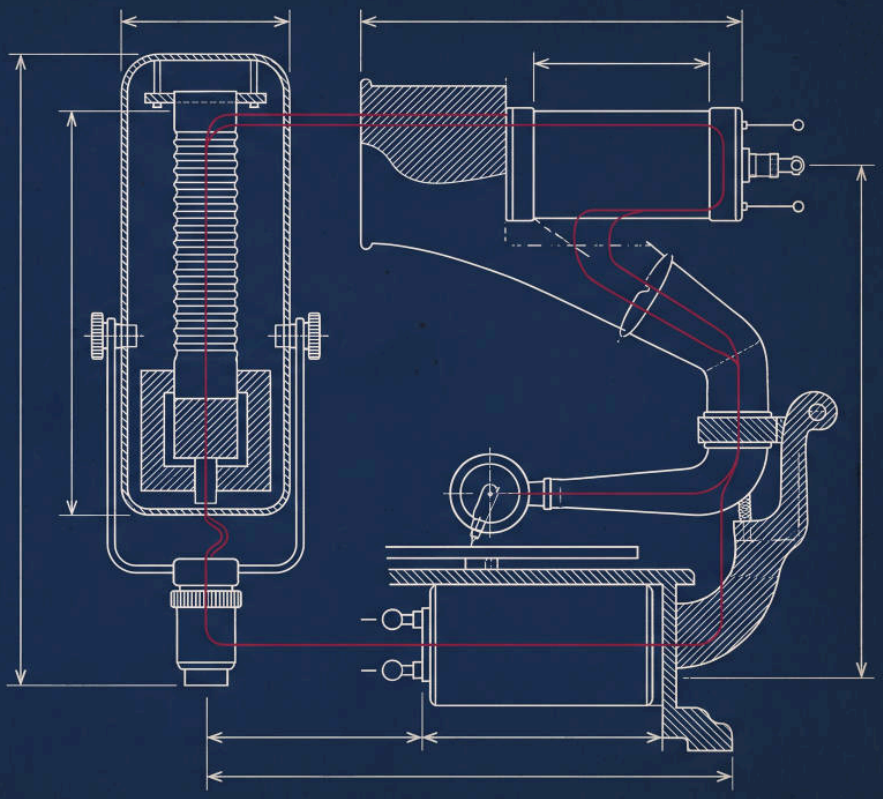

CHAPTER THIRTEEN

Voice and Conversational Agents

SECTIONS 13.1 TO 13.10

Text tolerates a slow answer. Conversation does not. Silence past roughly eight hundred milliseconds reads as a broken system, and the whole pipeline of Parts I to III has to fit inside it.

CHAPTER THIRTEEN

Contents

PART III · AGENTS

Every interface so far has been text, where a slow answer is an annoyance. Conversation removes that tolerance. Silence past roughly eight hundred milliseconds reads as a broken system, and the entire pipeline built across Parts I to III has to fit inside it.

