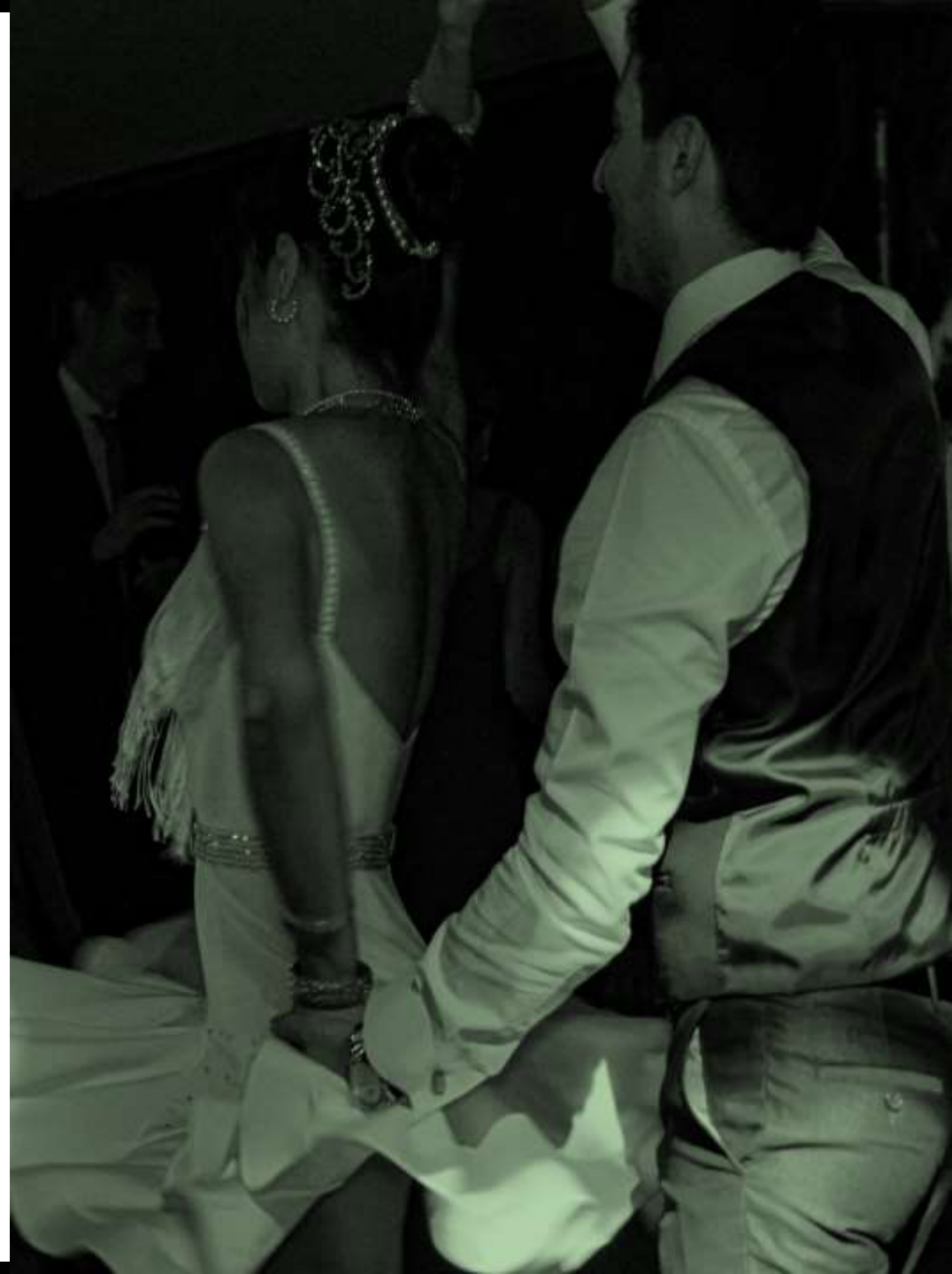
webRTC.ventures

UNVEILING THE **TECH SALSA** OF **LAMS** WITH **JANUS** IN REAL-REAL-TIME APPLICATIONS

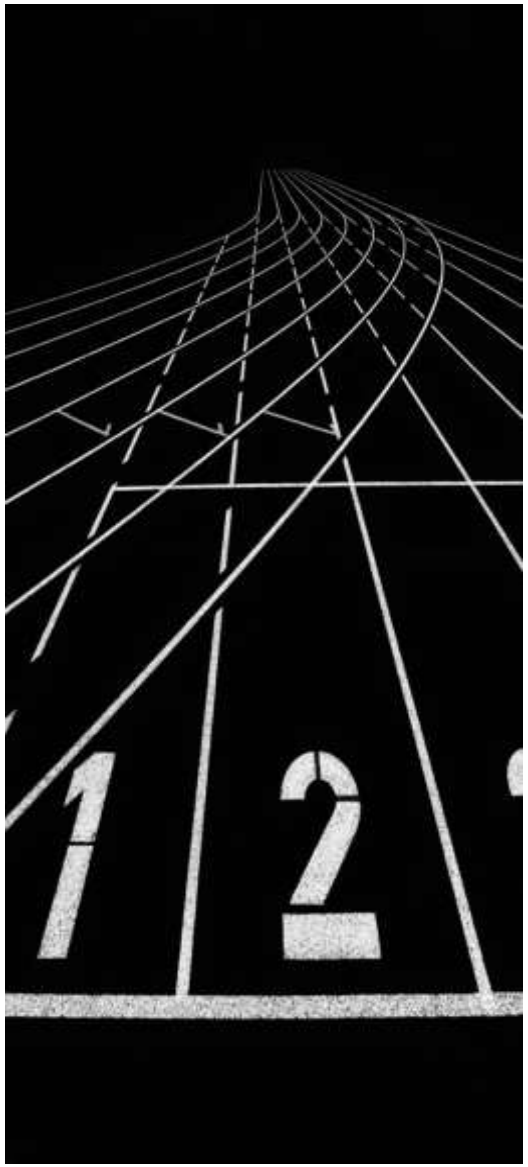


@agonza1_io

Alberto Gonzalez Trastoy

WebRTC.ventures

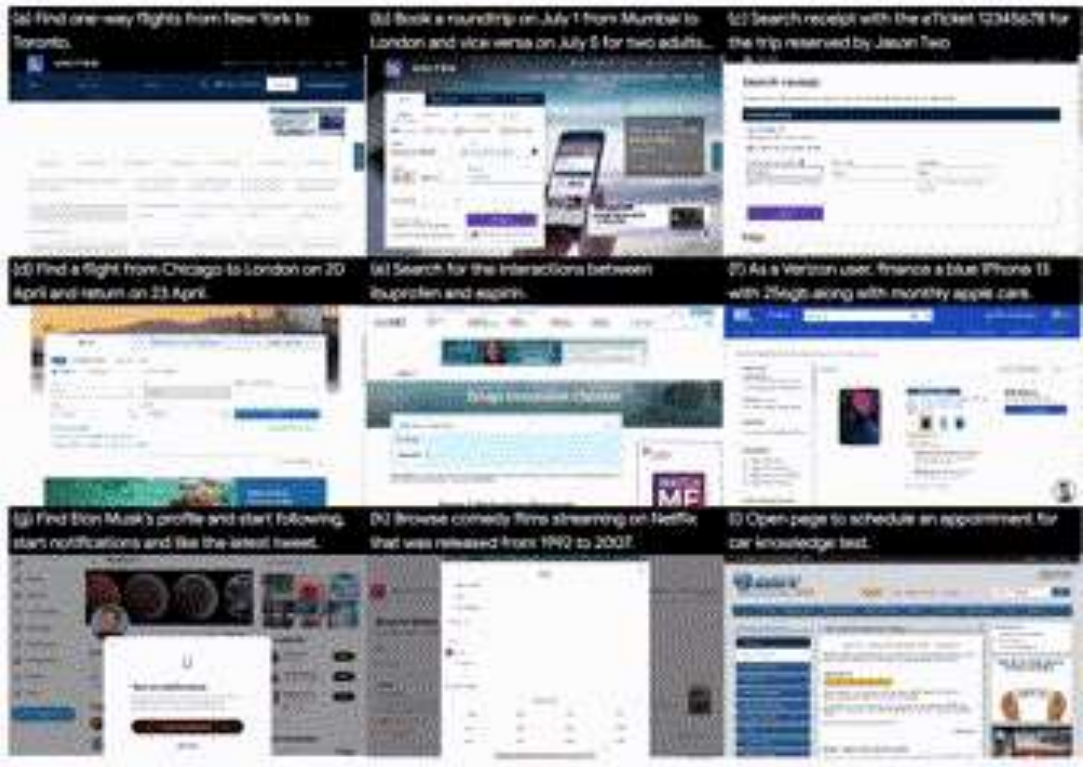
It all starts with this CES 2024 gadget presentation



My Discovery of LAMs, a new AI term

What are LAMs?

- Combine symbolic reasoning with neural networks.
- Directly model application actions. They learn by observing human interactions.
- Understand language like LLMs but also translate it into concrete actions (e.g.: UI actions).



*If you don't like new marketing terms you can just call them **LLMs that perform actions***

Gif source: Mind2Web (osu-nlp-group.github.io)

VERTICALS: LAM USE CASES

How They Will Unlock Value Across Industries

WebRTC.ventures 2024

Use Cases: Automated Trip Preparation

A “**Get ready for my trip**” could include searching email and calendar for the flight information, checking into the flight and booking a ride to the airport (cross checking ride sharing apps).

