StenoPhone

Author: Mitchell Yang, Ellen Seeser, Cambrea Earley: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable real-time generation of meeting transcripts and transmission of audio between conference rooms. The current systems that are in place best support communication between individuals, but not clusters of people. During a meeting, microphone arrays interfaced to Raspberry Pis communicate with a centralized server to generate people identified transcripts and relay the audio to the other rooms.**

***Index Terms* — Application Layer Networking, Audio Processing, Audio Streaming, Direction of Arrival, Machine Learning, Microphone Array, Speaker Identification, Speech to Text, Voice Over IP, Web Conferencing**

I. Introduction

In recent years, technology has facilitated a rise in the prevalence of geographically distributed workplaces. Even within a single team, one portion of the members may work in one city, a second portion in a second city, and so on. Responding to this phenomenon, the corporate world has adopted web conferencing tools such as Zoom, Microsoft Teams, and Google Meet. However, these solutions are optimized for the experience of individuals calling other individuals. With StenoPhone, we present a solution for audio-based web conferencing that's tailored towards the record-keeping and communication needs of semi-decentralized teams. Consisting of a wireless microphone and an associated web application, the product provides audio transmission and automatic transcription that includes the identification of speakers. StenoPhone supports multiple users per microphone and multiple microphones per meeting.

The goal of this project is to provide intelligible and low-latency audio transmission, as well as fast and accurate transcription, to the end users. The system must provide end-to-end audio latency of less than 150ms while maintaining a dropped audio packet rate of less than 5%. The system must also provide transcription of audio in less than three seconds, and the transcript's error must not exceed 25%.

II. Design Requirements

StenoPhone is intended for use in a conference room setting. Based on measurements we have taken during a survey of conference rooms, we define a conference room as a room with maximum dimensions of thirty-five square meters and featuring a large table. With the microphone placed in the middle of the table, the system will be tested with participants sitting or standing no more than two meters away, in any direction around the table. We will test with a maximum of three users per single microphone and a maximum of two microphones per meeting.

As described in the Introduction, the latency of audio across the system, also known as mouth-to-ear (M2E) latency, must be less than 150ms. This is a standard commonly accepted as maximum M2E latency in IP telephony; a round-trip delay greater than 250ms is noticeable to the users [1]. This latency may be tested by directing a captured audio packet from one microphone to the web server and back to the same microphone, capturing self-consistent timestamps at the beginning and end of this process. The second requirement related to the user's audio experience is that limiting the percentage of dropped packets to below 5%. IP telephony theory again advises this rate to avoid audio distortion experienced by end users [2]. The dropped packet rate may be estimated using a long audio transmission of a known number of packets to the webserver and then to an end microphone-speaker, which can count the number of packets received.

Requirements pertaining to the speed and quality of transcription are also essential to meeting the goals of our project. The 3s transcript latency, also known as average word delay, that we require is a measure that comes from FCC guidelines pertaining to live closed-captioning [3]. This latency may be tested by comparing the time at which a spoken phrase was captured by the microphone to the time at which a packet containing the transcript arrived at the browser of the end user.

The accuracy of the transcript can be measured with several different error rates. The first, word error rate (WER), measures the accuracy of the speech to text element itself. $WER = (S + I + D) / N$ where S is the number of erroneous substitutions, I is the number of word insertions, D is the number of deleted words, and N is the total number of words in the original text. The accuracy of the speaker identification system is measured by speaker identification error; because the participants in our system are known, this metric is simply computed as the number of misattributions divided by the total number of speaker changes. We require that neither WER nor speaker identification error rate (SIER) exceed 25%; research has shown that users find transcripts with error up to and including this rate to be useful [4]. We've also identified formatting errors that can arise from the combination of transcripts from multiple microphone streams — the chronology of the combined sections may be incorrect, or attribution may be missing or incorrect upon a switch between streams. A rate of these formatting errors greater than 5% will not be acceptable.

Similar tests may be used to measure the three aspects of transcript accuracy; all three tests require a 'known text.' A text may be considered 'known' either if it's predetermined before it's read aloud or if it's recorded as it is spoken. These texts will remain constant over multiple tests; they will consist of common English words formed into sentences, and they

should be at least 100 words long. To test WER, a known text is spoken into the microphone; the transcript and the correct text are compared, and WER is calculated. To test SIER, a known text that features speaker changes and speaker movement is spoken into the microphone; the transcript and the correct text are compared, and SIR is calculated. To test formatting error rate (FER), a known text, one that features speaker changes where the speakers use microphones in different rooms, is spoken into the microphone; the transcript and the correct text are compared, and FER is calculated.

Additional tests will be conducted regarding the transcript error rates. We will need to test the three error rates in the situation of one group of at least two speakers using one microphone and another speaker using a second microphone. We'll also test our beamforming capabilities in the following way: two speakers will be positioned on opposite sides of the microphone. One will loudly read the test text; the other will quietly read a text containing different words from the test. If the beamforming works correctly and the transcript shows the test text within the given error rates, then we will have passed the test.

Finally, the website will be required to provide certain functionality to the user: creating meetings, joining meetings, adding microphones, and viewing transcripts of current or past meetings.

Speaker Identification Accuracy	Speaker Identification Error (%) < 25%
Formatting Accuracy	Formatting Error Rate (%) < 5%

III. Architecture and/or Principle of Operation

Fig. 1 below shows the flow of information through our comprehensive system, including hardware and software modules as indicated. The system consists of a single web server, to which are connected an arbitrary number of audio devices and web browsers.

Audio data originates at a microphone array, which performs some processing in hardware before passing the audio to a small WiFi-capable computer device. There, further processing is performed to improve and compress the audio, detect voice content in the audio, and configure the microphone. A networking module then sends the audio information to the web server. Additionally, the web server sends audio from other microphones to each individual audio device, allowing the audio to be played on an attached speaker.

