Software Participants in WebRTC Calls

Neil Dwyer, LiveKit

LiveKit

My Background In RTC

- Blab.Im
  - WebRTC (Janus) (Clubhouse with video)
- Bebo
  - GStreamer (Composition, Transcoding, Egress)
  - CV For Kills and Victories in Fortnite
- (Detour)
- Start Up
  - Unity LiveKit SDK
  - WebRTC for game state
  - Spatial Audio
  - In-Game Video
- LiveKit

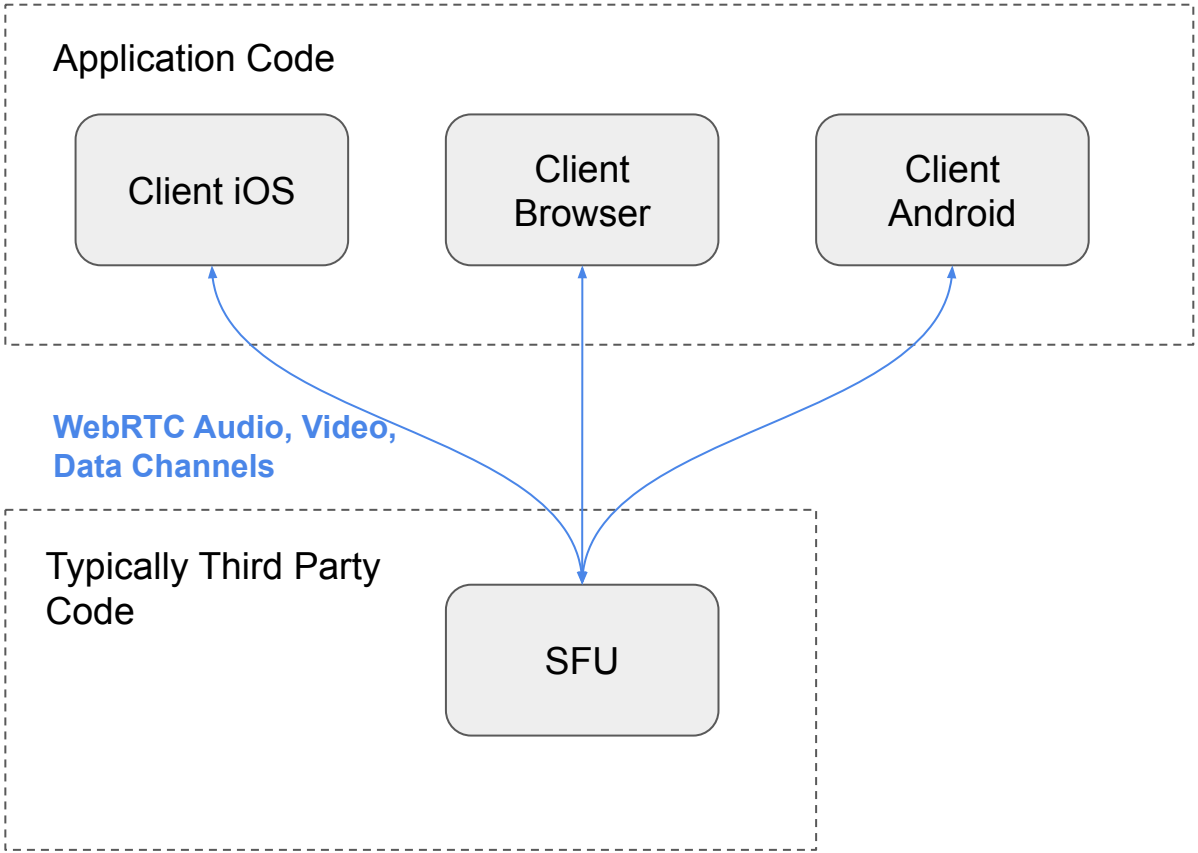
LiveKit

LiveKit

- LiveKit
  - Open Source SFU
  - Open Source Client SDKs
  - Maintain fork of libwebrtc
  - **Server-Side “Client” SDKs**
- LiveKit Cloud
  - Mesh Network of SFUs
    - Horizontally Scalable Sessions
    - Globally Distributed
    - Fault Tolerant

