>_ CONF42 · LLMs · 2026

Voice agents that think while they talk

From STT–LLM–TTS pipelines to realtime parallel reasoning.

A technical talk on real-time voice-agent architecture

24 slides

>_ whoami

Bohdan Snisar

Founding Engineer, building **Solea AI** an AI Operating System for home-service businesses

- 13 years of experience
- Co-founder of several startups in past
- Home-service operations · voice & AI infrastructure · ex-Wix, ex-Revolut and etc.

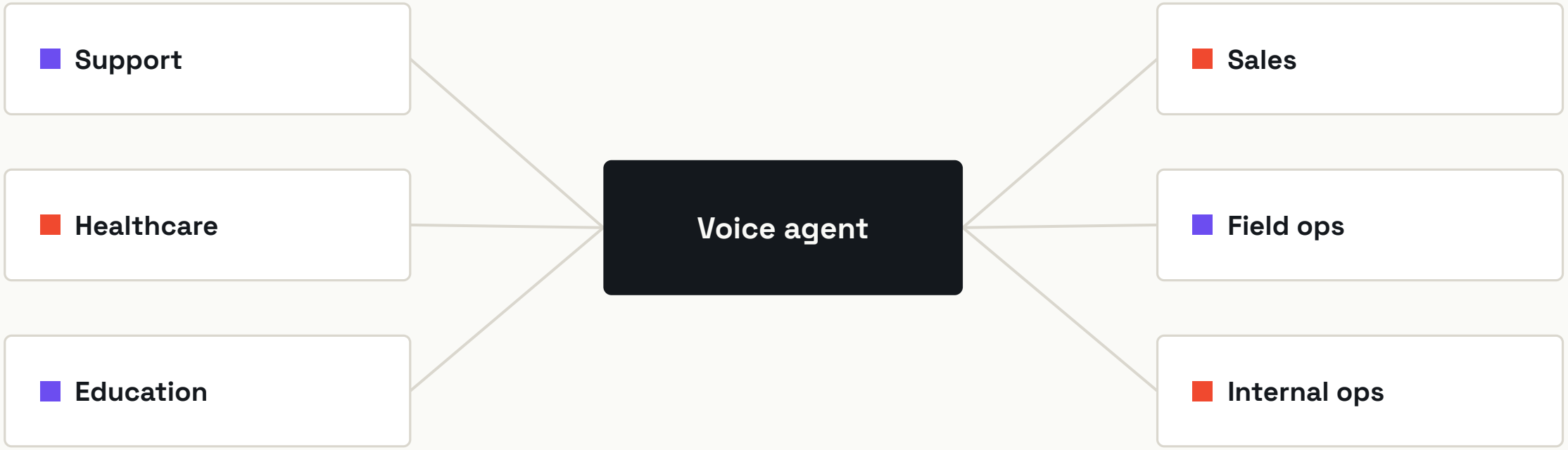
linkedin.com/in/bsnisar

x.com/BSnisar

// 02

Voice is becoming a production interface

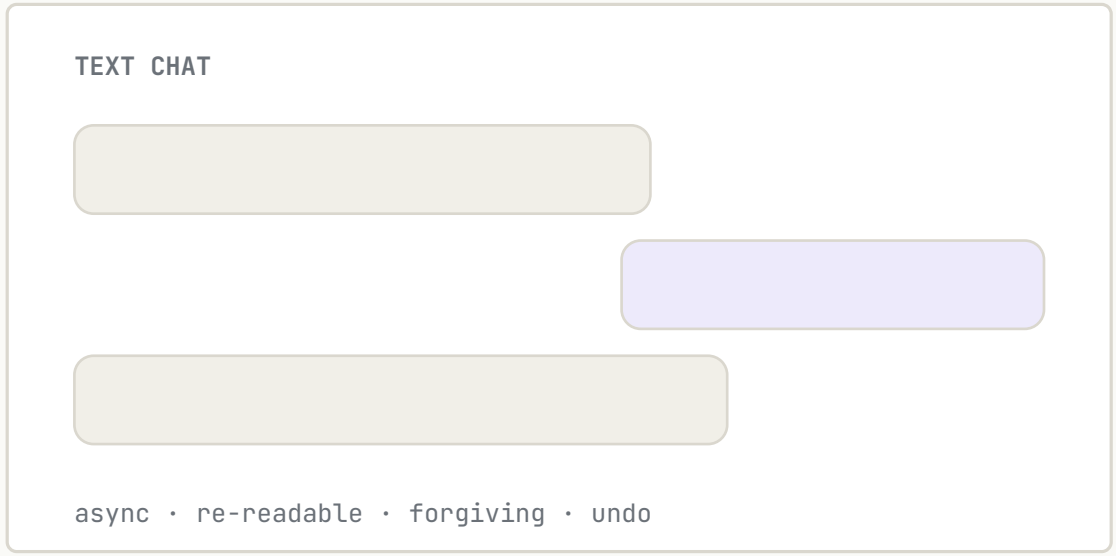
Voice works where text chat is too slow, too unnatural, or too detached from the task.