The networking component of the web server receives audio packets from the connected microphone devices. This audio is both sent out to other microphones and sent to the transcript generator module, where speech to text and speaker

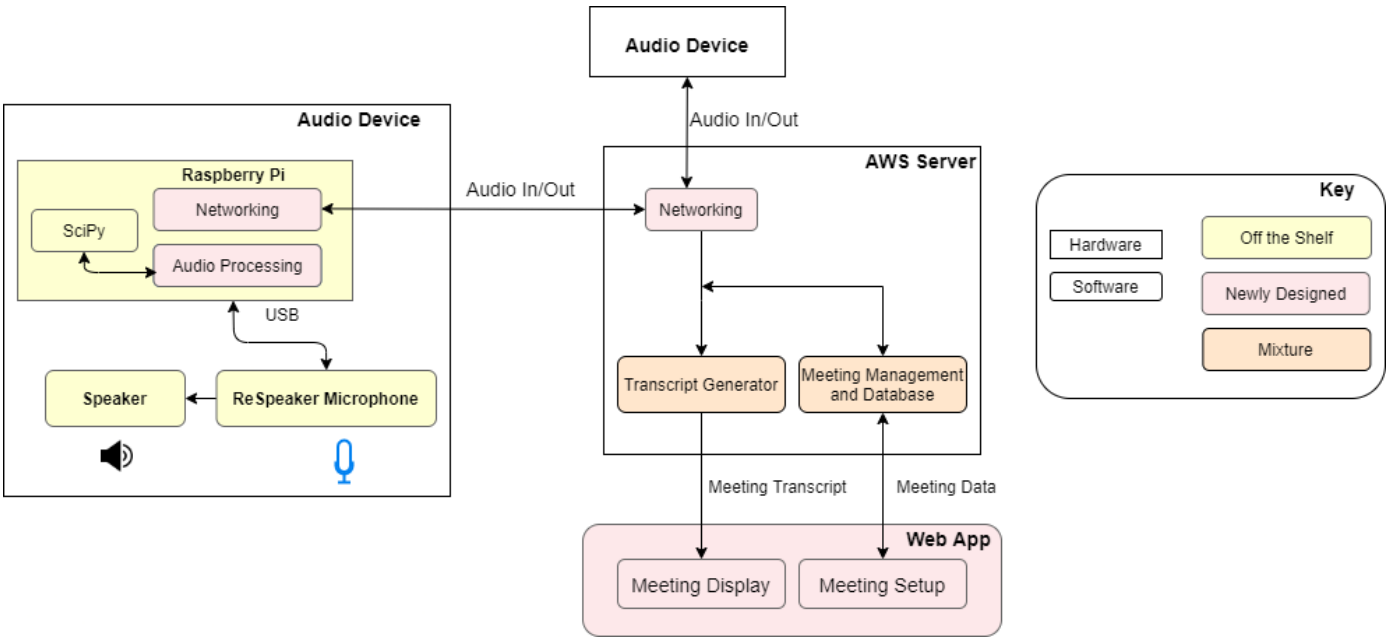

Fig. 1. System Data Flow Diagram

TABLE I. Summary of Requirements

Requirement	***Metric***
Audio Transmission Latency	Mouth-to-Ear Latency (ms) < 150 ms
Audio Quality	Dropped packets (%) < 5%
Transcript Latency	Average Word Delay (s) < 3s
Transcript Accuracy	Word Error Rate (%) < 25%

identification processing are performed with the assistance of machine learning solutions. Speakers are identified from a pool of users who vocally identify themselves to the microphone during the process of adding a microphone to a meeting.

Transcripts are compiled from multiple microphone streams by the meeting manager, and adding transcripts to the database allows them to be viewed by users in the web application. Web application users can also create meetings, join existing

meetings, and connect a microphone to a meeting on the website. Users who join meetings, whether or not they connect a microphone, will be able to view the meeting transcript, whether or not the meeting is ongoing.

IV. Design Trade Studies

A. Audio Device: Microphone Array

The microphone array is one of the main advantages of using the StenoPhone over other existing conference room meeting products. So for this microphone we needed to choose one that would work for our context; ideally it would sit in the center of a conference room table and detect audio from all surrounding angles. We were able to eliminate many microphone arrays that do not have this capability such as the Kinect Mic Array.

We then focussed in on the circular Respeaker Microphone arrays. These arrays have microphones placed in a circular formation for collecting audio surrounding the device at 360 degrees. Out of the ReSpeakers we decided to use the Mic Array V2.0. We researched and found that this array had higher quality audio than the other array pi-hats. The array v2.0 also has an on board microcontroller IC, XVF-3000 from XMOS. This chip handles processing algorithms that come with the board, such as automatic gain control and detection of direction of arrival of audio.

The other option of respeaker that we were looking at but ultimately decided against was the ReSpeaker Core V2.0. This mic array has similar capabilities to the mic array V2.0 but it also has its own linux system on the board with wifi capabilities so that the mic array functions as a single device without the need for a Raspberry Pi or computer to help with data transfer. Though this board is more powerful we decided against it since we prefer to have a raspberry pi connection to the speaker to handle more processing of the audio and to package the data to send to the server over the network. The ReSpeaker Core was also a more expensive choice and seemed to be a higher level product which leaves less opportunity for us to configure and build upon the given microphone implementation for our specific project needs.

B. Audio Device: Computer

For the audio device we needed to have a main system for processing the audio and sending and receiving data over the network. For this job we had two main choices, the Jetson Nano or the Raspberry Pi 3 B. We first started by looking at how much processing we would need to do on the device before we sent it over the network and found that we mainly need to filter the audio, detect direction of arrival, route output audio to the server, and route input audio to the speaker. This design of the software allows for most of the long processing for both the speech to text and speaker identification solutions to be performed on the server which greatly reduces the amount of work that the audio device needs to perform locally. The specs for the Jetson Nano are 1.42 GHz processor, with 2GB RAM. This is comparable to the Raspberry Pi 3 B which has 1.4GHz processing speed, with 1GB RAM. Both of these devices also offer UBS connection which we will use to connect our microphone array and speaker.

