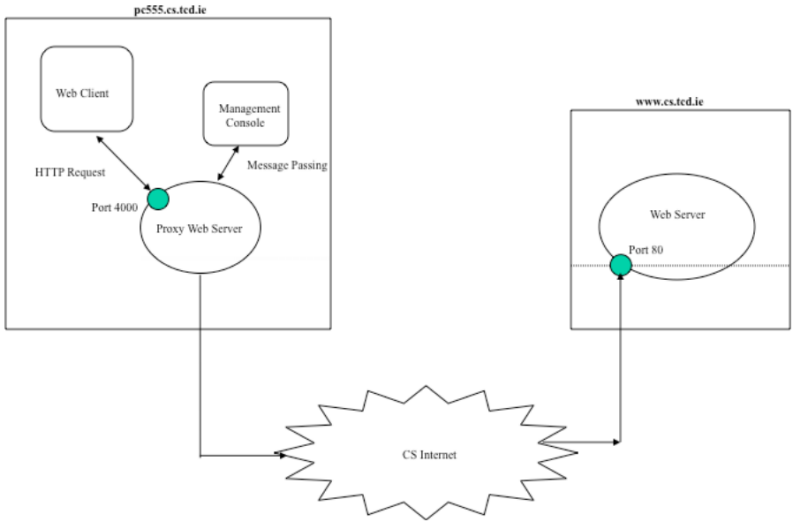
Assignment Report

This assignment is to implement a Web Proxy Server. A Web proxy is a local server, which fetches items from the Web on behalf of a Web client instead of the client fetching them directly. This allows for caching of pages and access control. In my project, I have Webclient, Webserver and Web Proxy Server java files. The basic instruction is the same as the below figure.



The basic process is firstly we run Webserver and then we run Web proxy Server, when we get ready all those part, we finally run our Webclient file, and then Webclient will send an request to Web proxy Server, in my Web proxy Server part, when the Web proxy Server check whether it receives the Webclient request or not. After Web proxy Server check process, it will extract the information and then send the information to Webserver. My Server port number is 8080, Server will work when connected via a web browser, http://localhost:8080/100 would be a valid URL if the Server runs in the same host. Server returns an HTML document according to the requested URI. Server will send the HTML back to Web proxy Server, after Web proxy Server receive the HTML document. After that, Web proxy Server will send response back to Webclient, and then Webclient will receive the result and then show HTML document information. That's the process of my Web proxy Server.

There are three main java part in my project. The first one is Webclient. As the figure below shown, That's my Webclient part, which has the same port number 8080 with Web proxy Server. The first part of Webclient is OutputStream output to write the get http url information to Web Proxy Server in order to let Web Proxy Server send to Server to find HTML document back to it, which means then send an HTTP request to the server. we're sending a HTTP GET request to the root URL (/) of the server, along with some headers indicating the hostname. We then read the Web Proxy Server response using a BufferedReader, which will read the response from the Web Proxy Server

line by line and prints it to the console.

```
public class Webclient {

    public static void main(String[] args) throws IOException {
        Socket cSocket = new Socket("localhost", 8080);
        OutputStream output = cSocket.getOutputStream();
        InputStream input = cSocket.getInputStream();
        output.write("GET /100 HTTP/1.1\r\n".getBytes());
        output.write("Host: localhost\r\n".getBytes());
        output.write("\r\n".getBytes());
        output.flush();

        BufferedReader read = new BufferedReader(new InputStreamReader(
        String line = read.readLine();
        while (line != null) {
            System.out.println(line);
            line = read.readLine();
        }
        cSocket.close();
    }
}
```

The second java part is Webserver. As the figure shown below, server has the different port number 8000 from web proxy server and webclient. Firstly create a ServerSocket object to listen for incoming connections on a specific port and start an infinite loop to continuously accept incoming connections. What's more for each incoming connection, create a new thread to handle the HTTP request and response.

```
public final class Webserver {

  public static void main(String[] args) throws IOException {
    int port = 8000;
    ServerSocket server = new ServerSocket(port);
    System.out.println("Opening Web Server and readyto connections...")
    Socket client = null;
    while (true) {
      client = server.accept();
      System.out.println("Connect..." + client.toString());
      Handler request = new Handler(client);
      Thread thread = new Thread(request);
      thread.start();
    }
  }
}
```

And then we have the sever handle part implements the Runnable interface to handle the connection in a separate thread. We use BufferReader to read the incoming request and parse the request to determine the requested resource. Using if statement to check whether the request is coming from client Get localhost request or not. After checking construct an appropriate response to the request based on the requested resource, and send the response using a PrintWriter (out). HTTP/1.0 200 means all good, HTTP/1.0 400 means bad request. Also using printHTML to print the HTML document information.

```
class Handler implements Runnable {
    private Socket clientSocket;

      public Handler(Socket c) {
        this.clientSocket = c;
      }

      public void run() {
        boolean check = true;
        try {
        InputStream input = clientSocket.getInputStream();
        OutputStream output = clientSocket.getOutputStream();
        BufferedReader br = new BufferedReader(new InputStreamReader(in
        String line = br.readLine();
        System.out.println(line);
        StringTokenizer tokens = new StringTokenizer(line);

          if (!tokens.nextToken().toUpperCase().equals("GET")) {
            check = false;
          }
          String string = null;
          if (check) {
              String fileName = tokens.nextToken();
                fileName = fileName.substring(1);
                string = fileName;
              }
              PrintWriter out = new PrintWriter(output);
              if (check) {
            PrintWriter out = new PrintWriter(output);
            if (check) {
              out.println("HTTP/1.0 200 OK");
              printMessage("Sending " + 100+string.length() + " bytes"
              printHTML(out, string);
              printMessage(100+string.length() + " bytes sent");
            } else {
              printMessage("Client Bad Request response message sent."
              out.println("HTTP/1.0 400 Bad Request");
              out.println();
              out.print("Bad Request.");
            }

            out.close();
            output.close();
            br.close();
            input.close();

       } catch (IOException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
       } catch (Exception e) {
           e.printStackTrace();
       }
     }

  private void printHTML(PrintWriter out, String numOfBytes) {
    out.println("Content-Type: text/html");
    out.println("Content-Length: " + 100+numOfBytes.length());
    out.println();
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<TITLE>This is " + 100+numOfBytes.length() + " byte
    out.println("</HEAD>");
    out.print("<BODY>");
    out.print(numOfBytes);
    out.println("</BODY>");
    out.print("</HTML>");
  }
}
```

