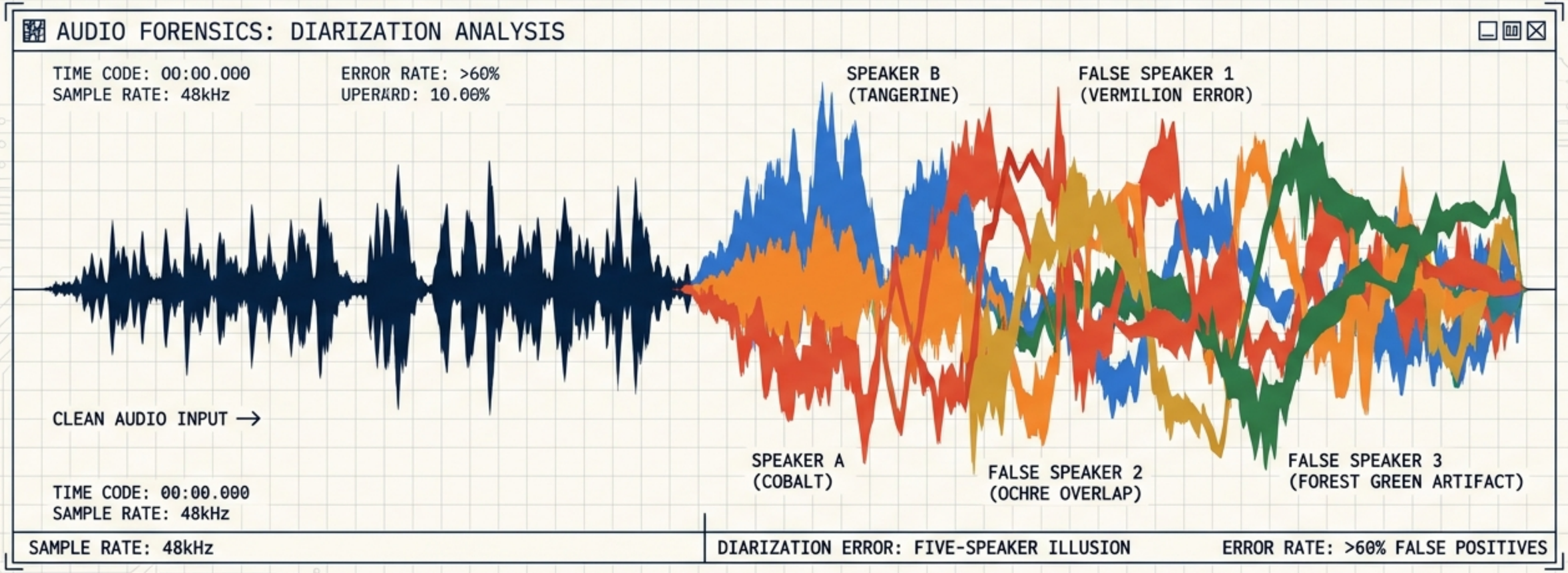
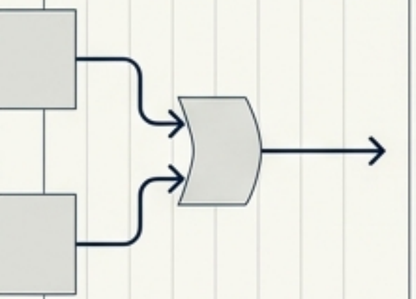
The Five-Speaker Illusion in a Two-Person Call

Building an automated voice memo pipeline revealed a hidden truth: turning sound into words is a solved problem. Figuring out exactly who spoke those words is the genuinely hard half.



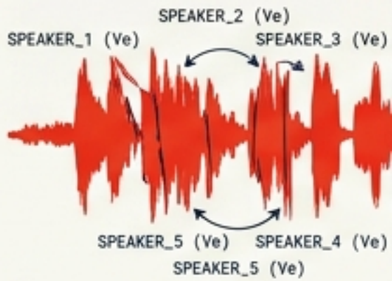
A 28-minute conversation between two people should yield two speaker labels.

