

CS3205: Computer Networks

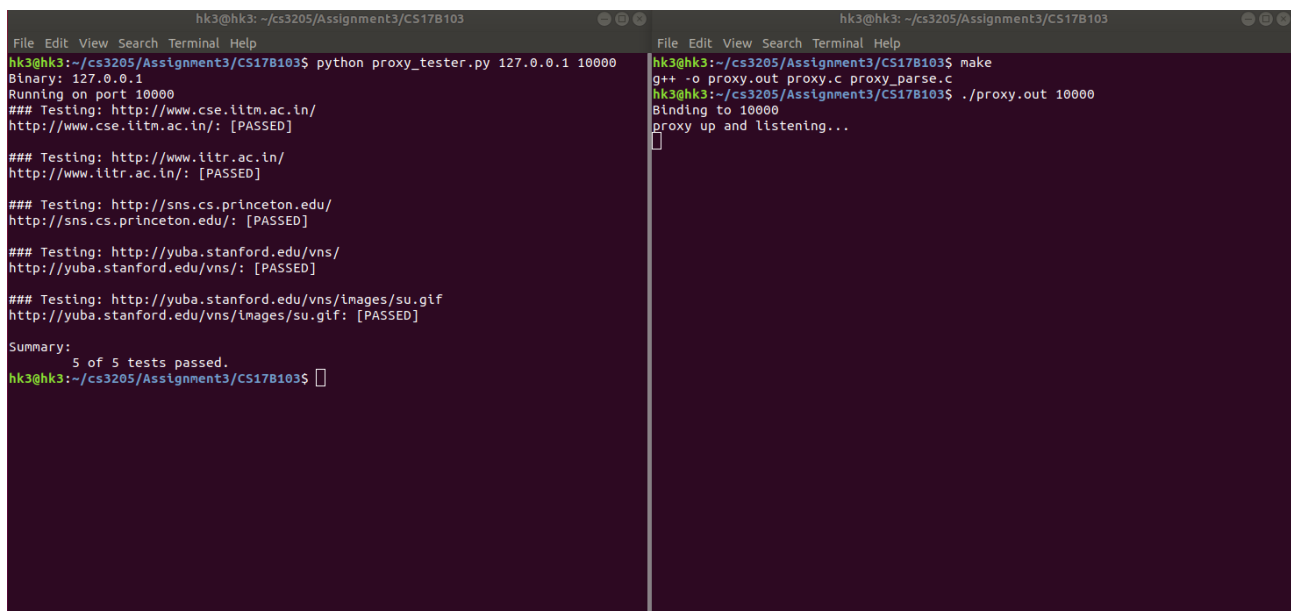
Assignment-3

HTTP Proxy Server in C

Harshit Kedia
CS17B103

Source Files : 1)proxy.c
2)proxy_parse.c
3)proxy_parse.h
4)proxy_tester.py
5)proxy_tester_conc.py
6)README
7)Makefile

Proof of working of proxy server :



The image contains two side-by-side terminal screenshots. The left terminal shows the execution of a Python script, proxy_tester.py, which tests the proxy server against five different URLs. All tests passed, resulting in a score of 5 out of 5. The right terminal shows the compilation of the proxy server using the make command, followed by running the proxy.out binary, which successfully binds to port 10000 and starts listening for connections.

```
hk3@hk3: ~/cs3205/Assignment3/CS17B103
File Edit View Search Terminal Help
hk3@hk3:~/cs3205/Assignment3/CS17B103$ python proxy_tester.py 127.0.0.1 10000
Binary: 127.0.0.1
Running on port 10000
### Testing: http://www.cse.iitm.ac.in/
http://www.cse.iitm.ac.in/: [PASSED]

### Testing: http://www.iitr.ac.in/
http://www.iitr.ac.in/: [PASSED]

### Testing: http://sns.cs.princeton.edu/
http://sns.cs.princeton.edu/: [PASSED]

### Testing: http://yuba.stanford.edu/vns/
http://yuba.stanford.edu/vns/: [PASSED]

### Testing: http://yuba.stanford.edu/vns/images/su.gif
http://yuba.stanford.edu/vns/images/su.gif: [PASSED]

Summary:
5 of 5 tests passed.
hk3@hk3:~/cs3205/Assignment3/CS17B103$

hk3@hk3:~/cs3205/Assignment3/CS17B103$ make
g++ -o proxy.out proxy.c proxy_parse.c
hk3@hk3:~/cs3205/Assignment3/CS17B103$ ./proxy.out 10000
Binding to 10000
proxy up and listening...
█
```

The screenshot above shows that our proxy is listening and for TCP connections.
The script proxy_tester.py scores 5 out of 5.