13.1 The Latency Budget of a Conversation

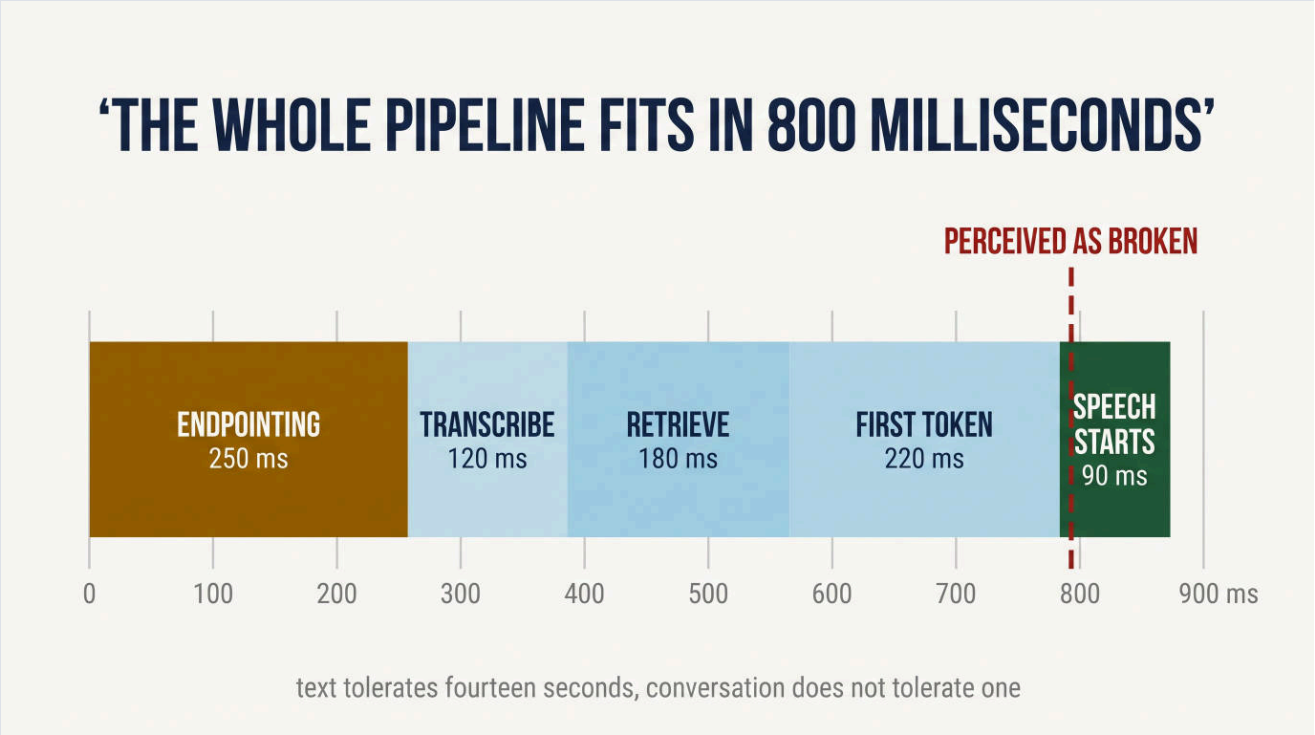

Figure 13.1 Indicative allocation. Every stage of Parts I to III competes for the same budget.

Human turn-taking runs on gaps of roughly two hundred milliseconds. A gap beyond about eight hundred stops reading as thinking and starts reading as failure: the caller says "hello?", talks over the reply, or hangs up. That number is the specification, and it is brutal.

Stage	Typical	Where it comes from
Endpointing	200 to 300 ms	Section 13.2. Usually the largest single slice, and the least discussed.
Transcription	80 to 150 ms	Streaming, so most of it overlaps the speech.
Retrieval	100 to 250 ms	Chapter 7's hybrid plus reranking, under real pressure.
Time to first token	150 to 400 ms	Section 2.11's prefill, which is why prompt length is now a latency decision.
First audio	60 to 120 ms	Synthesis of the first sentence only.

Only time to *first audio* matters. Total answer length is nearly free. This inverts section 2.11's finding for text, where output length dominated total latency. In voice, the user is listening while the rest generates, so a longer answer costs nothing perceptually and a slow start costs everything. Optimise the first sentence and stop optimising the rest.

Two consequences follow immediately. **Reranking may not fit**: section 7.7's cross-encoder at 150 milliseconds is a fifth of the budget, so voice often runs a cheaper retrieval than the text product on the same corpus. And **the prompt must be shorter**, because prefill is quadratic and sits directly in the critical path.

The table above describes a cascade: speech to text, text through the pipeline of Parts I to III, text back to speech. In 2026 that is a decision rather than a default. Native-audio realtime models take audio in and produce audio out with no text in between, and they beat the eight hundred millisecond budget easily, without any of the engineering this chapter teaches.

What the cascade buys is the text seam. Every control this book has built lives in text: chapter 6's grounding checks compare a text answer against text evidence, the output policy inspects a string before it leaves, citations attach to sentences, and the audit trail records what was said in a form a reviewer can read. Collapse the cascade and that seam disappears. You can still record the audio, but you cannot run a grounding check on a waveform in the middle of a turn.

So the trade is concrete. Native audio wins the latency budget for free and pays in controllability. The cascade keeps

every check you have built and pays in latency engineering, which is what the rest of this chapter is. Neither is right in general; the cost of a wrong answer decides.

The hybrid is often where products land: a native-audio model runs the conversational shell, the greeting, the turn-taking, the small talk, and anything consequential becomes a tool call into the text path, where retrieval, grounding and the output policy still apply. The seam moves from between every stage to around the decisions that need it. Measure how many turns actually cross it; if most do, you have rebuilt the cascade with extra steps.

