

Project 1

Parshva Jhaveri, Mohan Kukreja, Cameron McCarty

January 21, 2023

0.1 Multiple Clients

Our server uses a multithreaded response design. The program starts by initializing and entering in a main loop indefinitely. In this loop when new client connects a socket is created for it and passed on to a thread. Our server class extends thread which calls the run() method .

0.2 Server Design

Each thread only deals with one client request. The thread starts off by initializing and building out its response buffer. Then the thread enters into a control structure for requests for: GET, GET a file, GET a partial file, or an unsupported request. Once the server knows what the request is for it begins to send its response with sendResponse(). Each of these has different behavior.

- GET: GET requests without a file name are responded to with a 200 OK and a HTML string welcome text with and details about the connection from the response buffer.
- GET file: GET requests for a file are responded to with a 200 OK and the associated file.
- GET partial file: GET requests for a file with a range header are responded with 206 Partial Request and the relevant bytes from the file.
- Unsupported requests: Requests for anything else are responded to with a 404 File Not Found and a string "The Requested resource not found".

The response is started out by filling out the required headers: statusLine, serverdetails, contentTypeLine, contentLengthLine, lastModifiedTime, acceptRange, contentRange, range, date, and/or connection. Many of these headers are measuring the response buffer or file.

Then all response headers are sent followed by the data. The data sent is a string for responses without a string like plain GET and 404. The method sendFile() is used for full files and cutFile() for partial files. Or sendFile() method simply passes bytes from the file to the output stream until there are none to send. cutFile() interprets the ranges from the range request header and randomly accesses those bytes from the file before sending them.

0.3 Libraries Used

No external libraries were used only java packages.

- java.io
- java.net
- java.nio.file.Files
- java.text.SimpleDateFormat
- java.time.ZonedDateTime
- java.util.

0.4 Extra Capabilities

Our design focused on being extremely scaleable. Our implementation of multithreading should allow for many simultaneous requests at once without using up resources at idle.

Another extra capability is requesting multiple byte ranges. Our server can respond to any set of ranges, including ranges that overlap and will send the data in each range in the order the ranges where in. For example when requesting from a file with "0123456789", a request of "Range: bytes=8-,0-1,0-4,-1" would respond with "8901012349".

0.5 Extra Instructions

v Extra instructions on how to execute the code, if needed (WARNING: if we cannot compile and run your server on our cluster machine with the above execution method, your logistic points will be deducted. However, please do tell us any runtime parameter your server may have which may help in tuning)