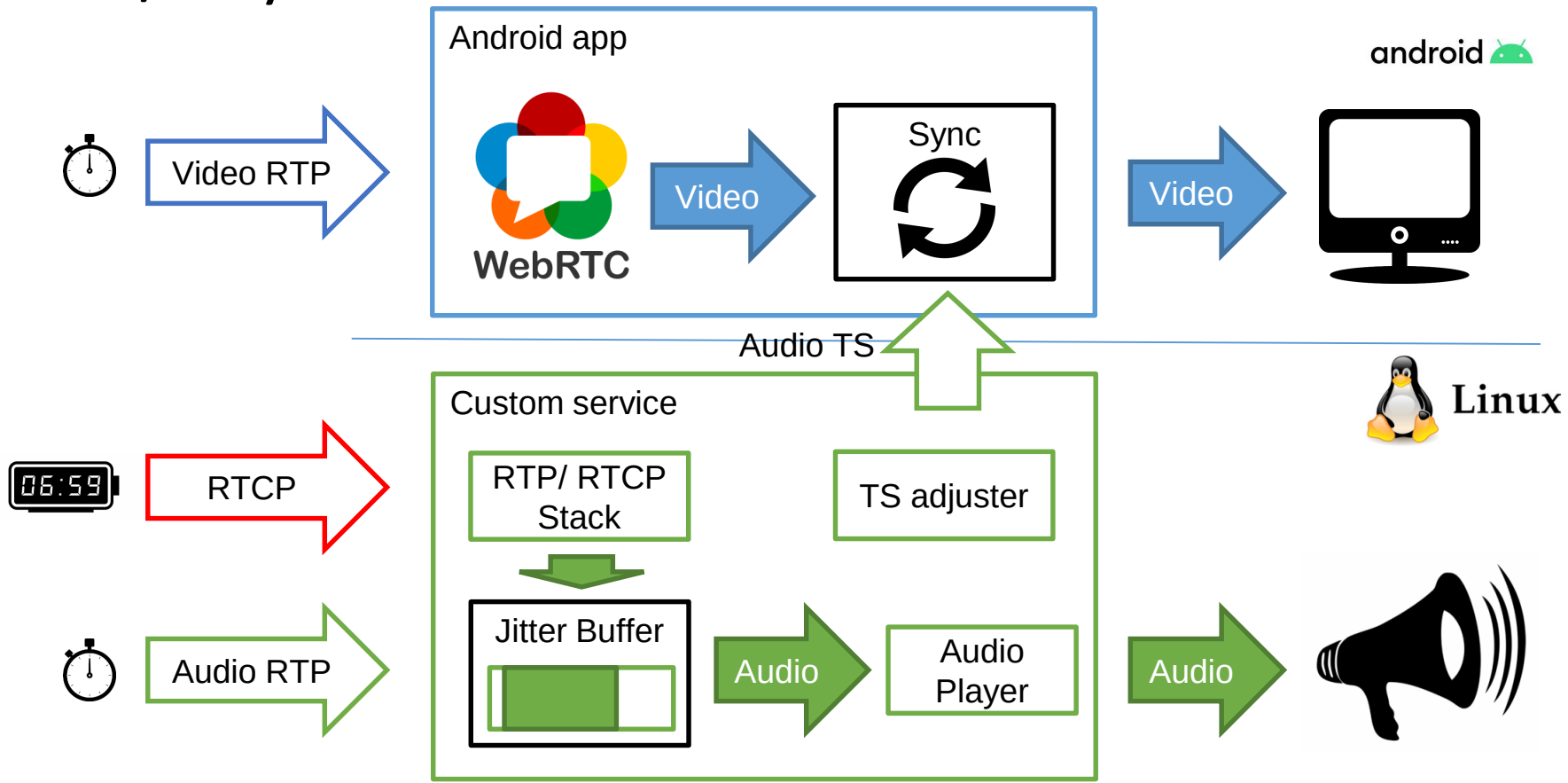
A/V Sync Architecture



A/V Sync Architecture



A/V Sync Architecture



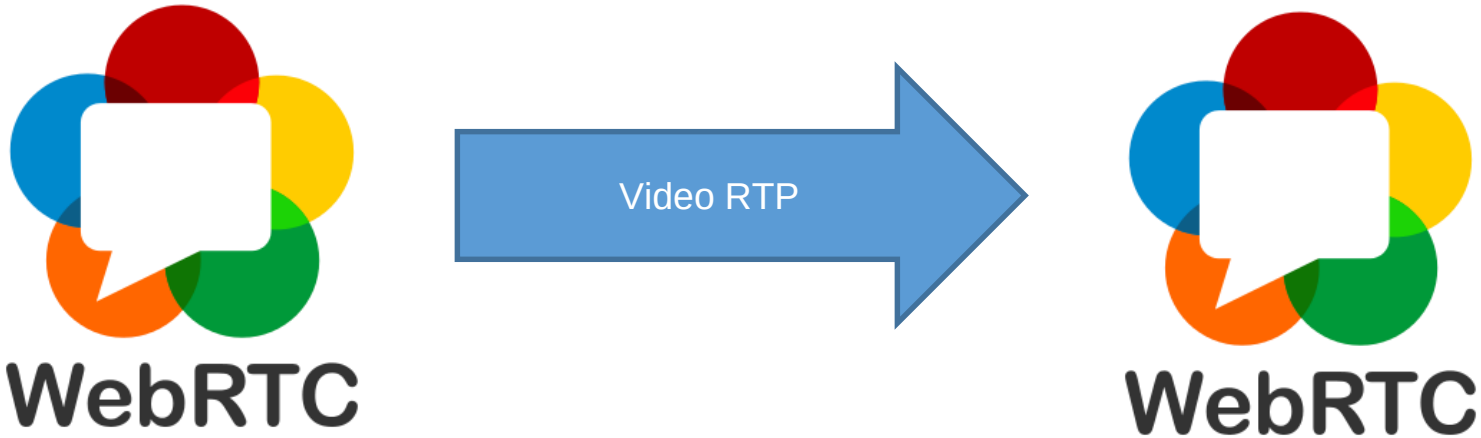
WebRTC

WebRTC captures video

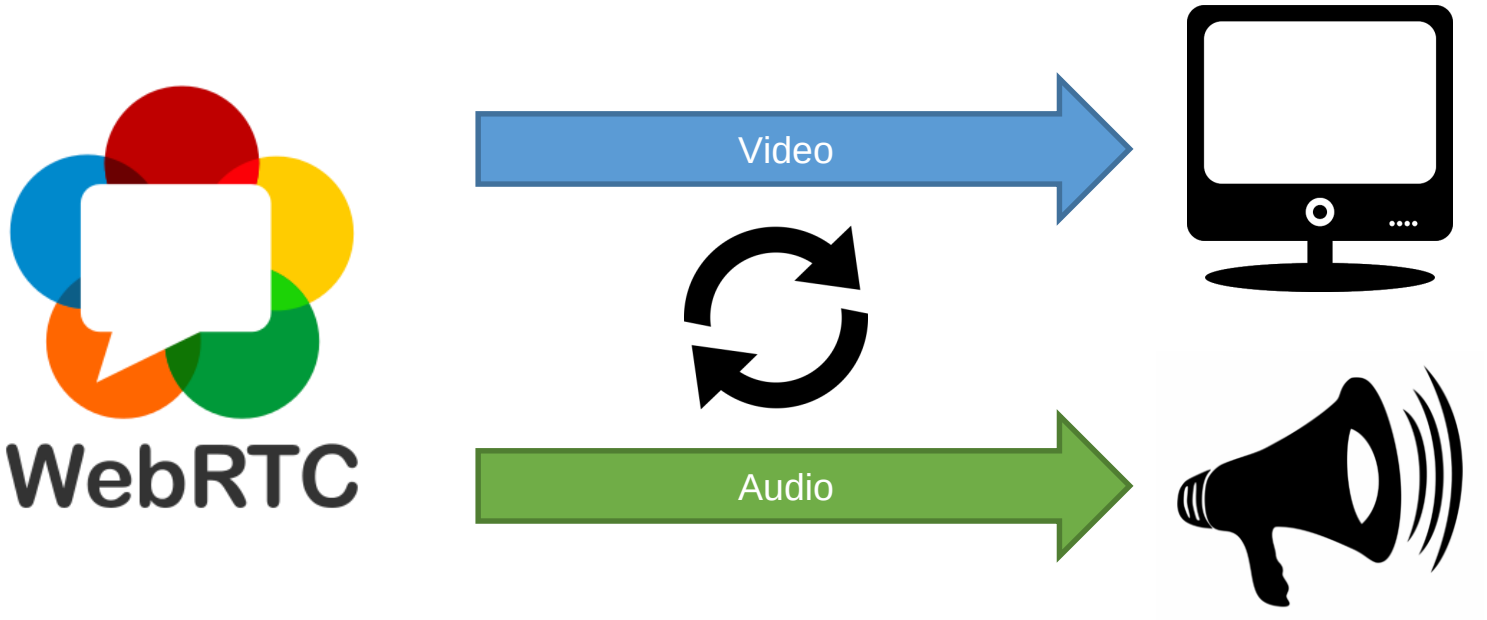
MediaRecorder

WebRTC

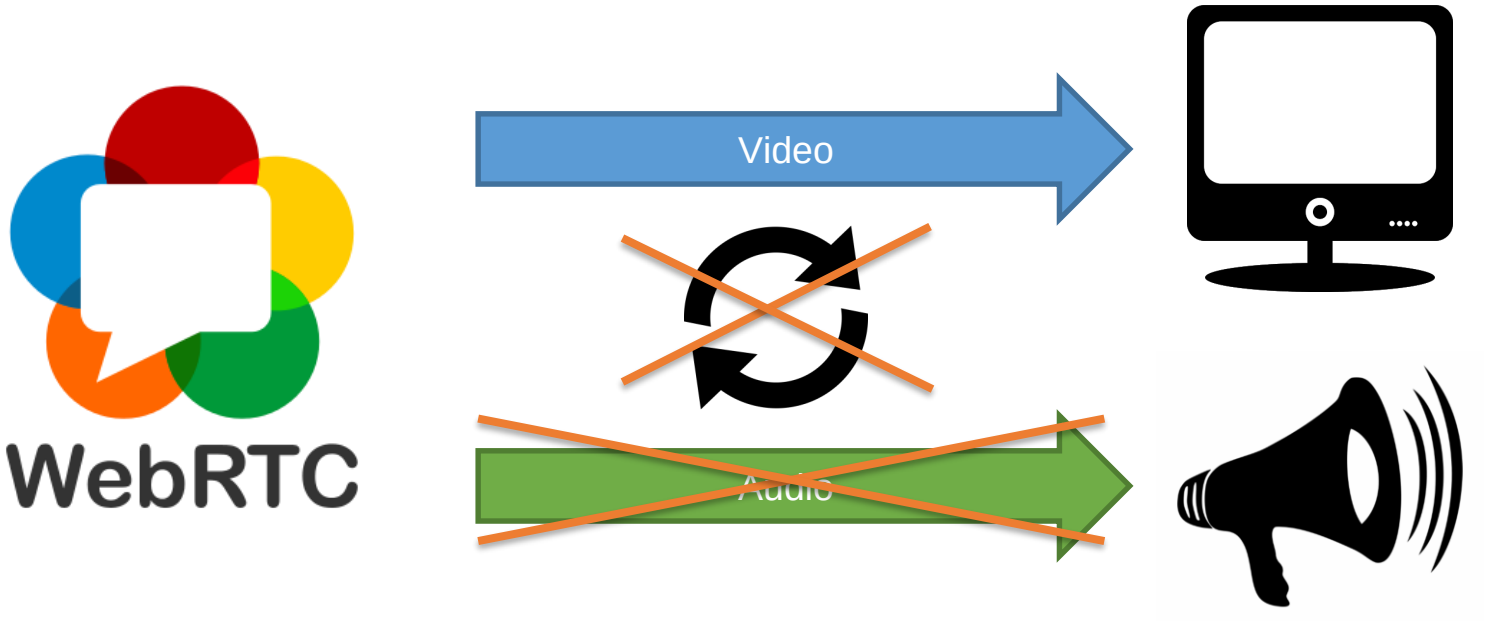
WebRTC transfers video



WebRTC can A/V Sync…



… only if it handles both audio and video



MediaSync

MediaSync

- Can be used to synchronously play audio and video streams
- Can be used to play audio-only or video-only stream
- Works like this:
  1. Take frame from buffer
  2. Adjust frame timestamp
  3. Send it for Playback/Rendering

MediaSync

```
MediaSync sync = new MediaSync();

sync.setSurface(outputSurface); // Output Surface
Surface inputSurface = sync.createInputSurface();  // Input Surface
mVideoDecoder.configure(mediaFormat, inputSurface, null, 0);

sync.setAudioTrack(audioTrack); //Set Audio Track for playback
mAudioDecoder.setCallback(new MediaCodec.Callback() {
    …
    @Override
    public void onOutputBufferAvailable(MediaCodec mediaCodec, int i, MediaCodec.BufferInfo bufferInfo) {
        …
        mMediaSync.queueAudio(copyBuffer, i, bufferInfo.presentationTimeUs); // Queue decoded audio for playback
    }
}

sync.setSyncParams(…);
sync.setPlaybackParams(new PlaybackParams().setSpeed(1.0f));
```

MediaSync

```