The last part of java file is Web Proxy Server, which is the most important part in this project. In my Web Proxy Server part, I have mainhandle, response, request three parts in Web Proxy Server. The proxy server listens for incoming connections on a specified port, using a ServerSocket. When a

client connects, the server creates a new Handler thread to handle the client request and response. The Handler reads the incoming HTTP request from the client's socket input stream, using a BufferedReader. And then it extracts the URL from the request, and determines the destination host by parsing the URL. In the response part, it has Socket("localhost", 8000) to set the Server part to send the information. The Handler reads the response from the destination server's socket input stream, using another BufferedReader. The Handler writes the response back to the client's socket output stream, using send function. For the cache function, in the web proxy server, store the file called cache, when we use the cache function, the web proxy server will check whether the cache file has the html file or not. If we have the cache file in the project, then we don't need to go to the server part, when the web proxy server receive the client's request, it will directly send back the cache file. This will be more efficient and effective, which means reduce lots of running time as my many times computational calculation.

In conclusion, that's my basic process of web proxy server and the explanation of code.

```
public class WebProxy {
    public static void main(String[] args) throws Exception {
        ServerSocket server = new ServerSocket(8080);
        System.out.println("Proxy Server start 8080");

        while (true) {
            Socket socket = server.accept();
            new Thread (new MHandler(socket)).start();
        }
    }
}
```

```
class MHandler implements Runnable {
    Socket s;

    MHandler(Socket s) {
        this.s = s;
    }

    public void run() {
        try {

            DataOutputStream output = new DataOutputStream(s.getOu
            BufferedReader read = new BufferedReader(new InputStre

            Request request = new Request(read);

            if (!"GET".equals(request.method)) {
                s.close();
                return;
            }
                System.out.println("//" + request);

                Response r = request.getResponse();

            if (r != null)
                r.send(output);
            read.close();
            output.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
            try {
                s.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

class Response implements Runnable {
    ArrayList<String> line1 = new ArrayList<>();

    Response(BufferedReader read) throws Exception {
        String line;
        while ((line = read.readLine()) != null) {
            line1.add(line);
        }
    }
    void send(DataOutputStream out) throws IOException {
        for (String line : line1) {
            System.out.println(line);
            out.writeBytes(line + "\r\n");
            out.flush();
        }
        out.writeBytes("\r\n");
        out.flush();
    }
    public void run() {
    }
}
```

```
class Request implements Runnable {
    int port;
    String method = "GET";
    String path = "/";
    String version = "HTTP/1.0";
    final String cacheFolderPATH = "cache";
    String cacheFileName;
    HashMap<String, String> headers = new HashMap<>();
    Socket socket;
    DataOutputStream outStream;
    BufferedReader read;
    PrintWriter out;

    Request(BufferedReader reader) throws Exception {
        String rLine = reader.readLine();
        if (rLine == null) {
            throw new Exception("Invalid");
        }

        String[] split = rLine.split(" ");
        try {
            path = split[1];
            URL url = new URL(path);
            port = url.getPort();
            cacheFileName = path.substring(1) + ".cache";
            headers.put("Host", url.getHost());
        } catch (Exception e) {
        }
    }
```

```
    private boolean checkCache() {
        File folder = new File(cacheFolderPATH);
        File[] listOfFiles = folder.listFiles();

        for (int i = 0; i < listOfFiles.length; i++) {
            if (listOfFiles[i].isFile() && listOfFiles[i].getName().eq
                return true;
            }

        }
        return false;
    }

    Response getResponse() throws Exception {
        boolean isCached = checkCache();
        System.out.println(isCached);
        if (!isCached) {
            try {
                socket = new Socket("localhost", 8000);
                outStream = new DataOutputStream(socket.getOutputStrea
                read = new BufferedReader(new InputStreamReader(socket
                out = new PrintWriter(outStream);
            } catch (Exception e) {

            }
            write(method + " " + path + " " + version);
            for (String key : headers.keySet()) {
                write(key + ": " + headers.get(key));
            }
            write("");
            Response response = new Response(read);
            File fileToCache = new File(cacheFolderPATH + "\\" + cache
            FileOutputStream fileOutStream = new FileOutputStream(file
```

```
            for (String key : headers.keySet()) {
                write(key + ": " + headers.get(key));
            }
            write("");
            Response response = new Response(read);
            File fileToCache = new File(cacheFolderPATH + "\\" + cache
            FileOutputStream fileOutStream = new FileOutputStream(file
            DataOutputStream outData = new DataOutputStream(fileOutStr
            response.send(outData);
            return response;
        } else {
            File cacheFile = new File(cacheFolderPATH + "\\" + cacheFi
            InputStream fileStream = new FileInputStream(cacheFile);
            BufferedReader cacheFileReader = new BufferedReader(new In
            Response cacheResponse = new Response(cacheFileReader);
            return cacheResponse;
        }
    }
    private void write(String line) throws Exception {
        System.out.println(line);
        out.println(line + "\r\n");
        out.flush();
    }
    public void run() {
    }
}
```