

Design and Implementation of a Custom Network Proxy Server

Shreyansh Kuntal
shreyansh.k@ph.iitr.ac.in

22324018

Contents

| | | |
|----------|--|----------|
| 1 | System Architecture | 2 |
| 1.1 | Main | 2 |
| 1.2 | Network | 2 |
| 1.3 | Log | 3 |
| 1.4 | ParseRequest | 3 |
| 1.5 | RequestHttp | 3 |
| 1.6 | RequestHttps | 4 |
| 1.7 | Filter | 4 |
| 1.8 | Config files | 4 |
| 2 | Concurrency Model and Rationale | 4 |
| 2.1 | Rationale for Model Choice | 5 |
| 3 | Error Handling | 5 |
| 4 | Limitations | 6 |
| 5 | Demonstrations | 6 |

1 System Architecture

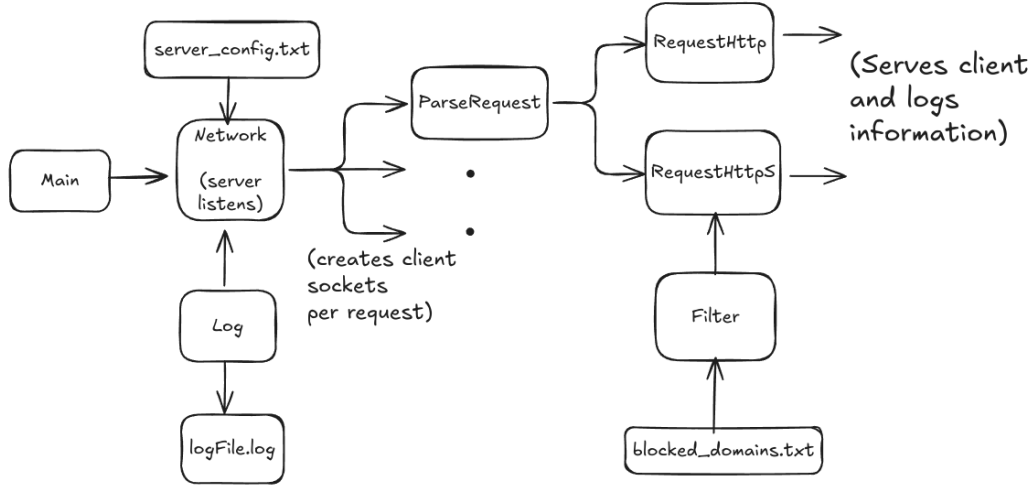


Figure 1: High-Level Architecture of the Proxy Server

Description of the various components in Figure 1 is given below:

1.1 Main

The **Main** component serves as the entry point of the application. It creates an instance of **Network** class and calls its **runProxyServer** method.

1.2 Network

The **Network** component is responsible for setting up the server socket on the port specified in **server_config.txt** file and listening for incoming client connections.

Further it accepts headers from client and uses **ParseRequest** to figure out the protocol(HTTP or HTTPS) and other stuff such as target host, port, etc. Based on the protocol it calls **RequestHttp** or **RequestHttps** component to carry out further transfer of data.

1.3 Log

Uses the standard `java.util.logging` API to log information of every request to a text file.

By default it rotates the log files in a count of 5 and a maximum size of 1 MB.



```
Jan 08, 2026 2:52:18 PM Log put
INFO: Client IP:Port /127.0.0.1:45546
Requested Host:Port www.codeforces.com:443
Requested Target www.codeforces.com:443
Action Allowed
Response Status HTTP/1.1 200 Connection Established
```

Figure 2: Example of information logged

1.4 ParseRequest

The `ParseRequest` component parses client's first request's headers and extracts information which can be accessed through its methods:

- `getHost` - such as `www.codeforces.com`
- `getMethod` - GET, POST, CONNECT
- `getRequestTarget` - requested webpage URL
- `getHostPort` - by default 80 for HTTP, 443 for HTTPS

1.5 RequestHttp

The `RequestHttp` component handles standard HTTP requests. It uses `Filter` component to check for blocked domains. If found blocked it responds with code 403. If valid it forwards body of the exact number of bytes specified by `Content-Length` header received from server. If `Content-Length` header is not found(in case of chunked encoding) it forwards 0 bytes(apart from headers). Finally it logs the information.