MediaSync sync = new MediaSync();

sync.setSurface(outputSurface); // Output Surface
Surface inputSurface = sync.createInputSurface();  // Input Surface
mVideoDecoder.configure(mediaFormat, inputSurface, null, 0);

sync.setAudioTrack(audioTrack); //Set Audio Track for playback
mAudioDecoder.setCallback(new MediaCodec.Callback() {
    …
    @Override
    public void onOutputBufferAvailable(MediaCodec mediaCodec, int i, MediaCodec.BufferInfo bufferInfo) {
        …
        mMediaSync.queueAudio(copyBuffer, i, bufferInfo.presentationTimeUs); // Queue decoded audio for playback
    }
}

sync.setSyncParams(…);
sync.setPlaybackParams(new PlaybackParams().setSpeed(1.0f));
```

MediaSync

```
MediaSync sync = new MediaSync();

sync.setSurface(outputSurface); // Output Surface
Surface inputSurface = sync.createInputSurface();  // Input Surface
mVideoDecoder.configure(mediaFormat, inputSurface, null, 0);

sync.setAudioTrack(audioTrack); //Set Audio Track for playback
mAudioDecoder.setCallback(new MediaCodec.Callback() {
    …
    @Override
    public void onOutputBufferAvailable(MediaCodec mediaCodec, int i, MediaCodec.BufferInfo bufferInfo) {
        …
        mMediaSync.queueAudio(copyBuffer, i, bufferInfo.presentationTimeUs); // Queue decoded audio for playback
    }
}

sync.setSyncParams(…);
sync.setPlaybackParams(new PlaybackParams().setSpeed(1.0f));
```