A huge drawback of the Jetson Nano is that it does not have wifi capabilities on the board, and we would need to buy an external wifi connection device to use with the Jetson Nano. Though the Nano does have an ethernet port, we were planning on using wifi for our project since in conference rooms there are not always ethernet connections available. The Raspberry Pi 3 does have dual-band wifi capabilities and comes at a lower price than the Jetson Nano with comparable specifications for processing speed making it the best choice for our project.

C. Audio Device: Processing

The audio is to be processed to remove outlying noise before being sent in MP3 format to the server for further processing for ML and sent to the other devices to output audio in other rooms. We considered two different libraries for scripting the audio processing: Audacity and SciPy. Audacity is supposedly good for larger files, while our implementation will use small chunks of audio. SciPy has a lot of support and documentation as well, making it our tool of choice.

There are several options for compressing the audio to prepare it for transmission over the network. The point of compression is to reduce the size of the audio for easier transmission, we are especially focussed on latency of the audio transmission over other parameters. Because of this we are planning on using a lossy audio compression format over a lossless audio compression format, since these algorithms are faster and produce smaller compressed files on average.

D. Web Server: Server

We chose to use a server for the site for this project's more complicated processing, such as speech to text and speaker identification ML. These components have higher latency so we decided to use the more powerful server to process this instead of processing this on the raspberry pi. Another reason why we chose to use the server is because we have structured our audio streaming system to be centralized, so each audio stream travels through the centralized server. This allows for us to quickly have access to all audio streams on the server so that we can generate the transcript in one place.

Amazon AWS was the obvious option for our web deployment because of its support from the course staff in terms of both knowledge and funding. To choose a specific Amazon EC2 instance we started by narrowing it down to the M5, M5a, and M4 classes of servers. We know that our system has audio streaming both to and from two devices potentially simultaneously (consisting of both audio and metadata at up to 44.1kHz with 16 bits per sample) as well as connections to a website, so we needed to focus on the network bandwidth with a lower limit of 3 Gbps. We also are running the speech to text and speaker identification ML, so we needed a balance of CPU power as well.

Comparing the EC2 servers we found that M5 had the highest processing speed 3.1 Ghz, with comparable network

bandwidth up to 10 Gbps, but was the most expensive. The M4 had the lowest processing speed at 2.3 GHz with network performance defined as “moderate” and was the lowest cost. The M5a was right between these two servers with 2.5GHz processing speed and up to 10 Gbps network bandwidth with economical prices. We ultimately decided to use the M5a EC2 instance; we have run several ping tests on this server as well, to test the network. We found that hosting the server at the Ohio site option results in the best latency from our location in Pittsburgh.

E. Web Server: Networking

In Fig. 1 we show 2 types of connections that need to be made by the server: the first is the connection to the audio devices, and the second is the connection to the website. Both of the connections are handled in different ways with consideration to the type of data being sent and to the latency bounds of each.

Firstly for the connection to the audio devices, here we are sending the audio, direction of arrival metadata, and the micID so that the server can distinguish where the data is coming from. One copy of this data is sent directly to the other audio device and the other copy is sent to the speech to text and speaker identification systems. It is important that this connection has very low latency since the audio stream output must appear at the second audio device in real-time and be intelligible to the users there. To determine the latency, we did a ping test. Ohio, us-east-2 region, is consistent at around 42 ms, median 41ms. North Virginia, us-east-1, spikes at beginning with around 550 ms and decreases to 27-32 ms and stays there, median 32. We like consistency so are choosing to use the Ohio region, as latency spikes can cause inconsistent and poor audio.

We decided to use a UDP connection for our implementation since this connection allows us to send our data with low latency; we also found that UDP connection is common with real-time audio transmission. One issue that this connection could add to our project is that UDP allows for packets to be dropped on the network. Because of this we have made a requirement for less than 5% packet drop rate of our system to ensure that the audio stream is intelligible to the users. Since packet size plays a role in how many packets can be dropped on a network, we can tune the packet size to help enforce this requirement.

The next connection that the server has is the connection to the website. Over this connection we will be sending the transcript to the website so that users can view it during and after the meeting, for this we will be using HTTP. We are most focussed that the http packets arrive and that they arrive in order to the website so that is why we need to use HTTP with a TCP network transport layer to ensure this.

F. Web Server: Database

The server information is to be stored using a SQLite database. For the database, we have identified other solutions such as MySQL and PostgreSQL. A SQLite database was chosen as it is portable and that we would rather spend server cycles in processing than using a DBMS which requires an additional server process like MySQL or PostgreSQL. In addition, most of the website development will be done locally, so it is convenient to have the database contained and stored in the repository for ease of sharing and use.

G. Web Server: Speech to Text

The speech to text portion of the system is to be implemented by integrating a preexisting machine learning model or solution into the transcript generator software module. Tradeoffs in this implementation come down to the tradeoffs between different solution options.

Solutions that have ‘streaming’ constructs are automatically favored over others. These tools allow additional audio content to be appended to previous audio with the understanding that the new audio could be a continuation of the previous sentence or word. It’s very helpful for our system since we receive small amounts of audio in each packet over the network, and these vocal fragments probably wouldn’t make much sense if transcribed individually.

We have identified three reputable speech to text solution options that contain streaming tools: Mozilla DeepSpeech, which is a software package; CMU PocketSphinx, another software package; and Google Speech-to-Text, which is a paid service. Because of our requirements associated with transcription, these options should be compared based on their latency and their accuracy, measured by word error rate (WER). DeepSpeech reports 7.5% WER [5]; PocketSphinx reports 10% WER [6]; Google reports a very low 4.9% WER [7]. Self-reported WER and timing information for each of these solutions, however, are not directly comparable; the best results on various different datasets are going to be reported.

