流媒体播放器——实验报告

罗梓璋 1800012729 岳鹏云 1800012711 郑凌骁 1800013110

实验目的

研究流媒体传输与播放的原理，在实践中理解流媒体传输与播放中的困难所在和解决方法。

实验要求

- 实现简单的流媒体客户端播放器，及其配套的流媒体服务器。
- 播放器要求：
  - 实现获取远程文件列表和选择播放文件功能。
  - 实现接收流媒体片段文件并缓存。
  - 实现流量控制功能。
- 服务器要求：
  - 实现流媒体的储存功能。
  - 配合客户端完成流媒体片段文件的传输。
  - 配合客户端完成流量控制。

实验原理

流媒体储存格式

每一个多媒体文件以.ts格式分片段储存，使用ffmpeg将多媒体文件划分为约10s长的片段（由于关键帧问题片段长度无法固定），依次命名为0.ts、1.ts、2.ts……

分片命令为：

```
ffmpeg -i {in_file} -codec copy -map 0 -f -segment -segment_time 10s %d.ts
```

其中`{in_file}`为输入文件名。

储存位置在`.\media`文件夹中，每一个多媒体文件对应于一个子文件夹，在这个文件夹中储存其流媒体片段文件。

通过读取`.\media`文件夹，可以获取远程文件列表，并通过读取相应子文件夹，实现流媒体片段的依次传输。

服务器-客户端简易控制协议

客户端发送	服务器回应	含义
`list\n`	`{length}\n {file_names}\n`	客户端询问远程文件列表，服务器回应长度`{length}`并一行一个文件名地回复
`file {file_name} {ip} {port}\n`	`ok {fragments}\n`	客户端要求服务器连接`{ip}:{port}`并发送`{file_name}`对应的流媒体文件的片段，服务器回应片段数量`{fragments}`。
`continue\n`	`ok\n` | `no\n`	客户端要求服务器继续发送流媒体片段，服务器根据是否可行进行回复。
`pause\n`	`ok\n` | `no\n`	客户端要求服务器暂停发送流媒体片段，服务器根据是否可行进行回复。
`end\n`	`ok\n` | `no\n`	客户端要求服务器彻底停止此次传输，服务器根据是否可行进行回复。

服务器-客户端简易传输协议

流媒体客户端在发送`file`控制指令之后，同时创建一个文件接收服务器，并将其IP地址与端口号一并发送给流媒体服务器，要求流媒体服务器以文件接收客户端的身份参与文件传输。

流媒体服务器传输格式为：32bit的文件长度+32bit的片段编号+片段文件。其中文件长度仅指片段文件的长度。格式均为二进制。

流媒体客户端接收到一个片段文件后，以ASCII的形式回应对应的片段编号。

客户端缓冲区与流量控制

客户端缓冲区是一个线性表，表中每一项储存：片段编号，到此片段开头为止的总Bytes数，以及指向片段文件的指针（或引用，取决于Python的具体底层实现）。

每当收到一个片段文件，为它动态分配一片内存，并将其指针加入缓冲区中。

客户端的缓冲区有2个阈值：max_size和continue_size。当片段文件数超过max_size，则认为客户端内存可能不足，需要向服务器发送`pause\n`控制指令（发送后并不会立刻停止接受，因此max_size要给内存留有一定空间）。当片段文件少于continue_size且之前已经要求服务器暂停发送，则发送`continue\n`控制指令让服务器继续发送。

实现时令continue_size=0.8×max_size。

播放器的实现

窗口及用户界面使用wxpython实现，流媒体片段文件的播放使用libvlc的python接口实现。

程序结构

实现程序分为3个部分：流媒体服务器、流媒体客户端、播放器。

其中流媒体客户端与播放器在同一个进程中运行。进程内部用asyncio进行异步调度，使用消息队列和缓冲区作为二者之间的交流通道。其中消息队列用于将播放器的用户操作传达给客户端，缓冲区用于将客户端收到的文件传递给播放器。

实验步骤

1.分工

流媒体服务器：郑凌骁 流媒体客户端：岳鹏云 播放器：罗梓璋

2.开发步骤

在划分好程序结构之后，制定控制和传输协议，约定客户端和播放器之间的接口，经过多次调整后完成了程序实现。

3.测试

在局域网连接下，使用一台计算机作为服务器，另一台作为客户端。 客户端通过下拉目录请求远程文件列表，选择播放文件。 在播放过程中，通过控制台观察缓冲区是否正常运作，缓冲区满后服务器暂停传输，缓冲区有足够空间后服务器继续传输。 在播放过程中停止播放并打开另一远程文件，服务器停止上一次传输并开始新的传输。

开发心得

- 在开发之前需要先约定好协议与接口，以保证最后各个分工可以合并在一起。
- 在开发之前要预先研究好实现原理，原本打算以0.1s为单位对多媒体文件进行分段，开发之后才发现多媒体文件并不能随意分段，只好改成10s为单位的简化版本。
- 要注意多线程之间的同步问题，由于libvlc是C语言实现的，在python和C之间转换时偶尔会出现不同步的问题，需要在实现中加以解决。