1.6 RequestHttps

The `RequestHttps` component handles HTTPS requests using the `CONNECT` method. It also uses `Filter` component. After responding with 200 OK, it acts as a tunnel between client and server and forward bytes bidirectionally using threading without interpreting them. Finally it logs the information.

1.7 Filter

The `Filter` component uses `isGood` method to determine whether a request should be blocked based on domains specified in the file `blocked_domains.txt`.

1.8 Config files

Config files should strictly follow the format specified in Figure 3 and Figure 4.

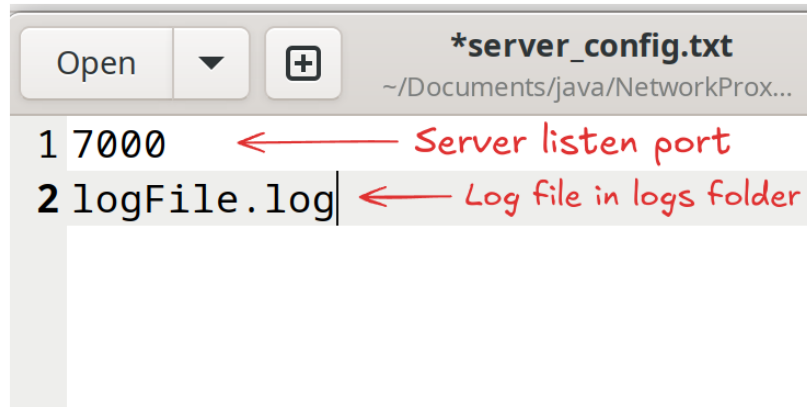


Figure 3: `server_config.txt`

2 Concurrency Model and Rationale

The proxy server is designed to handle multiple client connections concurrently using a **thread-per-connection** concurrency model. In the thread-per-connection model, the server listens for incoming client connections on a

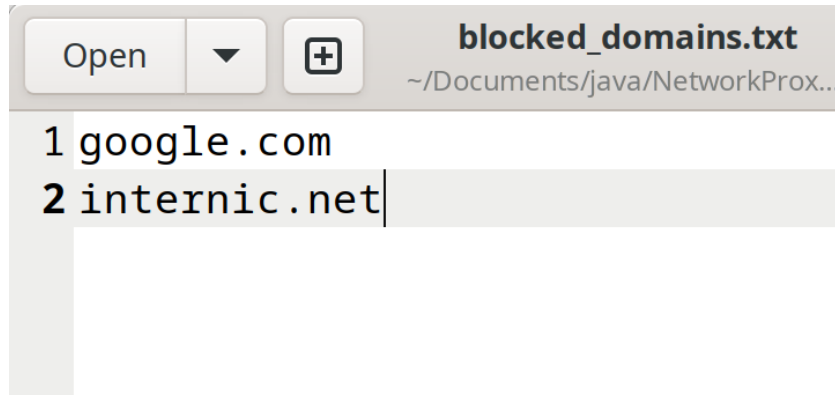


Figure 4: `blocked_domains.txt`. Each domain on new line.

server socket. For each accepted client connection, a new thread is created to handle the request independently.

2.1 Rationale for Model Choice

- **Simplicity:** The model is straightforward to implement and understand.
- **Adequate Performance for Moderate Load:** For a limited number of concurrent clients, this approach provides acceptable performance.

3 Error Handling

The proxy server handles basic exceptions to ensure stability and graceful failure during runtime such as:

- **Invalid Requests:** Requests that do not conform to the expected HTTP format or unresolved URL are rejected without forwarding. Code 403 is sent back to client.
- **I/O Exceptions:** Input and output operations are enclosed in exception handling blocks to prevent the server from crashing.
- **Graceful Termination:** Sockets and streams are closed automatically by JVM when SIGINT/SIGTERM are raised.