LiveKit

Typical WebRTC Session



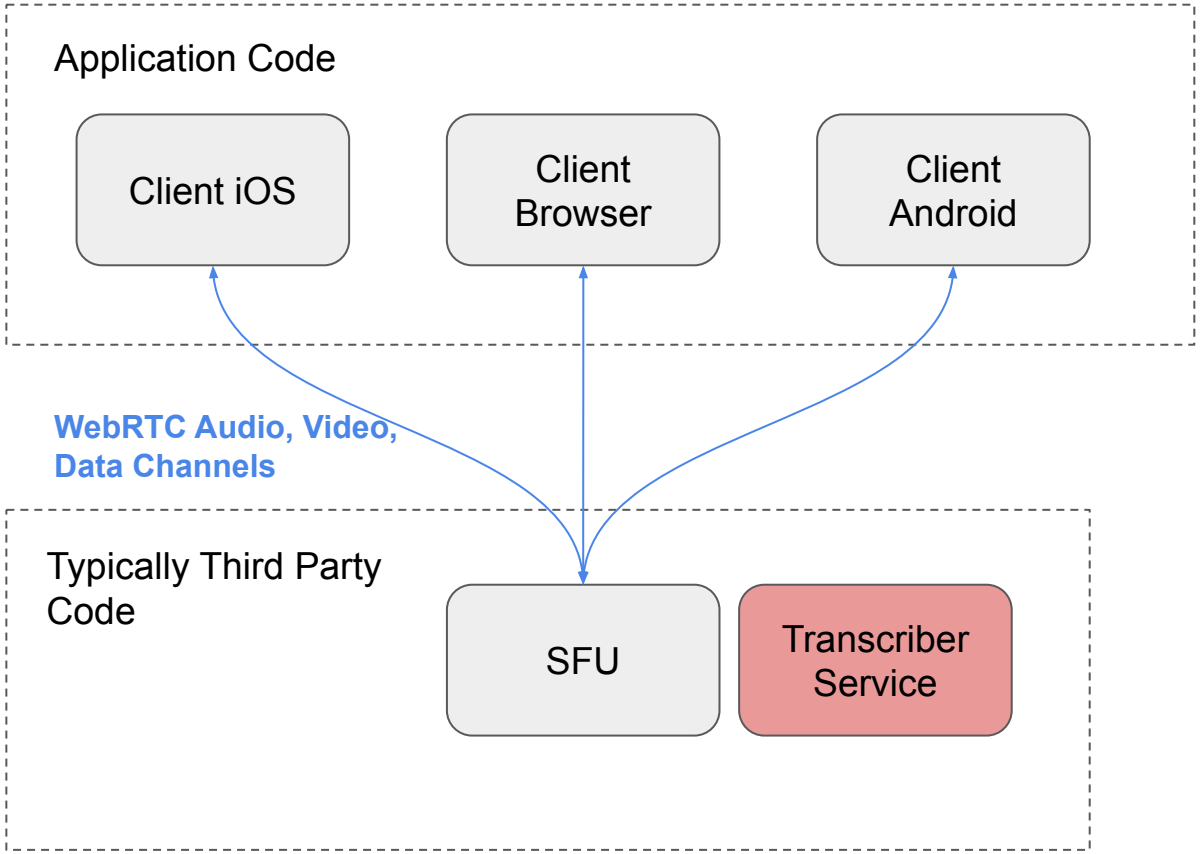
LiveKit

What if I need to process media on the server?

Example: Live Audio Transcription (Speech to Text)