Image source: DALL-E3

Note: WebRTC is well suited to be a tool to provide real time feedback to humans in this type of automations

Use Cases: Customer Service Bots

FAQs in the past

FAQs today/soon

FAQs in the future

In customer service scenarios, a bot that can help users or agents perform actions. It could handle a wide range of tasks such as helping with cloud services management, updating account information, generate video documentation or troubleshooting issues. This would reduce the workload on humans and provide faster results.

Image source: DALL-E3

Other Use Cases: Scheduling, Filling Out Forms, Testing, Trading, and More…

“Automated Appointment Scheduling”. Managing appointments can be time-consuming. A bot that can schedule appointments and send reminders could be used.

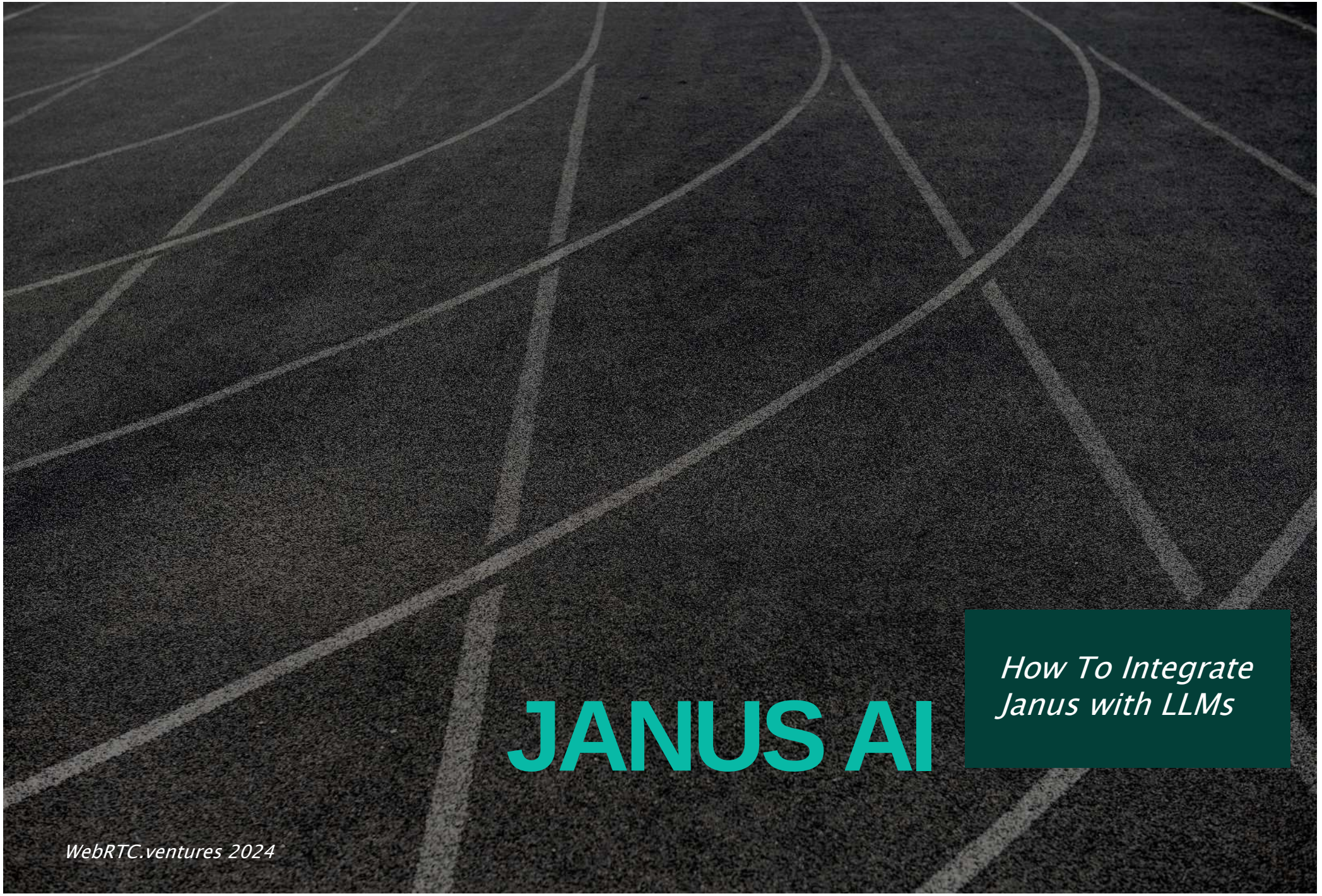
Could offer a “**Quick Tax Filing**” feature, retrieving financial data, filling in tax forms and submitting the return, streamlining the tax filing process for the user.