MediaSync

```
MediaSync sync = new MediaSync();

sync.setSurface(outputSurface); // Output Surface
Surface inputSurface = sync.createInputSurface();  // Input Surface
mVideoDecoder.configure(mediaFormat, inputSurface, null, 0);

sync.setAudioTrack(audioTrack); //Set Audio Track for playback
mAudioDecoder.setCallback(new MediaCodec.Callback() {
    …
    @Override
    public void onOutputBufferAvailable(MediaCodec mediaCodec, int i, MediaCodec.BufferInfo bufferInfo) {
        …
        mMediaSync.queueAudio(copyBuffer, i, bufferInfo.presentationTimeUs); // Queue decoded audio for playback
    }
}

sync.setSyncParams(…);
sync.setPlaybackParams(new PlaybackParams().setSpeed(1.0f));
```

MediaSync

```
MediaSync sync = new MediaSync();

sync.setSurface(outputSurface); // Output Surface
Surface inputSurface = sync.createInputSurface();  // Input Surface
mVideoDecoder.configure(mediaFormat, inputSurface, null, 0);

sync.setAudioTrack(audioTrack); //Set Audio Track for playback
mAudioDecoder.setCallback(new MediaCodec.Callback() {
    …
    @Override
    public void onOutputBufferAvailable(MediaCodec mediaCodec, int i, MediaCodec.BufferInfo bufferInfo) {
        …
        mMediaSync.queueAudio(copyBuffer, i, bufferInfo.presentationTimeUs); // Queue decoded audio for playback
    }
}

sync.setSyncParams(…);
sync.setPlaybackParams(new PlaybackParams().setSpeed(1.0f));
```

MediaSync

```
MediaSync sync = new MediaSync();

sync.setSurface(outputSurface); // Output Surface
Surface inputSurface = sync.createInputSurface();  // Input Surface
mVideoDecoder.configure(mediaFormat, inputSurface, null, 0);

sync.setAudioTrack(audioTrack); //Set Audio Track for playback
mAudioDecoder.setCallback(new MediaCodec.Callback() {
    …
    @Override
    public void onOutputBufferAvailable(MediaCodec mediaCodec, int i, MediaCodec.BufferInfo bufferInfo) {
        …
        mMediaSync.queueAudio(copyBuffer, i, bufferInfo.presentationTimeUs); // Queue decoded audio for playback
    }
}
```

sync.setSyncParams(…);

```
sync.setPlaybackParams(new PlaybackParams().setSpeed(1.0f));
```

Sync Source for Video

- *SYNC_SOURCE_AUDIO*
  uses *AudioTrack.getTimestamp()*

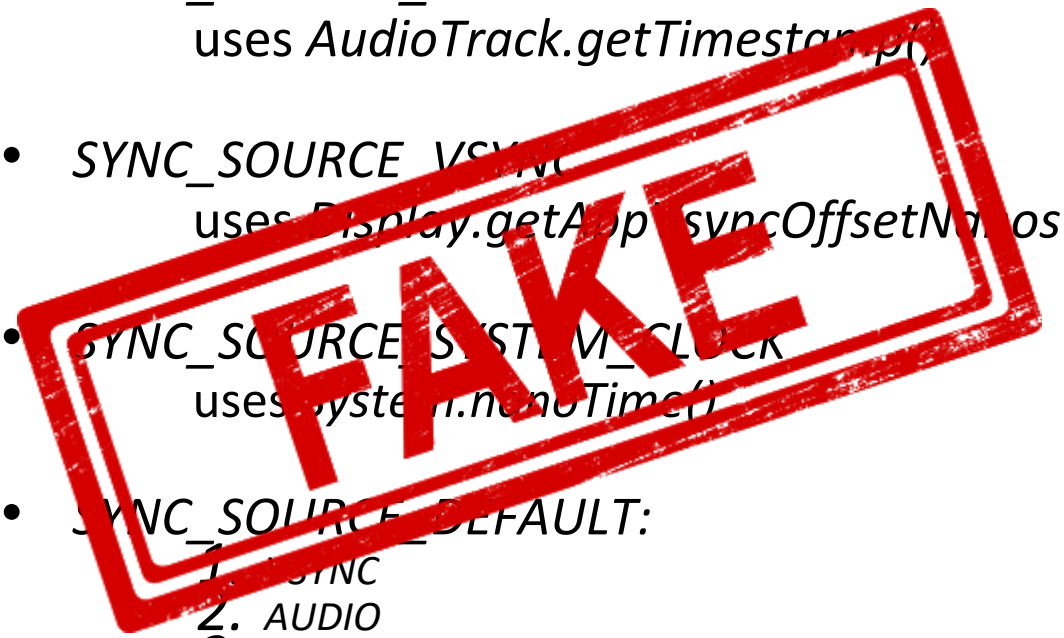
- *SYNC_SOURCE_VSYNC*
  uses *Display.getAppVsyncOffsetNanos()*

- *SYNC_SOURCE_SYSTEM_CLOCK*
  uses *System.nanoTime()*

- *SYNC_SOURCE_DEFAULT:*
  1. *VSYNC*
  2. *AUDIO*
  3. *SYSTEM_CLOCK*

Sync Source for Video

- *SYNC_SOURCE_AUDIO*
  uses *AudioTrack.getTimestamp()*
- *SYNC_SOURCE_VSYNC*
  uses *Display.getAppVsyncOffsetNanos()*
- *SYNC_SOURCE_SYSTEM_CLOCK*
  uses *System.nanoTime()*
- *SYNC_SOURCE_DEFAULT:*
  1. *VSYNC*
  2. *AUDIO*
  3. *SYSTEM_CLOCK*



How MediaSync adjusts timestamps

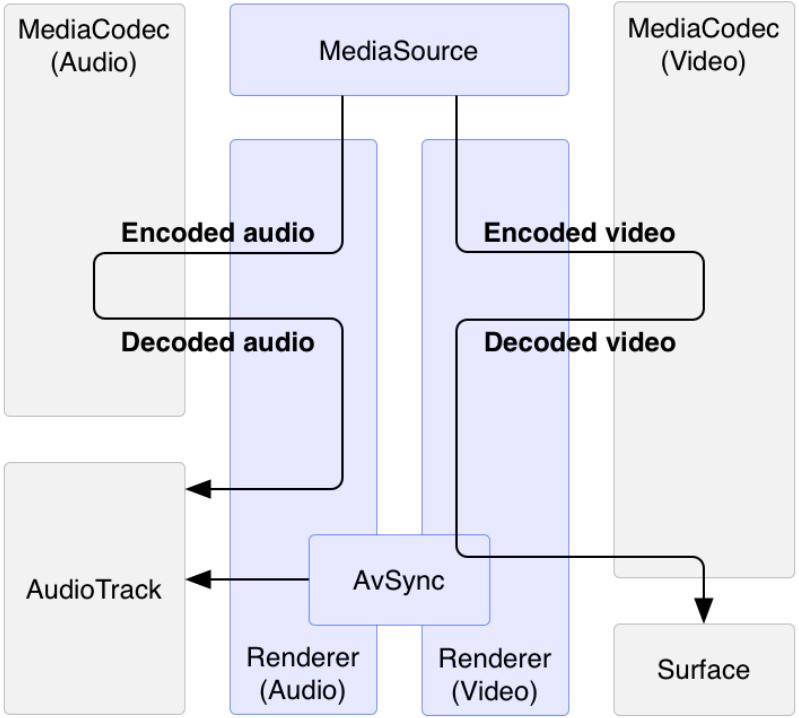
1. Sync source is ignored
2. System monolithic clock is used to get “now” time
3. Video frame presentation time is calculated based on “now” time and VSYNC
4. If audio is present and video is more than 40ms behind – skip video frame to catch up

Conclusion

- MediaSync is not configurable for video playback
- MediaSync only helps with adjusting to VSYNC really, if you only have one stream
- We can MediaSync for video playback, but it does not solve our problem

