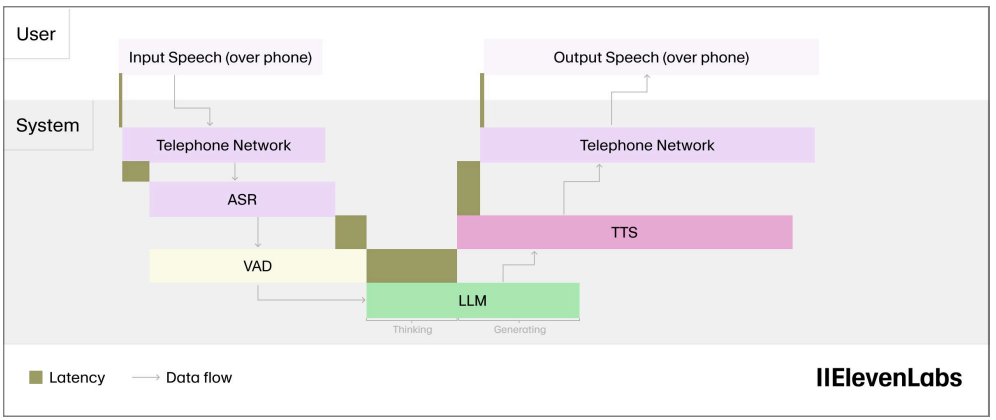
IIElevenLabs

Latency in Conversational AI

How do you optimize latency for Conversational AI?

Latency is what separates good Conversational AI applications from great ones

Oswin Kruger Ruiz - Forward Deployed Engineer



For most applications, latency is a mild concern. However, for conversational AI, latency is what separates good applications from great ones.

For starters, the goal of conversational AI is fairly aspirational—to provide the same feeling, touch, and voice as a human conversation, while surpassing a human in intelligence. To accomplish this, an application must converse without long silent gaps. Otherwise, the realism is shattered.

Conversational AI's latency challenge is compounded by its piecemeal nature. Conversational AI is a series of intermediate processes, all considered state-of-the-art in their respective fields. Each of these processes incurs additive latency.

As a generative voice company, we've spent a long time studying how to minimize latency for conversational AI. Today, we want to share our learnings, out of hopes that they'll be helpful for anyone interested in building conversational AI applications.

The Four Core Components

Every conversational AI application involves at least four steps: speech-to-text, turn-taking, text processing (i.e. LLMs), and text-to-speech. While these steps are executed in-parallel, each step still contributes some latency.

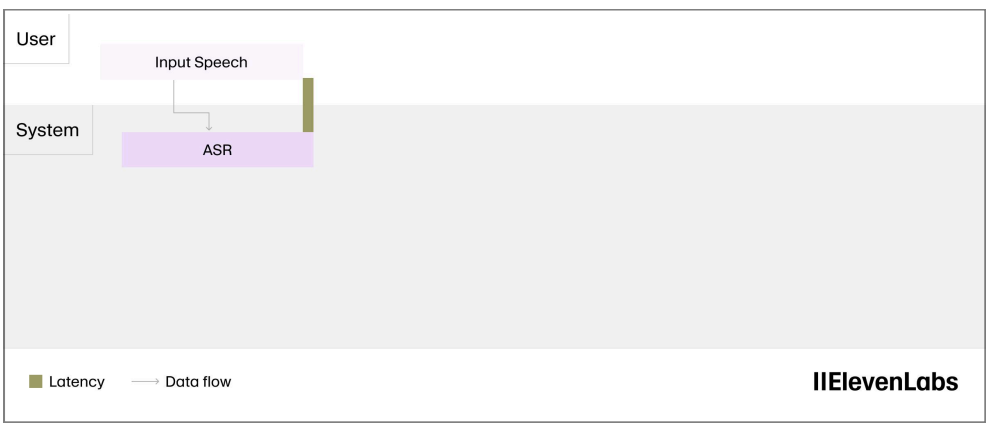
Notably, conversational AI's latency equation is unique. A lot of process latency problems can be reduced to a single bottleneck. For instance, when a website makes a database request, the web's network latency drive the total latency, with just trivial contributions from the backend's VPC latency. However, conversational AI's latency components aren't drastically varied. They are uneven, but each component's latency contribution is within a degree of the others. Accordingly, latency is driven by a sum of parts.