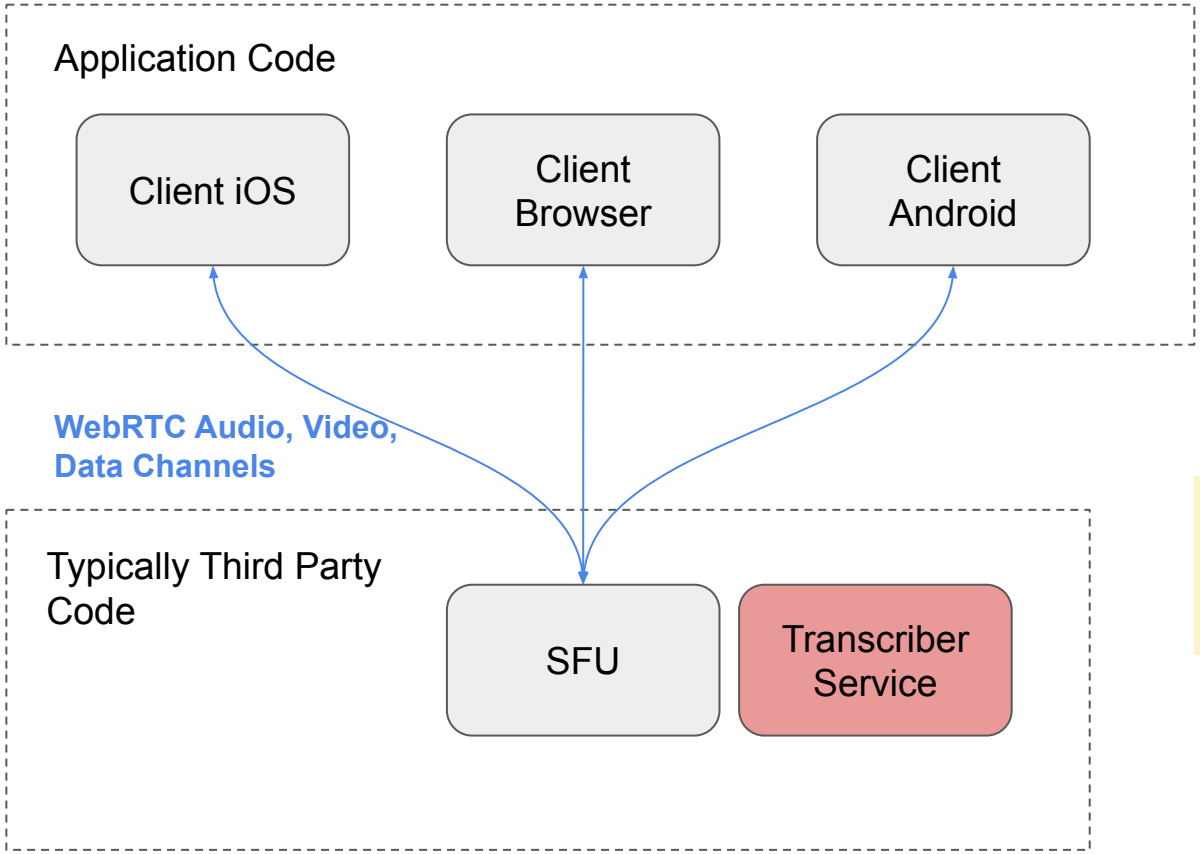
LiveKit

Live Transcription - Option 1



LiveKit

Live Transcription - Option 1

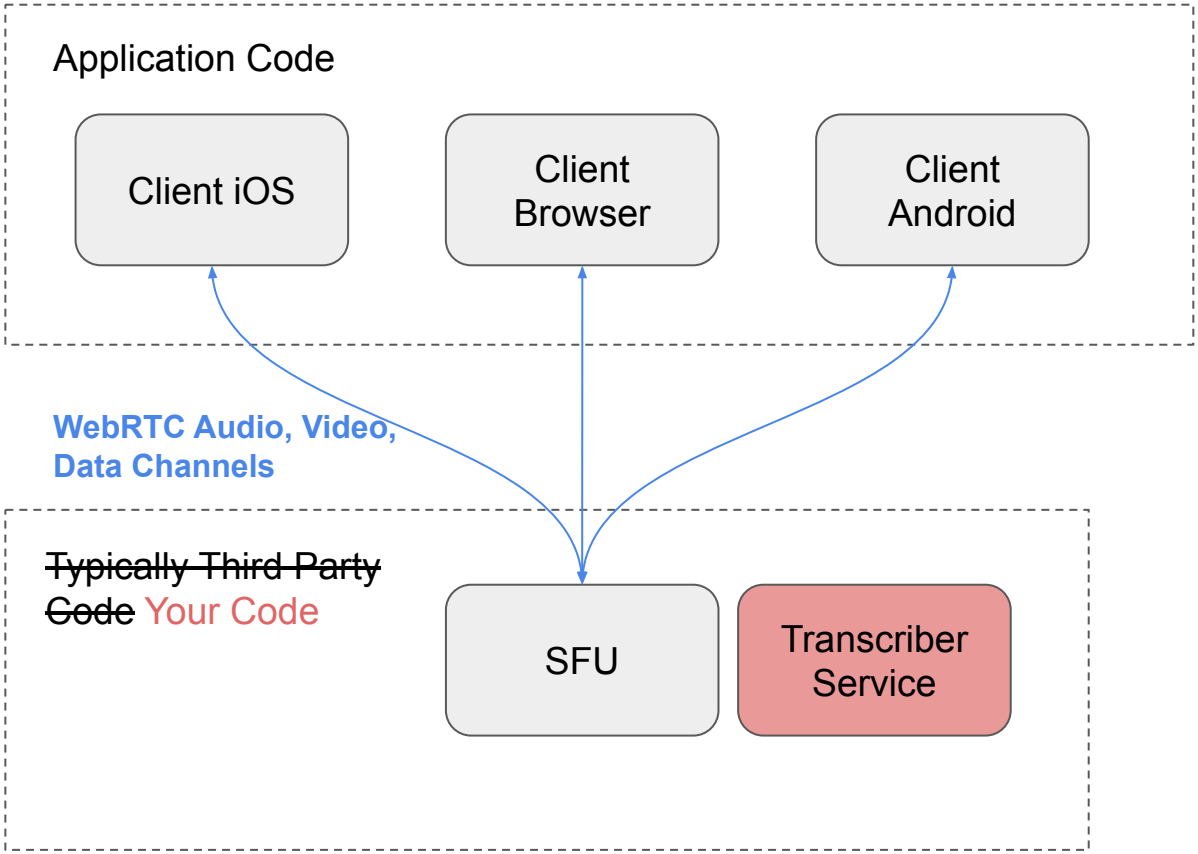


Problems

- You don't own it
- $$$