Therefore, in order to obtain a fair comparison, each option must be implemented inside the transcript generator and assessed on the same audio. Ideally, the entire audio processing pipeline of the StenoPhone would be in place so that test results would additionally reflect adherence or nonadherence to the requirements. Initial, unofficial results on ReSpeaker audio indicate that Google may be the best choice for our system.

H. Web Server: Speaker Identification

The situation with speaker identification ML solutions mirrors that of the speech to text. Multiple options have been identified, each with their own advertised diarization error rate (DER). Several will be implemented and tested inside of our system before the best one is chosen.

A table has been included below to describe the self-reported DER of the speaker identification solutions under our consideration.

TABLE II. SPEAKER IDENTIFICATION ML SOLUTIONS

Reported DER of Selected Solutions	
Speaker ID Solution	***DER***
pyannote audio	25% [8]

pyBK	12% [9]
Hitachi Speech EEND	15% [10]
BUTSpeech FIT VBx	22% [11]

I. *Web Server: Website*

The interface for users will be a website served through AWS using the Django Framework. Django was chosen as it is well established, so resources are readily available if Django specific problems arise. It is also based in Python, so it is easier to integrate with other Python libraries like the ML solutions mentioned above.

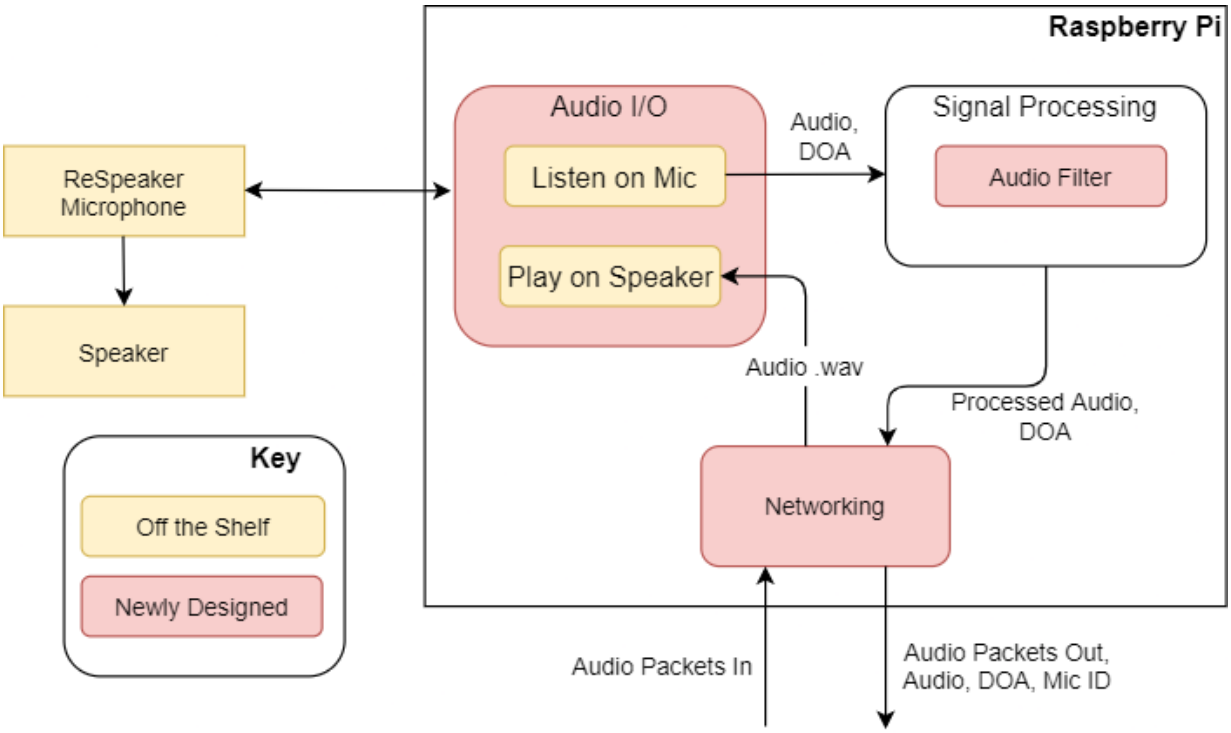

Fig. 2. Audio Device Diagram

V. System Description

A. *Audio Device*

The main function of the audio device, shown in Fig. 2, is to handle input and output audio. The three components to this are the direct Audio I/O to the microphone and speaker, the signal processing piece, and the networking layer to transmit audio to and from the server. Each of these components will operate on its own thread so that each component can keep working simultaneously. Each audio device has its own microphone ID so that users can register their microphone when they create and join meetings. This microphone ID is also used in the transmission of audio data, further discussed below.

Starting with the control flow of audio from the ReSpeaker mic to the server: the users will speak into the microphone; this audio is streamed, using the pyaudio streaming library; and it's put into a queue to be processed by the signal processing component.

The signal processing component will filter each chunk of audio to improve it. The frequencies 20Hz - 8kHz will be kept as those are the frequencies important to intelligibility [12].

The networking layer will read from the queue if there is a processed audio packet ready to send to the server. This packet contains the audio segment, direction of arrival information, and the microphone ID. We are using a UDP connection for this transmission to ensure low latency.

As for control from the server to the speaker, this is audio from users using a different audio device. The server sends the audio packet to the other audio devices in the meeting. The software on the Raspberry Pi will play this audio on the speaker connected to the mic array. The ReSpeaker Mic array will block stop listening while audio is played on the speaker to prevent a feedback loop here.

B. *Web Server*

Two separate connection types are used on the AWS Server, shown in Fig. 3, since we need to connect to both audio devices and the website; first we discuss the connection to the audio devices.