Could assist traders by automating the process of “**Preparing for Market Open**.” This could involve aggregating news articles, social media, and pre-market trading data.

“**Automated Form Testing**” could involve the LAM filling out web forms with various inputs to test validation rules, error messages, and submission processes.

…

Image source: DALL-E 3



JANUS AI

How To Integrate Janus with LLMs

WebRTC.ventures 2024

How Can We Extract Janus Real Time Media Server Side

RTP Forwarding

- Unidirectional forwarding of WebRTC media (RTP/RTCP) to specific UDP ports
- Available in video room plugin or using RTP forward plugin independently
- UDP broadcast/multicast support
- Easiest to integrate with ffmpeg or gstreamer rtp bin

WHEP (WebRTC-HTTP Egress Protocol)

- WHEP player communicates with WHEP endpoint to get unidirectional media server media
- Available in video room plugin

WebRTC clients

- Bidirectional option that can be used with any plugin.
- Some examples:
  - Pion (Go)
  - Aiortc (Python)

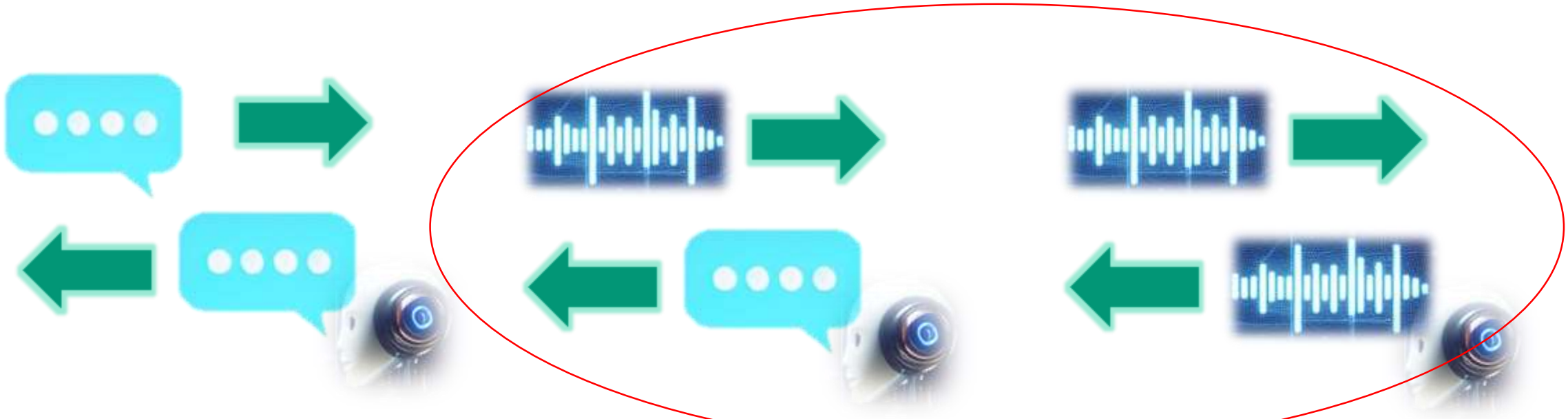
Repos: https://github.com/michaelfranzl/janus-rtpforward-plugin, https://github.com/meetecho/janus-gateway and https://github.com/meetecho/simple-whep-client

We Got The Media, Now, How Do We Want To Interact And Get Feedback From The LLM?