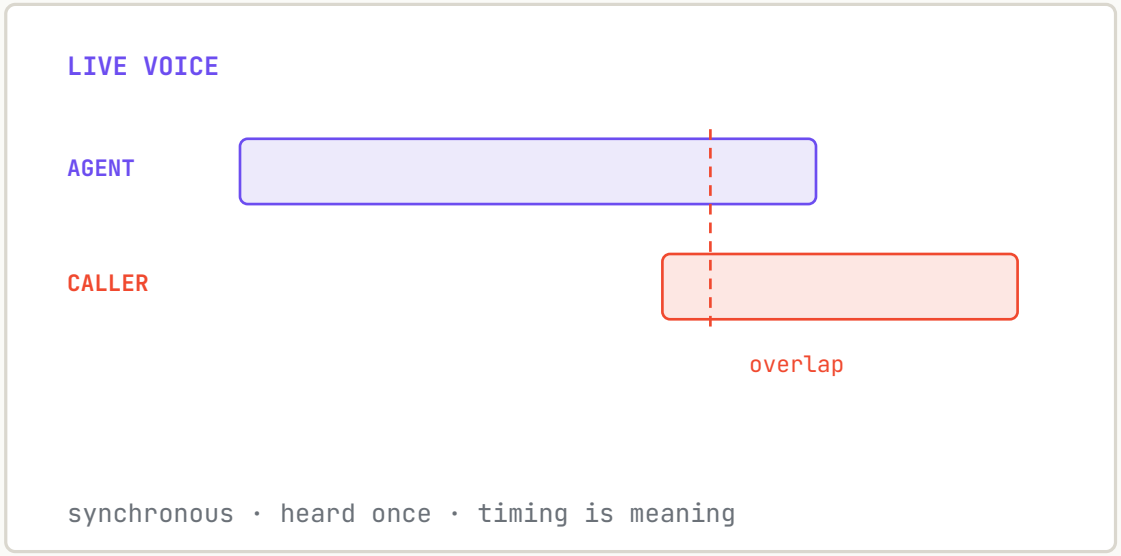


// 03

Voice is not chat with audio

Text is forgiving. Voice runs on a clock, and there is no easy undo.





Latency · Interruption · Turn-taking · Emotion · Silence · Timing · No undo

A good answer is not enough. *It must arrive at the right moment.*

// 04

What is a voice agent?

It listens, understands, reasons, acts, and speaks — as one loop.

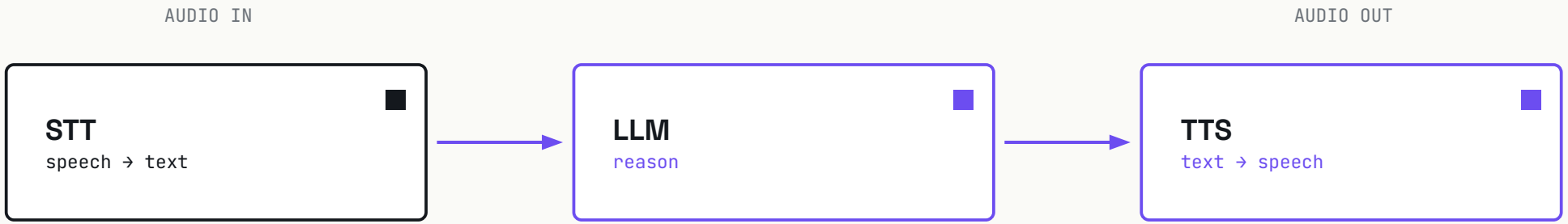
// speaker note this loop looks simple — but each step gets much harder when it must happen in realtime.