Automatic Speech Recognition

The System's "Ear"

Automatic speech recognition (ASR)—sometimes referred to as speech-to-text (STT)—is the process of converting spoken audio into written text.

ASR's latency is not the time it takes to generate text, as the speech-to-text process runs in the background while the user speaks. Instead, the latency is the time between the end of the speech and the end of the text generation.



Accordingly, short and long speaking intervals can incur similar ASR latency. Latency can vary across ASR implementations (In some cases, there is no network latency whatsoever as the model is embedded in the browser, such as Chrome/Chromium). The standard open source model, Whisper, adds 300ms + latency. Our custom implementation adds <100ms.

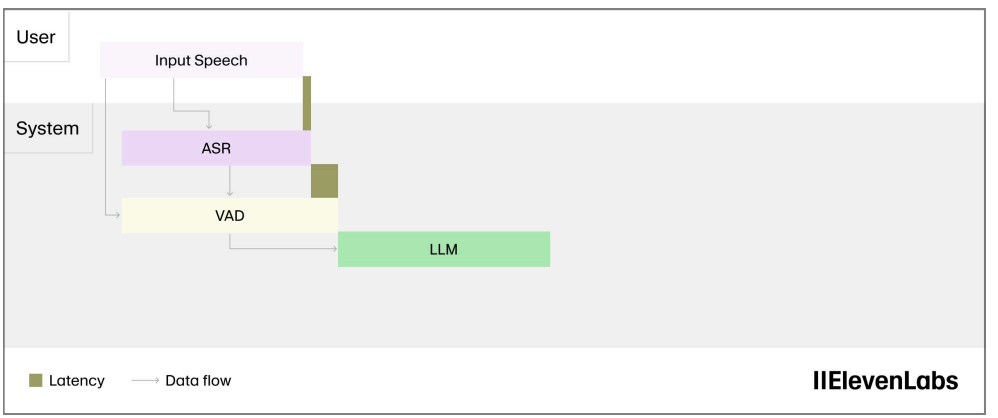
Turn-Taking / Interruption

The System's "Right Brain"

Turn-taking / Interruption (TTI) is an intermediary process that determines when a user is finished speaking. The underlying model is known as a Voice Activity Detector (or VAD).

Turn-Taking involves a complex set of rules. A short burst of speech (e.g. "uh-huh") shouldn't trigger a turn; otherwise, conversations would feel too staccato. Instead, it must assess when a user is actually trying to fetch the model's attention. It also must determine when the user is finished relaying their thoughts.

A good VAD will not signal a new turn whenever it detects silence. There is silence between words (and phrases), and the model needs to be confident that the user is actually done speaking. To accomplish this reliably, it needs to seek a threshold of silence (or more specifically, a lack of speech). This process introduces a delay, contributing to the overall latency experienced by the user.



Technically speaking, if all of the other conversational AI components incurred zero latency, the latency attributed to TTI would be a good thing. Humans take a beat before responding to speech. A machine taking a similar pause gives realism to the interaction. However, since other components of conversational AI already incur latency, minimal TTI latency is ideal.