13.2 Turn Detection and Endpointing

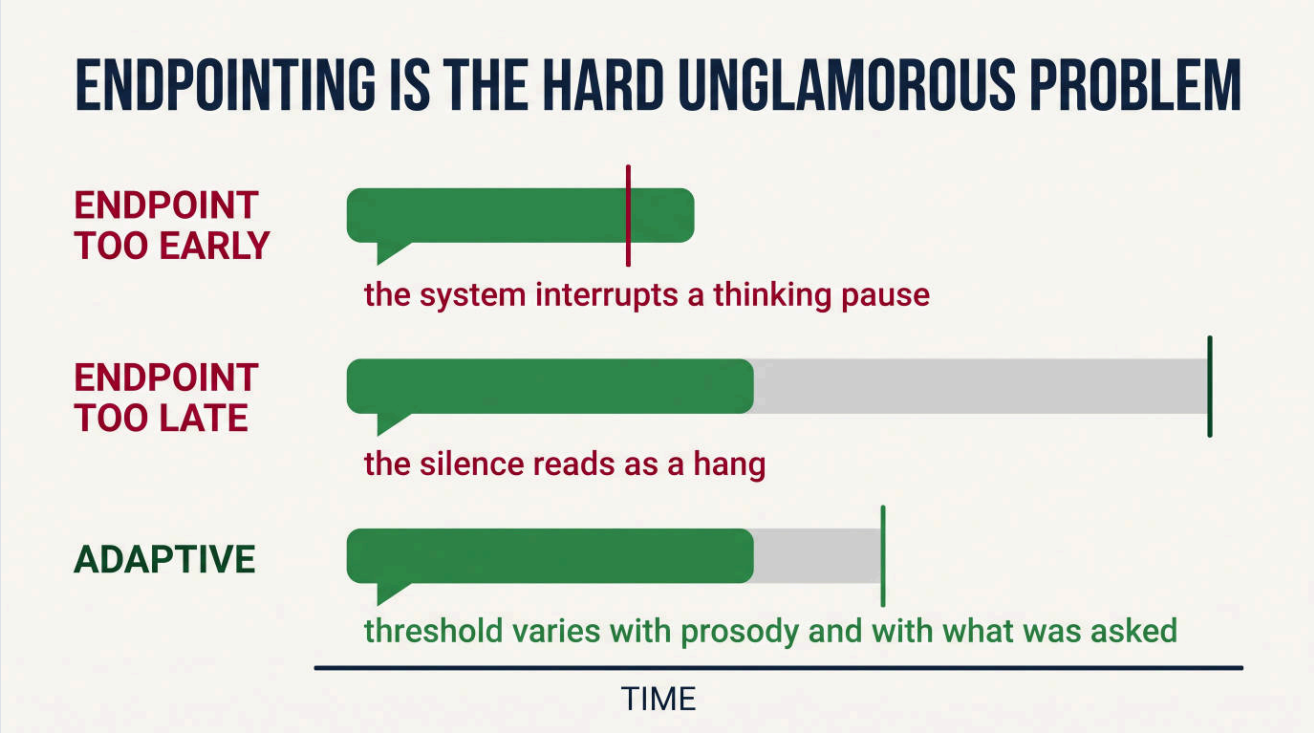

Figure 13.2 Both errors are worse than they look, and the correct threshold is not a constant.

Endpointing is deciding that the user has finished speaking. It is the largest slice of the budget, it receives the least design attention, and both of its failure modes are severe.

Too early and the system interrupts a thinking pause. The user was mid-thought, the assistant answers half a question, and the recovery costs a full turn.

Too late and the silence reads as a hang. The user repeats themselves, which now produces a doubled transcript and an answer to a question asked twice.

A fixed silence threshold cannot serve both. Three signals do better, and none requires a model:

Signal	What it contributes
Prosody	A falling pitch contour signals completion; a level or rising one signals continuation. Cheap and language-dependent.
Syntactic completeness	A partial transcript ending mid-clause is not a turn. The streaming transcript already gives you this for free.
What was asked	After "what is your account number", expect digits and a longer pause. After a yes-or-no question, endpoint aggressively. The *system* knows what it asked, so this is state you already hold.

The third is the one to build first. It costs nothing, it is deterministic, and it is unavailable to any general-purpose endpointer because only your graph knows what question it just asked.

