CPS 512.01 Spring 2015 Final Project
Synchr

Miles Oldenburg, Bobo Bose-Kolanu
Duke University

April 14, 2015

Contents

1 Introduction **3**

2 Technology **3**

3 Server Nodes **3**

4 Client Nodes **3**

5 Current Limitations **3**
5.1 Server Node Robustness 3
5.2 Server Node Tracklist Memory 3

6 Future Work **4**
6.1 Control Opt Out 4
6.2 Pre-Fetch 4
6.3 Contributing 4

1 Introduction

Skeleton Introduction

2 Technology

Skeleton Technology

3 Server Nodes

Skeleton Server Nodes

4 Client Nodes

Skeleton Client Nodes

5 Current Limitations

5.1 Server Node Robustness

The server network is not robust to node failures. Each node can be connected to many other nodes in order to share tracklist data. If a node fails, then all clients of the current node who have tracks from the failed node in their tracklist would show an error when attempting to play tracks from the failed node.

The solution to this problem would be for each node to listen for disconnects from connected nodes and then notify all connected clients to remove tracks from the failed nodes from their tracklists.

5.2 Server Node Tracklist Memory

Server nodes do not store the combined tracklist data in memory. If a new node joins the network and they receive new tracklist data, the node simply forwards the new data on to other nodes and clients. This means that if a client has joined the network after a server node it would not receive the combined tracklist for both server nodes. It would receive only the tracklist

for the server node it connected to in addition to all tracklist updates from any subsequent server node joins.

The solution to this problem would be for each server node to store combined tracklist data in memory. Then when a new client connects to it, the client can immediately be aware of all tracks the server node knows of.

6 Future Work

6.1 Control Opt Out

In the current version of the software, any client connected to the network will receive control messages. This feature may not be always desirable so an option to opt out of sending and receiving control messages could be added to the client interface.

6.2 Pre-Fetch

In a normal listening session we can expect the next song in the library to play after the current song has finished. In the current version, the upcoming song would only be streamed after the current song has finished. One optimization would be to look ahead in the potential play queue and fetch a number of songs into browser memory so when the current song finishes playing the next song can be played immediately.

6.3 Contributing

Source code of the application is available at
`https://github.com/milesoldenburg/music-synchr`