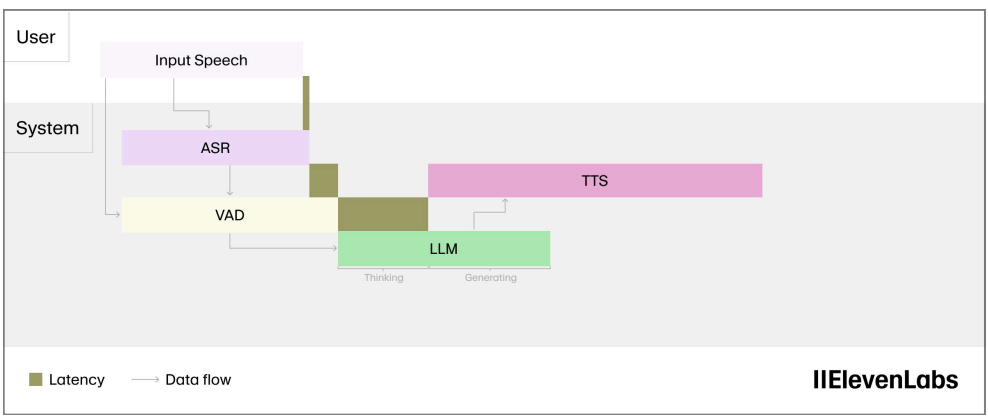
Text Processing

The System's "Left Brain"

Next, the system needs to generate a response. Today, this is typically accomplished with a Large Language Model (LLM), such as GPT-4 or Gemini Flash 1.5.

The choice of the language model makes a significant difference. Models like Gemini Flash 1.5 are incredibly fast—generating outputs in less than 350ms. More robust models that can handle more complex queries—such as GPT-4 variants and Claude—could take between 700ms to 1000ms. Choosing the right model is typically the easiest way to target latency when optimizing a conversational AI process.

However, the latency of the LLM is time it takes to start generating tokens. These tokens can be immediately streamed to the following text-to-speech process. Because text-to-speech is slowed by the natural pace of a human voice, the LLM reliably outpaces it—what matters most is the first token latency (i.e., time to first byte).

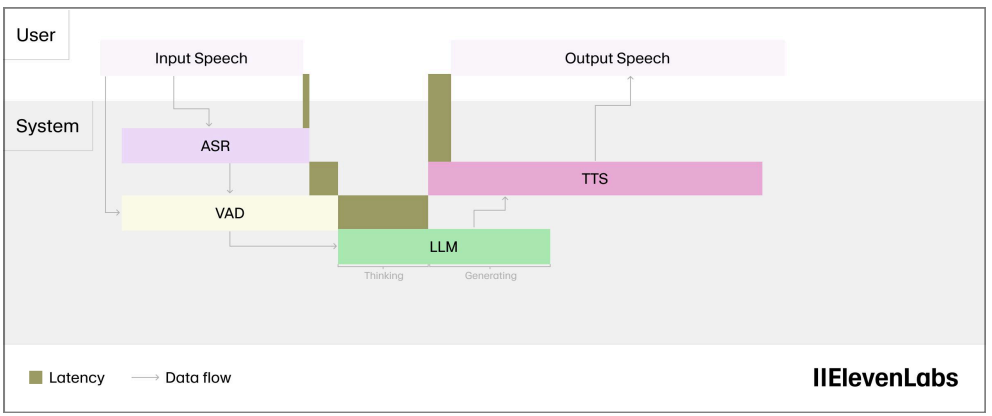


There are other contributors to an LLM's latency beyond the model's choice. These include the prompt length and the size of the knowledge base. The larger of either, the longer the latency. It boils down to a simple principle: the more that the LLM need to consider, the longer it takes. Accordingly, companies need to strike a balance between a healthy amount of context without overburdening the model.

Text to Speech

The System's "Mouth"

The final component of conversational AI is text-to-speech (TTS). Text-to-speech's net latency is the time it takes to begin speaking after receiving input tokens from text-processing. That's it—because additional tokens are made available at a rate faster than human speech, text-to-speech's latency is strictly the time to first byte.



