Agent Mode Architecture

Figure 1: Agent Mode Architecture

# Agent Mode (LiveKit WebRTC)

## Overview

**Agent Mode** is the primary, recommended architecture for high-performance real-time speech AI. It leverages **LiveKit's WebRTC Audio Tracks** for audio transport and **Data Channels** for receiving transcriptions.

This method completely decouples the audio capturing logic (Frontend) from the processing logic (Backend Agent), allowing for ultra-low latency and network resilience.

---

## End-to-End Architecture

### System Flow

1. **Frontend (UI)** initiates a room connection.
2. **LiveKit Cloud** acts as the central relay.
3. **Backend Agent** automatically joins the room as a heavy-computational peer.
4. **Audio Flow:** Microphone (Opus 48kHz) -> Cloud -> Agent (PCM 16kHz).
5. **Data Flow:** Agent (Text JSON) -> Cloud -> Frontend (UI Update).

---

## Micro-Interaction Walkthrough

### 1. User Speaks

- **Input:** The user speaks into the microphone.
- **Visual:** The `AudioVisualizer` component reacts instantly to local MediaStream energy levels.
- **Transport:** The audio is encoded into efficient Opus packets and transmitted via UDP (WebRTC) to the nearest LiveKit Edge server.

### 2. Cloud Relay

- **Processing:** LiveKit Server distributes the packets.
- **Resilience:** If packets are lost, PLI (Picture Loss Indication) or NACKs handle recovery, though for audio it's mostly forward error correction.

3. **Agent Processing**
    - **Reception:** The Python backend receives the track.
    - **VAD:** A Voice Activity Detector checks if speech is present.
    - **Gating:** If energy < -40dB, it is ignored (Silence).
    - **Inference:** The audio buffer (0.6s) is fed into `faster-whisper`. `python segments, _ = asr_engine.model.transcribe(buffer)`
4. **UI Update**
    - **Transmission:** The transcript `{"text": "Hello world", "isFinal": true}` is sent via Data Channel.
    - **Display:** The React `TranscriptDisplay` receives the JSON and pushes it to the `segments` array.
    - **Feedback:** A small " 120ms TAT" badge appears, indicating the turnaround time.

---

## Performance Characteristics

| Metric | Value | Notes |
|---|---|---|
| **Latency** | **< 300ms** | Fastest mode. Direct memory stream to inference. |
| **Bandwidth** | **Low** | Uses efficient Opus codec (vs raw PCM/WAV). |
| **Reliability** | **High** | WebRTC handles network fluctuations better than TCP/WS. |
| **CPU Usage** | **Low (Client)** | Client only handles encoding; Server handles inference. |

---

## Key Code References

- **Frontend Controller:** `frontend/src/hooks/useLiveKitAgent.ts` (Lines 1-240)
- **Backend Agent Logic:** `backend/main.py` (Lines 394-568)
    - `spawn_agent()`: Manages room connection.
    - `process_audio_track()`: Handles the audio stream loop.