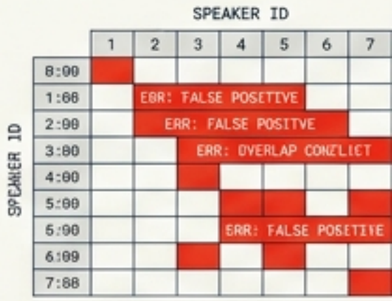
To solve this, I built a series of iteratively failing architectures before finding the root cause.



Actual Speakers (Ground Truth)

AI Output

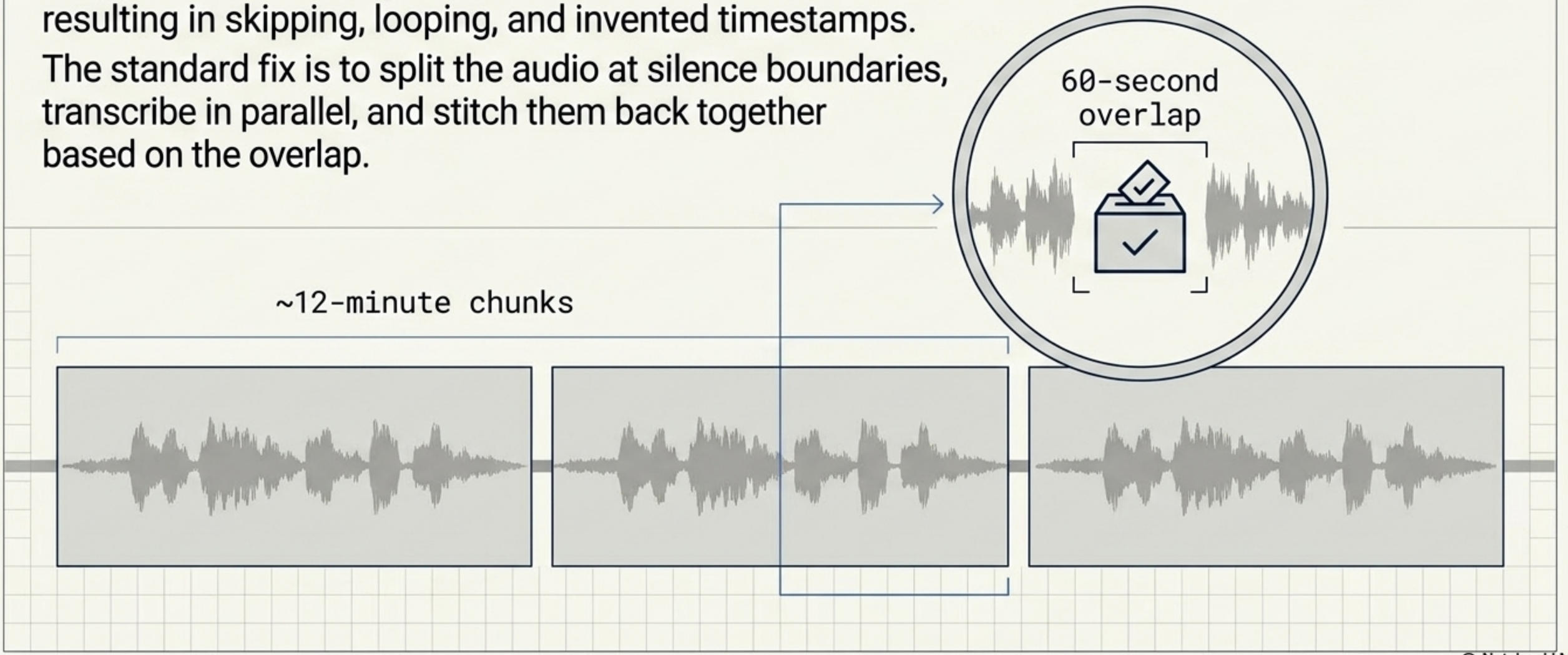




Attempt 1: Bypassing long-context hallucination with Gemini chunking.