// 05

Classic STT → LLM → TTS pipeline

BUILD 1 / 3 · NAIVE

Start with the simplest version — three models in a row.



ALTERNATIVE

Speech-to-Speech (S2S): one model, audio → audio.

GPT-Realtime-2 · Gemini Live · Moshi · Qwen-Omni — lower latency, less control & observability.

This talk builds the cascaded path — for control, tool-use and observability.

Modular and observable — but every stage waits for the one before it.

// 06

First challenge: realtime audio transport

Before any model runs, you must move audio both ways — continuously, low-latency, over a lossy network.

WebRTC transport

Media over WebRTC (or SIP / Twilio / LiveKit). NAT traversal, encryption, adaptive bitrate.

Streaming, not request/response

Audio flows as ~20 ms Opus frames, continuously — there is no single upload to wait for.

Jitter buffer

Reorders and smooths late / out-of-order packets. Every ms buffered is latency you add.

Packet loss

No time to retransmit — lost frames are concealed (PLC), never fully recovered.

Echo cancellation

The agent's own TTS leaks into the mic. AEC + AGC stop VAD / STT from hearing the bot.

Clocking & resampling

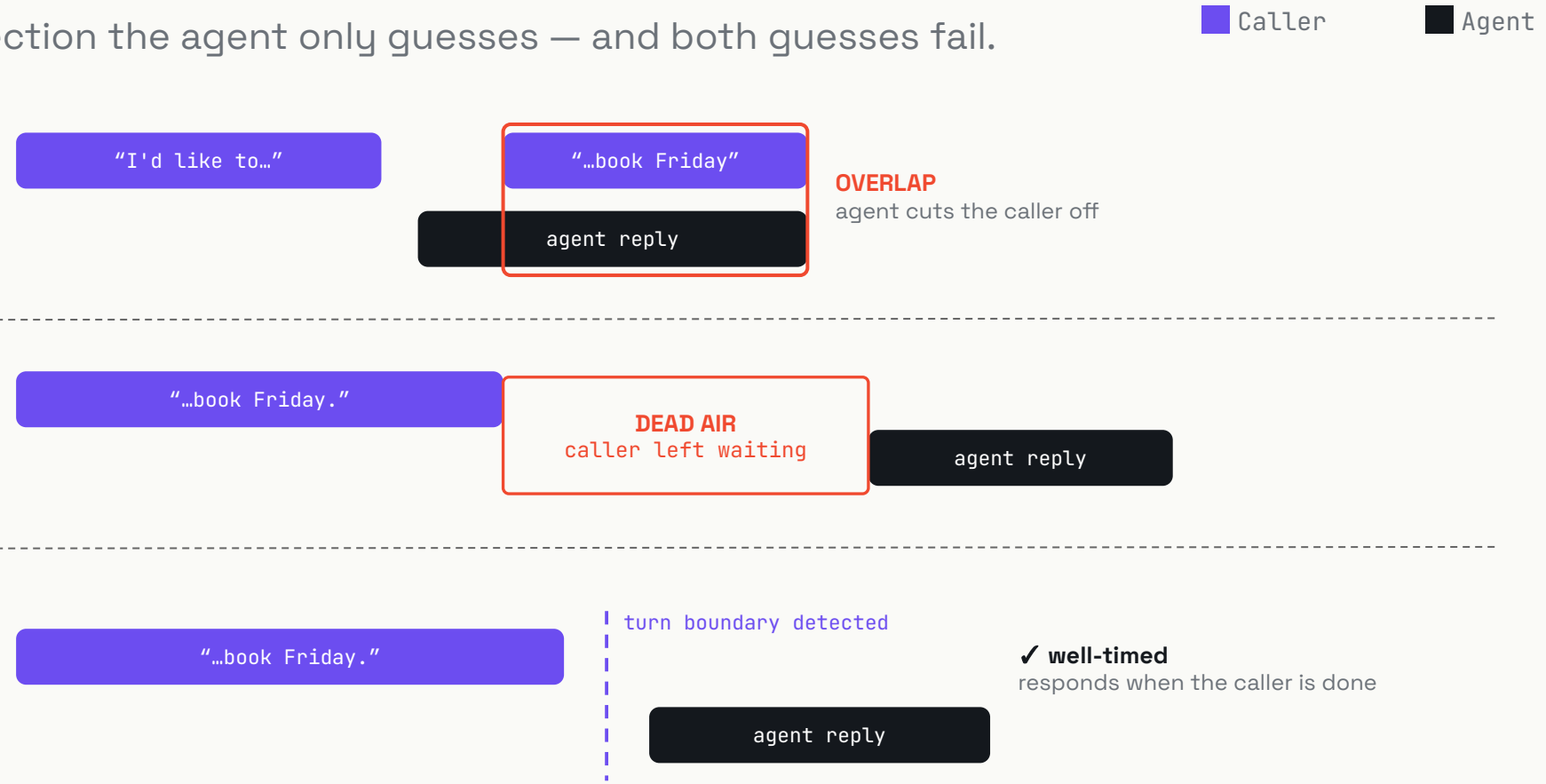
A continuous 16–48 kHz stream; sample-rate sync and resampling between every stage.

