**18441/18741: Design report for HTTP server project**

**Name: Xinyu Bao Andrew ID: xinyub**

1. **Project structure**

The project structure is simple. The main thread is basically responsible for creating, binding, listening from a socket, and repeatedly accepts new TCP connection through the given socket. After accepting and a new connection fd is generated, the main thread will create a child handling thread and pass the new fd through as a parameter.

Each child thread first receives and parses a new HTTP request, then try to find and open the file specified in the request. If failed, a 404 error response is sent. After successfully opening the file, the server will decide the response type based on the file size and type. If it is not a file of very large size (larger than 5 MB) or it is not a video file, then the whole file will be sent along with a 200 OK response. Otherwise only the first 5MB chunk will be responded with a 206 Partial Content response. If the request contains a “range” header in the first place, my server will set the fd to the specified file position and then send the next chunk of file or all of the remaining file based on the size left to send. After a child thread exits, both the file fd and the connection fd will be properly closed.

1. **Multiple clients handling**

Basically I just used multiple threads to handle multiple clients’ connection. The main thread is just responsible for accepting new connections, which would make sure that each connection could be established quickly. The server sleeps for a very short amount of time after creating a new connection fd because I find it often creating same connection fd without sleeping.

My server is able to handle 10 concurrent connections, however when it comes to around 2000 connections the server would always time out with no response for the latter part of requests. One assumption for this is that as the number of connections grows, time split for each thread becomes pretty small so that it would take a much longer time to respond timely. This is a potential improvement in the future.

1. **How to run the server**

My server can be run just as described in the handout. First invoke “make” and generate a *vodserver* binary file. Then run the file with an argument specifying the port number and then the server is up and run.