ExoPlayer

A/V Sync in ExoPlayer

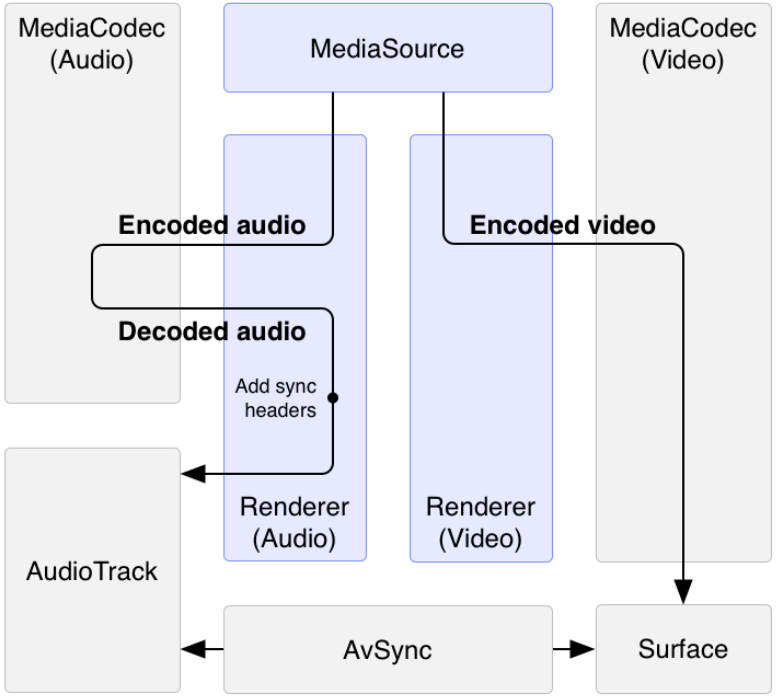


- ExoPlayer implements A/V Sync using standard Android APIs: *AudioTrack* & *MediaCodec*
- Audio and Video from *MediaSource* must carry synced timestamps for A/V Sync to work

A/V Sync in ExoPlayer

```
...
audioTrack.getTimestamp(audioTimestamp);//Get audio timestamp from AudioTrack
...

//Some very complicated logic to adjust this timestamp

...
protected void renderOutputBufferV21(
    MediaCodec codec, int index, long presentationTimeUs, long releaseTimeNs) {
  ...
  //Using adjusted timestamp to render video frame
  codec.releaseOutputBuffer(index, releaseTimeNs);
  ...
}
```

Multimedia tunneling in ExoPlayer



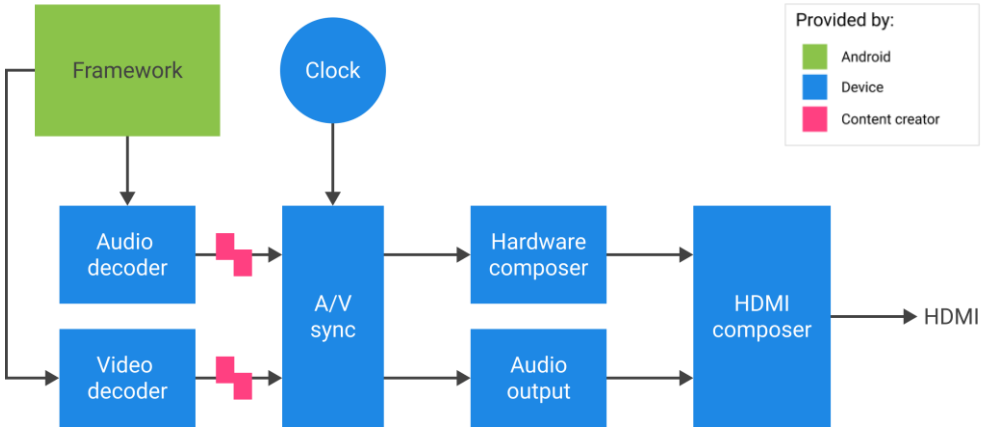
- ExoPlayer supports Multimedia tunneling
- This is more efficient and it offloads A/V Sync to the underlying platform

Conclusion

- By default ExoPlayer implements it’s own complex A/V sync logic, but it only handles playback
- ExoPlayer also supports multimedia tunneling (if device can provided this functionality)
- In either case it only works if Audio and Video in MediaSource already synced somehow
- We could’ve used ExoPlayer, but decided against it:
  1. Only works for playback
  2. Complex sync logic too hard for us to understand and adjust

Multimedia Tunneling

Multimedia Tunneling



- This is an optional feature that is present on some devices (mostly Android TV)
- Usually not all codecs support it
- Useful feature if you want to play video in 4K

Multimedia Tunneling

```
SurfaceView sv = new SurfaceView(mContext);

AudioManager am = mContext.getSystemService(AUDIO_SERVICE);
int audioSessionId = am.generateAudioSessionId()

AudioAttributes.Builder aab = new AudioAttributes.Builder();
aab.setUsage(AudioAttributes.USAGE_MEDIA);
aab.setContentType(AudioAttributes.CONTENT_TYPE_MOVIE);
aab.setFlag(AudioAttributes.FLAG_HW_AV_SYNC);

AudioAttributes aa = aab.build();
AudioTrack at = new AudioTrack(aa);
```

Multimedia Tunneling

```
SurfaceView sv = new SurfaceView(mContext);

AudioManager am = mContext.getSystemService(AUDIO_SERVICE);
int audioSessionId = am.generateAudioSessionId()

AudioAttributes.Builder aab = new AudioAttributes.Builder();
aab.setUsage(AudioAttributes.USAGE_MEDIA);
aab.setContentType(AudioAttributes.CONTENT_TYPE_MOVIE);
aab.setFlag(AudioAttributes.FLAG_HW_AV_SYNC);

AudioAttributes aa = aab.build();
AudioTrack at = new AudioTrack(aa);
```