13.3 Streaming the Whole Pipeline

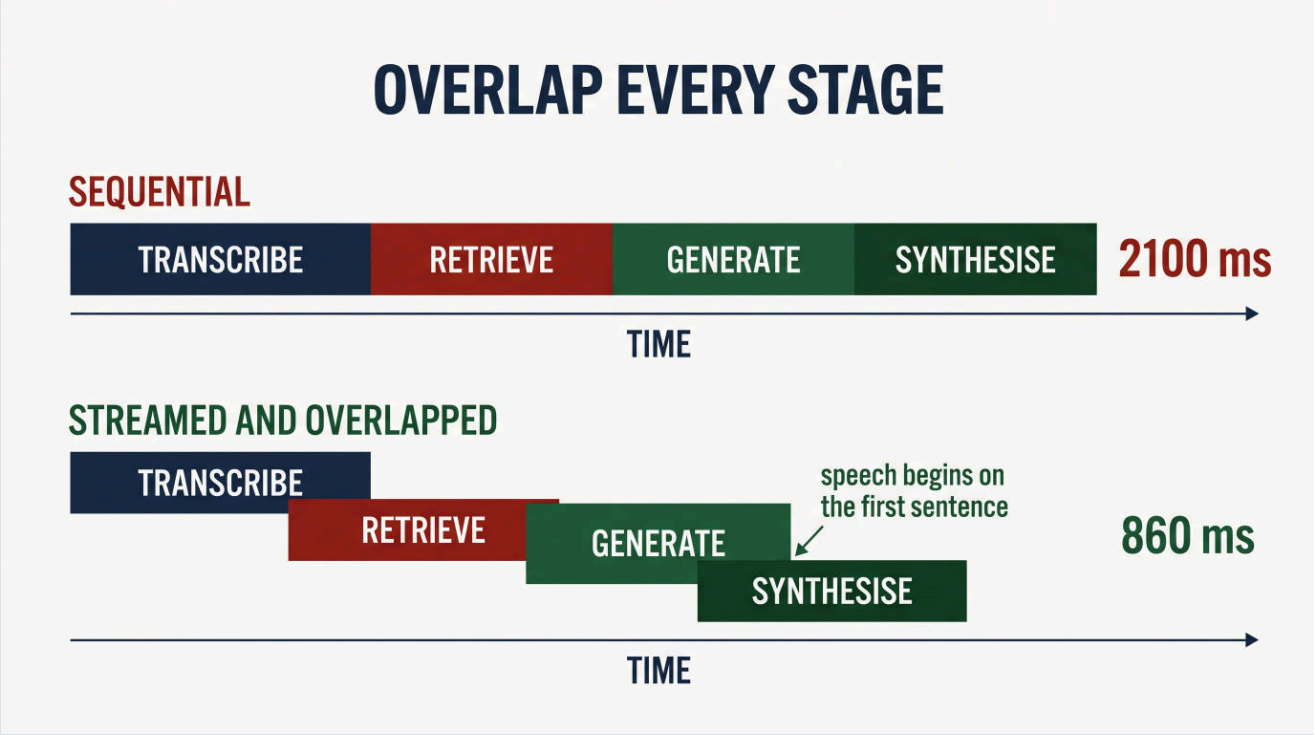

Figure 13.3 The same work, sequenced two ways. Only one of them is a conversation.

Run the stages end to end and the budget is gone before generation starts. The architecture is not "make each stage faster", it is **make each stage begin before the previous one finishes**.

```
# Retrieval starts on the PARTIAL transcript, not the final one.
async for partial in stt.stream(audio):
    if partial.is_stable and looks_like_a_question(partial.text):
        retrieval_task = asyncio.create_task(retrieve(partial.text))

# When the turn ends, retrieval is often already done.
# close_enough: normalised token overlap above a threshold set from logged transcript pairs.
final = await stt.final()
evidence = await retrieval_task if close_enough(final, partial) else await retrieve(final)

# Synthesis starts on the first complete SENTENCE, not the whole answer.
async for sentence in sentences(gateway.stream(messages=msgs)):
    await tts.speak(sentence)
```

Two details in that sketch matter more than the structure.

Speculative retrieval on a partial transcript is usually right and occasionally wasted. The waste is a retrieval you discard; the win is two hundred milliseconds off every turn. Guard it with a similarity check so a materially different final transcript re-retrieves.

Sentence-boundary synthesis, not token-boundary. Speech synthesised per token sounds wrong, because prosody spans a clause. The first sentence is the entire latency win, and everything after it is free.