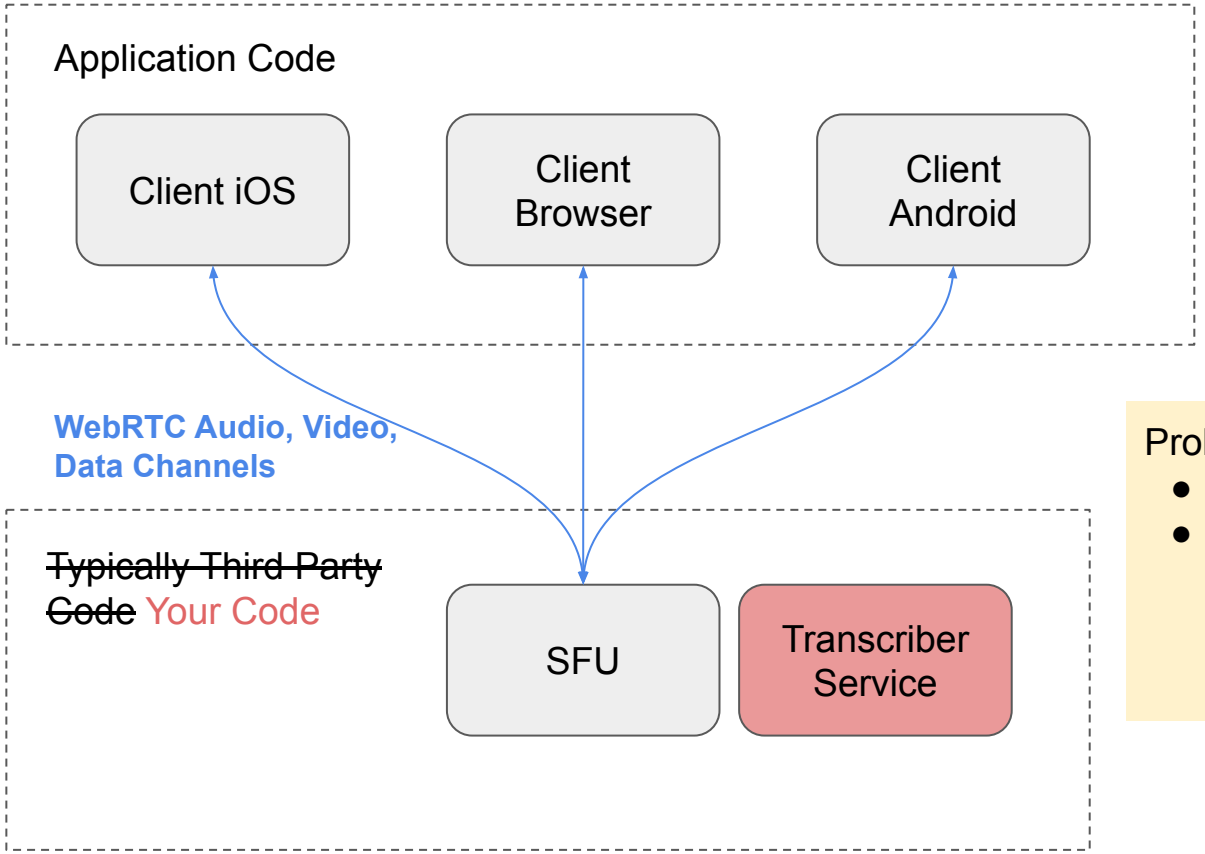
LiveKit

Live Transcription - Option 2



LiveKit

Live Transcription - Option 2

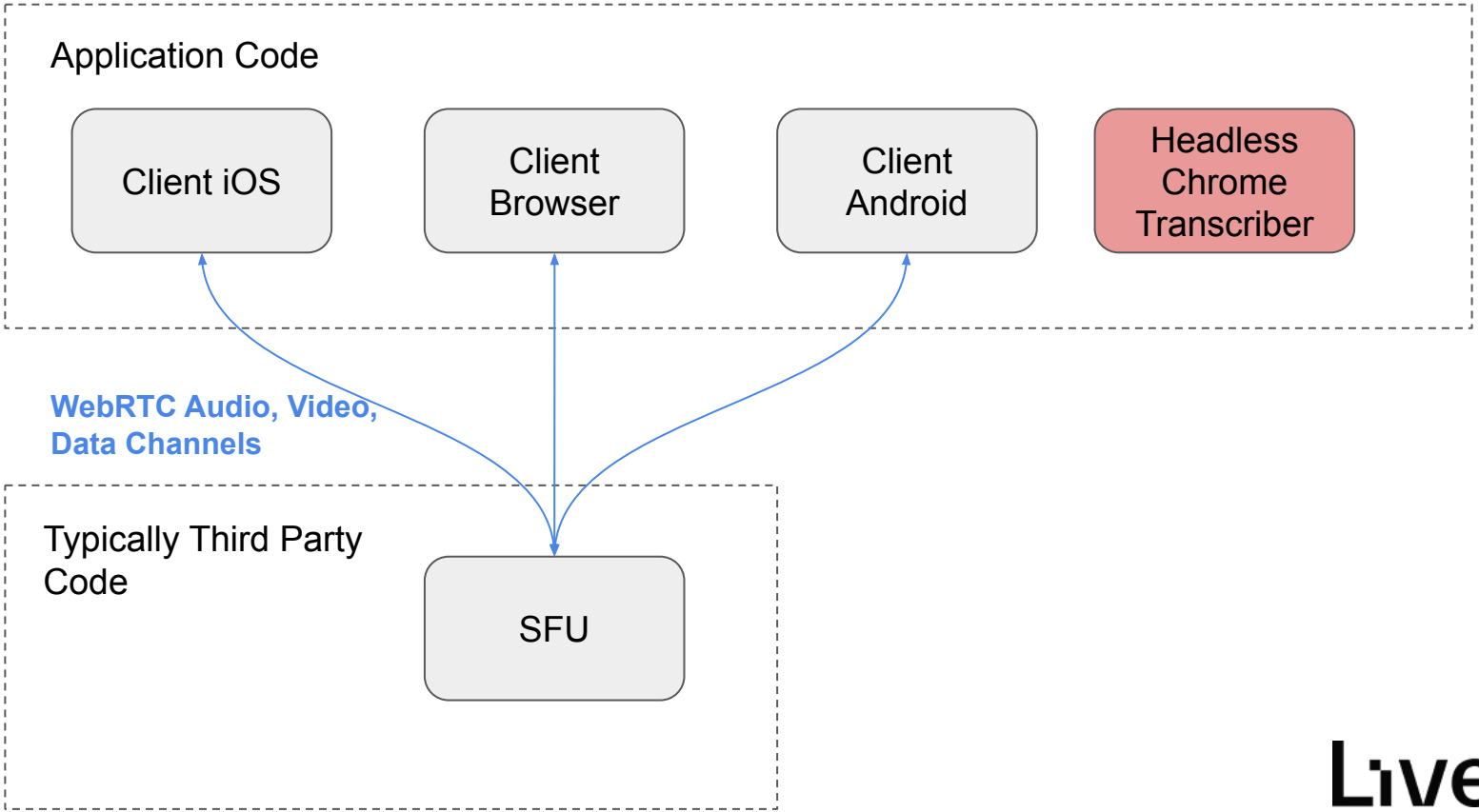


Problems

- Complexity
- Application engineers need to also be WebRTC engineers