We use UDP to send and receive the audio from the audio devices; this operates on a single thread that holds both of these actions. One stream of audio is from one audio device to the other, for example, this audio is spoken by users in room 1 and will be heard by users in room 2. We also need to make a copy of the input audio stream from a microphone that will be sent to the speech to text and speaker identification ML portion of our project. To do this, we will use 2 separate queues to store input audio data, so that the server can

continue sending and receiving data on its own thread, and will not be blocked by the transcript generation. One queue will hold audio meant to be sent out by the server, this is defined as audio that will be sent to an audio device to be played to the users. The other queue will hold the audio packets that will be used by the transcript generator. Both of these queues are from the python Queue library, we are using the non-blocking and thread safe queues.

The transcript generator software module receives short audio files from the networking layer. A microphone identifier and the audio's direction of arrival (DOA) additionally accompany the file. The transcript generator keeps a dictionary of information pertaining to each active microphone connection, including the timestamp of the last time audio from this microphone was processed and the DOA of that audio. These two pieces of information help with the transcript generator's first task: speaker change detection. A speaker change is detected when the DOA of the new audio differs from the old DOA by more than ten degrees *or* it has been more than five seconds since the last audio. When a speaker change is detected, the ML submodules are reset so that the new audio is considered independent from the old audio; otherwise, if no speaker change is detected, the new audio is appended to the old preexisting stream of audio.

Before audio can be sent to speech to text and speaker identification, though, some preprocessing is performed on the file. First, a low pass filter with cutoff frequency about 8kHz is applied. Human listeners may find high frequency audio information useful or interesting, but the machine learning models have less use for the higher frequencies. This filter is implemented as a fifth order Butterworth filter using scipy. The scipy package is also used to resample the audio to 16kHz, the sampling frequency required by DeepSpeech. Then, execution is split into two threads; one executes speech to text and the other executes speaker identification.

The speech to text submodule has its own dictionary containing references to model instances for each active microphone connection. The information that needs to be kept differs for each model type, but always includes a lock that protects the model state while it is being used by one thread. Given the new bit of audio, the speech to text submodule returns the latest projection of what has been said since the last speaker change.

The speaker identification submodule, again, has a dictionary of information related to each microphone. This includes not only references to model instances but also information about the number, last known location, and identifiers of each individual in the room with that microphone. Speaker ID does not run during meeting initialization while this information is being collected by the meeting manager. If the speaker ID ML component assigns less than 50% confidence to a speaker prediction, then the speaker is chosen based on confidence ranking from the two individuals physically closest to the new audio DOA.

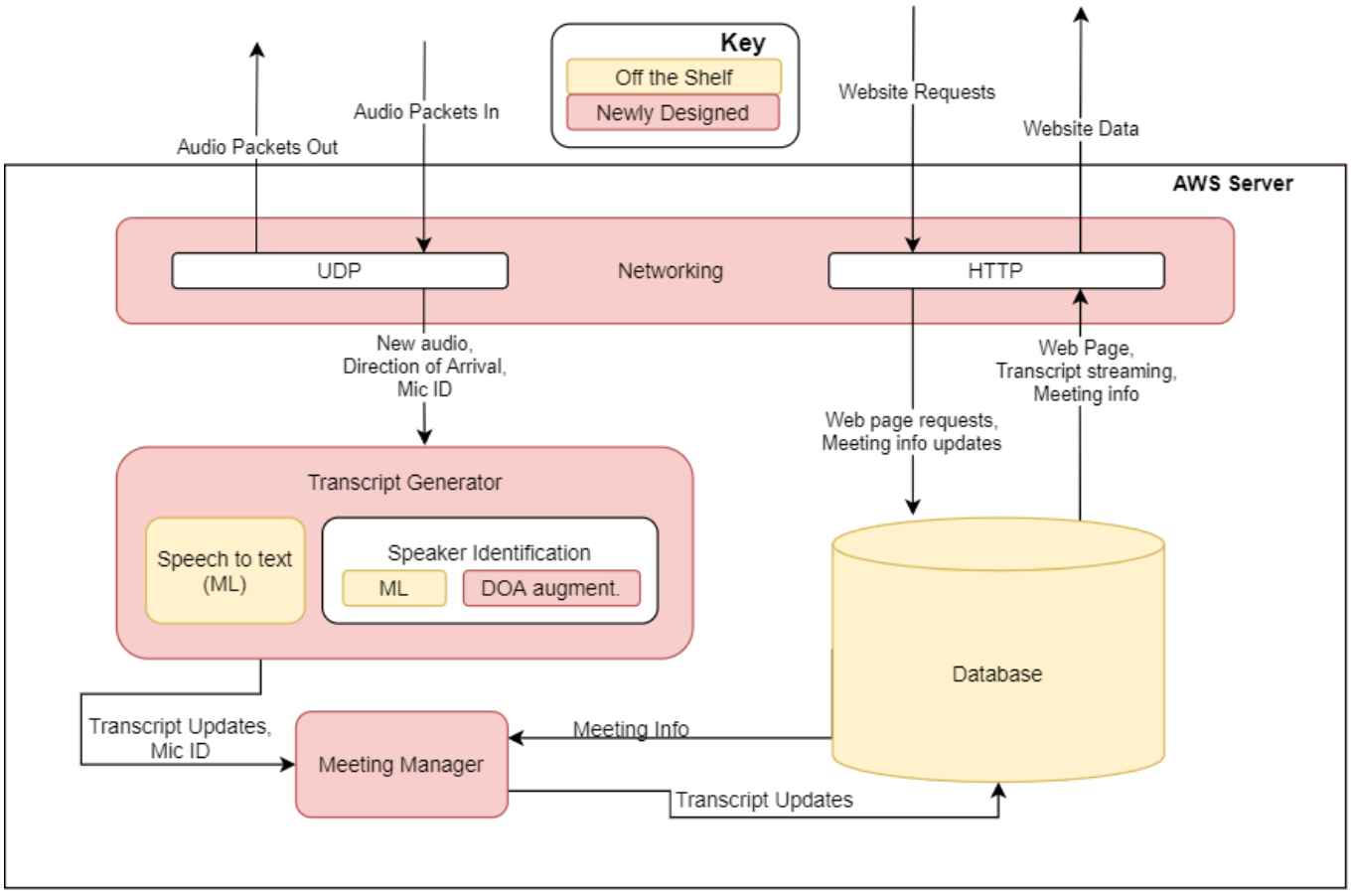

Fig. 3. AWS Software Diagram

The transcript generator joins the second thread and passes the microphone identifier, speaker change status, new speech to text result, and new speaker ID result to the meeting management module.

