Michael Chang (machang), Mitchell Lee (mklee1)

18-441 Project 1

Server Design

For our server, we created it using one singular object. Within the object, there were several methods and variables that we used to implement it. Below is our class structure for the VodServer class:

```
public class VodServer {

    public static void main(String[])

    static String getContentType(String)

    public static void serve(Socket)

}
```

The main method creates a server socket and waits for a connection. If a connection is able to be forged, then it attempts to create a thread (by checking threadPool) and serves the client.

The getContentType method is a helper function that parses through the request in order to determine the content type (txt, css, htm/html, gif, jpg/jpeg, png, js, mp4/webm/ogg/ogv). This is then fed back to the serve method.

The serve method is where the core of our implementation was. Within the serve function, we open a connection and get the input (request). Within the request, we look in particular for "Range" and "keep-alive" in order to satisfy range requests and to maintain the connection to the client instead of closing it.

Concurrency

We chose to handle concurrent requests using java.util. We also intentionally limited the number of active threats using an executor service. The services creates a thread pool (threadPool) so that too we will not slow down performance when many threads are created. The executor service reuses a fixed number (in this case, 12) threads.

Libraries

We mainly relied on java.io to communicate between the server and the client. Other libraries were built into Java. For example, java.net includes several Socket and ServerSocket, and some java.util functions for general uses like getting the time and date or implementing concurrency with the ExecutorService.