Multimedia Tunneling

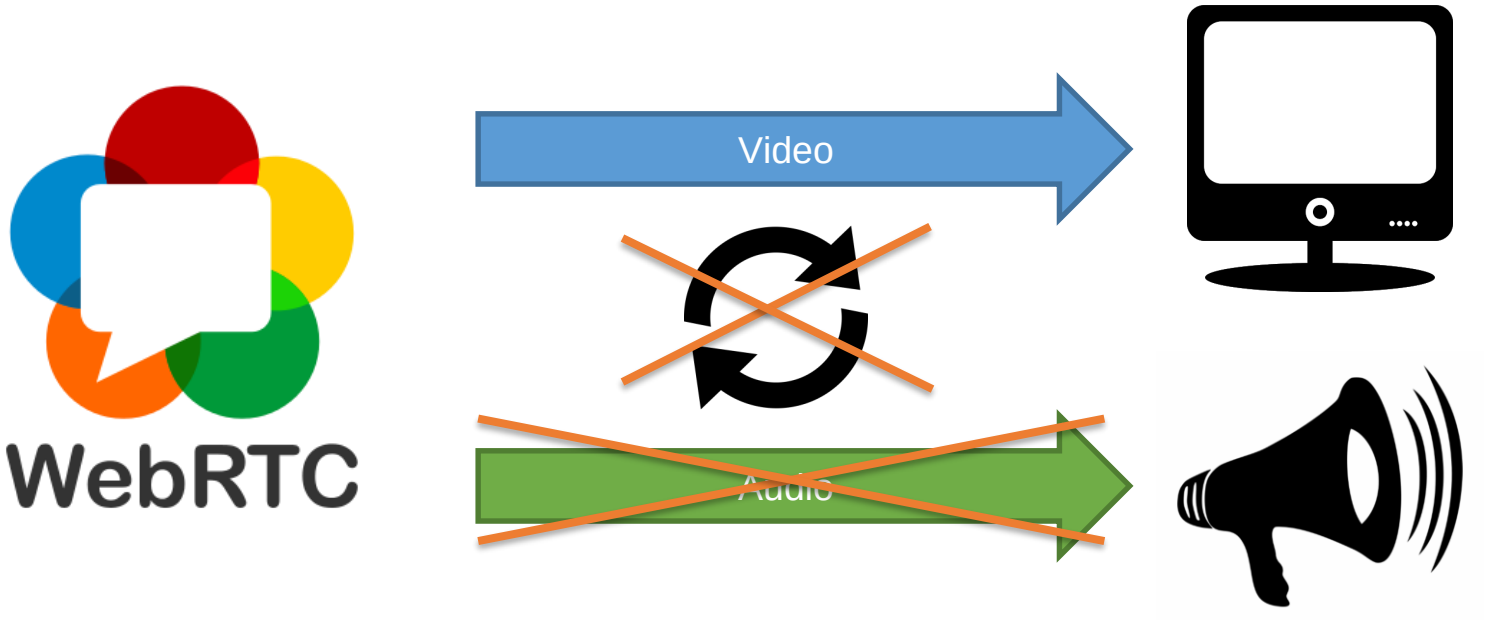
```
// retrieve codec with tunneled video playback feature
MediaFormat mf = MediaFormat.createVideoFormat(“video/hevc”, 3840, 2160);
mf.setFeatureEnabled(CodecCapabilities.FEATURE_TunneledPlayback, true);
MediaCodecList mcl = new MediaCodecList(MediaCodecList.ALL_CODECS);
String codecName = mcl.findDecoderForFormat(mf);

// create codec and configure it
mf.setInteger(MediaFormat.KEY_AUDIO_SESSION_ID, audioSessionId);


MediaCodec mc = MediaCodec.createCodecByName(codecName);
mc.configure(mf, sv.getSurfaceHolder().getSurface(), null, 0);
```

Multimedia Tunneling

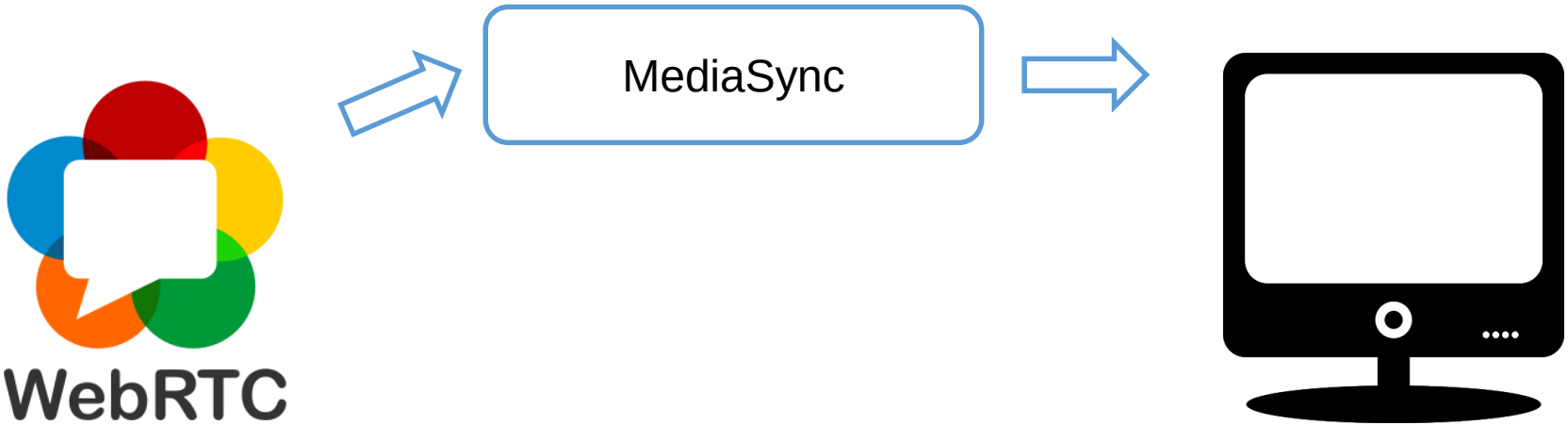
```
// retrieve codec with tunneled video playback feature
MediaFormat mf = MediaFormat.createVideoFormat("video/hevc", 3840, 2160);
mf.setFeatureEnabled(CodecCapabilities.FEATURE_TunneledPlayback, true);
MediaCodecList mcl = new MediaCodecList(MediaCodecList.ALL_CODECS);
String codecName = mcl.findDecoderForFormat(mf);

// create codec and configure it
mf.setInteger(MediaFormat.KEY_AUDIO_SESSION_ID, audioSessionId);


MediaCodec mc = MediaCodec.createCodecByName(codecName);
mc.configure(mf, sv.getSurfaceHolder().getSurface(), null, 0);
```

WebRTC again?

