Lab 4: Designing a Jukebox Protocol (Week 3)

Due: Monday, Nov 16th, 11:59 PM

How to play music…

Look at mp3-example.py as an example of the libraries we will use…

```
import ao
```

- library that gives you access to audio output
- needs raw audio output

How to play music…

Look at mp3-example.py as an example of the libraries we will use…

`import ao`

- library that gives you access to audio output
- needs raw audio output

`import mad`

- library that can decode MP3 data into the raw audio output that ao needs

`mad.MadFile(wrapper)`

- to play a new song, need to create this object `mad.Madfile`
- the input to this object must be an object that has a `.read()` method

`mad.MadFile(wrapper)`

- to play a new song, need to create this object `mad.Madfile`
- the input to this object must be an object that has a `.read()` method

`wrapper = FakeFile()`

- creates an object of type `FakeFile` that has a constructor and a `.read()` method
- to add music data to wrapper wrapper.data = music_data (or += music_data)
- get decoded music data: buf = wrapper.mf.read()
- if buf is None, then there is no music data left

`mad.MadFile(wrapper)`

- to play a new song, need to create this object `mad.Madfile`
- the input to this object must be an object that has a `.read()` method

`wrapper = FakeFile()`

- creates an object of type `FakeFile` that has a constructor and a `.read()` method
- to add music data to wrapper wrapper.data = music_data (or += music_data)
- get decoded music data: buf = wrapper.mf.read()
- if buf is None, then there is no music data left

`dev.play(bytes(buf), len(buf))`

- will play bytes

Let’s look at the client-side code and how you can interleave messages and playing music…

See Section 2.3 of writeup for Client Workflow….

Playing thread vs. receiving thread

- Playing music may block while playing music and receiving thread may block on `recv()` so we need two different threads for these operations to happen concurrently

Playing thread vs. receiving thread

- Playing music may block while playing music and receiving thread may block on `recv()` so we need two different threads for these operations to happen concurrently
- These two threads will share the `FakeFile` object (`wrapper`)

Playing thread vs. receiving thread

- Playing music may block while playing music and receiving thread may block on `recv()` so we need two different threads for these operations to happen concurrently
- These two threads will share the `FakeFile` object (`wrapper`)
- They need to use condition variables so only one of these threads is ever accessing this shared object

