ClouDJ: A Music Sharing Application

Amanda Hittson, Rebecca Lam, Josh Slauson
University of Wisconsin-Madison

ABSTRACT

ClouDJ enables music sharing. Listening to music is largely a social activity that is best enjoyed with others. However, there is currently no easy way to listen to music together online. We plan to make listening and sharing music easier with our system ClouDJ. ClouDJ will allow users to play their own music while sharing it with other users in real-time.

1. INTRODUCTION

For our project we will design and implement ClouDJ, a music sharing application that uses a centralized server system. The ClouDJ application allows multiple clients to simultaneously listen to music together in shared sessions. To this end, we designed a system with the following goals:

- Performance: users that are participating in a session together should be able to listen to music at a similar rate.
- Availability: users must be able to continuously listen to music in a session assuming the user hosting the session is available.
- Scalability: adding clients should not affect system performance.

2. DESIGN

To create ClouDJ, we will implement the design described in subsections 2.1 and 2.2 by integrating Google App Engine. Figure 1 depicts the overall system and how each element interacts during a normal session. If we have enough time, we will consider multi-host sessions, dynamic access control lists, and adding remote files to session playlists.

Figure 1: Depiction of typical message flow for one music session. 1) Host client sends a message to the master server telling it to create a session. 2) The master server chooses a session server to serve the session and sends it a create session message. 3) The chosen session server sends a message back to the host client to establish the connection. 4) The host then sends data received from the host client to client listeners.

2.1 Front End

A session is an abstraction in which multiple users may listen to one song that is hosted by a single user. Our front end application will allow a user to either create a session and become a host, or join an existing session and become a listener. When acting as a host, a user may add or remove songs from the session playlist, select the currently playing song, and play/pause the current song. When acting as a listener, the user has no control over what song is being played. Users can see all sessions that they are able to join. Only members of a user's access control list (ACL) may see or join any session in which that user is a host.

2.2 Roles

There are three major roles in our system: the master server, session server, and client. Clients own and store music, can share their music with others and can listen to music shared by others. There is a single master server that provides access control, starts sessions for users who want to host and provides minimally acts as a liaison between the clients and session servers. Session servers' main functionality is streaming music from host-client to listener-clients.

2.3 Back End

The backend infrastructure is more complicated than the frontend. The master server keeps track of users currently online, user ACLs, and user membership lists (ACLs it is a member of). It also is responsible for maintaining a session table, a table that maps host users to the session servers (this is a one-to-one mapping). When a client logs on, the master server informs each session server associated to a host on the client's membership list that this client is a potential listener.

The session server is the workhorse of the system. It services requests for sessions it is in charge of by routing data from the host client to relevant clients (listeners). It maintains the list of listeners and potential listeners. A potential listener is a client who could listen to this session, but is currently not. In other words, these are the clients on the host client's ACL that are logged in but not listening to this session. Session servers also take care of session cleanup when a session ends (or fails).

Finally, the client exists on the user machine has access to its user's music and playlists and also keeps track of data such as the user's current session and the user's potential sessions (sessions this user can access). For the current session, the client keeps it's session server address and stream. For the potential sessions, the client keeps the host, session server address, and currently playing song.

Sessions are created when a client contacts the master server about hosting a new session. The master server then contacts a session server to create a new session. On success, the session server contacts the client, which updates its current session information. Clients may join sessions by contacting the session server, at which point, data from the host will be sent to the session server, which will forward the stream to any listeners. Notice that after a session is established, the client no longer communicates with the master server and the master server handles no client data, only client meta-data.

3. IMPLEMENTATION

Here is how to cite things, see [1], [2] or [3].

4. EVALUATION

We will evaluate our system on the metrics of performance, availability and scalibilty. To measure performance, we will design benchmarks to measure the data rate of a listener versus the data rate of the host. To evaluate availability, we will examine how our system behaves with various failure modes by artificially taking down each role (client, master server, session server). For scalability, we will measure system performance as we add clients to the system. We can achieve this by launching our application on an increasing number of client machines.

5. FUTURE WORK

<ADD FUTURE WORK INFORMATION HERE>

6. CONCLUSION

We will be developing a music sharing application called ClouDJ. With ClouDJ, users will be able to host music sessions that other users can listen to in real-time. It will consist of many client users as well as some centralized servers to handle session and user state. We will make use of Google App Engine for the implementation of the system.

7. REFERENCES

[1] `https://developers.google.com/appengine/`.

[2] `http://www.google.com/+/learnmore/hangouts/`.

[3] `http://www.csc.kth.se/~gkreitz/spotify-p2p10/spotify-p2p10.pdf`.