13.4 Barge-In and Interruption

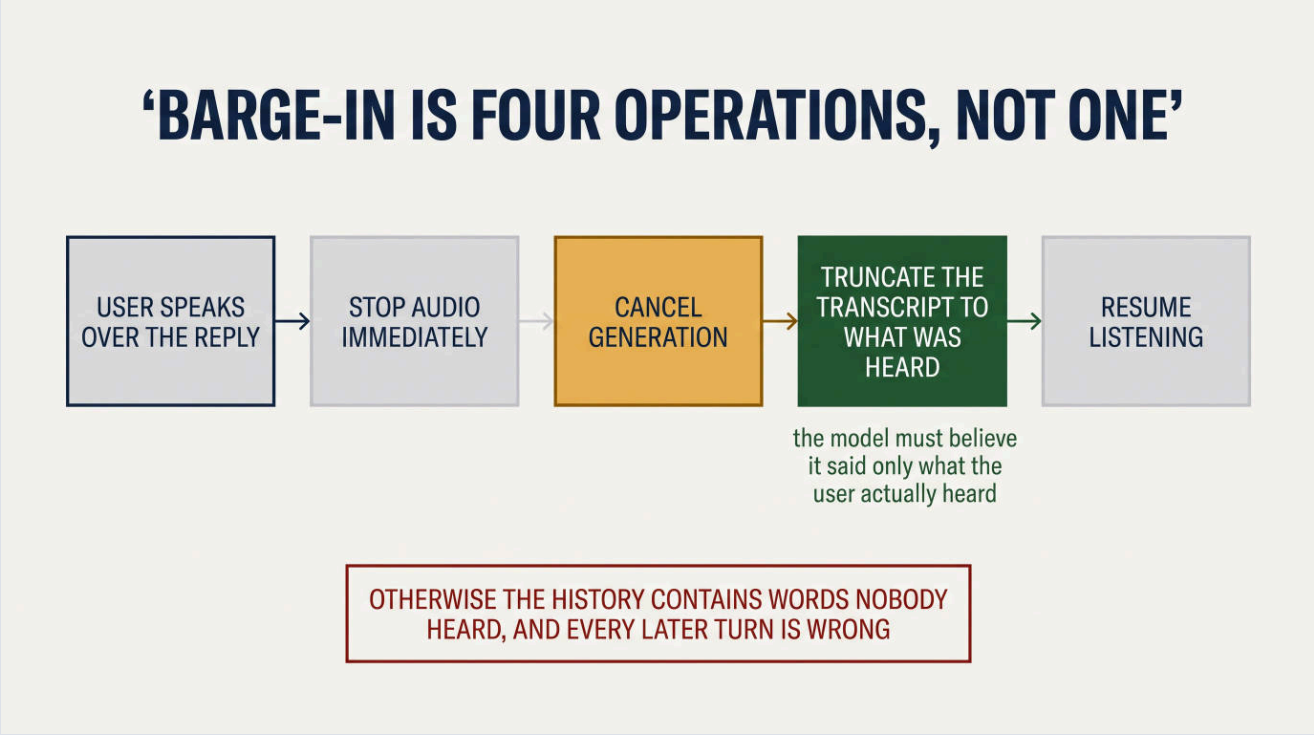

Figure 13.4 The fourth operation is the one everyone forgets, and it corrupts every later turn.

Users interrupt. Handling it well is table stakes; handling it correctly is subtler than it appears, and the subtlety is in the state rather than the audio.

Stopping playback and cancelling generation are obvious. The third and fourth are not:

```
# Truncate the assistant turn to what was ACTUALLY HEARD.
spoken = tts.bytes_played()          # not what was generated
history.append(Turn(role=ASSISTANT, text=text_for(spoken), truncated=True))
```

> THE FAILURE THAT POISONS EVERY LATER TURN
>
> If the transcript records the full generated answer while the user heard only the first eight words, the conversation history now contains words nobody heard. The model will refer back to them: "as I mentioned, the cap is 30 days." The user never heard that, and the assistant appears to be inventing its own past.
>
> This is a memory-correctness bug wearing an audio costume, and section 11.3's point applies: it is silent, and no metric reports it.

Cancelling generation also connects back to Chapter 3: the upstream call must actually be cancelled, not merely ignored, or you pay for tokens nobody will hear. Section 3.5 made that a cost control for text; here it fires on a large fraction of turns.

13.5 TTS and the First Sentence

Synthesis is the last stage and the one with the most headroom, because only its beginning is on the critical path.

Three decisions, in order of impact:

Synthesise the first sentence with the fastest available voice, and the rest with the best one. The switch is inaudible if the voices match, and it buys the difference between a fast model's latency and a good model's quality without choosing between them.

Constrain the first sentence in the prompt. "Begin with a direct one-sentence answer, then elaborate" is a section 5.1 checkable constraint, and in voice it is a latency control: a first sentence that is a preamble wastes the entire streaming advantage.

Pre-synthesise the fillers. "Let me check that" and "One moment" are fixed strings, so they can be cached as audio and played at zero latency while retrieval runs. Used sparingly this is honest and buys three hundred milliseconds; used on every turn it becomes a verbal tic.

Numbers, dates and identifiers must be normalised before synthesis, not after. "A-1042/B" read as "a dash one thousand and forty two slash b" is a support call. Section 7.5's tokenizer discussion has an audio twin: the exact identifiers that matter most are the ones both the recogniser and the synthesiser handle worst.