1.Typed with text feedback

2.Spoken with text feedback

3.Spoken with voice feedback

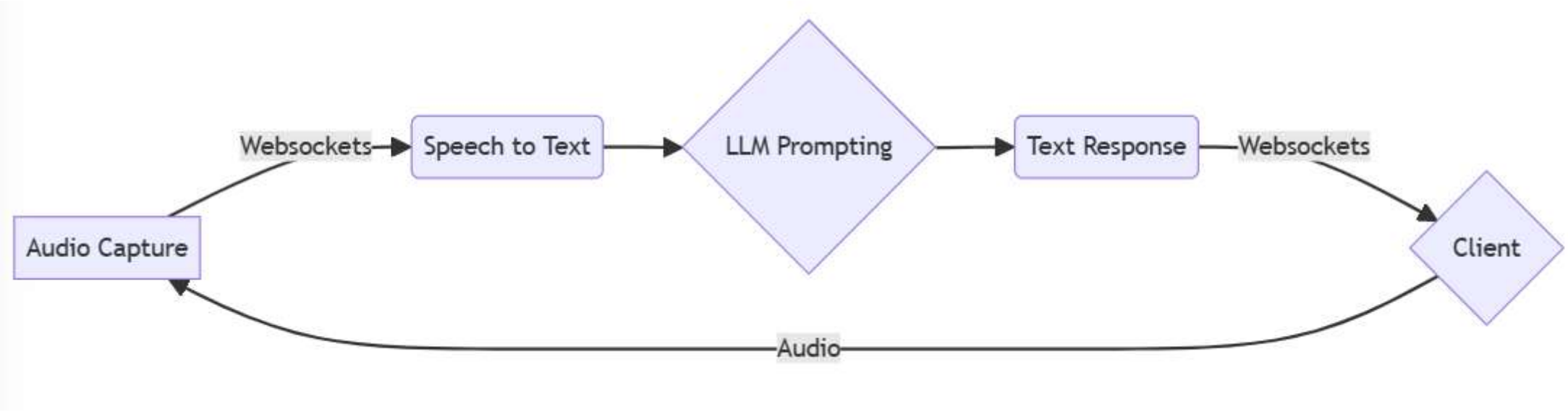


When WebRTC makes more sense to be involved

Even images or video instead of audio?

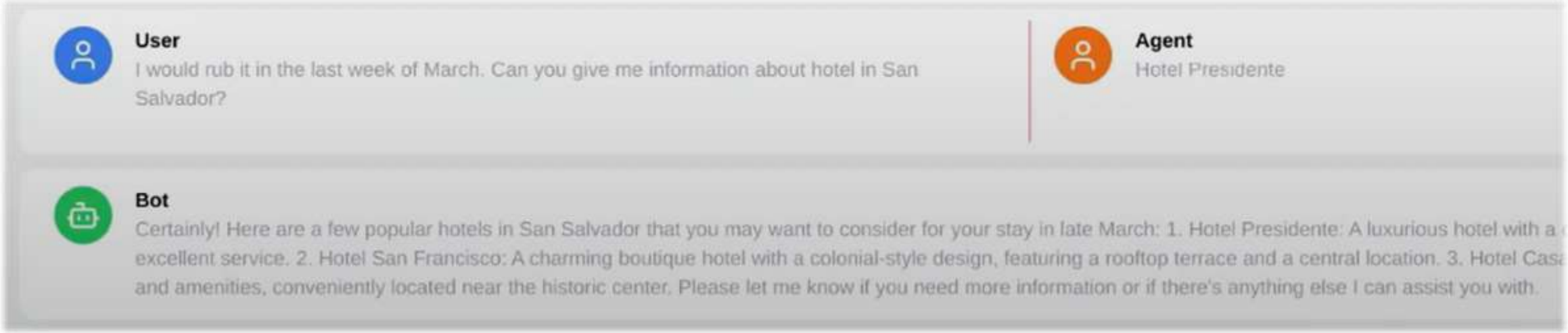
An Architecture Alternative for capturing audio and interacting with LLMs

The most common approach is capturing audio client side (simplified)



And That's How We Did It! Using a server side LLM in Janus based 1 to 1 audio calls

An Agent Assist / Real Time Copilot for a Call Center



This image is not our original project but is a basic representation of the use case through a demo we developed.

Note: In 2023 we developed our first production application combining LLMs with RAG and Janus

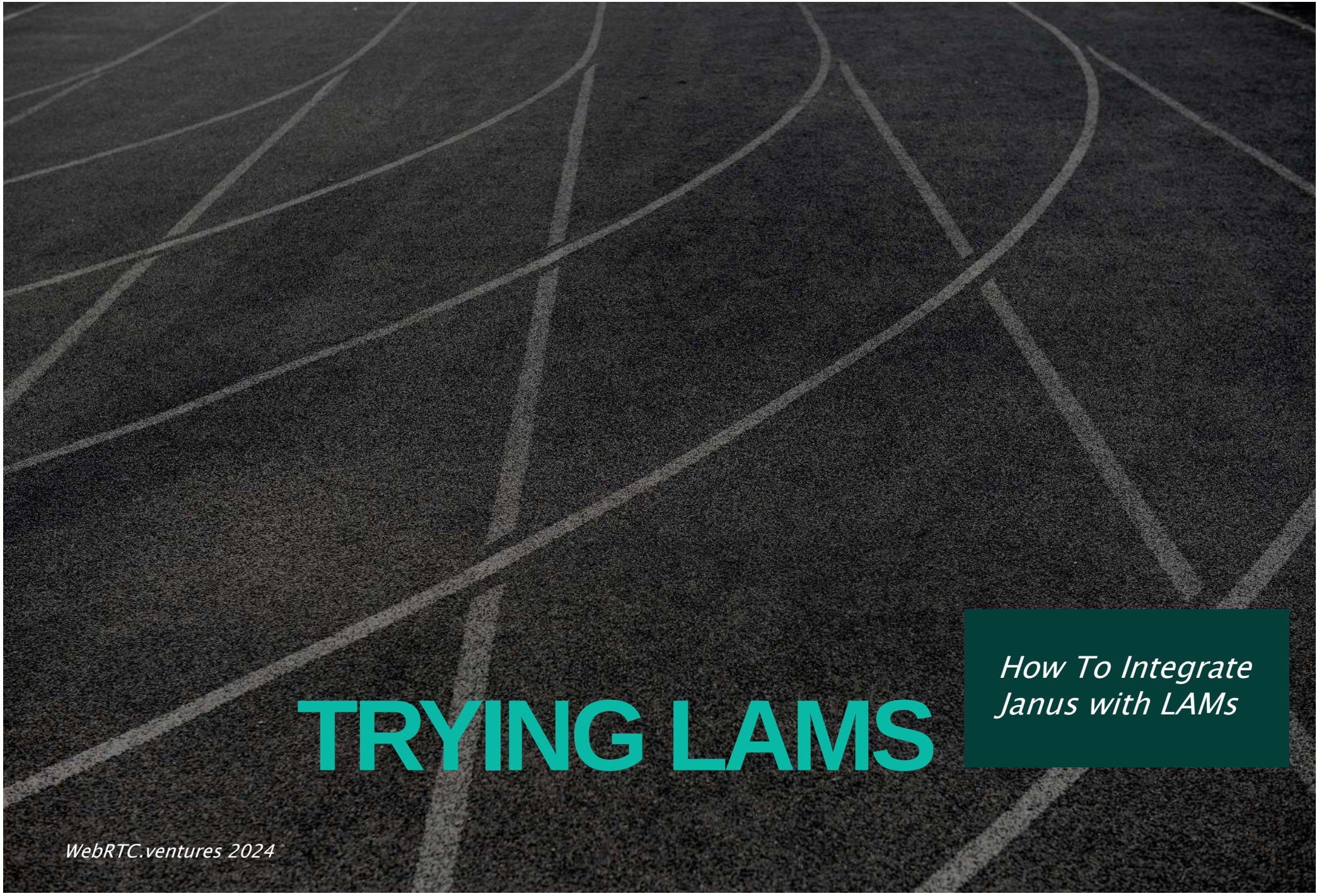
When To Prompt When Building an Agent Assist Like Solution?