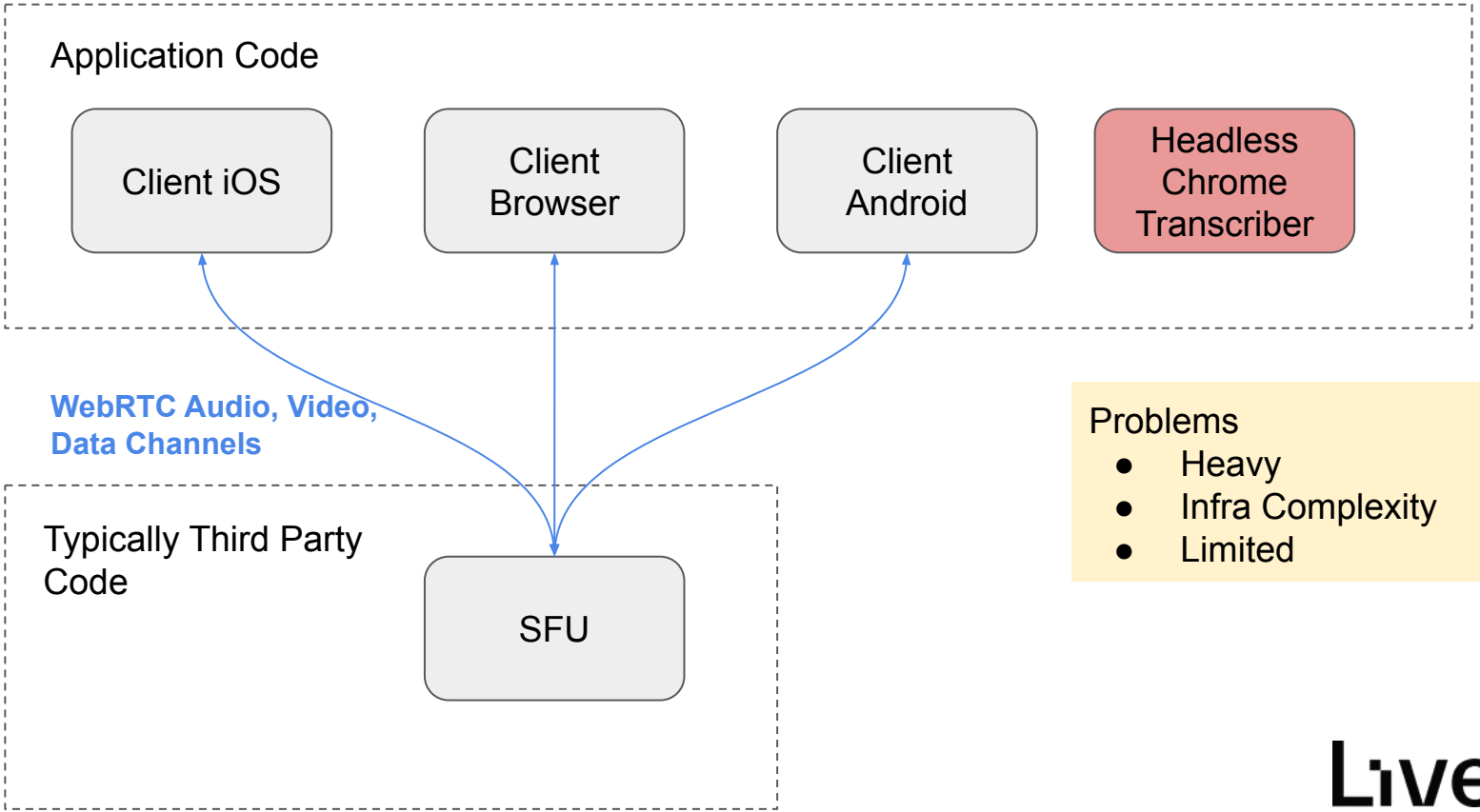
LiveKit

Live Transcription - Option 3



LiveKit

Live Transcription - Option 3

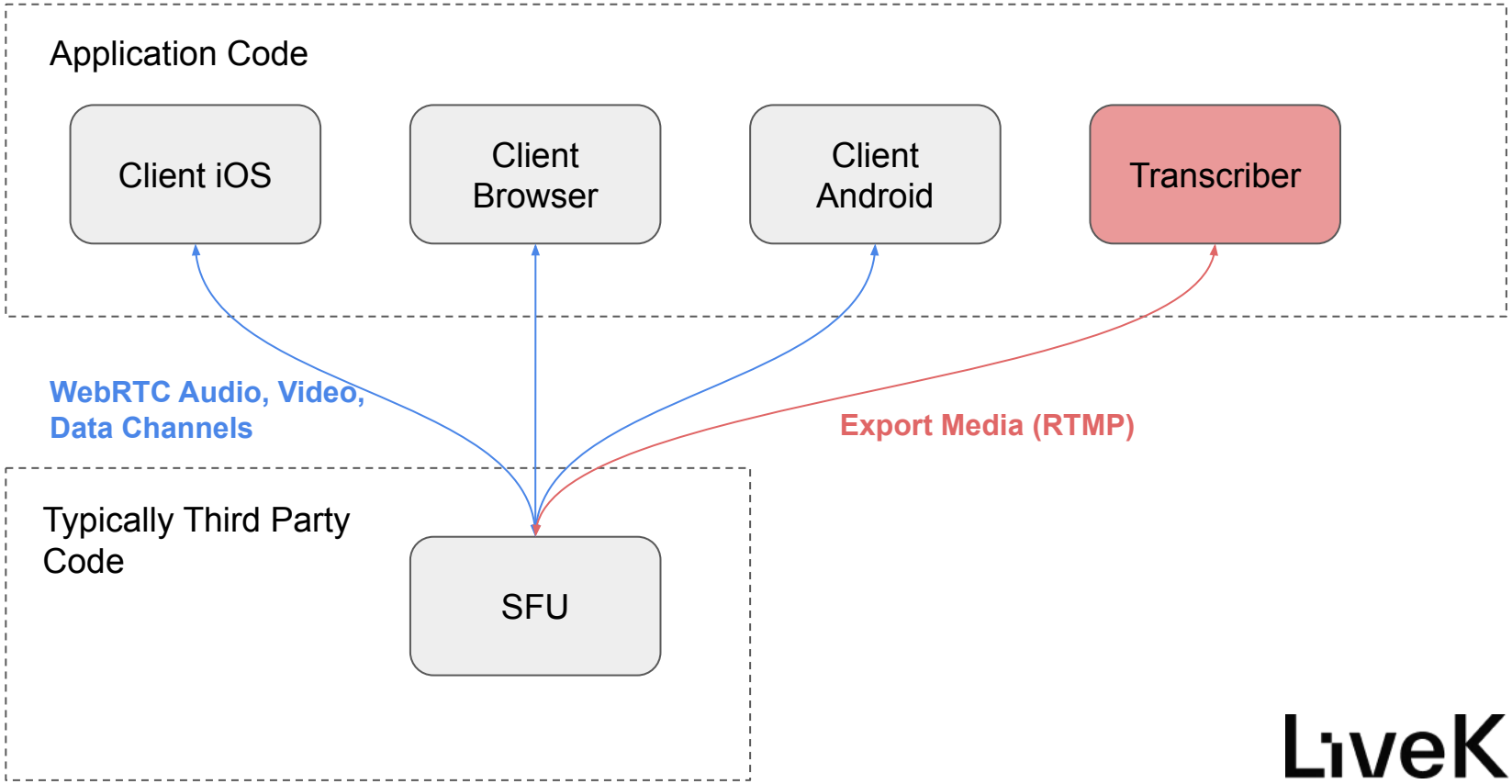


Problems

- Heavy
- Infra Complexity
- Limited