Before a single token is generated — you're already fighting the network.

// 08

When should the agent talk?

Without turn detection the agent only guesses — and both guesses fail.



VAD: detecting speech activity

The lowest layer of turn management — it hears **sound**, not meaning.

EXAMPLE

VAD ANSWERS

“Is there speech right now?”

yes / no — per ≈20 ms frame

VAD CANNOT ANSWER

- Has the user finished their thought?
- Should the agent respond now?
- A real interruption, or just a backchannel?

TOOLS

Silero VAD | WebRTC VAD | Picovoice Cobra | Deepgram (built-in)

≈10–30 ms / frame

VAD detects sound boundaries — not conversational meaning.

Comparing VAD tools

WebRTC, Silero and Pyannote — a trade-off between speed, robustness and accuracy.

WebRTC VAD

signal-processing · GMM

- Fast & lightweight (CPU)
- True realtime, ~10–30 ms
- Struggles with noisy audio

BEST FOR — lowest latency

Silero VAD

deep learning · tiny model

- Robust to background noise
- Still CPU-realtime
- Slightly higher latency

BEST FOR — balanced default

Pyannote Audio

deep learning · diarization

- Highest accuracy
- Speaker diarization too
- Heavy · GPU · HF token

BEST FOR — accuracy / offline

No single winner — match the VAD to your latency and accuracy budget.

Endpointing and turn detection

This is where realtime voice starts to get hard.

VAD
Is there speech?

Endpointing
Is the utterance complete?

Turn detection
Should the agent take the turn?

silence is not the same as completion.

EXAMPLE

User: “I need to change my appointment…”

[pause]

✕ *bad endpointing: the agent answers here, too early*

User: “…because I’m travelling next week.”

EXAMPLE

User: “I need to change my appointment…”

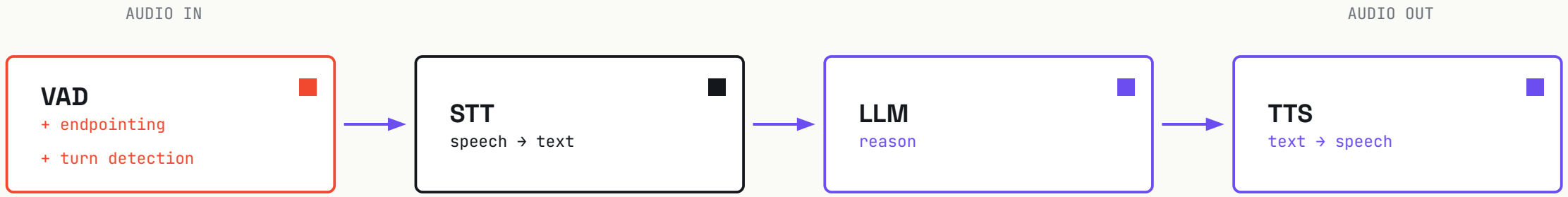
[pause]

User: “…because I’m travelling next week.”