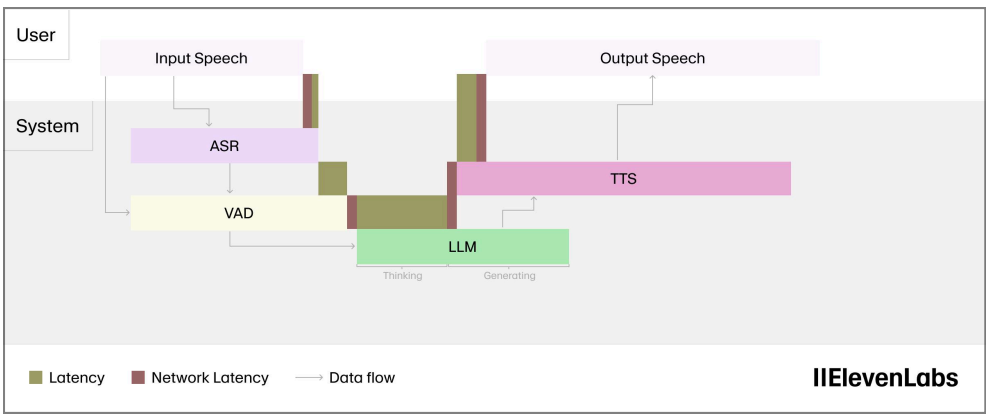
Previously, text-to-speech was particularly slow, taking as long as 2-3s to generate speech. However, state-of-the-art models like our Turbo engine are able to generate speech with just 300ms of latency the the new Flash TTS engine is even faster. Flash has a model time of 75ms and can achieve an e2e 135ms of time to first byte audio latency, the best score in the field (we have to brag a little!).

Additional Contributors

Beyond the four components, there some additional contributors to conversational AI's net latency.

Network Latency

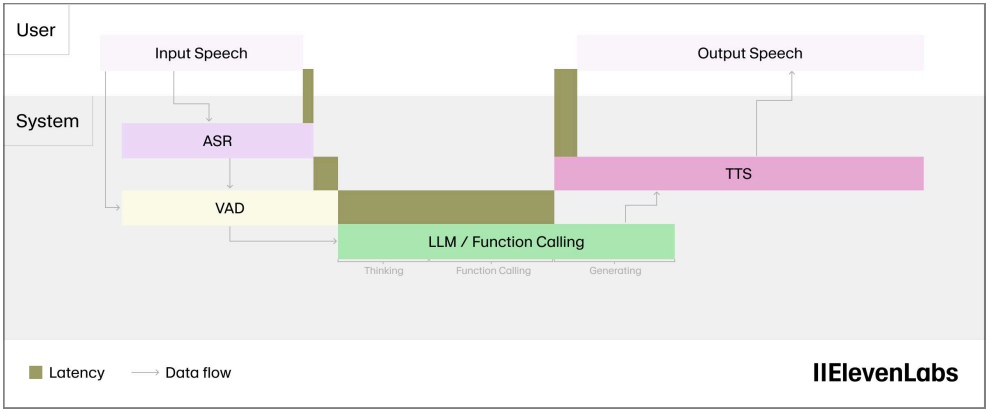
There will always be latency associated with sending data from one location to another. For some conversational AI applications, the ASR, TTI, LLM, and TTS processes should ideally be co-located, so the only source of non-trivial network latency is the paths between speaker and the entire system. **This gives us an advantage on latency as we can save two server calls since we have our own TTS and an internal transcription solution.**