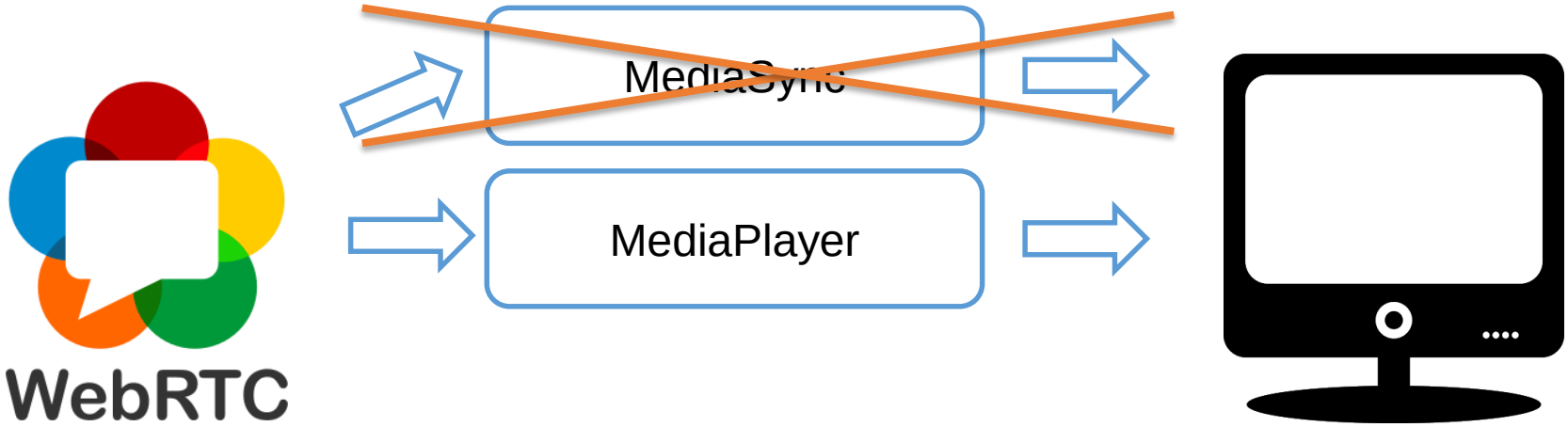
WebRTC can’t sync just video



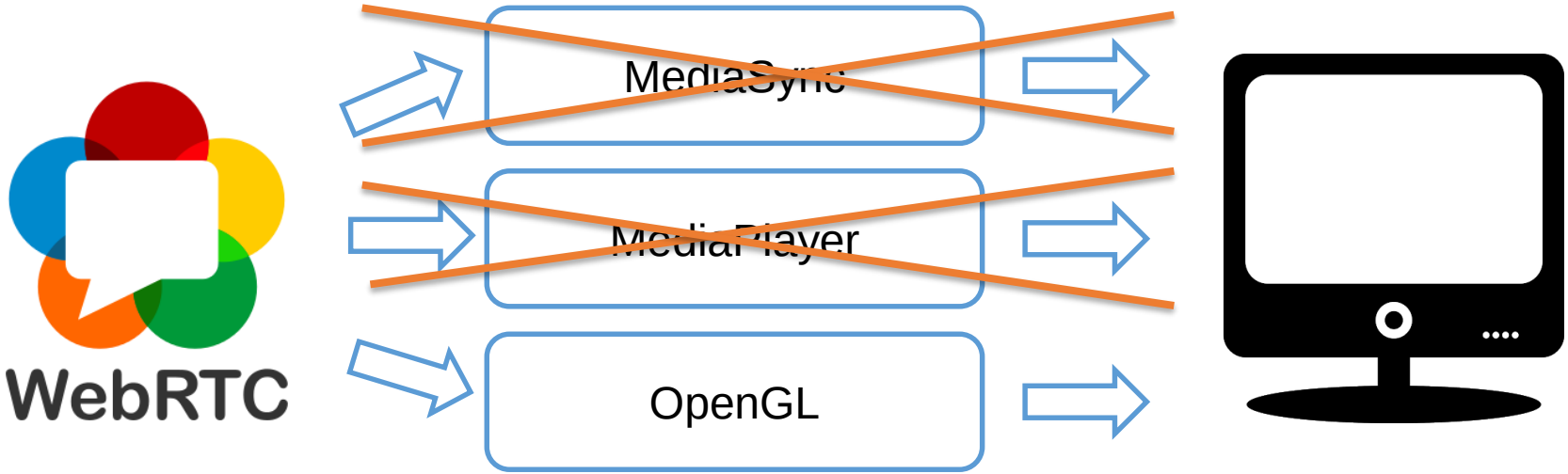
WebRTC renders video



WebRTC renders video



WebRTC renders video



WebRTC renders video with OpenGL

```
//defined in EglRenderer.java
private void renderFrameOnRenderThread() {
    ...
    eglBase.swapBuffers(frame.getTimestampNs());
```

```
//defined in EglBase14Impl.java
public void swapBuffers(long timeStampNs) {
    ...
    EGLExt.eglPresentationTimeANDROID(eglDisplay,eglSurface, timeStampNs);
    EGL14.eglSwapBuffers(eglDisplay, eglSurface);
```

WebRTC renders video with OpenGL

```
//defined in EglRenderer.java
private void renderFrameOnRenderThread() {
    ...
    eglBase.swapBuffers(frame.getTimestampNs());
```

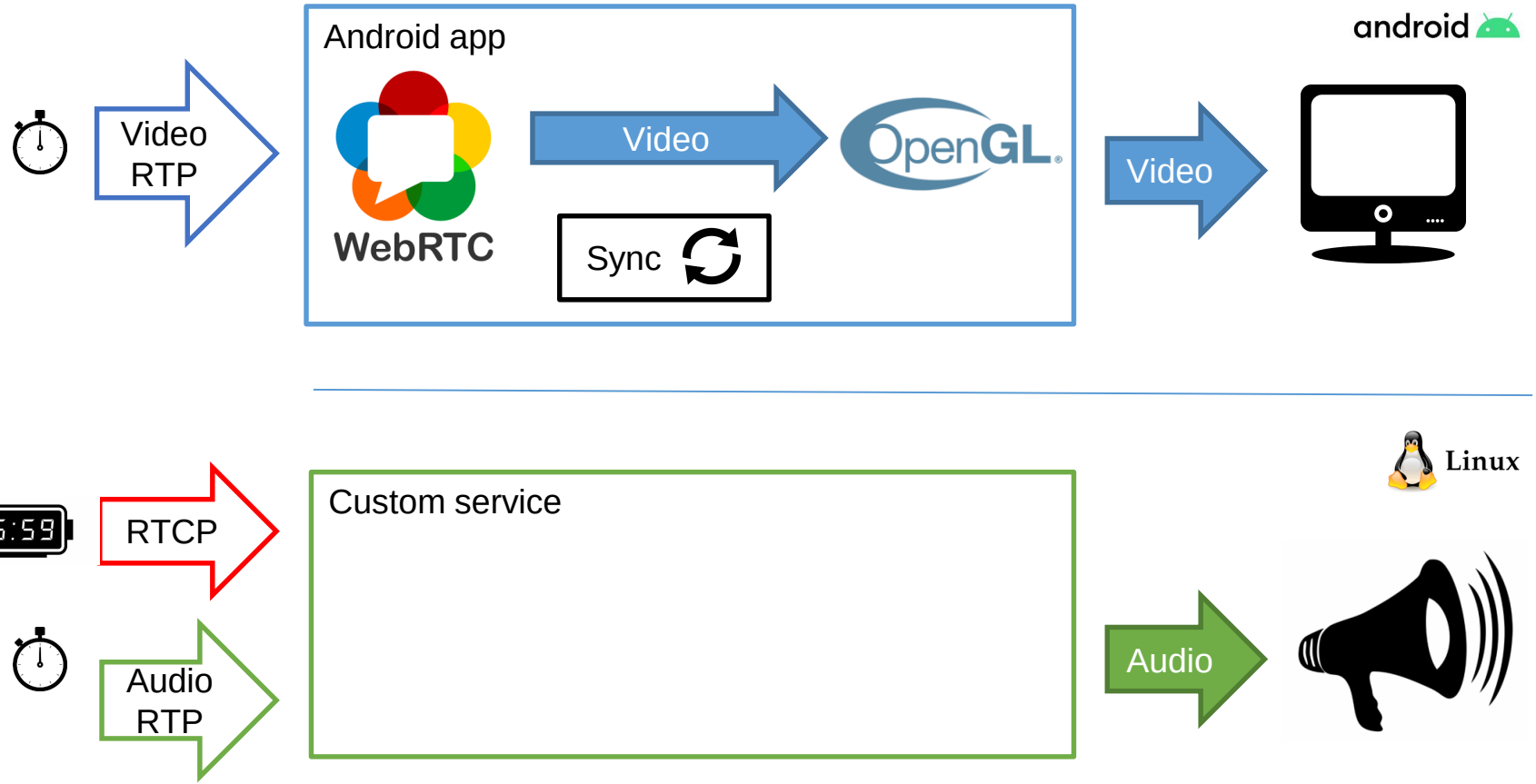
```
//defined in EglBase14Impl.java
public void swapBuffers(long timeStampNs) {
    ...
    EGLExt.eglPresentationTimeANDROID(eglDisplay,eglSurface, timeStampNs);
    EGL14.eglSwapBuffers(eglDisplay, eglSurface);
```