13.6 Grounding a Voice Answer

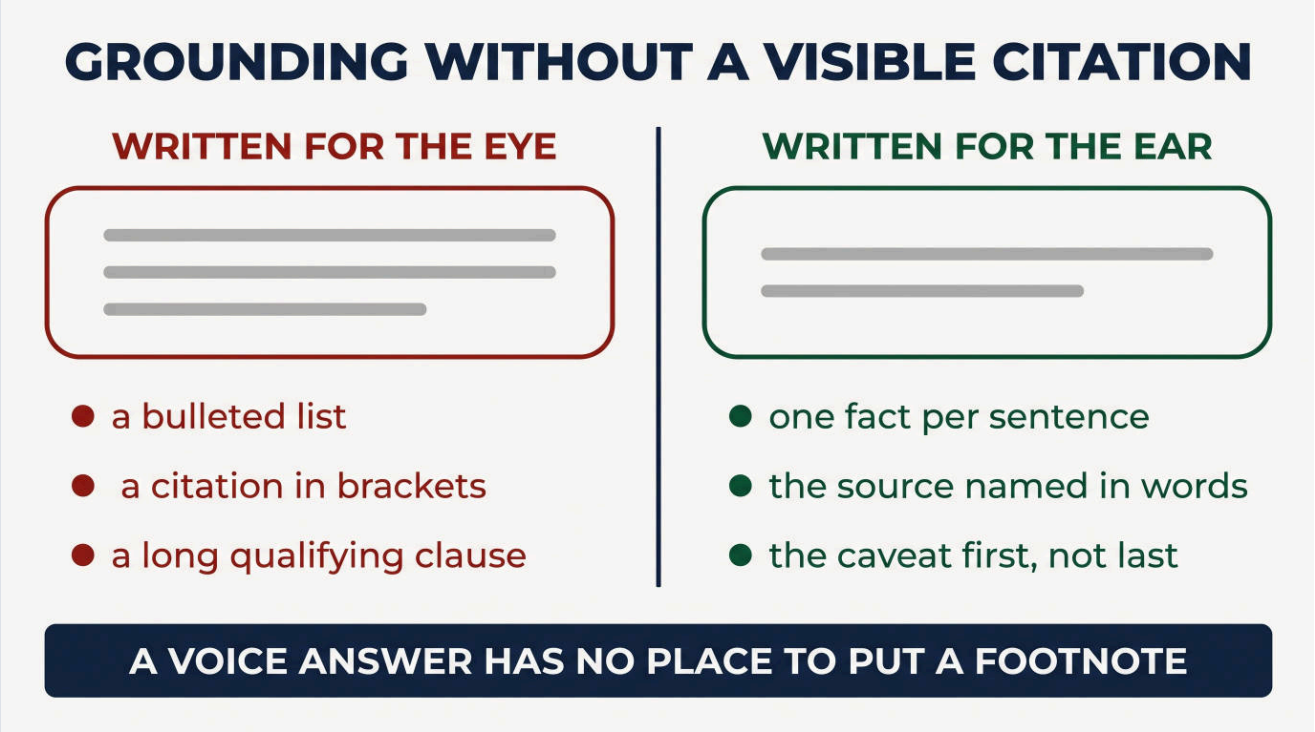

Figure 13.5 Chapter 6's citations have nowhere to render, so the grounding has to move into the sentence.

Chapter 6 made citations the mechanism by which a user can check a claim. Voice removes the surface: there is no bracket to render, no link to follow, and no page to open.

Three adaptations, and they are product decisions as much as engineering ones.

Name the source in words, sparingly. "According to the master agreement, sixty days" carries provenance in a form the ear accepts. Naming a source on every sentence is unbearable; naming it on the load-bearing claim is not.

Put the caveat first. In text a qualifier at the end is fine because the reader sees the whole answer at once. In speech, a listener may act on the first clause before the qualifier arrives. "This is from the 2023 version, and the cap is 30 days" is safe; the reverse order is not.

Send the citation to a second channel. If there is a screen, a transcript, or a follow-up email, the verifiable citation belongs there. The voice carries the answer; the channel carries the proof.

Abstention has to sound different

Section 6.9 argued that an abstention must be visibly distinct. In voice, "I could not find that" delivered in the same tone and cadence as an answer is heard as an answer. Make it audibly different: shorter, a distinct opening, and always followed by what the system *did* search. "I checked the master agreement and the two amendments and found nothing about notice periods" is actionable in a way that "I'm not sure" is not.

13.7 Errors You Cannot See

Figure 13.6 Four voice-specific failures, none of which appears in a text log as an error.

Every failure mode in Parts I to III still applies. Voice adds four of its own, and their common property is that they are *invisible to the observability you already built*.

Failure	Why the trace looks healthy
Mistranscribed identifier	The pipeline received a valid string and answered it correctly. The trace shows a successful, well-grounded answer to the wrong question.
Speaker confusion	On a shared line or a handover, turns from two people merge into one history. Every downstream component behaves correctly on corrupt input.
Barge-in truncation	Section 13.4. The history diverges from what was heard, and nothing compares the two.
Abstention heard as answer	The system correctly abstained. The metric records an abstention. The user acted as though it were an answer.

The mitigation that covers three of the four is **carrying transcription confidence into the pipeline as state**, rather than discarding it at the STT boundary.

```
# Confidence is not a log field, it is an input to the decision.
if turn.stt_confidence < 0.7 and contains_identifier(turn.text):
    return confirm(f"Did you say {spell_out(identifier)}?")
```

Confirming a low-confidence identifier costs one turn. Answering the wrong invoice costs a support case and the user's trust in every answer that follows. Section 6.9's asymmetry between a confident wrong answer and a refusal is sharper here, because the user cannot see the input the system worked from.

The threshold itself is population-dependent. Background noise, accents and code-switching all depress confidence, so a cutoff tuned on clean office audio confirms constantly in a warehouse and never protects a quiet line; evaluate it per deployment population, not globally. Speaker confusion has a partial mitigation too: diarisation can separate voices on a shared line, but its errors are of exactly the same invisible kind and belong on the same list.

13.8 Telephony Constraints

A phone call is not a WebRTC session with worse audio. Four constraints change the architecture rather than degrading it.

Constraint	Consequence
Narrowband audio	Roughly 300 to 3400 Hz. Recognition accuracy on digits and letters falls, which is exactly the material section 13.7 said you cannot afford to get wrong.
No visual channel	Nowhere to send the citation from section 13.6, no way to render a form, no place for a disambiguation list. Everything must be sayable.
Hard session limits	Carriers time out. A long approval pause (section 10.7) cannot hold a call open, so an interrupt has to become a callback.
Regulatory recording	Consent, retention and redaction obligations from section 15.4 attach to the audio itself, not only to the transcript.