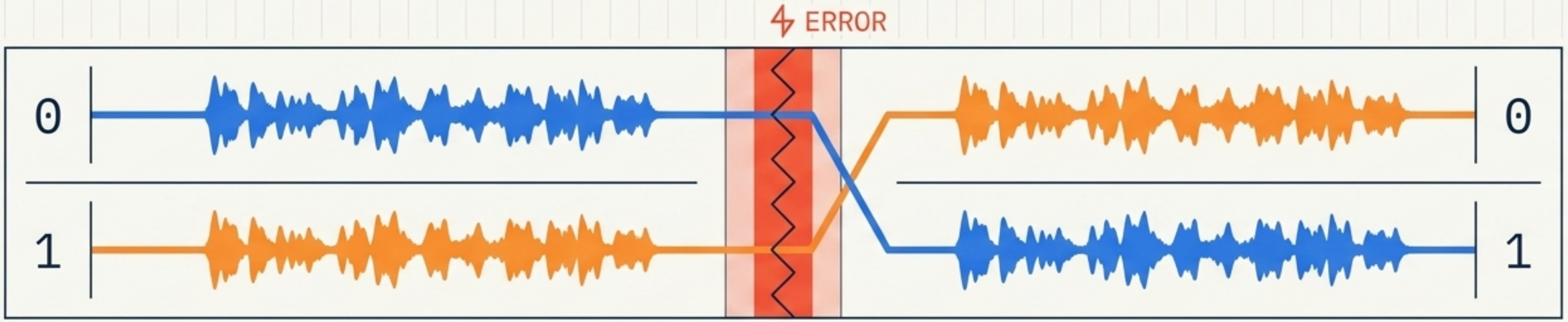
Long audio fed directly to an LLM breaks down past the 15-minute mark, resulting in skipping, looping, and invented timestamps.

The standard fix is to split the audio at silence boundaries, transcribe in parallel, and stitch them back together based on the overlap.



The Seam Problem: One bad alignment **vote ruins the entire file.**

Every chunk numbers its own speakers from zero. If the stitching algorithm makes a **single voting error** in the 60-second overlap region, the labels invert. The count slowly creeps upward as the drift compounds across multiple seams.



Attempt 2: Eliminating seams with a global CoreML pass via Senko.

By processing the entire file locally through Senko in a single global diarization pass, the numbering stays mathematically consistent from start to finish. There is nothing to stitch.

Gemini Method

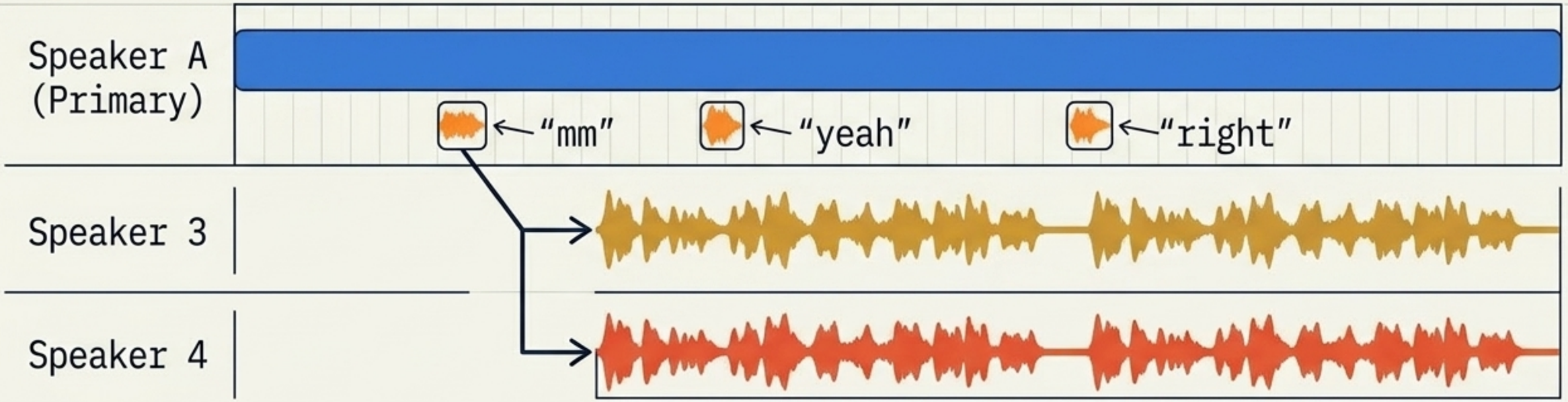
Fractured, misaligned chunking.

