CS 2620

Prof Waldo

Spring 2025

Distributed Computing - Final Project

Remixing Spotify:

Synchronized Music Jams!

Eric Gong, Ryan Jiang, Evan Jiang, Charlie Chen

CS 2620

Prof Waldo

Spring 2025

01

Problem Overview

What's so hard about synchronizing music?



Project Goal

Create a music player with the following traits:

- Any user can start/stop/skip a song
- Any user can add a song
- All users hear the exact same thing



Technical Challenges

Similar to many other distributed systems, we must:

- Define a synchronized notion of time across multiple devices
- Maintain a consistent truth shared among clients
- Limit processing overhead to maintain user experience

Commercial Implementations

These aren’t new problems:

Obstacles and Considerations

This is a non-trivial task

- Varying network and hardware delays vary
- Humans are good at hearing subtle offsets in sound
  - The average Gap Detection Threshold is 4.19ms https://pubmed.ncbi.nlm.nih.gov/18465408/

CS 2620

Prof Waldo

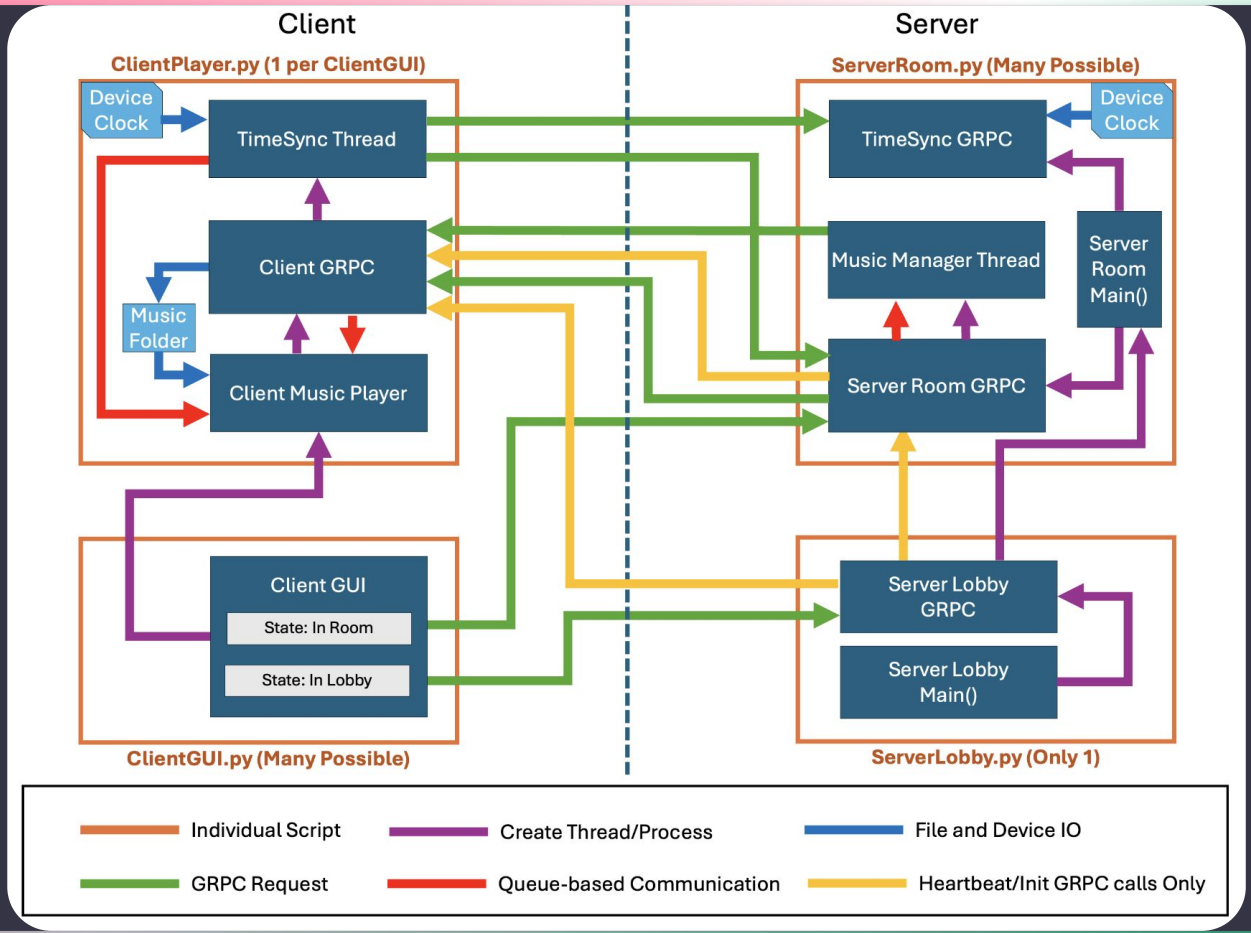
Spring 2025

Distributed Computing - Final Project

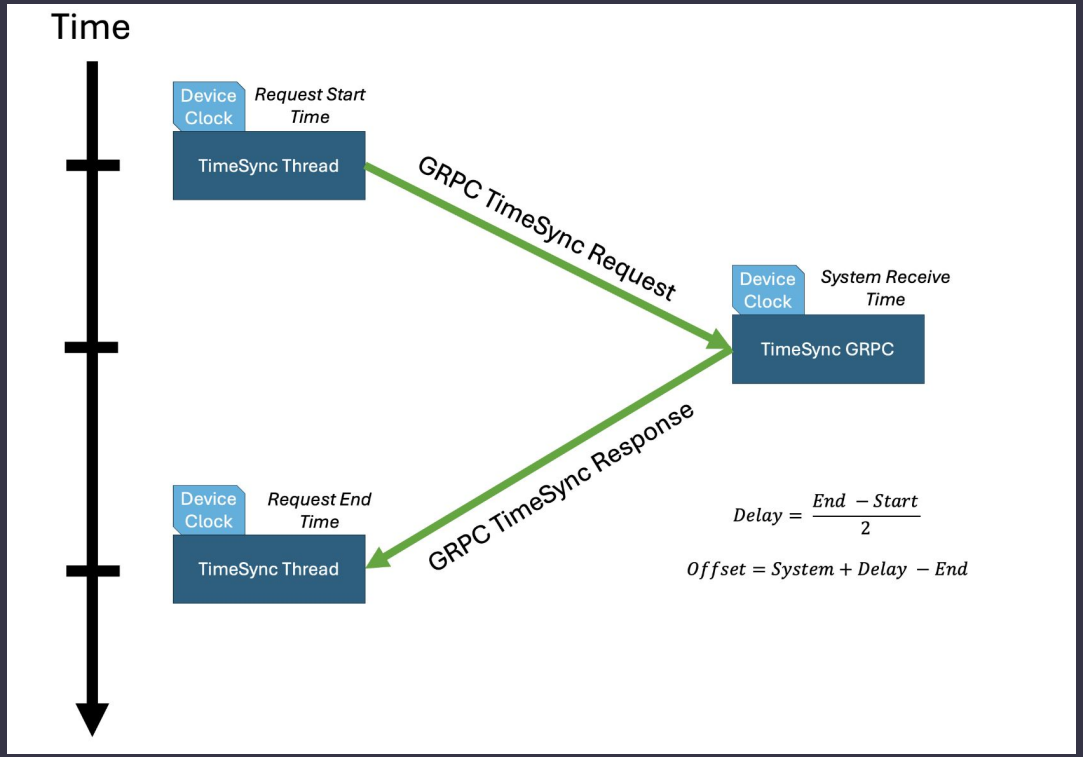
02

Design Choices

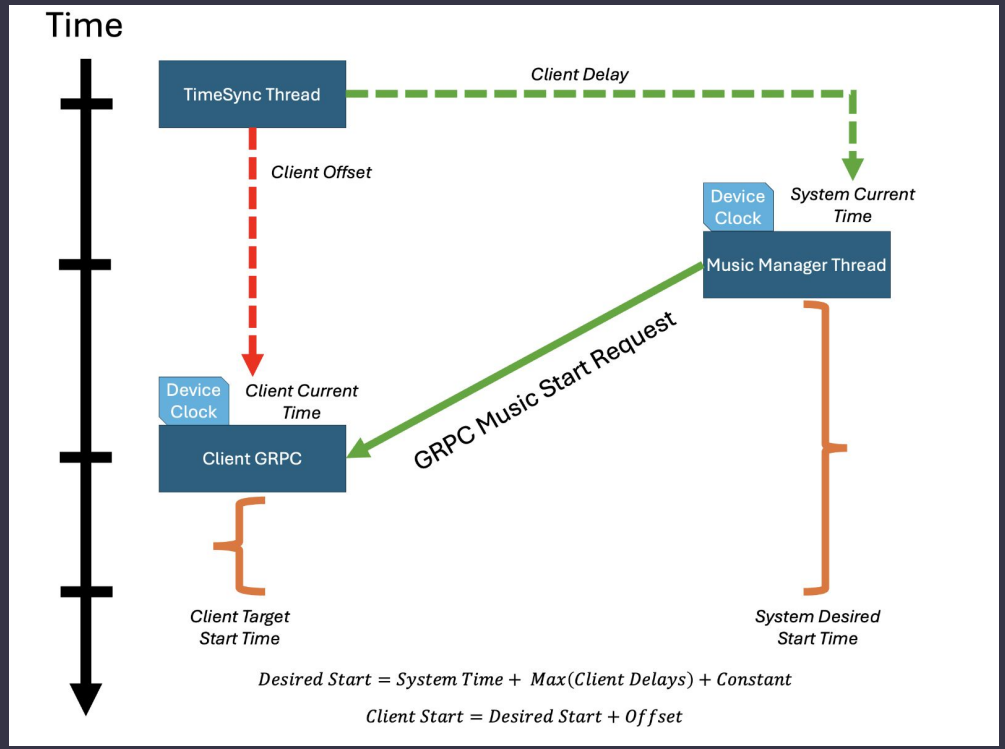
What implementation choices were made?


Client
Server
ClientPlayer.py (1 per ClientGUI)
ServerRoom.py (Many Possible)
Device Clock
TimeSync Thread
TimeSync GRPC
Device Clock
Client GRPC
Music Manager Thread
Server Room Main()
Music Folder
Client Music Player
Server Room GRPC
Client GUI
State: In Room
State: In Lobby
Server Lobby GRPC
Server Lobby Main()
ClientGUI.py (Many Possible)
ServerLobby.py (Only 1)
Individual Script
Create Thread/Process
File and Device IO
GRPC Request
Queue-based Communication
Heartbeat/Init GRPC calls Only