The third is the one that reshapes the graph. In a text product an interrupt suspends and waits. On a call, **the interrupt must resolve within the call or become an asynchronous commitment**: "I have sent that to a colleague for approval and we will call you back." Chapter 10's checkpoint makes the second option cheap, because the run is already a durable row rather than a held thread.

Transport is a decision that comes before any of these constraints. For browser and app clients, WebRTC is the default for a reason: jitter buffers, packet loss concealment and echo cancellation come with it, and a raw WebSocket carrying audio frames gives you none of them, so you either rebuild them badly or ship a client that stutters on any imperfect network. Telephony arrives over SIP trunking through a carrier or a CPaaS provider, which is where the four constraints above enter. Either way, the transport's own latency sits inside section 13.1's budget; it is not free time before the pipeline starts.

The first constraint has a standard fallback the industry solved decades ago: for exactly the identifiers narrowband mangles, offer keypad entry, because DTMF digits survive any line and callers already know how to use them.

The fourth constraint has a sibling. Several jurisdictions now require disclosing that the caller is speaking with an automated system, not only that the call is recorded. Treat it the same way: a per-jurisdiction flag in the call policy, with the disclosure designed into the greeting rather than bolted on.

13.9 Measuring Voice Quality

The text metrics from Chapter 14 still apply to the answer. Voice needs five more, and the first two are the ones to alert on.

Metric	What it catches
Time to first audio, p95	The only latency number that matters, per section 13.1. Total answer duration is not a quality metric.
Barge-in rate	Rising means answers are too long, too slow to start, or wrong. It is the cheapest continuous signal a voice product has.
Endpoint error rate	Sampled and human-labelled: too early, too late, correct. Section 13.2's two failure modes need separating.
Repeat rate	The user saying the same thing twice. Catches endpointing failures and mistranscription together.
Confirmation rate	How often section 13.7's low-confidence path fires. Too low means the threshold is not protecting anything; too high means the recogniser is the problem.

Sample and listen to complete calls weekly. This is the voice equivalent of section 7.2's failure labelling, and it is equally unglamorous and equally the thing that actually improves the product. Reading transcripts is not a substitute, because three of section 13.7's four failures do not appear in one.

Voice also changes the shape of the bill. A voice stack is metered per minute in three or four places at once: transcription per audio minute, generation per token, synthesis per character, and telephony per connected minute. None of these is the number to manage. The number is cost per resolved call, the same discipline as section 14.8's cost per successful task: a cheaper recogniser that raises the repeat rate lengthens calls and loses money, and a longer answer is nearly free in latency per section 13.1 but is not free on the synthesis and telephony meters. Track the per-minute meters as inputs and gate on the per-resolution number.