Senko Method

Single, continuous global pass.

The Backchannel Trap inflates speaker counts through minor interjections.

Real conversations involve active listening. Senko cannot conceptually map one-word interjections back to the primary listener. It routinely scores them as entirely new people, immediately inflating the speaker count.



Attempt 3: Doubao ASR delivers perfect bilingual text but aggressive speaker splitting.

Volcano Engine's Doubao perfectly solved the original transcription pain points, handling code-switching flawlessly. Yet, its speaker counting was fundamentally worse than Senko.

Transcription Quality (Pros)

- Clean Chinese
- Perfect Mixed Chinese-English
- Natural Spoken Rhythm

Speaker Counting (Cons)

28-minute clip:

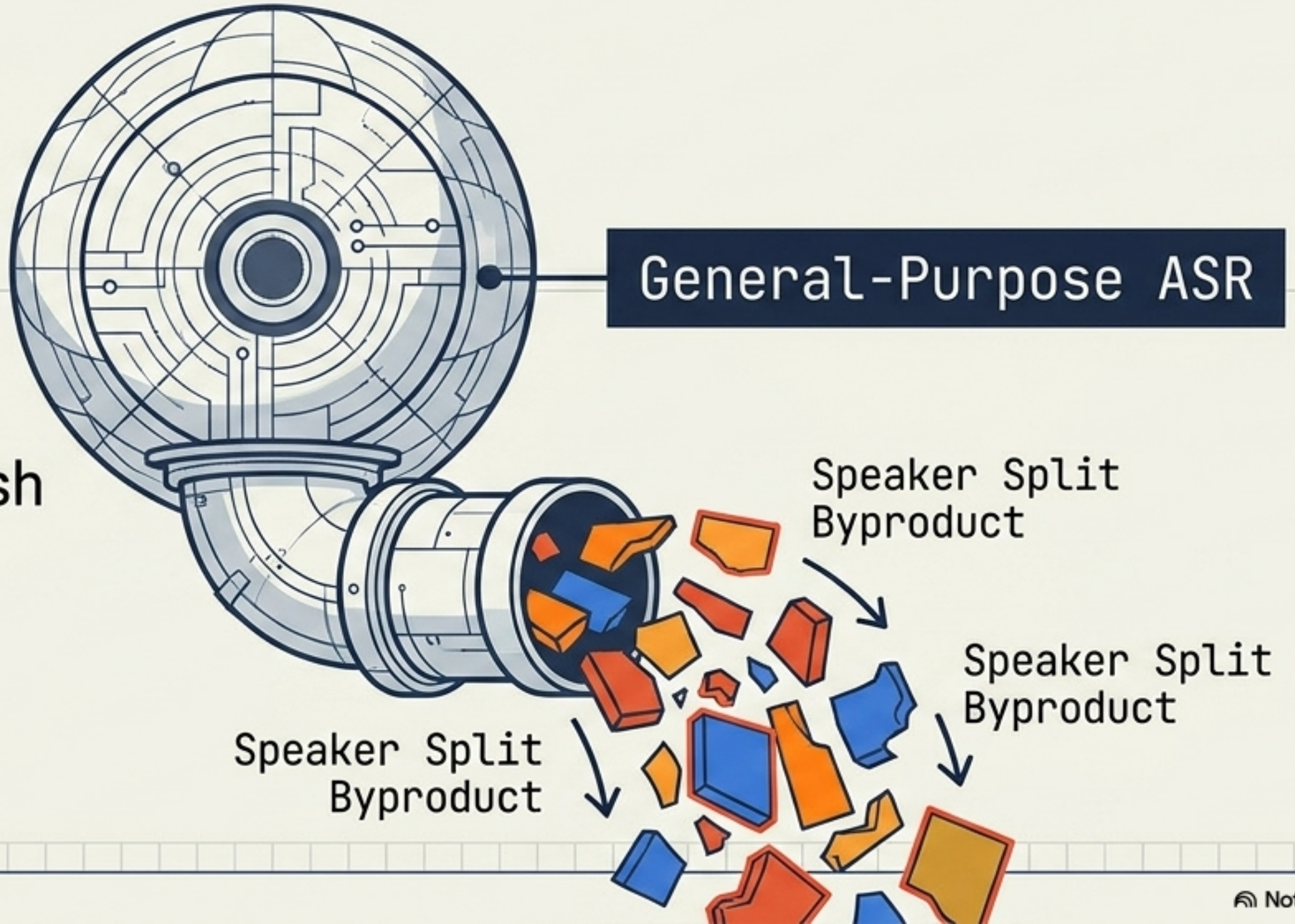
Fast Edition: 5 Speakers

2.0 Edition: 4 Speakers

Diarization cannot be an afterthought of transcription.

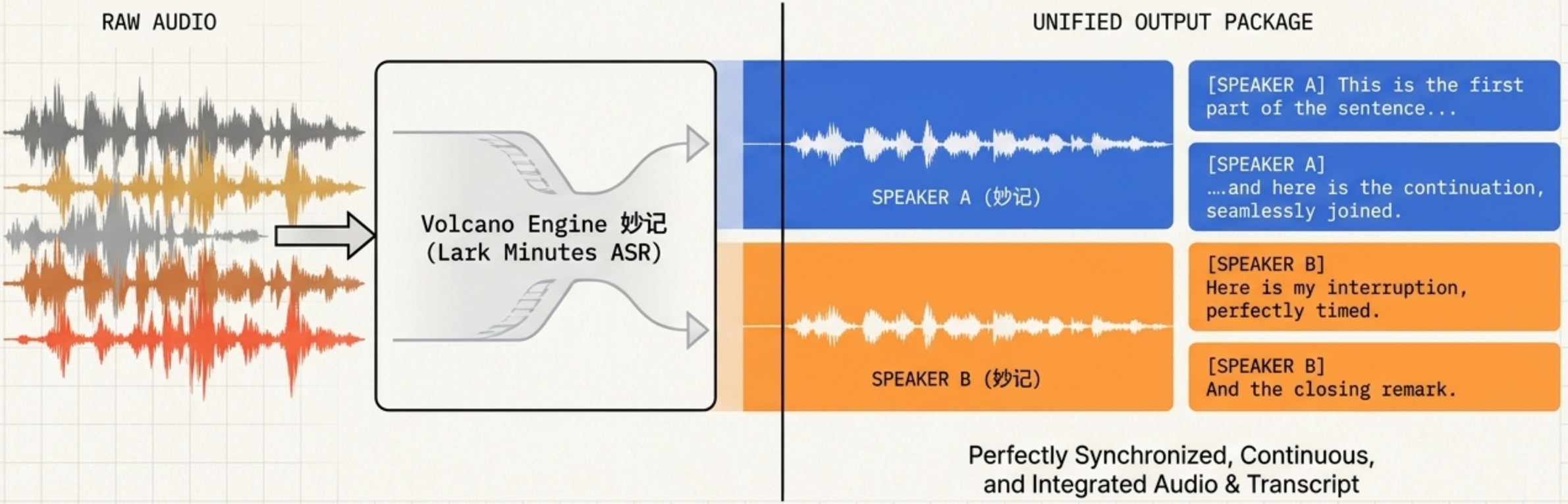
The common thread across Gemini, Senko, and Doubao is that they treat diarization as a side-effect.