Conclusion

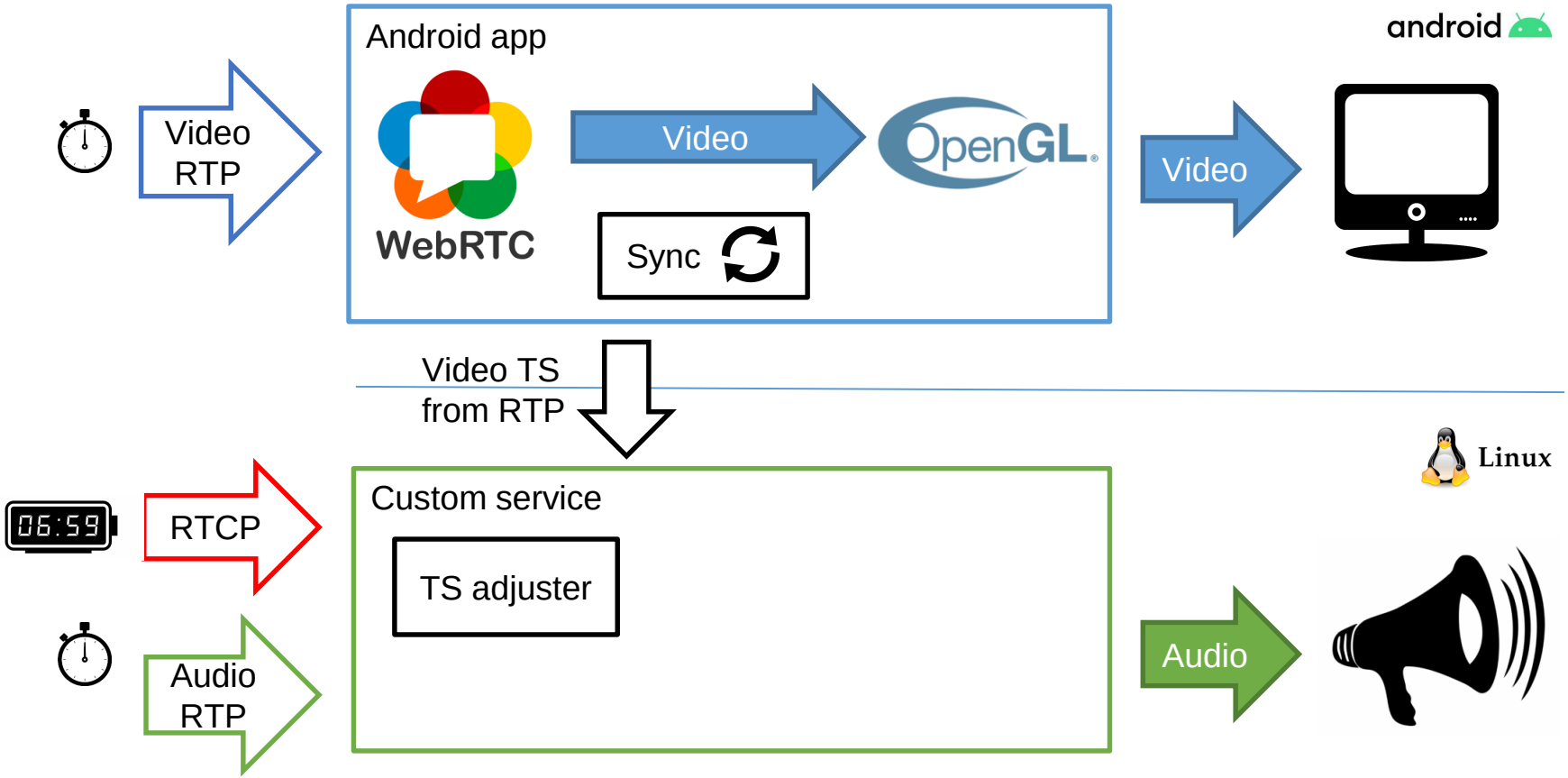
- WebRTC was the obvious choice for us, since it took care of recording video, transferring and playing it
- WebRTC has an internal A/V sync logic that we couldn’t use, since we only WebRTC for video
- Fortunately, WebRTC uses OpenGL for rendering video and it had interface for modifying video frame presentation time