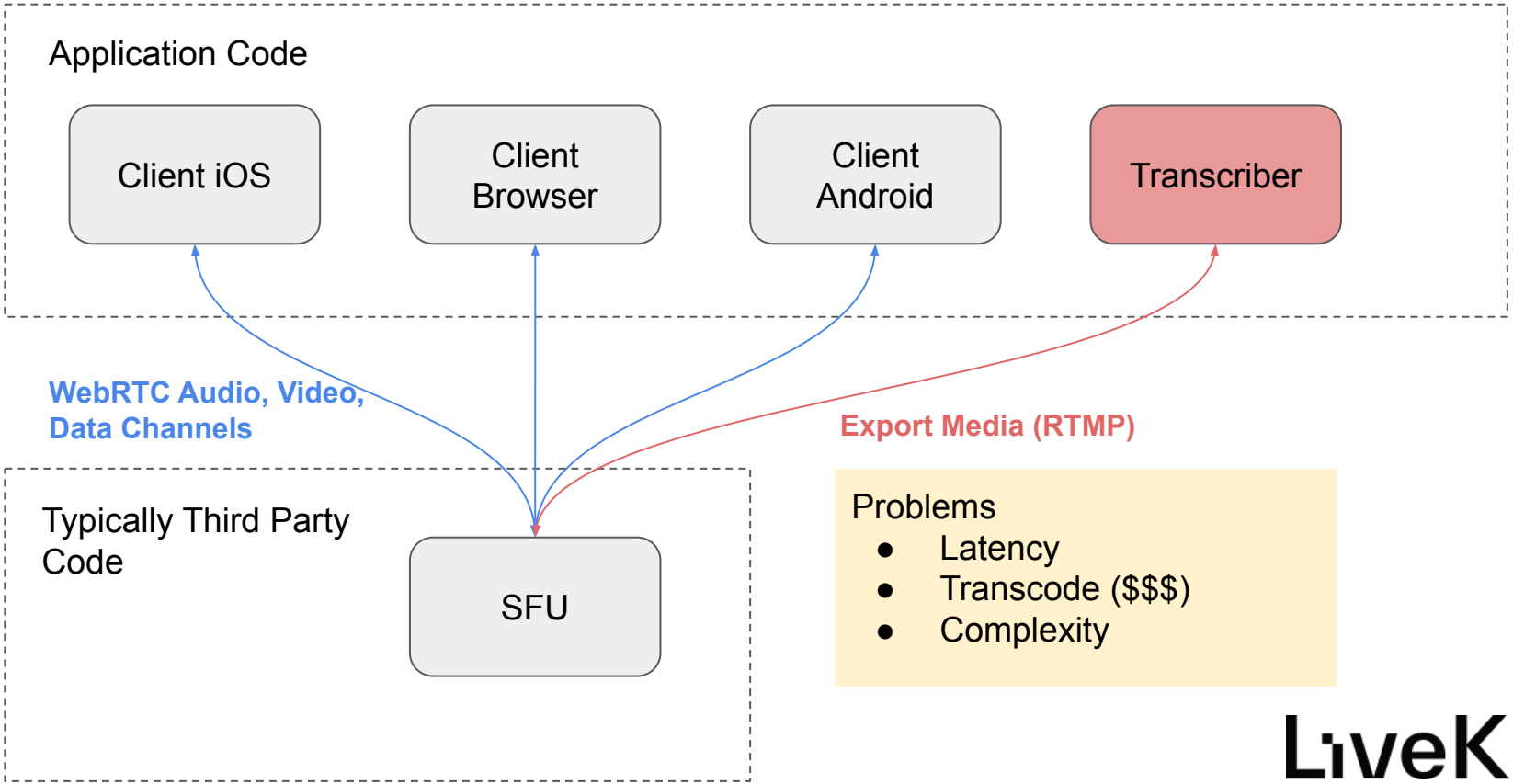
LiveKit

Live Transcription - Option 4



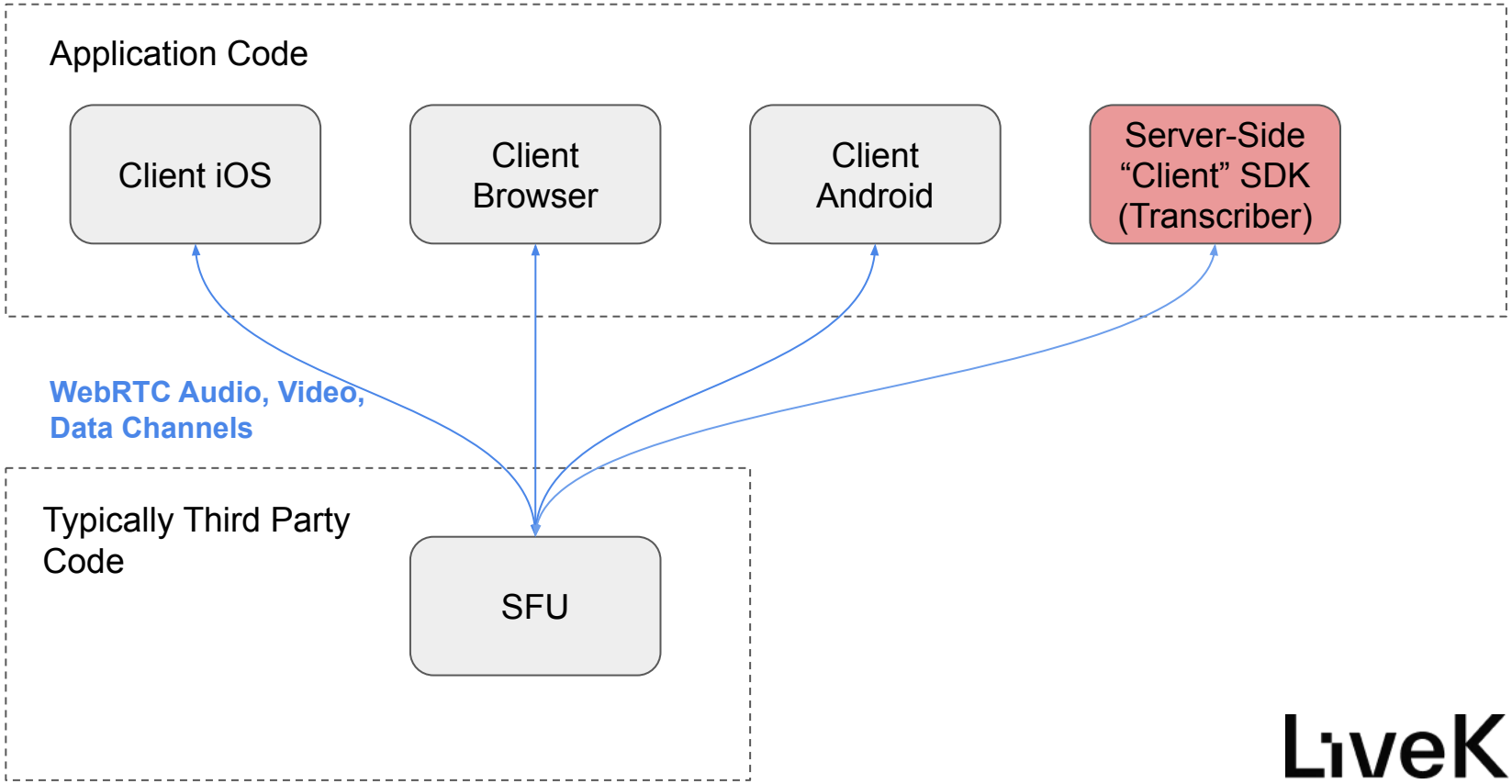
LiveKit

Live Transcription - Option 4



LiveKit

Live Transcription - How It Should Be



LiveKit

What does a Server-Side Client SDK Look Like?

- Completely Open Source
- PCM, not OPUS
- RGBA, not H264/VP8/etc.
- User should not need to know what an ICECandidate is
- User should not need to know what an SDP is