Generalist models run aggressive—because they cannot confidently distinguish a backchannel from a new voice, their default behavior is to **over-split** rather than merge.



The Solution: Hand the entire job to a model built specifically for diarization.

妙记 functions differently. Diarization is executed entirely server-side and returned as a unified package. It treats 'who spoke' as a first-class task, entirely eliminating client-side stitching and aggressive backchannel splitting.



Performance Scorecard: 妙记 perfectly isolates the true speaker count.

On a massive 3.45-hour multi-character drama track, 2.0 returned 10 speakers. 妙记 swallowed it in one pass and returned exactly 3.

Recording Length	Fast Edition	2.0 Edition	妙记
28min	5	4	2
38min	3	3	2
57min	3	3	2
68min	4	3	2
3.45h (drama)	—	10	3

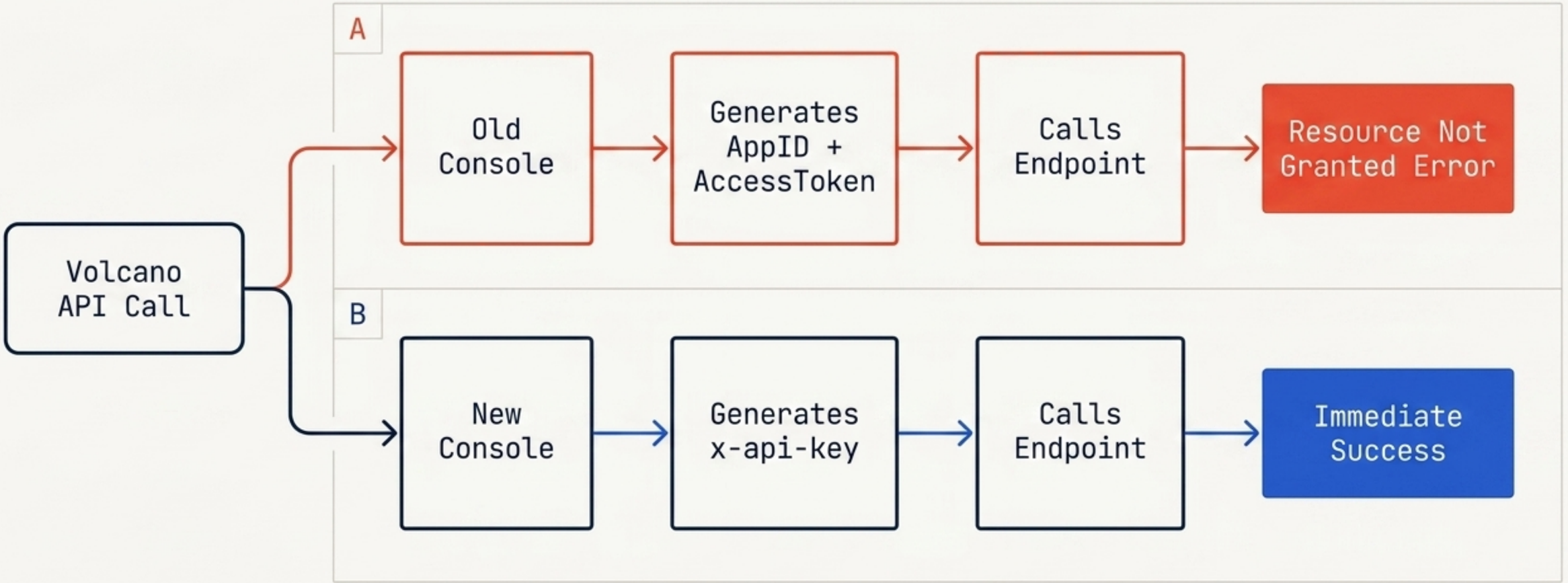
The Architectural Diagnostic Matrix

A definitive reference summarizing the engineering journey.