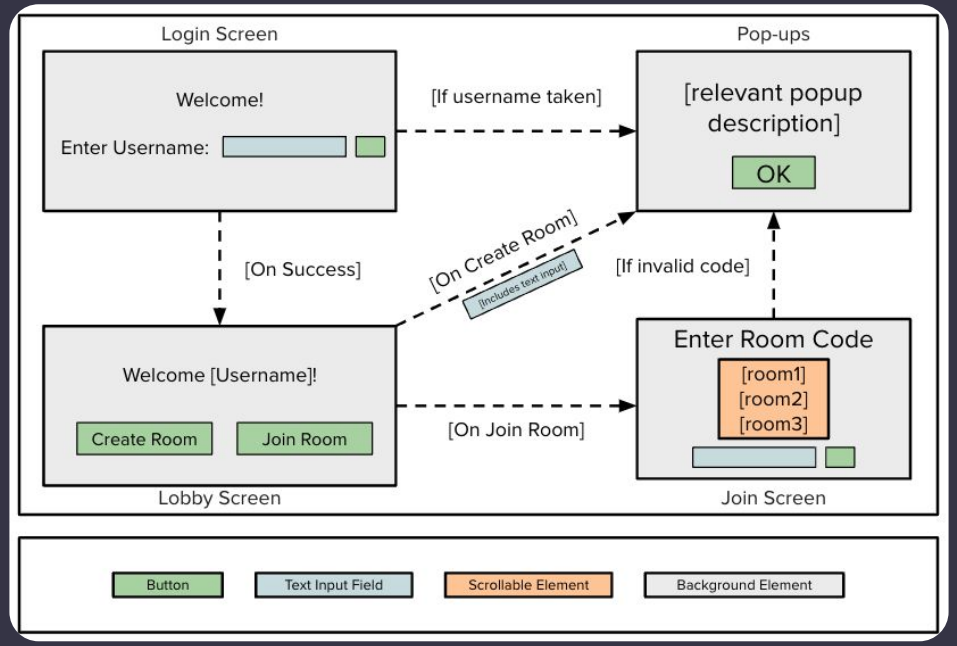
Custom Network Time Protocol



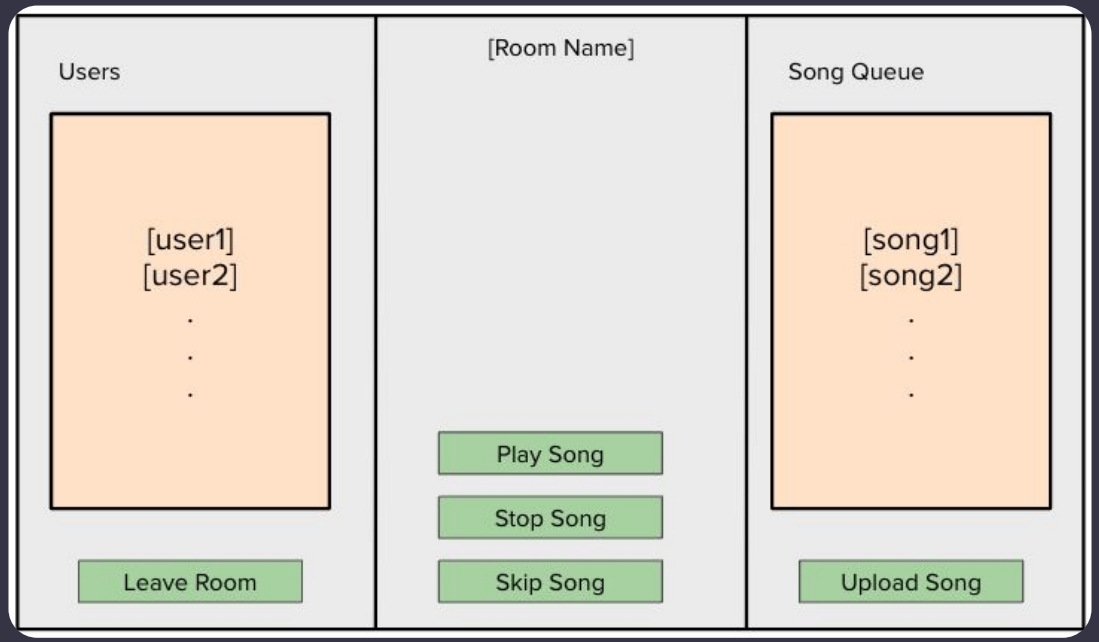
Command Delivery (at most once)



Front End Sequencing [Part 1]