1. **Manual request done by the agent.**

2. **Using real time topic or question detection.** This is typically powered by TSLMs (Task Specific Language Models) which can generate topics based on the context of the language content in the transcript.

Other Considerations for RTC-STT-LLM Integrations

- **Architecture Considerations:** If more than one participant interacts with a bot/agent we can't handle all client side
- **Latency**: Server-side STT and LLM operations near media server for reduced delay. We are experiencing above 1s latencies for first character LLM response to voice conversations.
- **Audio Quality**: Clear, high quality audio capture is assumed for most STT models, that's the opposite of what WebRTC optimizes for.
- **Audio format**: PCM audio is usually required for most ASRs (transcoding may be needed if using Opus)
- **LLM Use and Data Flow**: Ideally, should be all run in your own servers. But it is expensive and not trivial to run an optimal LLM API server today, for text it might be an acceptable compromise.



TRYING LAMS

How To Integrate Janus with LAMs

WebRTC.ventures 2024

How To Perform Browser Actions

Ingredients:

- **Mind2Web**: A dataset for developing and evaluating generalist agents for the web
- **A LMM (Large Multimodal Model)** that combines NLU with computer vision:
  - LLaVA (Open Source)
  - GTP-4 Vision
- A **headless browser** to perform the actions
- **App logic to manage the operations: SeeAct.** It is generalist web agent that autonomously carries out tasks.