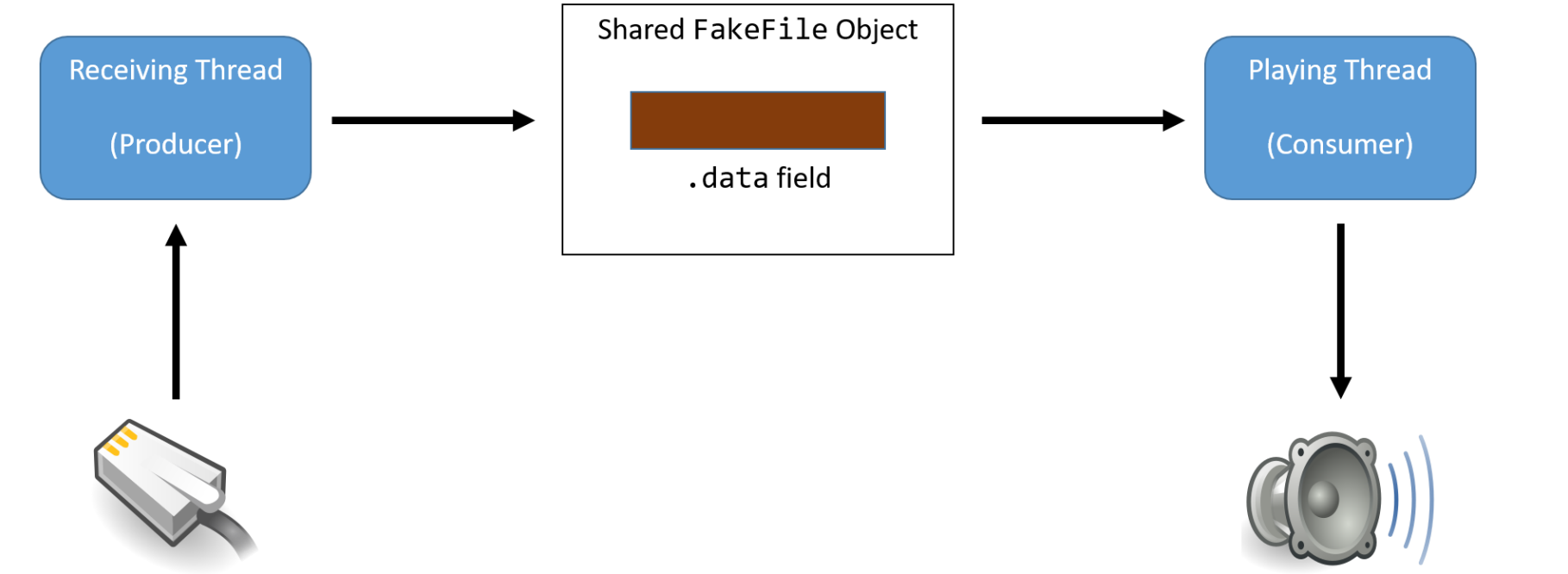
Playing thread vs. receiving thread

- Playing music may block while playing music and receiving thread may block on `recv()` so we need two different threads for these operations to happen concurrently
- These two threads will share the `FakeFile` object (`wrapper`)
- They need to use condition variables so only one of these threads is ever accessing this shared object
- Reading thread will be producer thread that adds data to the wrapper and playing thread will be the consumer thread that reads data from the wrapper

Playing thread vs. receiving thread

While **holding the lock**
- add song data to the shared `.data` field. (+=)
- notify playing thread that data was added

While **holding the lock**
- check length of `.data` and wait if empty
- when non-empty, call `mf.read()` to decode song data



Receive data from network (while **not** holding lock).

Send decoded song data to audio device (while **not** holding lock).

Using conditional variables in Python…

- `acquire`
  - acquire the lock
  - if a thread already has the lock, block until it is available
- `release`
  - release the lock currently holding
- `wait`
  - if you are holding the lock you are allowed to call wait to give up the lock and block until someone else notifies me
- `notify`
  - if you are holding the lock you are allowed to call notify that will wake up one of the threads waiting on this condition

Testing on catnip

- As of today, I've slowed ports 5001 to 5100 to be much slower to about 256kbps
- Run your server on catnip on one of these ports (Do not run your client on catnip!)

ssh catnip
./server 5099 /home/rware/music

You’ll need to test your client playing music from a CS lab machine

- You need to test your client from a CS lab machine
- You cannot ssh into the machines and test your client like you may have been able to for other labs

You'll need to test your client playing music from a CS lab machine

- You need to test your client from a CS lab machine
- You cannot ssh into the machines and test your client like you may have been able to for other labs
- Make sure the client is running on the machine you are currently sitting at. Some students have lost a lot of time thinking their implementation wasn't working when they were running their client on a remote machine

You'll need to test your client playing music from a CS lab machine

- You need to test your client from a CS lab machine
- You cannot ssh into the machines and test your client like you may have been able to for other labs
- Make sure the client is running on the machine you are currently sitting at. Some students have lost a lot of time thinking their implementation wasn't working when they were running their client on a remote machine
- Make sure the volume is turned up on the machine you are at. Test by trying to play a YouTube video or something

Lab 4 grading: You need to meet with me to demo and discuss your solution.

- Will post signups for next week
- I may spread these out over 2 weeks, but your code will be graded by what you pushed at the usual deadline

Checkpoint

- Can you connect multiple clients and each can send a list or info command and get a response while staying connected?
- Does you server not crash if a client disconnects?
- Can you exchange song data with a client?