Direct Mode Architecture

Figure 1: Direct Mode Architecture

Direct Mode (WebSocket)

Overview

Direct Mode (formerly "WebSocket Only Mode") is a lightweight, peer-to-peer style architecture. It bypasses the LiveKit Cloud entirely for the audio path, sending audio data directly from the browser to the Python backend via a standard WebSocket connection.

This mode is ideal for local testing, offline-first scenarios (if backend is local), or simple deployments where WebRTC complexity is not required.

End-to-End Architecture

System Flow

1. **Frontend (Client Side)** captures microphone input.
2. **WebSocket Tunnel** is established directly to the Python Backend.
3. **Data Stream:** The browser encodes audio into small chunks and pushes them through the pipe.
4. **Backend (Server Side)** receives, decodes, and processes the stream.
5. **Response:** Transcriptions flow back through the same pipe.

Micro-Interaction Walkthrough

1. Initiation

- **Action:** User clicks "Start Recording".
- **System:** The `MediaRecorder` API initializes with `mimeType: audio/webm`.
- **Visual:** The UI shows a "Connecting…" state while the WebSocket handshake (`HTTP 101 Switching Protocols`) completes.

2. Streaming Loop (The Pipe)

- **Chunking:** Every 1 second (1000ms), the recorder slices a binary blob.
- **Encoding:**
  - Blob -> ArrayBuffer -> Base64 String.
  - *Note:* This adds ~33% bandwidth overhead but ensures safe transport over text-based sockets.

- **Transmission:** `ws.send(JSON)` executes. The "Pipe" arrow in the diagram lights up.

3. **Server Processing**

   - **Reception:** The FastAPI server receives the JSON packet.
   - **Decoding:** `base64.b64decode` -> `ffmpeg` converts WebM Opus to PCM WAV.
   - **Inference:** `faster-whisper` runs on the accumulated buffer.

4. **Feedback**

   - **Return Trip:** The server sends `{"type": "transcript", "text": "..."}`.
   - **Latency:** Because chunks are 1 second long, the user sees text appear in ~1.2s bursts.

Performance Characteristics

Metric	Value	Notes
Latency	**Medium (~500ms-1s)**	Higher latency due to `MediaRecorder` chunking intervals.
Bandwidth	**Medium**	Base64 encoding adds ~33% overhead to the payload.
Reliability	**Medium**	TCP-based WebSocket can suffer from Head-of-Line blocking on poor networks.
Simplicity	**High**	Requires zero external infrastructure (No LiveKit Cloud account needed).

Key Code References

- **Frontend Controller:** `frontend/src/hooks/useWebSocketOnly.ts`
- **Backend Endpoint:** `backend/main.py` (Lines 206-330)
  - `websocket_endpoint`: The core loop handling `receive_json` and `send_json`.