Model	Approach	Core Mechanism	Fatal Flaw
Gemini	Chunking	Stitching	Label Drift
Senko	Local Global	CoreML	Backchannel Inflation
Doubao	API Global	ASR Byproduct	Aggressive Over-splitting
妙记	Unified Server-side	Dedicated Task	None (Perfect alignment)

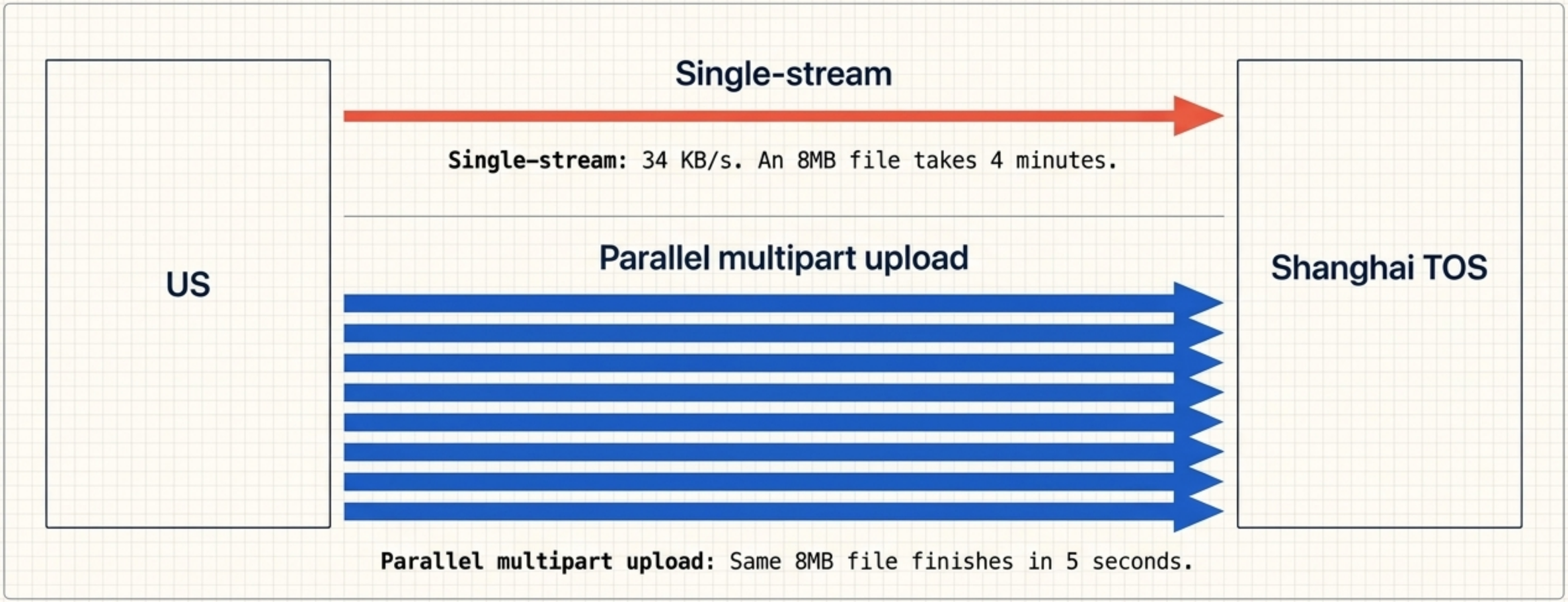
Implementation Gotchas: Navigating the dual authentication trap.

Volcano's speech services maintain two distinct account systems bound to different resources. The documentation buries this distinction.



Escaping the 34 KB/s cross-border upload throttle.

妙记 requires a publicly fetchable URL. Because cross-border links throttle per-connection, not by total bandwidth, you must split the file and open multiple connections simultaneously to bypass the bottleneck.



A reliable pipeline treats ‘who spoke’ with as much respect as ‘what was said.’

Looking back, transcription was never the bottleneck. The answer wasn’t in a smarter client-side stitching algorithm, but in offloading the work to a model willing to treat diarization as a real job.