The meeting management software module is the next stage in transcript processing. During the microphone initialization stage, the meeting manager compiles a list of participants and their locations instead of adding to the official transcript. After initialization is finished, this information is given to the speaker identification submodule.

During the normal course of a meeting, the meeting manager's job is to compile the official transcript. After receiving new information from the transcript generator, the manager looks up the meeting that this microphone is a part of and places a lock on that meeting's transcript so it can't be modified by two threads simultaneously. Then, the manager detects microphone changes—if the current end of the transcript file does not match what the manager remembers having written last from this microphone, then a microphone change is detected. In the case of a microphone change, a speaker change reset is sent to the transcript generator of each microphone in the meeting; also, the speaker change status of the new audio is considered to be 'true.'

Finally, the meeting manager modifies the transcript file to reflect the new information. If the speaker change status is 'true,' then the new speaker and speech content are simply appended to the end of the file. Otherwise, the end of the file is overwritten by first deleting the speaker and speech content last written to the transcript and then replacing it with the updated projections given by the transcript generator.

Next, a connection must be maintained with the website. We need to send the transcript using channels for the users to view during a meeting. This connection is HTTP so that we can ensure packets arrive at the website and are in order. We can ensure this since HTTP uses TCP as an underlying protocol; this is also ensured by HTTP, assuming normal network conditions. Here we have two different types of data being sent. One is the transcript that is generated at the time of a meeting. The next data contains the meeting data information since the meeting database is stored on the server.

The type of data received by the server on this connection are meeting updates such as the joining, starting, or closing of a meeting. The updates are handled by the database and meeting manager.

VI. Project Management

A. Schedule

As seen in Fig. 5 and 6 at the end of the document, our project schedule has been divided into four phases. In the first phase, system setup, we ordered parts, configured the hardware we obtained, we configured our web server, and we set up some initial speech-to-text functionality. Phase two, currently ongoing, is focused on the backend of the product that the user would not generally see. The networking layer both on the Raspberry Pi and the web server will be written. Non-moving transcriptioning, database and website backend, and filter design are also on the todo list for this phase. The third phase, then, is the frontend phase. The meeting setup and transcript streaming web pages will be developed. Transcription capability will be enhanced as well. Finally, the fourth phase leaves multiple weeks at the end of the project to account for system testing, making the site look nice, and whatever unaccounted-for tasks might arise. Integration is expected to be continuous, as is sub-module testing as each part is completed.

B. Team Member Responsibilities

Fig. 4 shows the general distribution of responsibilities among team members. Mitchell had prior experience with websites from his research; Ellen had used machine learning packages at previous jobs; and Cambrea had the most knowledge about hardware setup from the classes she had taken. The remaining tasks outside of these were divided equally between the members.

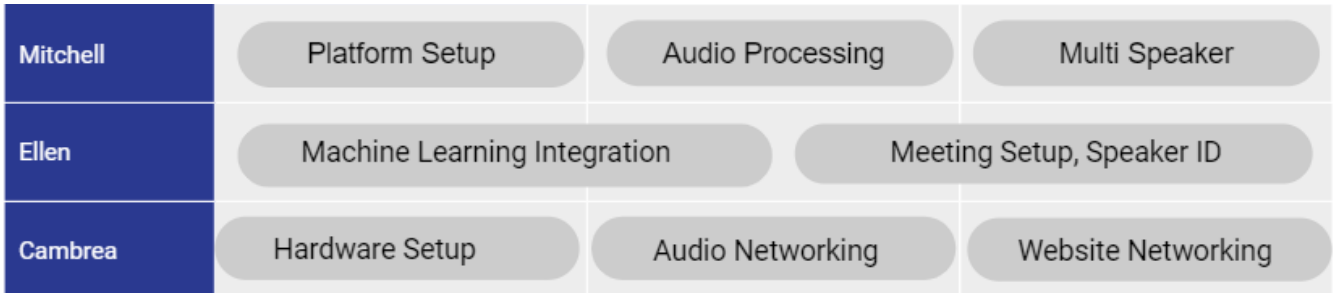

Fig. 4. Team Member Responsibilities

C. Budget

Our main source of spending was the ReSpeaker Mic array since we need to order 2, this totals to $158. Fortunately we were able to borrow the other high priced items from inventory, a JBL Speaker and a Raspberry Pi 3 B. Our budget and our table of parts can be found in Fig. 7. and Table 3.

D. Risk Management

The risks to our project come in multiple forms. There are some associated with the technical design and others that come from the project plan. We have mitigation strategies in place for all the risks we've identified, and we've in fact already had to enact one as risks actually come into play.

The first category of risk we may encounter is technical risk

relating to our solution design. As described in section ii, we have identified a set of requirements for our design, and we have tests in place to verify whether or not our solution meets the requirements after its implementation. We consider each requirement to be a potential source of risk, the risk being, of course, that our solution fails our tests. Because of this, we have planned fallback strategies corresponding to each requirement.