Conclusion

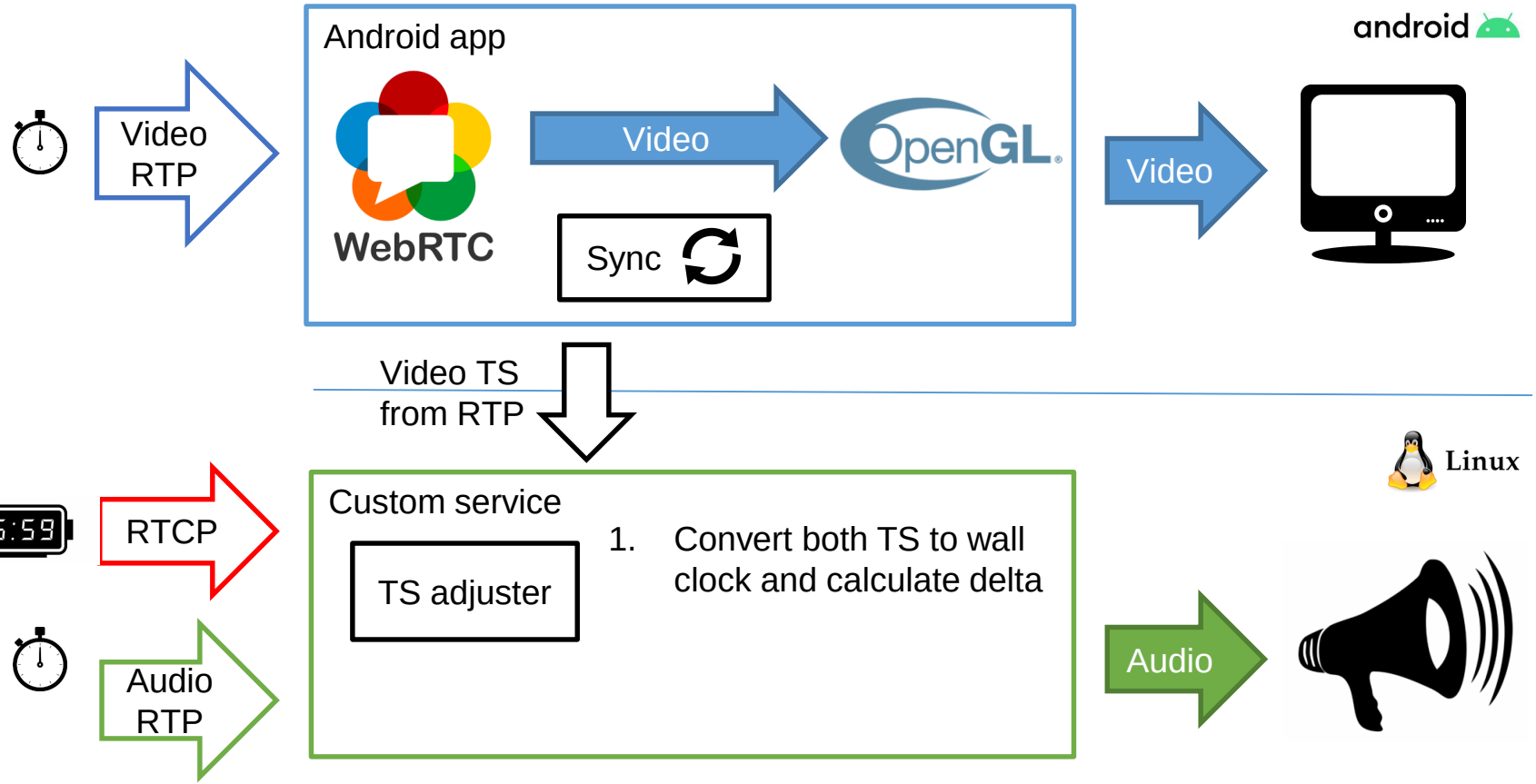
Final solution



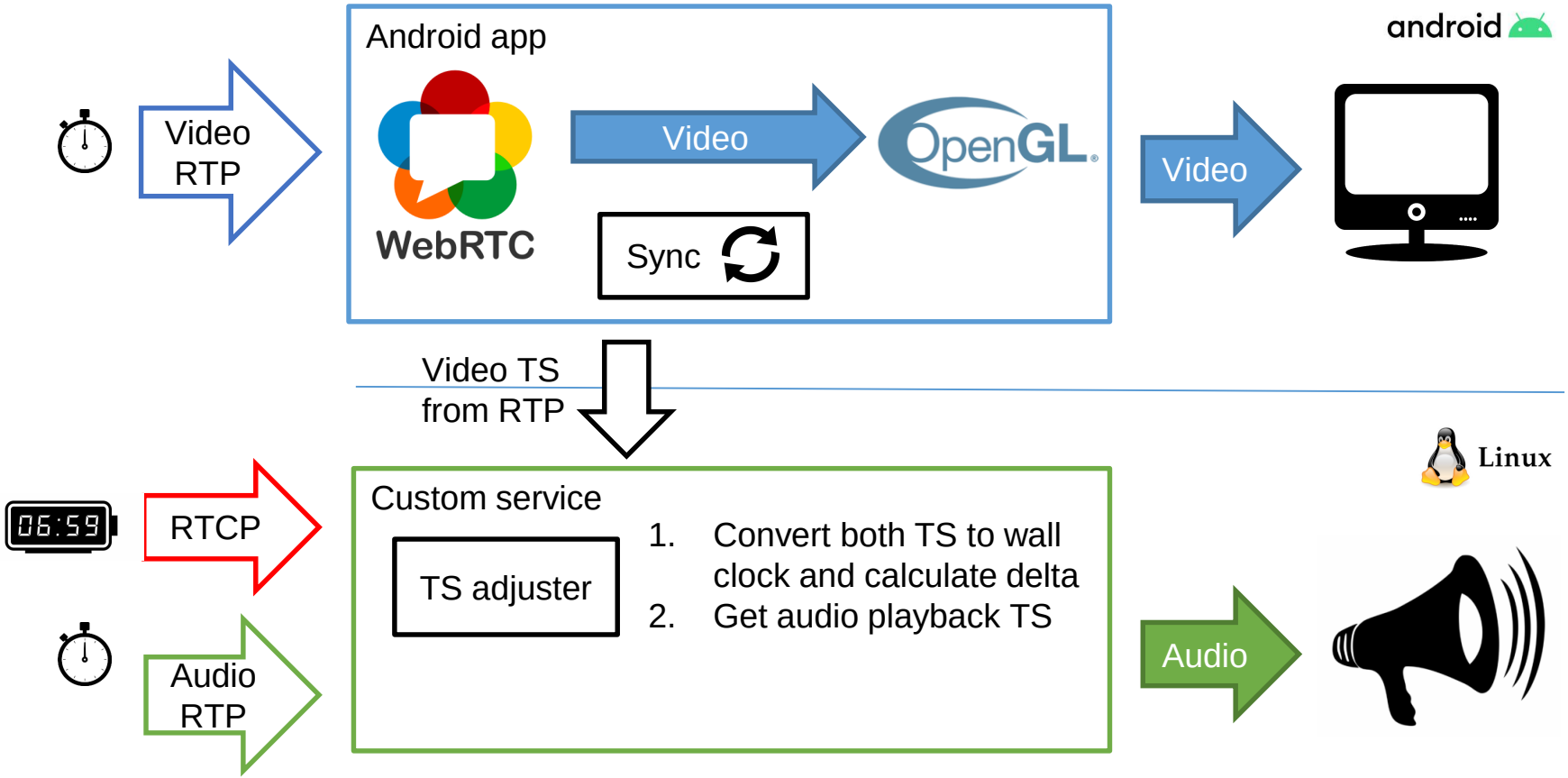
Final solution



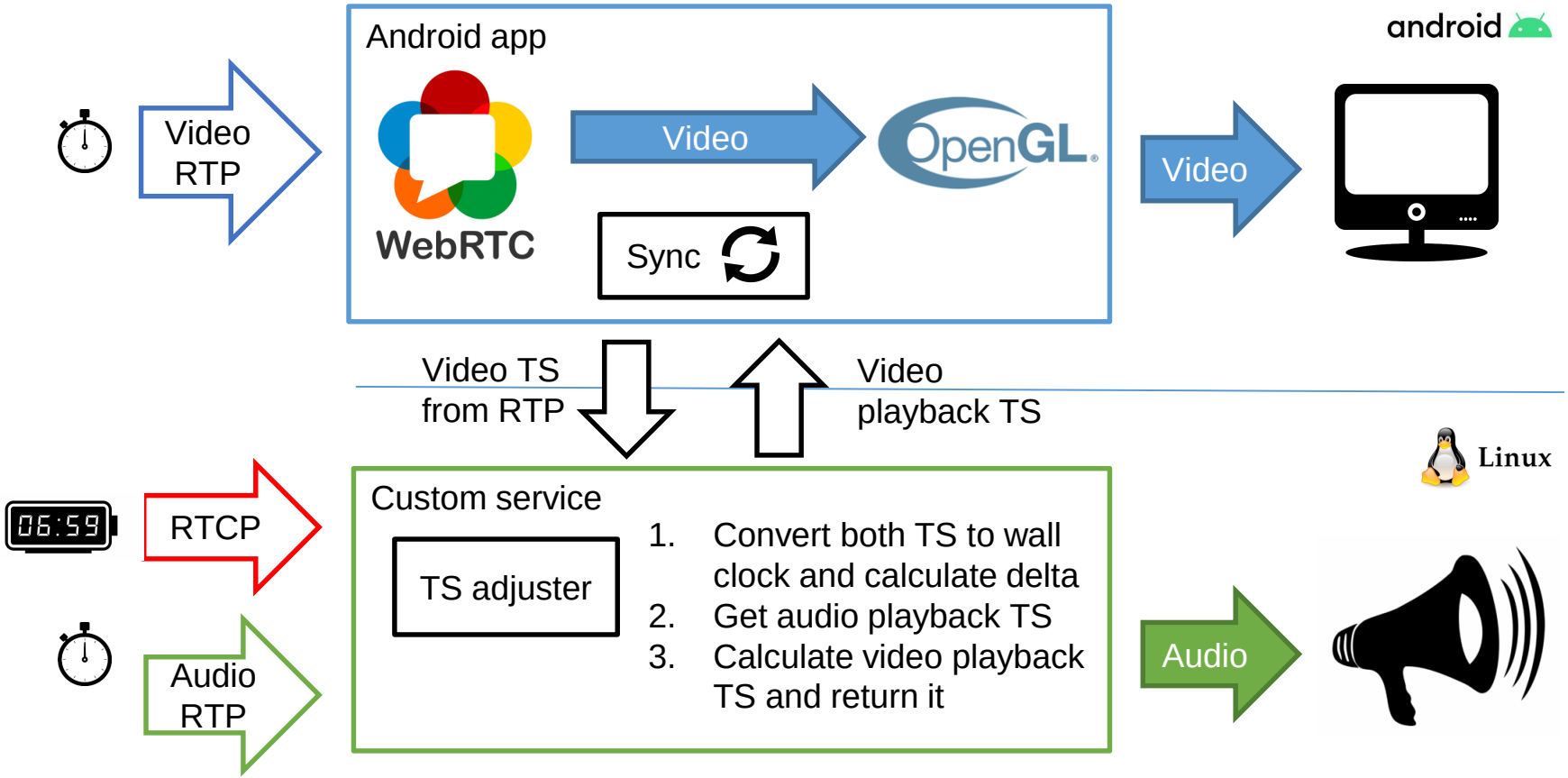
Final solution



Final solution



Final solution



Conclusion

- A/V sync is an advanced magic
- Better to use frameworks than implement it yourself
- MediaSync – can’t recommend
- ExoPlayer – excellent for playing video and supports multimedia tunneling
- WebRTC – works best if you want video calls synced

Thank You

Fedor Tcymbal

Technical Manager, CTO

fedor.tcymbal@orioninc.com

 @ftsymbal

orioninc.com