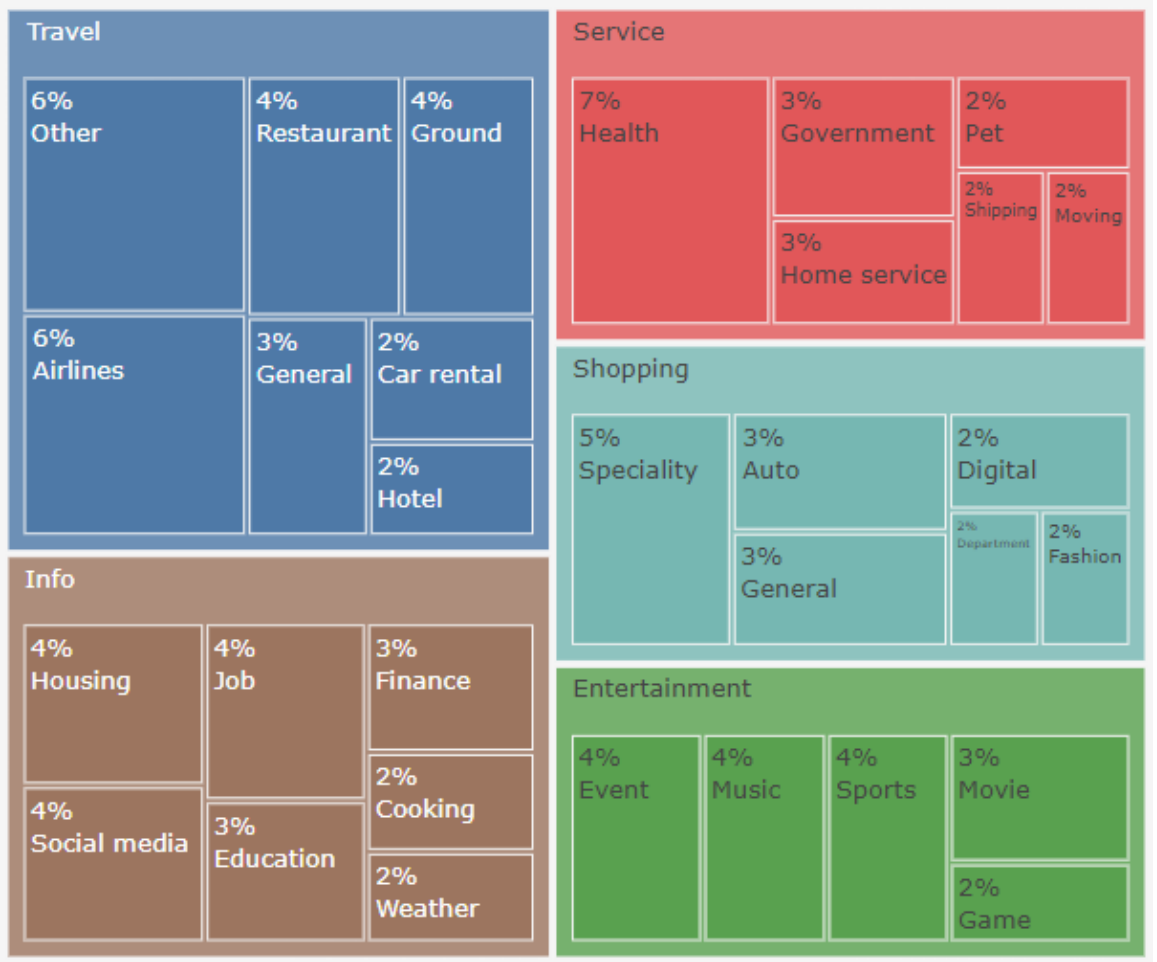


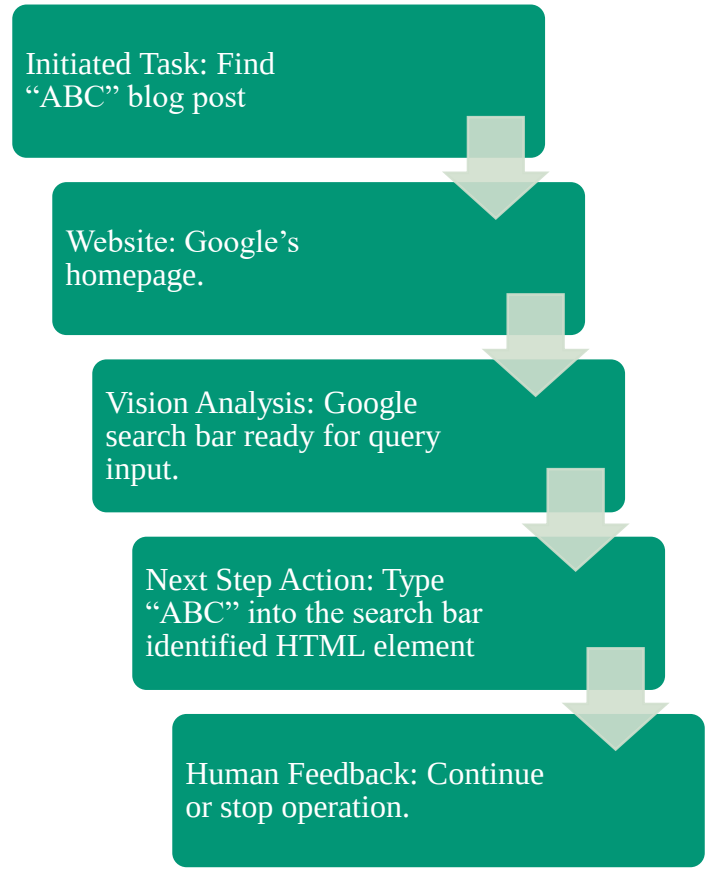
Image source: Mind2Web Dataset: https://osu-nlp-group.github.io/Mind2Web/ Repo Source: OSU-NLP-Group/SeeAct: SeeAct is a system for generalist web agents that autonomously carry out tasks on any given website, with a focus on large multimodal models (LMMs) such as GPT-4V(ision). (github.com)

How To Perform Browser Actions

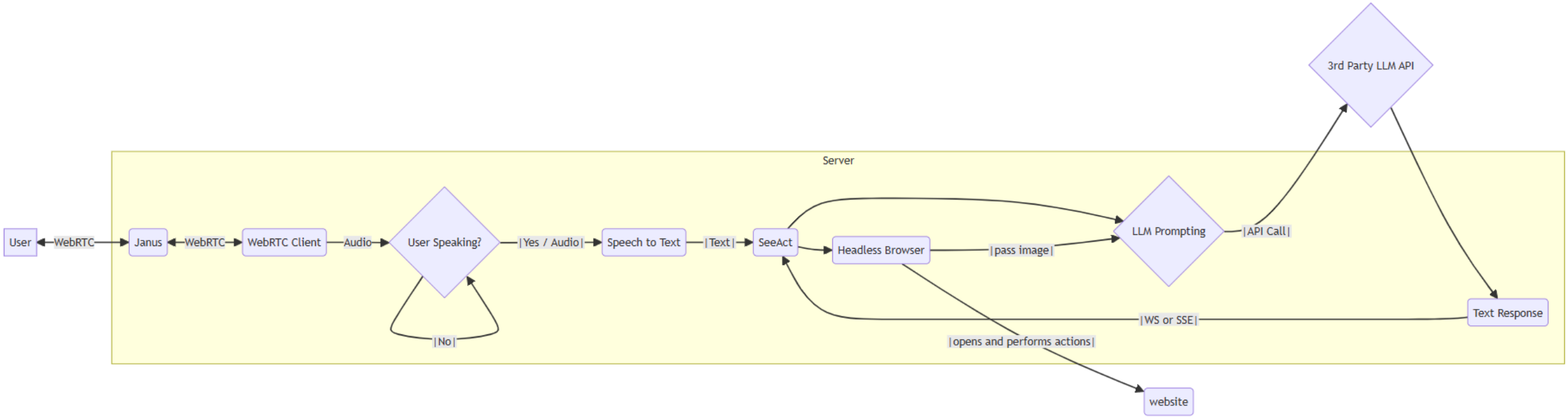
Steps:

1) **Action definition** including website and task
2) Playwright **headless browser** (open site)
3) Get interactive **HTML elements list**
4) Find **top candidates** from list using Cross-Encoder to compare elements to action (limiting list of HTML elements)
5) **Screenshot** of screen for elements identification
6) **LLM inference**:
    6.1) Using GPT vision to extract current site page information
    6.2) Using GPT to obtain action(e.g: CLICK button or TYPE “abc”) and programmatic grounding (connection of supported actions to html elements. E.g CLICK <a>)
7) **Browser action** with Playwright