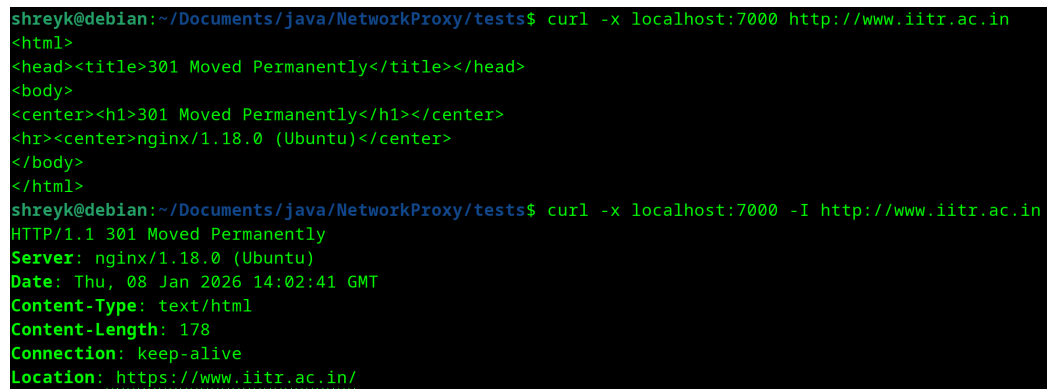
These mechanisms help maintain server robustness and prevent a single client error from affecting other active connections.

4 Limitations

- **Scalability:** The thread-per-connection model does not scale efficiently for a very large number of concurrent clients.
- **No Caching:** The proxy does not implement response caching, which could improve performance for repeated requests.

5 Demonstrations

Logs of all the tests are saved in logs/logFile.log.



```
shreyk@debian:~/Documents/java/NetworkProxy/tests$ curl -x localhost:7000 http://www.iitr.ac.in
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
shreyk@debian:~/Documents/java/NetworkProxy/tests$ curl -x localhost:7000 -I http://www.iitr.ac.in
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 08 Jan 2026 14:02:41 GMT
Content-Type: text/html
Content-Length: 178
Connection: keep-alive
Location: https://www.iitr.ac.in/
```

Figure 5: Simple tests.

```

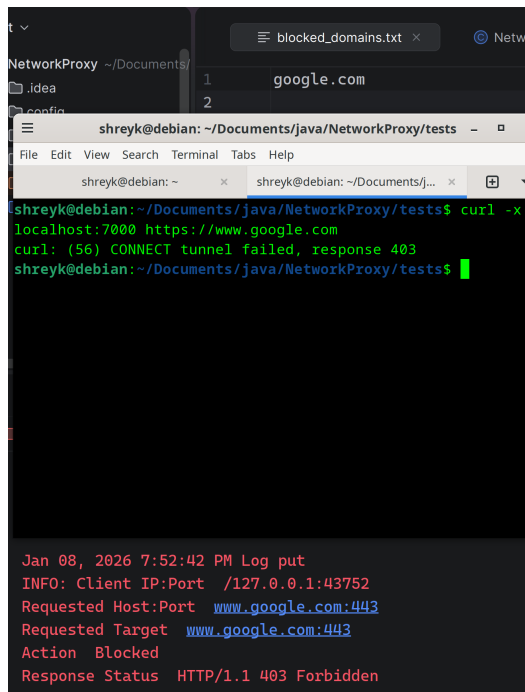
Jan 08, 2026 7:44:52 PM Log put
INFO: Client IP:Port /127.0.0.1:47358
Requested Host:Port www.w3schools.com:443
Requested Target www.w3schools.com:443
Action Allowed
Response Status HTTP/1.1 200 Connection Established

Jan 08, 2026 7:44:52 PM Log put
INFO: Client IP:Port /127.0.0.1:47348
Requested Host:Port www.w3schools.com:443
Requested Target www.w3schools.com:443
Action Allowed
Response Status HTTP/1.1 200 Connection Established

Jan 08, 2026 7:44:52 PM Log put
INFO: Client IP:Port /127.0.0.1:47406
Requested Host:Port www.w3schools.com:443
Requested Target www.w3schools.com:443
Action Allowed