13.10 Three Voice Architectures Compared

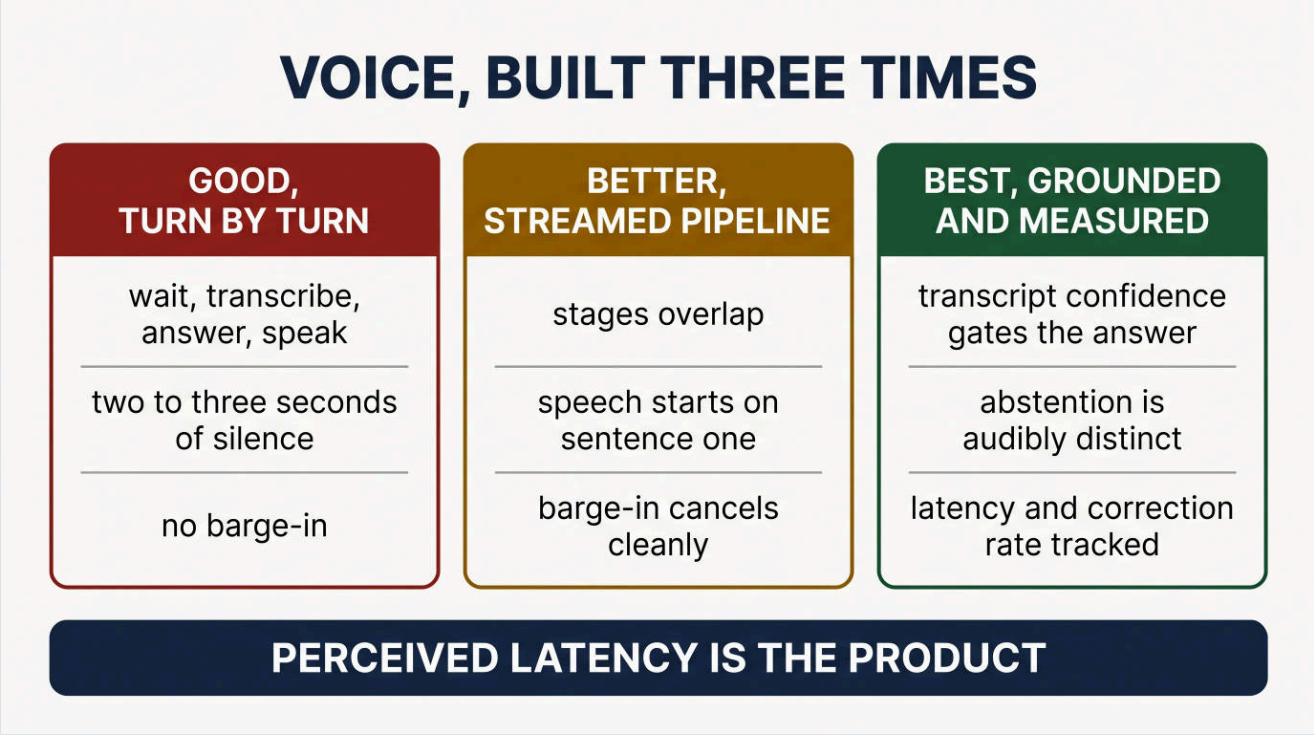

Figure 13.7 Perceived latency is the product, and it is decided by the first sentence.

GOOD

Turn by turn

Wait for silence, transcribe, retrieve, generate, synthesise, play. Every stage complete before the next begins. Two to three seconds of silence per turn, no barge-in, and the user learns to wait or gives up.

It is correct for one situation: a voice interface where turns are rare and long, such as dictation with occasional commands. As conversation it does not work, and no amount of per-stage optimisation makes it work, because the stages are additive.

BETTER

Streamed and interruptible

Sections 13.3 and 13.4: overlapping stages, speculative retrieval on partial transcripts, sentence-boundary synthesis, and barge-in that cancels generation and truncates the transcript to what was heard.

What it still does not solve. Transcription confidence is discarded, so section 13.7's invisible errors are all live. Abstention sounds like an answer. And endpointing is a fixed threshold, so it is wrong in both directions depending on the question.

BEST

Grounded, confidence-aware and measured

Everything above, plus confidence carried into the decision rather than logged, question-aware endpointing, audibly distinct abstention, citations routed to a second channel, and the five metrics of section 13.9 with alerts on the first two.

Axis	Good: turn by turn	Better: streamed	Best: grounded and measured
Time to first audio	2 to 3 s	Under 800 ms	Under 800 ms
Interruption	Not supported	Cancels and truncates	Cancels, truncates, measured
Mistranscription	Answered confidently	Answered confidently	Confirmed when confidence is low
Abstention	Sounds like an answer	Sounds like an answer	Audibly distinct, says what was searched
Endpointing	Fixed threshold	Fixed threshold	Varies with the question asked
Build cost	One to two weeks	Four to six weeks	Eight to twelve weeks

All three tiers are cascades. A native-audio realtime model (section 13.1) meets the latency row of the best tier on day one and fails the mistranscription, abstention and grounding rows differently, because those controls live at the text seam it removes; compare it as a separate architecture, not as a shortcut to the third tier.

PROMOTION TRIGGERS

Good to better, immediately, if the product is conversational at all. This is not a maturity step; a turn-by-turn conversational agent is a prototype regardless of how well each stage performs.

Better to best, at the first of: the system handles identifiers, numbers or names spoken aloud; a wrong answer has a cost beyond annoyance; or the barge-in rate from section 13.9 is above roughly one turn in five, which usually means answers are wrong rather than long.

What Chapter Thirteen Establishes

What exists now	What later chapters put through it
Time to first audio as the latency metric	SLOs and alerting *(16.7)*
Confidence carried into decisions	Abstention thresholds under evaluation *(14.5)*
Truncated transcripts on barge-in	Audit trails that match what was heard *(15.8)*
Interrupt becoming a callback	Human handoff patterns *(15.5)*
Sampled call review	Human review that scales *(14.10)*

Chapter checklist

Can you...	Section
State the budget and say which latency metric is the only one that matters	13.1
Explain why voice inverts the text finding about output length	13.1, 2.11
Name both endpointing failure modes and the signal only your system has	13.2
Say why retrieval starts on a partial transcript, and how to guard it	13.3
Give the four operations of barge-in and name the one that corrupts state	13.4
Explain why the caveat comes first in speech and last in text	13.6
Name four voice failures that a healthy trace will not show	13.7
Say what a telephony session limit does to a Chapter 10 interrupt	13.8
Give the two voice metrics worth alerting on, and why	13.9

What Chapter 14 builds

Part IV begins, and it begins with the discipline every preceding chapter has been deferring to. Chapter 2 established that correctness is statistical, so testing is sampling. Chapters 5 to 13 each pointed at an eval set as the thing that makes a decision defensible. Chapter 14 builds it.

Why a pass-or-fail test is the wrong unit of confidence. Building a case set from real traffic and from the dead letter queue rather than from imagination. Deterministic scorers before model judges, and how to calibrate a judge that will otherwise drift alongside the system it grades. Continuous regression detection against production traffic, because a release-time gate cannot catch a provider changing a model underneath you. And the gate itself: which thresholds, at what tolerance, and why injection resistance is the one metric with none.