To delve into the specifics of these strategies, we'll describe them in the order the requirements were originally introduced. Audio transmission latency, and similarly transcript latency, may be improved by investing more money into the AWS server's networking specs, or by relocating parts of the processing onto the server from the hardware. We might improve the dropped packet rate by reducing packet size or selecting a different transport layer protocol. Transcript error, both word errors and speaker identification errors, can be combated by trying alternative machine learning models or switching from free software to paid services -- this is a path we have already had to explore, adding paid Google services to our transcription arsenal given preliminary speech to text results. Transcript formatting error is a non-ML issue and can be corrected by sending more metadata, such as timestamps, to the meeting management software module.

The project planning risks are related to, but separate from, the technical risks. First of all, there is a danger that, even if our solution works, it takes longer to implement than we anticipated. A similar risk is that, having implemented our solution according to the schedule, we find that it doesn't pass some test in the end. Our schedule design is intended to help with this: we have allocated multiple weeks at the end of the semester for testing, and this purposefully includes extra time for revision or further development. A second project planning risk is related to our budget, which has a formal cap of $600, albeit with some informal wiggle room. The risk, naturally, is that our need for parts exceeds our budget. However, through careful research of the parts and services we request to buy, we believe we have mitigated this potential risk.

VII. Related Work

The field of web conferencing solutions is well-saturated at this point, with seemingly every major software company offering their own alternative as an element of their suite of products. Here we'll highlight just a few major examples. We'd also like to point out the work of previous ECE Capstone group iContact, who created a project in the same field of distributed meetings with multiple participants per room. Their solution, though, focused on the video portion more than the audio portion.

The current most famous web conferencing tool is Zoom. The focus of this software is audiovisual meetings between multiple individuals. Zoom allows live closed-captioning to be performed manually by participants or added by a third-party service [13]. It also offers live speech-to-text transcription in paid versions. Transcription does not indicate who is speaking [14]. Microsoft Teams, another prominent web conferencing solution, does offer a form of speaker attribution in its live transcription service. However, this very new feature assumes that there is only one participant per connected device [15]. Cisco's Webex provides the same level of speaker labeling as Microsoft, along with other interesting functionality using voice commands [16].

These and other common conferencing tools allow each instance of the application to run on different hardware. This is great for portability, but limits speaker identification capabilities.

The area of hardware microphone-speaker meeting tools is, however, one under active development in the industry. Only a few weeks ago, at the beginning of March, Microsoft demonstrated its plans to release "Intelligent Speakers, small puck-like devices that can identify up to 10 different voices in a Microsoft Teams meeting" [17]. The speakers can transcribe and translate meetings. It's interesting that a prominent company is developing their version of the product simultaneously as we develop ours. When we learned about this news we believed it demonstrated that there is a real demand for this technology.

References

[1] VOIP-info.org, "QoS." https://www.voip-info.org/qos/
[2] vyopta.com, "Troubleshooting Packet Loss: How Much is an Acceptable Amount?" Dec. 2018. https://www.vyopta.com/blog/video-conferencing/understanding-packet-loss/
[3] B. McLaughlin, "Recommended Practices for Closed Captioning Quality Compliance," Jan. 2015. https://s3.amazonaws.com/eegent-assets/resources/McLaughlinB190115.pdf
[4] C. Muneanu et al., "Measuring the Acceptable Word Error Rate of Machine-Generated Webcast Transcripts," Jan. 2006. https://www.researchgate.net/publication/221485939_Measuring_the_acceptable_word_error_rate_of_machine-generated_webcast_transcripts
[5] R. Morais, "DeepSpeech 0.6: Mozilla's Speech-to-Text Engine Gets Fast, Lean, and Ubiquitous," Dec. 2019. https://hacks.mozilla.org/2019/12/deepspeech-0-6-mozillas-speech-to-text-engine/
[6] D. Huggins-Daines, "PocketSphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices", 2006. https://www.cs.cmu.edu/~dhuggins/Publications/pocketsphinx.pdf
[7] E. Protalinski, "Google's Speech Recognition Technology Now Has a 4.9% Word Error Rate," May 2017. https://venturebeat.com/2017/05/17/googles-speech-recognition-technology-now-has-a-4-9-word-error-rate/
[8] H. Bredin et al., "Pyannote.Audio: Neural Building Blocks for Speaker Diarization," May 2020. https://ieeexplore.ieee.org/document/9052974
[9] J. Patino, "EURECOM Submission to the Albayzin 2016 Speaker Diarization Evaluation," 2016. https://www.semanticscholar.org/paper/EURECOM-subm

ission-to-the-Albayzin-2016-speaker-Patino/d8917a687edaaae061b59cf0baec95ced41e9dfc?p2df
[10] Y. Fujita et al., “End-to-End Neural Speaker Diarization with Self-Attention,” Dec. 2019. https://ieeexplore.ieee.org/abstract/document/9003959
[11] F. Landini et al., “Bayesian HMM Clustering of X-Vector Sequences (VBx) in Speaker Diarization: Theory, Implementation, and Analysis on Standard Tasks,” Dec. 2020. https://arxiv.org/abs/2012.14952
[12] DPA Microphones, “Facts About Speech Intelligibility,” March 2021. https://www.dpamicrophones.com/mic-university/facts-about-speech-intelligibility
[13] Zoom Help Center, “Closed Captioning and Live Transcription.” https://support.zoom.us/hc/en-us/articles/207279736-Closed-captioning-and-live-transcription
[14] Vanderbilt Brightspace, “How Can I Enable Live Transcription in Zoom?” Feb. 2021. https://www.vanderbilt.edu/brightspace/how-can-i-enable-live-transcription-in-zoom/
[15] R. Noureen, “Microsoft Teams Adds Live Transcription with Speaker Attribution for Meetings,” March 2021. https://www.onmsft.com/news/microsoft-teams-adds-live-transcription-with-speaker-attribution-for-meetings
[16] A. Barave, “Turning Talk into Action: Bringing Voice Intelligence into Webex Meetings,” Jan. 2020. https://blogs.cisco.com/collaboration/turning-talk-into-action-bringing-voice-intelligence-into-webex-meetings
[17] T. Warren, “Microsoft’s New Intelligent Speakers Deliver its Promised Meeting Room of the Future,” March 2021. https://www.theverge.com/2021/3/2/22308962/microsoft-intelligent-speaker-teams-translation-transcription-features