✕ *good endpointing: the agent answers here*

Classic STT → LLM → TTS pipeline

Add VAD, endpointing and turn detection — the agent must know when to listen and when to respond.



Now the pipeline knows when to start — and when the caller is done.

THE REALTIME LATENCY BUDGET

< 500 ms

Beyond this, the conversation starts to feel delayed or unnatural.

The latency budget — and where it goes

Realtime voice has a very small budget — and the chained pipeline already exceeds it.

Component	Best case	Typical
VAD	15–20 ms	20–30 ms
STT	200–300 ms	400–600 ms
LLM	100–200 ms	500–1000 ms
TTS	100–150 ms	200–300 ms
Total	**~415 ms**	**1.1 s – 2 s**

Approximate ranges. Actual latency depends on model, network, streaming mode, endpointing, provider and tool calls.

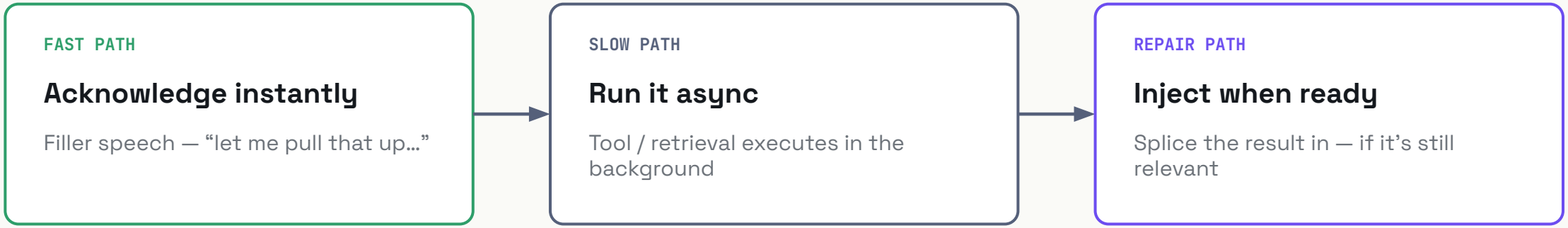
Tool calls: the latency wild card

A CRM, calendar or payment call can take 500 ms – 5 s — far beyond the speech budget.

EXTERNAL TOOLS

CRM · calendar · booking · payment · retrieval

500 ms – 5 s



Never let a tool call block the conversation — cover it with speech, run it async, inject the result.

The classic pipeline in production SDKs

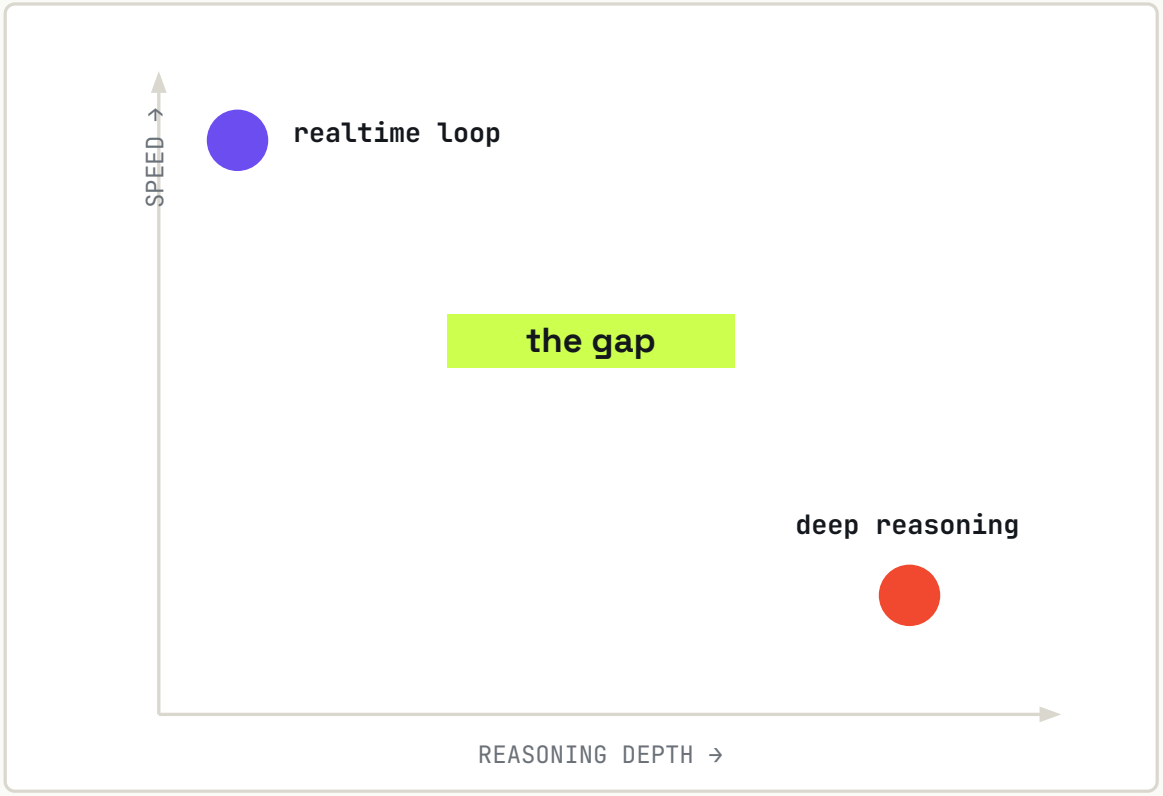
Real systems are an orchestrated chain of services — representative tools per layer.