```

Figure 6: Concurrent tests. `script` in tests folder make 20 concurrent request. 3 of which shown in figure with same timing. Output saved in tests/out.



```

shreyk@debian: ~/Documents/java/NetworkProxy/tests
File Edit View Search Terminal Tabs Help

shreyk@debian: ~ x shreyk@debian: ~/Documents/j... x

shreyk@debian:~/Documents/java/NetworkProxy/tests$ curl -x
localhost:7000 https://www.google.com
curl: (56) CONNECT tunnel failed, response 403
shreyk@debian:~/Documents/java/NetworkProxy/tests$

Jan 08, 2026 7:52:42 PM Log put
INFO: Client IP:Port /127.0.0.1:43752
Requested Host:Port www.google.com:443
Requested Target www.google.com:443
Action Blocked
Response Status HTTP/1.1 403 Forbidden

```

Figure 7: Blocking test. 403 response sent by proxy to client when it tries to access google.com which is in blocklist.

```

shreyk@debian:~/Documents/java/NetworkProxy/tests$ curl -x localhost:7000 https://www.google.com
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-IN"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleleg_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="K6sZFVAgx10TJczfKZt_w">(function(){var _g={kEI:'fcBfadzkBbWx5OUP1YW3sA8',kEXPI:'0,1304203,2935842,14112,64701,6397,354504,226399,2,74749,5231286,36811932,25306698,74361,57130,8041,30633,7033,2105,11153,1116,63048,23255,3292,34513,28334,48317,30997,7714,33385,3050,2,23178,2864,22727,10370,35967,21840,59,4372,6292,5321,1116,9742,2646,103,4,1,2143,4041,10618,7336,14505,1383,2,1,1515,3355,2,193,5815,2,4205,3,3212,7727,775,4384,2,874,2231,5089,19,3008,25,770,4081,4,5386,13783,12119,219,1031,1035,4302,311,534,846,1200,2,12,16,415,1,1129,3,2842,9,27,8,3055,1010,1747,4,15193,4,2170,73,240,297,5,1113,389,4,299,2306,42,5,5568,298,5201,4052,589,686,7,1784,7,997,52,2,290,1622,420,95,6034,1250,4,82,2,2570,2,5724,9,602,96,1553,4,795,1087,4,1,3,2,1,681,811,2525,934,346,251,1229,5,1809,1,2,592,235,3,314,573,472,2,2141,3,2767,70,10,215,441,1805,191,4,28,1900,4,1634,544,4,2054,547,1297,5,97,187,863,1242,5,179,80,5,197,1921,445,19,295,1126,25,94,4,235,137,13,1040,207,392,4,136,200,137,231,1995,5,40,535,6,147,5,143,1733,4,121,24,975,2186,141,398,4,586,4,1460,112,152,107,5,4,3070,3,2,1014,247,217,4,486,4,670,626,482,4,32,4,847,1020,134,12,690,4,40,4,1966,6,558,1,5,16,119,4,9,191,1,256,320,1776,1865,442,701,88,4,12,2,3,2,1,461,28,324,236,56,3,2,2,2,79,68,4,1940,4,61,606,387,4,32,894,676,3,2,1,328,1,1091,1,914,53,98,937,276,2,560,661,4,782,4,40,1488,1929,44,389,1155,4,746,4,33,60,1373,67,764,124,8,1048,98,2,1197,2,3,2,2,2,39,11,152,227,210,611,1940,181,81,48,3,1,399,64,411,3,2,2,2,51,3,2,2,2,213,5596,454,833,1789,1,2,190,1,2100,9604,5,2253,739,4,2960,3,2022,1215,5254,2,1558,3,11875,1303,1695,1195,3,746,6,1149,3,1371,4207,2,440,363,74,201,282,3,96,642,1199,6491328,2532,2,696,263,160,2166720,386546,1288385,11901908,2907,4857,1,194310',kBL:'ljJR',kOPI:89978449});(function(){var a;((a=window.google)==null?0:a.stvsc)?google.kEI=_g.kEI:window.google=_g;}).call(this);})();(function(){google.sn='webhp';google.kHL='en-IN';google.rdn=f

```

Figure 8: CONNECT test. curl request to https://www.google.com responds with webpage content indicating fine working of proxy.