Decentralized Music Rooms

Group Members: Seth Fuller, Blake Troutman, Umesh Raja

Objective

There are several different file-sharing systems that have been created and can be used to distribute music. These systems share files to anyone and everyone who is on either a large P2P network, like Soulseek and BitTorrent, or on a local network, like iTunes. But none of these systems support the concept of what we call a "music room." In our peer to peer application, any user can create a music room, and, once a user creates one, he or she can allow specified peers access to this music room. Within this private music room, any user can host music and stream or download music hosted by other peers in the music room. Since this application is built upon a peer to peer foundation, once a user goes offline, the music that he or she hosted is now unavailable. This is why our system also allows users to pin music to their computer.

There are a couple key quality attributes for this application that are especially important to us. This application will entail ensuring that each music room that is created is secure and cannot be tapped by other uninvited peers. Additionally, since all of the data throughout this application is transferred and stored in a distributed manor, the speed of streaming and downloading audio files should greatly increase since we aren't using a centralized server.

Approach

This application will be built upon the P2P hypermedia protocol called IPFS (Interplanetary File System). The main entry point to the site will be a SPA (Single Page Application) stored on an AWS S3 bucket. But there won't be a centralized server that stores and distributes the audio data that each user uploads and requests. Instead, each user will be an IPFS node and, when they "upload" audio files to a music room, those files will be hosted on their local machine but accessible to any of the peers in the music room. Users will also be able to pin audio files so that they can continue to listen to the music even if the original peer who hosted that music goes offline.

A music room will be an IPFS cluster and will be identified with a cryptographic hash, which is how each peer in the network is also identified. Whenever a user requests access to a music room, a message will be sent to the original peer who created the music room. If the peer accepts the request, the requester will gain access to the music room and all of the files that are hosted by the peers in that specific room.

Because each member within a music room is able to add other peers to the music room, there needs to be some way of tracking who added who. This will be done by using a BLT (blame tree) that is synchronized between each peer in the music room. A BLT is essentially a tree data-

structure that tracks who-added-who in the music room. Each node in the tree is identified by a peer's hash, contains a timestamp for when the peer was added to the music room, and includes a binary flag as to whether the peer is still active in the music room or not (which is useful if the peer happens to log off or leave a music room). The parent of each node is the peer that added them, and each child node is the peer that was added. Nodes are never removed from the tree, and the tree will continue to grow as peers are added to the music room. BLTs will be used to track how the music room population grows. If there is a peer that is added to the room that the room creator doesn't approve of, he/she can view the room's BLT and see who added the peer. He then can remove whichever members he/she sees fit.

The team has a large dataset of royalty free, custom-made audio files that will be used for developing and testing our application.

Limitations

- Since there are only three people in our team, we might not be able to test how the application scales as the number of users increases. This is a minor issue because we will be designing the application to be scalable from the beginning. Building our application upon an IPFS foundation will give us the ability to scale. As the number of user's increase, the number of nodes in the network will as well. If a user requests an audio file, the file will be retrieved from the nearest peer that has that audio file pinned. Therefore, our application is inherently decentralized with no central server having to mitigate all of the incoming requests. In addition to this, as the number of audio files increases in the app, storage space will not be an issue since all of the files are distributed throughout the peerspace.
- IPFS is a sizable technology to learn. So, each member will have to spend time digging into documentation and tutorials. This is also a minor issue because, since there is no sizeable backend to this application that needs to be built and the brunt of the IPFS logic is located on the frontend, the main development for this project will be on the frontend. So, all of our resources can be spent there.

Responsibilities of Individual Members

As already stated, the majority of the development will be on the frontend. So, all of the members will be contributing to this area of the application. Tasks that need to be completed on the frontend will be distributed to each member as seen fit.

Seth Fuller will also handle setting up the AWS S3 bucket for the SPA to reside.