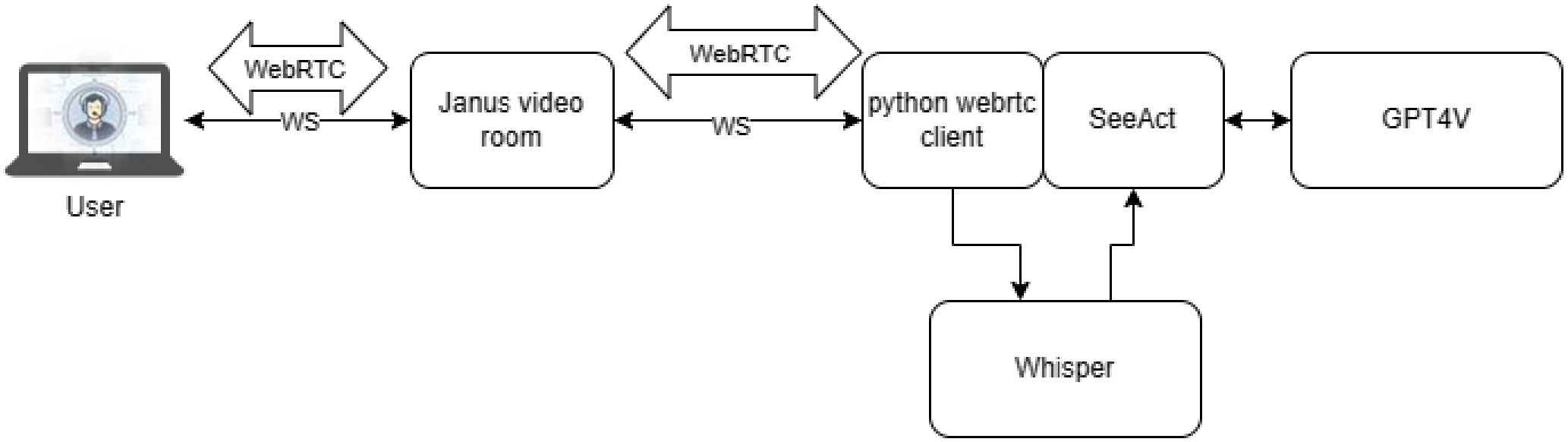
Example



A LAM/LMM Flow Diagram for the WebRTC Demo



A LAM/LMM High Level Architecture for a WebRTC Application



R-SeeAct - Tech Stack

- Videochat Web App: *agonza1/reunitus*
- WebRTC Media Server: Janus
- WebRTC Client: Aiortc
- Speech to Text: RealTimeSTT based on faster-whisper (base mode runs on CPU too)
- Multimodal LLM: GPT-4V
- Browser Action Core Logic: SeeAct

Source code: agonza1/R-SeeAct and agonza1/reunitus at seeact-bot-integration (github.com)

Demo

Image source: DALL-E 3

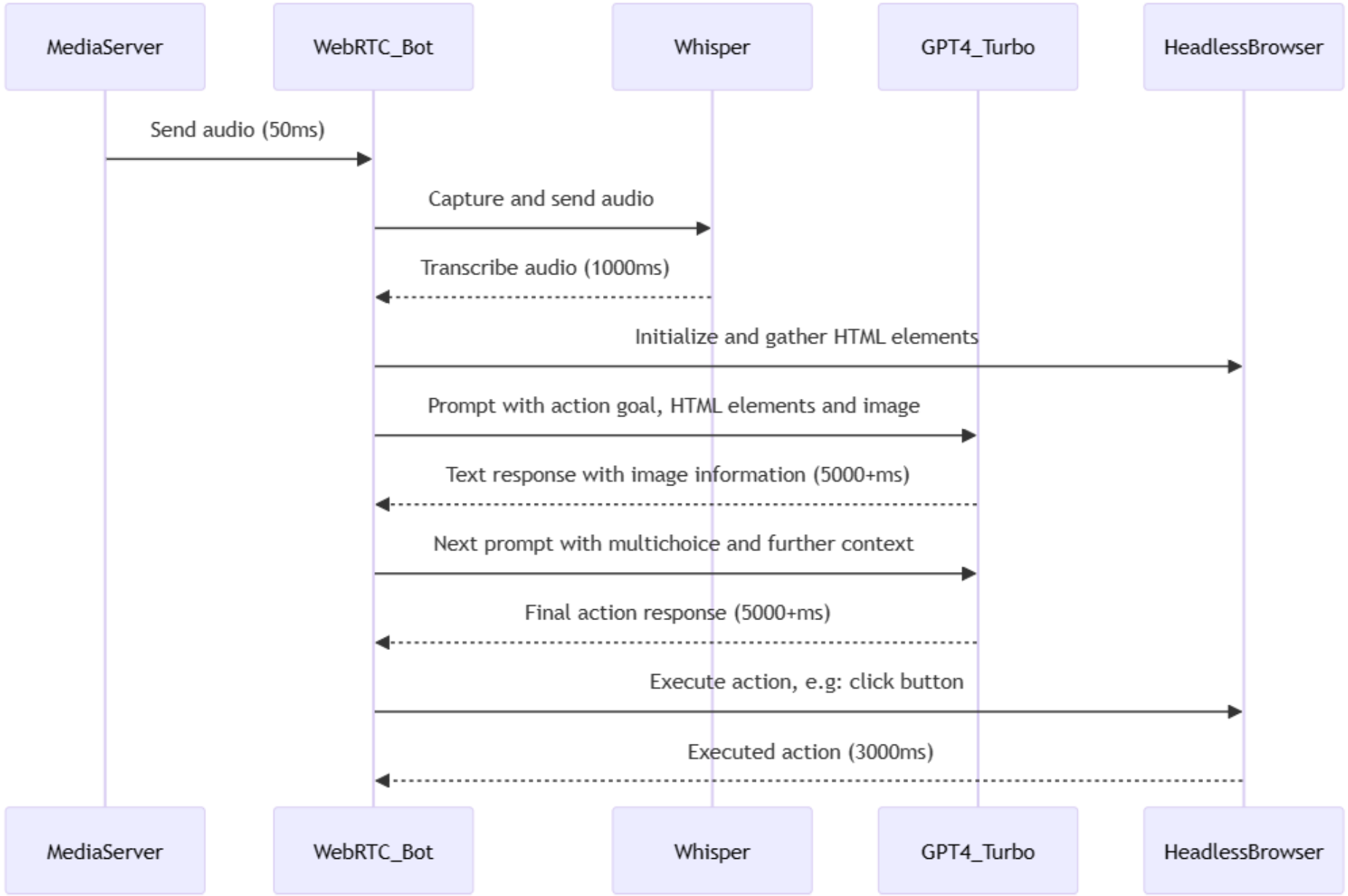
CHALLENGES AND OPPORTUNITIES

Experiences incorporating real time LLMs

WebRTC.ventures 2024

Latency

15+
Seconds!