Transport media in / out	LiveKit · Twilio · Daily · Vonage · WebRTC / SIP
Turn detection VAD · endpointing	Silero VAD · WebRTC VAD · LiveKit turn-detector
STT speech → text	Deepgram · AssemblyAI · Whisper · Azure · Google
LLM reason + tools	OpenAI · Anthropic · Gemini · Llama / vLLM · Groq
TTS text → speech	ElevenLabs · Cartesia · Deepgram Aura · PlayHT · Rime
Orchestration frameworks + backend	Pipecat · LiveKit Agents · Vapi · Retell · + CRM / calendar / DB

Not one model — a realtime orchestration of many specialised services.

Latency vs. intelligence

Fast answers aren't always good. Good reasoning isn't always fast. This is the core tension.



Push everything inline

Latency balloons; the agent stutters and pauses.

Keep the loop shallow

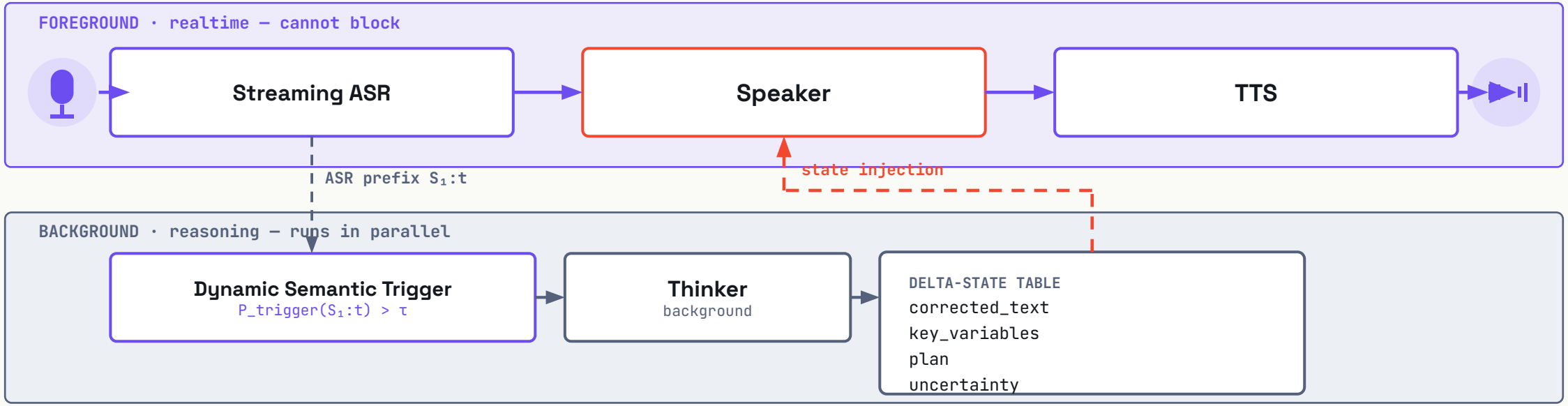
Fast, but it misses intent, policy and risk.

