Conversation 4D

Transforming multi-speaker audio into synchronized playback across devices

Approach

Diarization

1. Utilize PyAnnote, an open-source toolkit for speaker diarization, which uses a Bayesian hidden Markov model to find speaker clusters in a sequence of x-vectors.
2. Uses a pre-trained model and allows supplying parameters such as a preset number of speakers or auto-detection.
3. Gives results in segments with start and end times in seconds; to extract audio segments, I convert these times to sample indices by multiplying by the sample rate.

GUI

1. Uses ARP scanning via Scapy to discover devices on the local network by sending ARP requests and collecting responses.
2. Allows the user to select an audio file and executes deployment once the diarization is complete.

Deployment

1. Uses Ansible to automate the deployment of the separated audio tracks and playback scripts to the selected devices. Steps include:
    - Generating an Ansible playbook that copies audio files and scripts to the devices.
    - Executing the playbook to deploy the files.
    - Running the playback scripts on the devices to play the audio in sync.
2. Synchronization is managed by using Ansible's `wait_for` module to ensure all devices have the audio files before playback, and by running the playback scripts simultaneously across devices.

Strength

1. Utilizes existing libraries, avoiding reinventing the wheel.
2. Provides a GUI to improve accessibility.
3. Scalable by using Ansible to deploy across platforms and automate configuration.
4. Separation of concerns, such as the playbook generator can be run independently.

Areas of Improvement

1. Make the IP addresses more recognizable, such as parsing the MAC addresses and checking with a database.
2. Denoise with Librosa to improve accuracy, especially during quick speaker turns.
3. Create a one-liner command-line interface to improve efficiency.
4. Address security risks, such as using Ansible Vault to secure sensitive information.