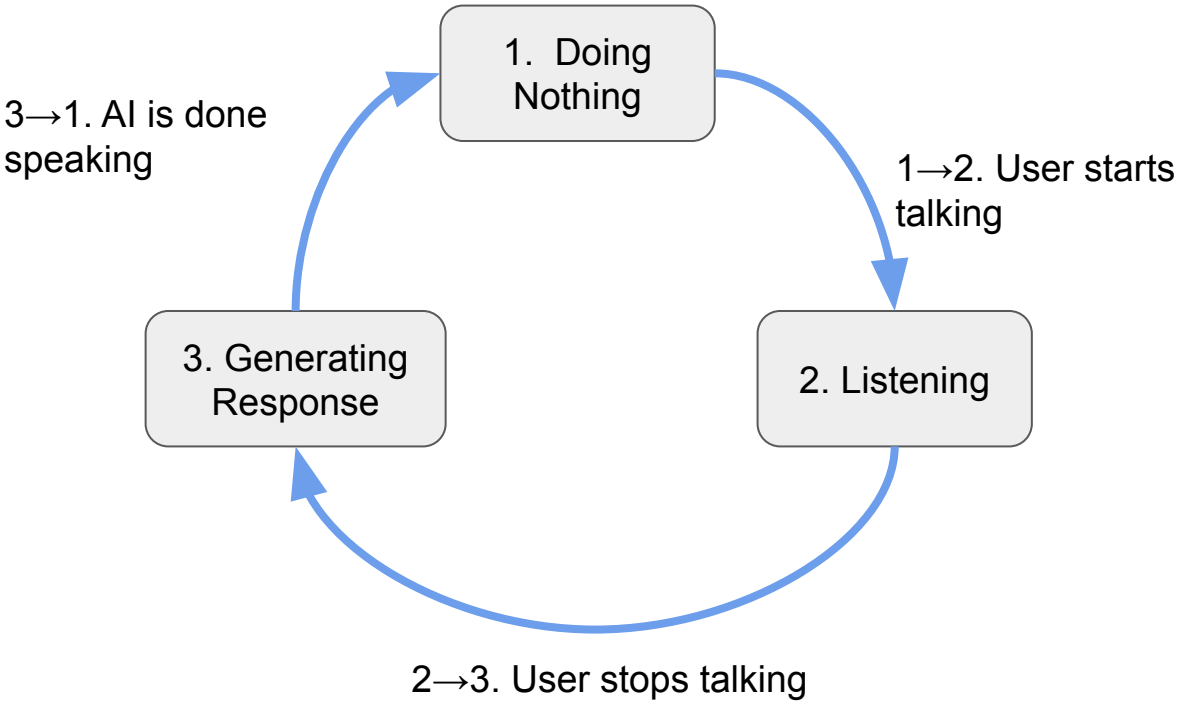
LiveKit

Live Transcription as a Server Side Participant

```python
class Transcription(Agent):

    def on_audio_track(
        self,
        track: livekit.Track,
        participant: livekit.Participant,
    ):
        transcriber = Transcriber(audio_track=track, callback=self._transcriber_cb)
        transcriber.start()

    def _transcriber_cb(self, event: Transcriber.Event):
        print(f"transcription event: {event.type} - text: {event.text} - seconds: {event.time_seconds}")

    def should_process(
        self, track: livekit.TrackPublication, participant: livekit.Participant
    ) -> bool:
        if participant.identity != "caller":
            return False
        return track.kind == livekit.TrackKind.AUDIO      You, now • Uncommitted changes
```

LiveKit

Let’s make Siri in 5 minutes



LiveKit

1. Doing Nothing

LiveKit

1 → 2. User Starts Talking

```python
def _transcriber_cb(self, event: transcription.Transcriber.Event, participant: livekit.Participant):
    if event.type == transcription.EVENT_TYPE_TALKING_STARTED:
        if self.state.type == states.StateType.DOING_NOTHING:
            self._set_state(states.State_Listening())
```

LiveKit

2. Listening

```
async def _state_listening(self):
    await self.tts.warmup()
```

LiveKit

2→3. I Stop Talking

```python
def _transcriber_cb(self, event: transcription.Transcriber.Event, participant: livekit.Participant):
    if event.type == transcription.EVENT_TYPE_TALKING_STARTED:
        if self.state.type == states.StateType.DOING_NOTHING:
            self._set_state(states.State_Listening())
    elif event.type == transcription.EVENT_TYPE_TALKING_FINISHED:      You, 3 days ago • Rename monologue t
        self.chat_gpt.add_message(Message(role=MessageRole.user, content=event.text))
        if self.state.type == states.StateType.LISTENING:
            self._set_state(states.State_GeneratingResponse())
```

LiveKit

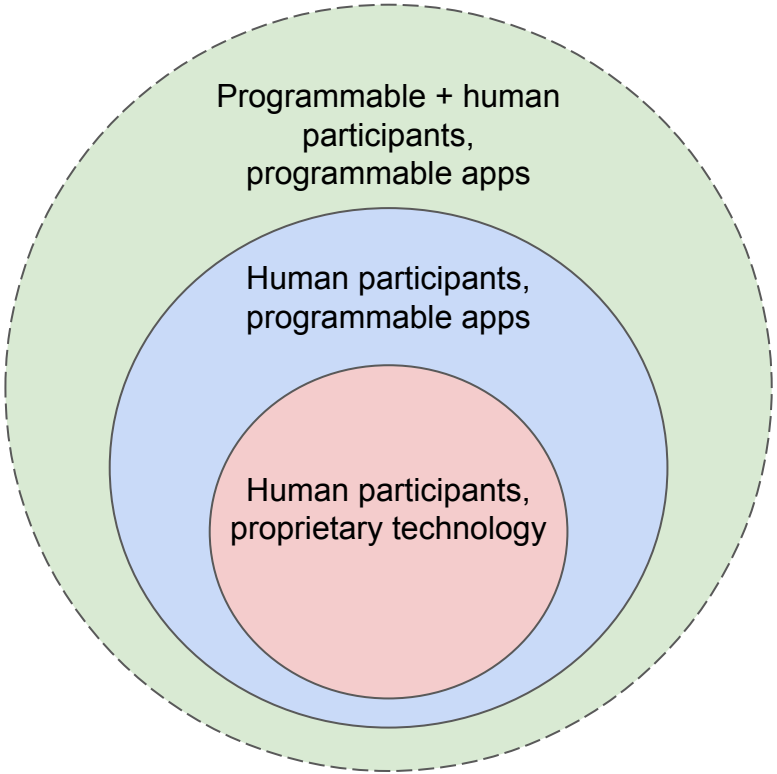
3. Generating Response

```
PROMPT = "You are KITT, a voice assistant in a meeting created by LiveKit. \
          Keep your responses concise while still being friendly and personable. \
          If your response is a question, please append a question mark symbol to the end of it."
```

```
async def _state_generating_response(self):
    text_queue = asyncio.Queue()
    asyncio.create_task(self.tts.stream_generate_audio(text_queue=text_queue))
    full_result = ""
    async for chunk in self.chat_gpt.generate_text_streamed(model='gpt-3.5-turbo'):
        await text_queue.put(chunk)
        full_result += chunk

    # Signal that we are done sending text
    await text_queue.put(None)
    print("full result", full_result)
    self.chat_gpt.add_message(Message(role=MessageRole.assistant, content=full_result))
    self._set_state(states.State_DoingNothing())
```

LiveKit

Future of RTC



LiveKit

Other Server Side Participants

- Existing features
  - Audio transcription
  - Transcoding
  - Recording
  - Egressing
- AI
  - AI meeting notes
  - AI friends
  - AI teachers / coaches
- Gaming
  - Game Server
  - AI NPCs
- Real Time Computer Vision
  - Surveillance
  - Robotics
  - Drones
  - Cars

LiveKit

Resources

- Online Demo
  - https://livekit.io/kitt
- Full Code From This Presentation
  - https://github.com/livekit-examples/python-agents-examples

LiveKit