The tension

You can't simply pick one — voice needs both speed and depth.

LTS Architecture Move: State Injection, Not Just Parallel Reasoning

The ASR prefix stream triggers background state updates; the Speaker consumes the latest valid state.



The Speaker doesn't reason from raw transcript fragments — it speaks from the latest injected state.

Dynamic Semantic Trigger

Don’t think on every chunk — think when the prefix becomes meaningful.

VAD

“Is there speech?”

Endpointing

“Did the user stop?”

Semantic Trigger

“Is there enough meaning to think?”

ASR PREFIX STREAM

“I…”	no trigger
“I need…”	weak signal
“I need to reschedule…”	TRIGGER
“…my appointment”	update state

The point: not to think earlier on every chunk — to think at the right semantic moment.

Thinker: delta-state update

The Thinker doesn't speak — it maintains a structured state, and corrections overwrite it.

STATE v1

```
intent:    reschedule_appointment
appt_id:   null
new_time:  null
plan:      ask for new date
missing:   appt_id, new_time
```

"also change the address" →

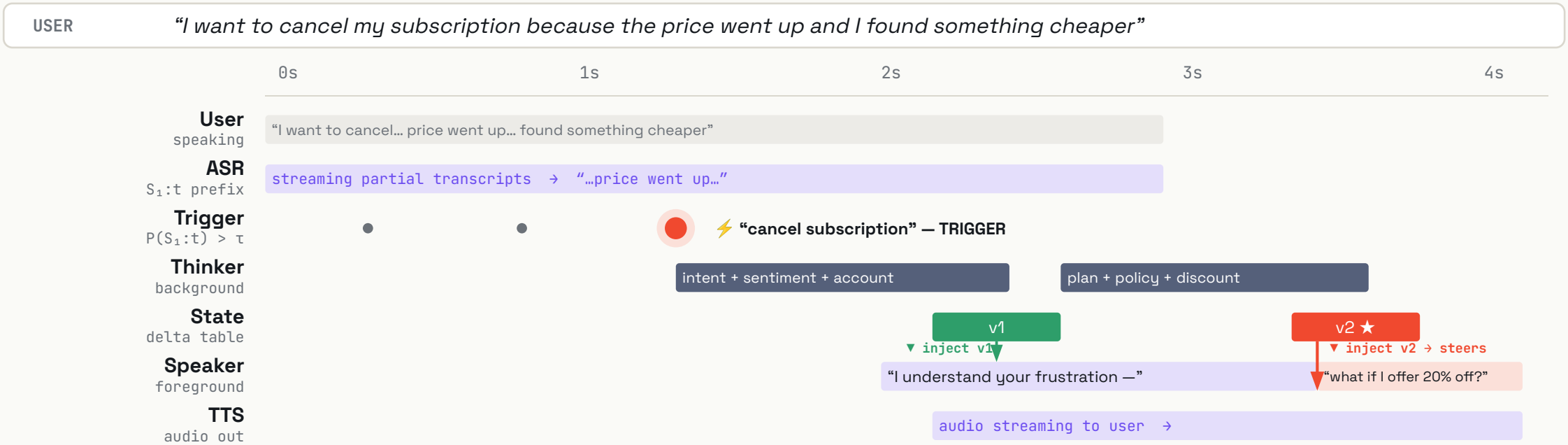
STATE v2

```
intent:    reschedule + change_address
missing:   new_address, new_time
plan:      new address first, then reschedule
```

Corrections overwrite variables — **the Thinker owns durable conversation state.**

Speaker starts early, then new state steers mid-generation

The Speaker starts on partial state (v1), then adjusts mid-utterance when richer state (v2) arrives.



At **v1** the Speaker shows empathy; at **v2** it steers mid-utterance to a retention offer — **the Speaker never stopped.**

Dual-Role Stream Orchestrator

One coordinator runs two independent roles — and decides what survives a shift.

ORCHESTRATOR — COORDINATES BOTH ROLES

- starts Thinker on semantic trigger
- injects state into Speaker
- cancels Speaker on semantic shift
- preserves valid Thinker state
- restarts response from updated context

Thinker

background state maintenance

Speaker

foreground spoken response

LTS is not just streaming — **it is stateful stream orchestration.**

Barge-in: kill the Speaker, keep the Thinker

When the caller interrupts, the Speaker is cancelled — but the Thinker state survives.