Front End Sequencing [Part 2]



CS 2620

Prof Waldo

Spring 2025

Distributed Computing - Final Project

03

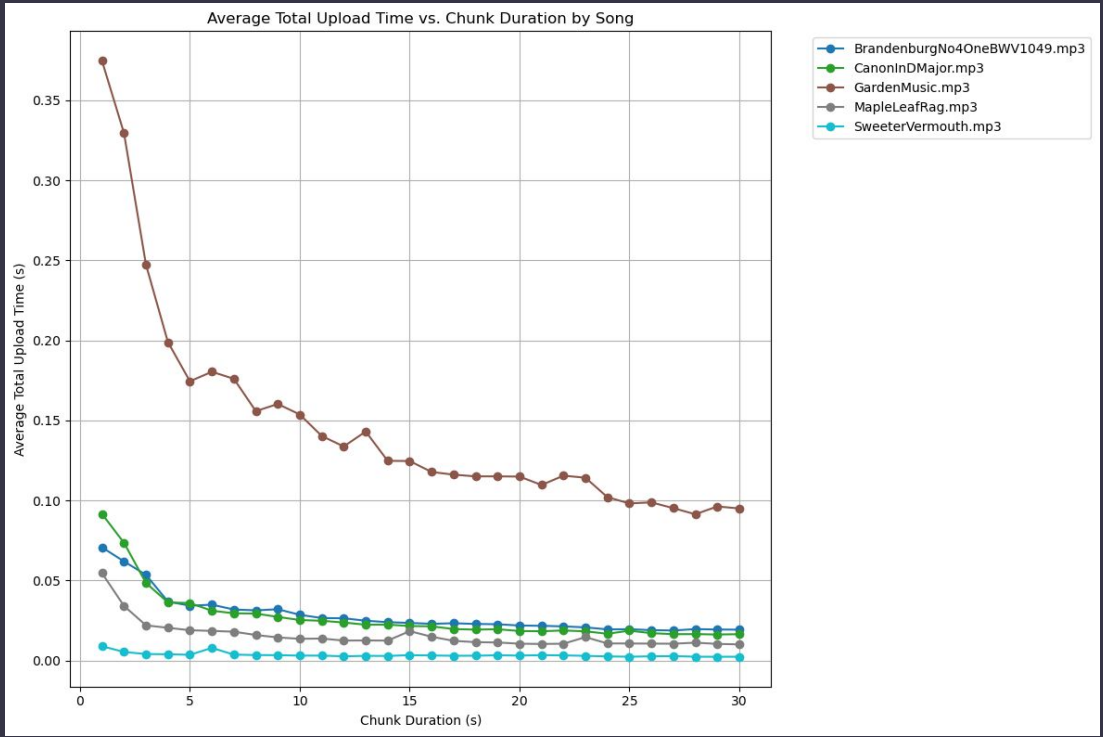
Testing Suite

Why are we sure it works?