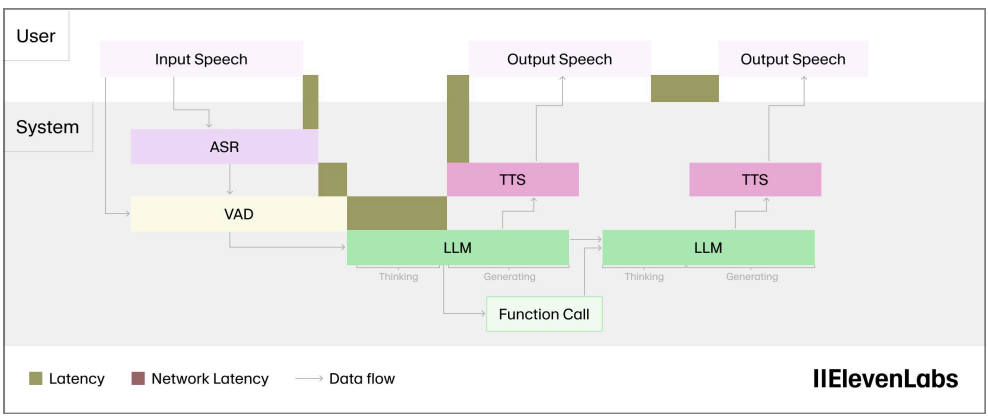
Function Calling

A lot of conversational AI applications exist to invoke functions (i.e. interface with tools and services). For instance, I might verbally ask the AI to check the weather. This requires additional API calls invoked at the text-processing layer, which can incur significantly more latency depending on the needs.

For example, if I need to order a pizza verbally, there might be multiple API calls that are necessary, some with excessive lag (e.g. processing a credit card).



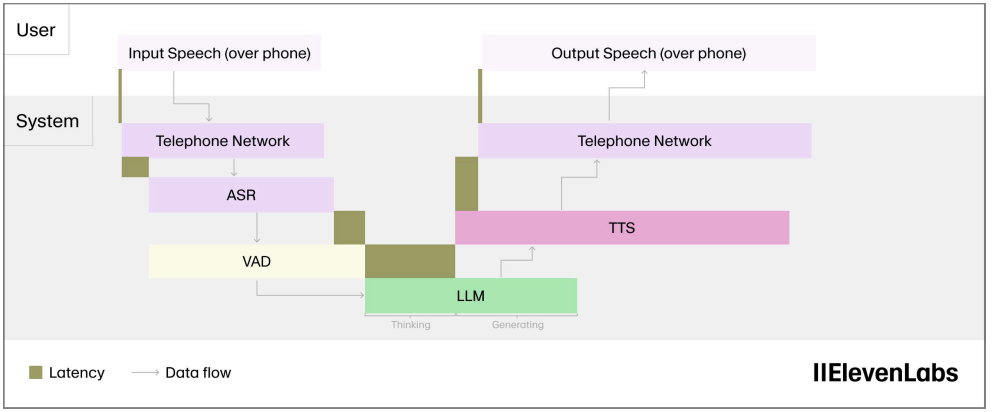
However, a conversational AI system can combat the delays associated with function calling by prompting the LLM to respond to the user before the function call is finished (e.g. “Let me check the weather for you”). This models a real-life conversation and doesn’t keep the user without engagement.



These async patterns are typically accomplished by leveraging webhooks to avoid long-running requests.

Telephony

Another common feature for conversational AI platforms is to allow the user to dial-in via the phone (or, in some cases, make a phone call on behalf of the user). Telephony will incur additional latency—and this latency can be severely geography dependent.



As a base, telephony will incur an additional 200ms of latency if contained to the same region. For global calls (e.g. Asia → USA), the travel time can increase significantly, with latency hitting ~500ms. This pattern could be common if users have phone numbers outside of the region that their located in—forcing a hop to their base country's phone networks.

A closing thought

We hope this round-trip exploration of conversational AI was interesting. In summary, applications should target a sub-second latency. This can typically be accomplished by choosing the right LLM for the task. They should also interface with the user whenever more complex processes are running in the background to prevent long pauses.

At the end of the day, the goal is to create realism. A user needs to feel the ease of talking to a human while getting the benefits of a computer program. By tightening the sub-processes, this is now possible.

At Elevenlabs we are optimizing every part of the piece of a conversational AI system with our state of the art STT and TTS models. By working on each part of the process, we can achieve seamless conversation flows. This top-down view on orchestration allows us to shave off a little latency—even 1ms—at every juncture.