WHAT THE CALLER HEARS

USER
"…reschedule my appointment…"

SPEAKER v1
"Of course, what day—" ✕ aborted

USER · barge-in
"actually, also change the address"

SPEAKER v2
"Got it — let's update the address first."

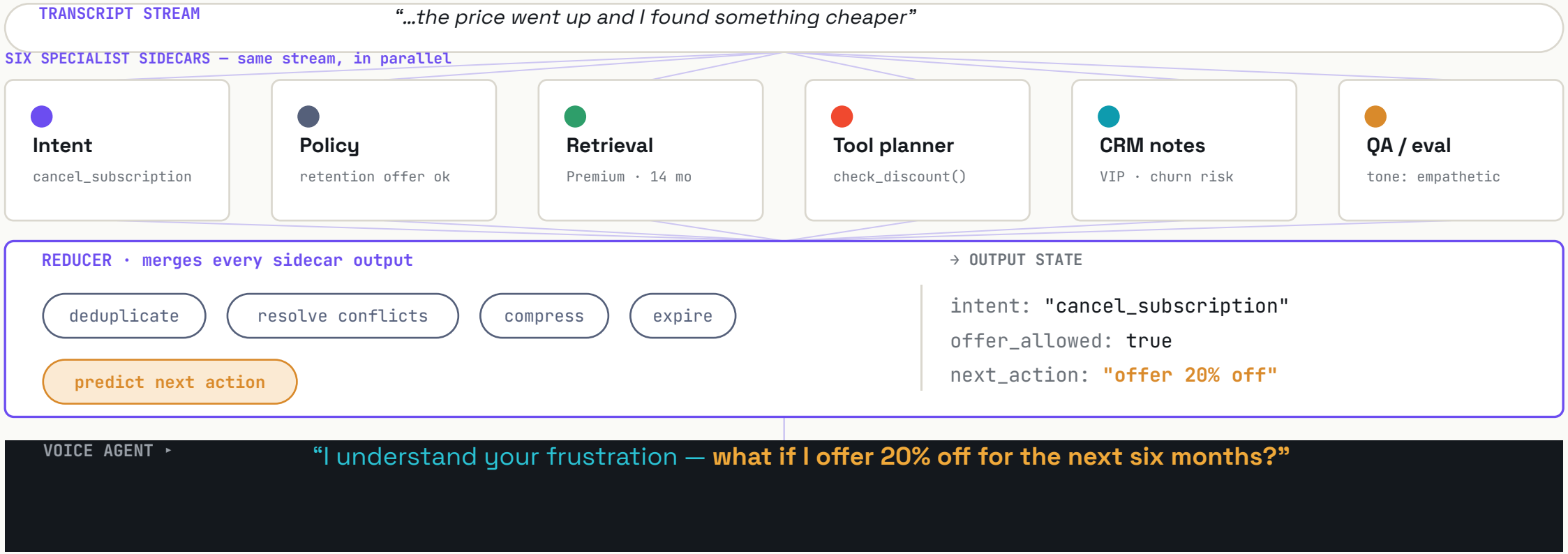
ORCHESTRATOR · ON BARGE-IN

1. abort Speaker — cancel TTS immediately
2. preserve Thinker state (don't restart)
3. merge the new intent into state
4. restart Speaker from the updated state

The Speaker is disposable, the Thinker state is durable — interruptions cancel speech, not state.

>_ FROM LTS TO YOUR ARCHITECTURE

From one Thinker to many sidecars



Freshness: injected state has a shelf life

Guidance that arrives too late is worse than none — match it to the turn.

TTL

Expire by age

Drop guidance older than ~N ms.

TURN MATCH

Tag the turn

State carries a turn-id; ignore on mismatch.

SUPERSEDE

Newest wins

A fresher snapshot replaces the stale one.

INJECT ONLY INSIDE THE WINDOW

Inject only fresh, turn-matched state — expire the rest.

Thank you.

Voice agents that think while they talk.

Bohdan Snisar

bohdan@solea.ai · solea.ai