Task	Team Member	Status	Week 3(Feb 15-21) M	W	F	Week 4(Feb 22-28) M	W	F	Week 5(March 1-7) M	W	F	Week 6(March 8-14) M	W	F	Week 7(March 15-21) M	W	F	Week 8(March 22-28) M	W	F
Phase 0: Design	all	done																		
Initial research	all	done	all	all	all															
Project Proposal Presentation	all	done	all	all	all															
Phase 1: System Setup	all	progress																		
Network Design: RPi to AWS	cne	done				cne	cne	cne												
Network Design: AWS to rpi	cne	done						cne	cne	cne										
Hardware Setup and Integration	cne	progress						cne	cne	cne	cne									
Respeaker Config on Raspberry Pi	cne	todo								cne	cne	cne								
AWS Database + Django Initialization	mfy	done				mfy	mfy	mfy												
AWS EC2 Setup	mfy + cne	done				mfy	mfy	mfy	cne	cne	cne									
Speech to Text ML Initial Version	eseeser	done				eseeser	eseeser	eseeser												
Design Presentation & Report (presentation due 7th, report 17th)	all	progress				all	all	all	all	all	all	all	all	all						
Backend Pre-ML Processing	eseeser	done						eseeser	eseeser	eseeser										
Test & Slack Time	all	done										all								
Phase 2: Backend	all	todo																		
Database Creation and Integration	mfy	progress						mfy	mfy	mfy	mfy	mfy	mfy							
Website Creation, Including Backend	mfy	progress						mfy	mfy	mfy	mfy	mfy	mfy							
Non-Moving Speaker ID (no ML)	eseeser	done								eseeser	eseeser	eseeser	eseeser							
Server connection at AWS	cne	progress										cne	cne	cne						
Server connection at RaspberrryPi	cne	progress											cne	cne	cne	cne	cne			
Speech to Text ML Integration (first pass at meeting manager)	eseeser	done											eseeser	eseeser	eseeser					
Audacity Filter Design + Audio Process	mfy + eseeser	progress								mfy	mfy	mfy	mfy	mfy	eseeser	eseeser	eseeser			
Raspberry Pi integration	cne	todo														cne	cne	cne		
Mic Initialization Backend	mfy	todo															mfy	mfy	mfy	mfy
Testing & Slack Time	all	todo																all	all	all

Fig. 5. Schedule, Phases 0-2 View

Task	Week 7(March 15-21) M	W	F	Week 8(March 22-28) M	W	F	Week 9(March 29-April 4) M	W	F	Week 10(April 5-11) M	W	F	Week 11(April 12-18) M	W	F	Week 12(April 19-25) M	W	F	Week 13(April 26-May 2) M	W	F	Week 14(May 3-5) M	W
Phase 3: Frontend																							
Create housing	mfy	mfy	mfy	mfy																			
Transcript Support for Multi-mic Meetin	eseeser	eseeser	eseeser																				
Full On-Website Meeting Setup			eseeser	eseeser	eseeser	eseeser	eseeser																
Simultaneous Speaker Handling						mfy	mfy	mfy	mfy	mfy	mfy	mfy											
ML + Hardware Integration							cne	cne	cne														
Moving Speaker ID (ML)							eseeser	eseeser	eseeser	eseeser	eseeser												
Transcript Web Streaming								cne	cne	cne													
Testing & Slack Time										all	all	all											
Phase 4: Touchup																							
Web CSS												all	all	all	all								
Integration and Complete System Testi							all	all	all	all	all	all	all	all	all	all							
Slack Time																all	all	all	all	all			
Final Report and Presentation													all	all	all	all	all	all	all	all	all	all	

Fig. 6. Schedule, Phase 3-4 View

Item	Quantity	Price	Total Price	Link
Respeaker Mic Array v2.0	2	$79.00	$158.00	https://www.amazon.com/Se
SD card for Raspberry Pi	1	$6.57	$6.57	https://www.amazon.com/Sa
Raspberry Pi case	1	$9.99	$9.99	https://www.amazon.com/Bla
Raspberry Pi Power cable	1	$9.99	$9.99	https://www.amazon.com/Ca
ReSpeaker Case	2	$10.00	$20.00	
AWS Credits				
$50				
Parts From Inventory				
JBL Speaker	1	$0.00	$0.00	Parts Inventory: EDE0076
Raspberry Pi 3 B	1	$0.00	$0.00	Parts Inventory: EDE0130
			$600.00	Budget (approved vendors)
			$204.55	Total spent
			$395.45	Total remaining

Fig. 7. Budget and Parts List

TABLE III. BILL OF MATERIALS

Material	**Type**	**Quantity**
Google Cloud Speech to Text	Software service	N/A
Mozilla DeepSpeech	Software package	N/A
CMU PocketSphinx	Software package	N/A
pyannote audio	Software package	N/A
pyBK	Software package	N/A
Hitachi Speech EEND	Software package	N/A
BUTSpeech FIT VBx	Software package	N/A
Django	Software package	N/A
SQLite	Software package	N/A
AWS EC2 M5a instance	Web deployment service	1

Material	**Type**	**Quantity**
Micro SD Card	Hardware	2
JBL Speaker	Hardware	2
Raspberry Pi 3 B	Hardware	2
Raspberry Pi Power Cable	Hardware	2
ReSpeaker Microphone Array V2.0	Hardware	2
ReSpeaker Microphone Array Case	Hardware	2
3.5 mm Audio Cable	Hardware	2
USB A to micro USB Cable	Hardware	2