Chunk testing [1/4]

- Initial Plan: pre-load chunks of X seconds
- Pros: in theory it reduces delays of uploading a song
- Cons: if the upload time exceeds chunk length, no point
- Questions:
  - Are chunk sizes fixed or dynamic w.r.t song length
  - If they are fixed, what is optimal chunk size?
    - W.r.t total upload length for all chunks?
    - W.r.t maximum time for one chunk upload

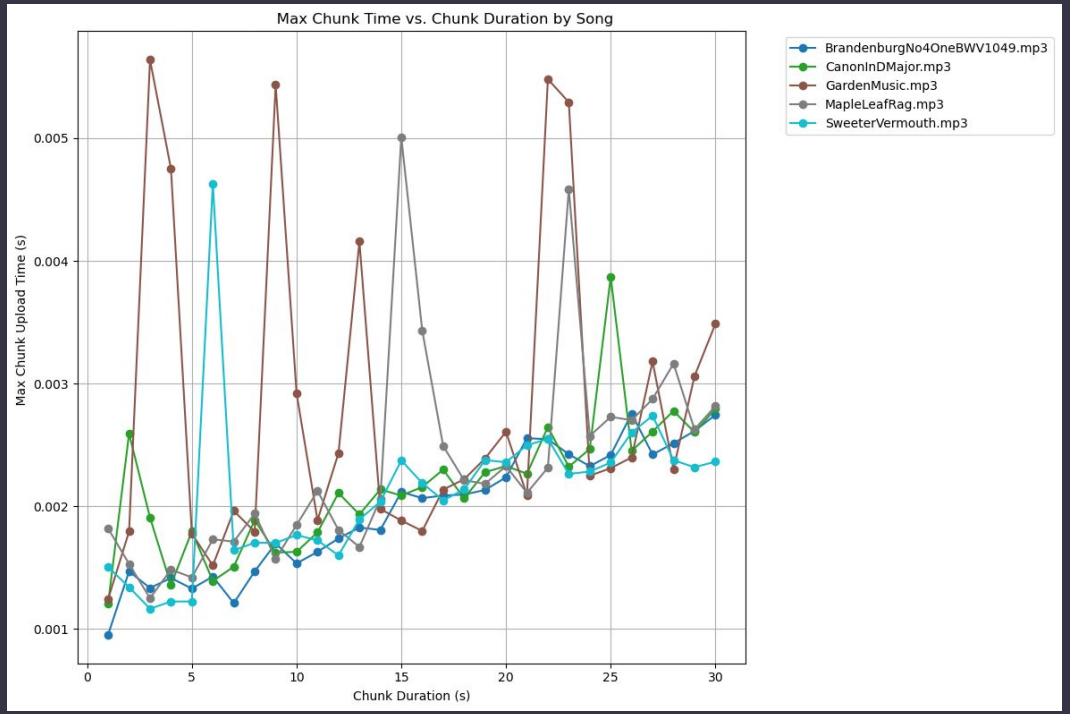
Chunk testing [2/4]



Conclusion:

Larger chunk durations actually **reduce** total upload time

Chunk testing [3/4]



Conclusion:

The variability and spikes indicate there is no benefit to have smaller chunk uploads

Chunk testing [4/4]

- Tested with songs ranging from 25s to 40m
- Results indicate that the optimal chunk length was ~28 for all songs, showing the optimal chunk length is fixed independent of song length
- Based on graphical trends, larger chunks are better

Overall conclusion: upload the entire song at once without chunking



Our tests

- Unit Tests
- Integration tests
- Metronome ear test
- Person counting tests
- Client role-playing tests

Limitations

- Too many users can slow it
- Some strange testing errors with specific computer users (Mac OS v15?)
- Can only upload mp3 files
- Must pause when joining the room and then play to hear music

CS 2620

Prof Waldo

Spring 2025

04

Demo

Does it work?

CS 2620

Prof Waldo

Spring 2025

Thanks!

Questions?

Repo:
https://github.com/ericgong2005/CS2620_Final_Project/tree/main