Main Bottleneck

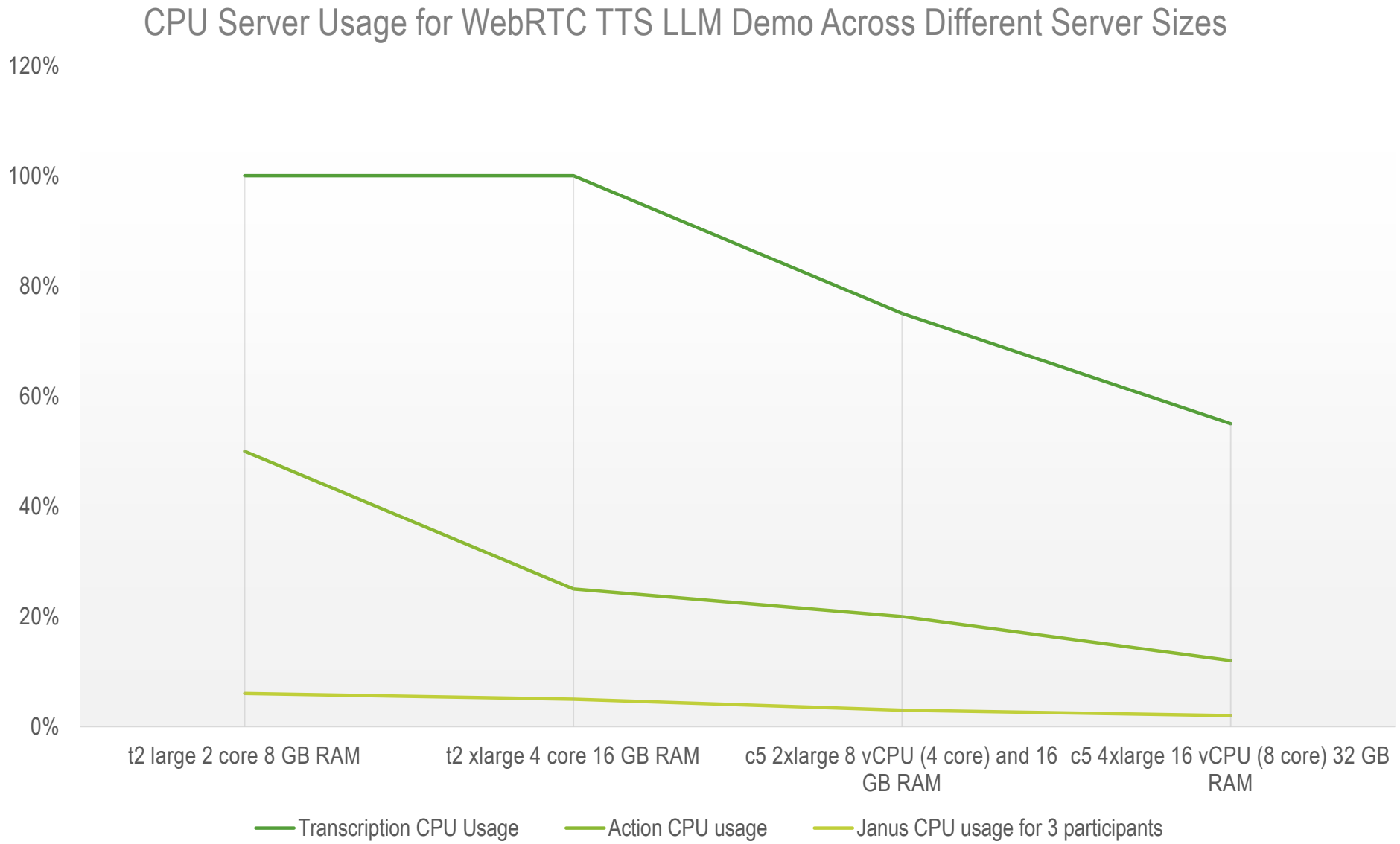
For prompt 2 we need the completion of the initial image LLM inference.

Potential Solutions:

1) Reduce size of response for each step ⟶ decrease quality
2) Usage of agents with some of the initial required context
3) Other LLM with lower latencies
4) Caching

Image source: DALL-E3

Resources


CPU Server Usage for WebRTC TTS LLM Demo Across Different Server Sizes
120%
100%
80%
60%
40%
20%
0%
t2 large 2 core 8 GB RAM
t2 xlarge 4 core 16 GB RAM
c5 2xlarge 8 vCPU (4 core) and 16 GB RAM
c5 4xlarge 16 vCPU (8 core) 32 GB RAM
Transcription CPU Usage
Action CPU usage
Janus CPU usage for 3 participants

Cost

1. Transcription:
   - Using 3rd Party Service Approximately **~$0.02/min**

   or
   - Your own NVIDIA Server Starts at **~$0.006/min**

2. Multimodal GPT-V4 requests: **~$0.01/Analyzed Browser Image**

3. GPT-4 Action/Context Prompts: **~900 input tokens which is ~$0.01**

4. GPT-4 Action Response: **~300 output tokens which is ~$0.01**

5. WebRTC Media Server and Headless Service Action Costs Disregarded

Cost per full tasks/request: ~$0.3
**Includes 1 min transcription + 10 image analysis + 10 Prompts*

CONCLUSIONS AND FUTURE

What's next?

Next Project Steps

Short term:
- Speech to text on GPU with CUDA support!
- Display of browser actions in real time

Long term:
- Improve applying partial results to the query (send prompt before full response)

- Use future ChatGPT enhancements like storing context of previous queries (stateful prompts)

- Alternatives using self hosted LLM servers (LLaVA) or leveraging other existing services that have 10x faster inference

- Implement something like GPTCache for frequent operations

Conclusion

WebRTC is well suited to be a tool **to provide real time feedback** to humans in this type of interactions.

New tech = Opportunities that will let us have better experiences and make interactions more inclusive.

Too Early for LAMs in RTC
The multi step process is what makes this unusable in RTC apps.

We are seeing RTC – LLM integrations **implemented internally first**. In call centers term is called call assist.

THANK YOU

Alberto Gonzalez Trastoy

@lbertogon

webrtc.ventures

Project: agonza1/R-SeeAct