The screenshot below is for the python script proxy_tester_conc.py, 11 out of 11 cases passed.

I had to edit the websites in script, to indian websites, so that the request is processed fast, also some of the sites had moved on to some other IP/location, hence were failing the test cases.

The websites chosen are:

- 1) <http://www.cse.iitm.ac.in/>
- 2) <http://www.iitr.ac.in/>
- 3) <http://netaccess.iitm.ac.in>

No preferences, just randomly chosen.

Experiments where multiple terminals were using telnet concurrently was also done, though I think, it is not exactly the best way to check concurrency, since human reactions take time. So, only script output attached.

```
hk3@hk3:~/cs3205/Assignment3/CS17B103$ python proxy_tester_conc.py 127.0.0.1 10000
Binary: 127.0.0.1
Running on port 10000
### Testing: http://www.cse.iitm.ac.in/
http://www.cse.iitm.ac.in/: [PASSED]

### Testing: http://www.iitr.ac.in/
http://www.iitr.ac.in/: [PASSED]

### Testing 2 concurrent connects to http://netaccess.iitm.ac.in/
Connect to http://netaccess.iitm.ac.in/, 2 concurrently: [PASSED]

### Testing 10 concurrent connects to http://netaccess.iitm.ac.in/
Connect to http://netaccess.iitm.ac.in/, 10 concurrently: [PASSED]

### Testing 2 concurrent fetches to http://netaccess.iitm.ac.in/
Fetch to http://netaccess.iitm.ac.in/, 2 concurrently: [PASSED]

### Testing 10 concurrent fetches to http://netaccess.iitm.ac.in/
Fetch to http://netaccess.iitm.ac.in/, 10 concurrently: [PASSED]

### Testing 2 concurrent split fetches
Fetch to http://netaccess.iitm.ac.in/, 2 concurrently: [PASSED]

### Testing 10 concurrent split fetches
Fetch to http://netaccess.iitm.ac.in/, 10 concurrently: [PASSED]

### Testing apache benchmark on args [-n 20 -c 1]
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking netaccess.iitm.ac.in [through 127.0.0.1:10000] (be patient).....done


Server Software:
Server Hostname:      netaccess.iitm.ac.in
Server Port:          80

Document Path:        /
Document Length:       0 bytes

Concurrency Level:     1
Time taken for tests:   0.014 seconds
Complete requests:     20
Failed requests:        0
Total transferred:     0 bytes
HTML transferred:      0 bytes
Requests per second:   1382.84 [#/sec] (mean)
Time per request:      0.723 [ms] (mean)
Time per request:      0.723 [ms] (mean, across all concurrent requests)
Transfer rate:          0.00 [Kbytes/sec] received
```

```
hk3@hk3: ~/cs3205/Assignment3/CS17B103
File Edit View Search Terminal Help
Benchmarking netaccess.iitm.ac.in [through 127.0.0.1:10000] (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:
Server Hostname:      netaccess.iitm.ac.in
Server Port:          80

Document Path:        /
Document Length:       0 bytes

Concurrency Level:     50
Time taken for tests:   0.062 seconds
Complete requests:     1000
Failed requests:        0
Total transferred:     0 bytes
HTML transferred:      0 bytes
Requests per second:   16178.09 [#/sec] (mean)
Time per request:      3.091 [ms] (mean)
Time per request:      0.062 [ms] (mean, across all concurrent requests)
Transfer rate:         0.00 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median    max
Connect:    0    1   0.2      1    1
Processing: 0    1   0.6      1   16
Waiting:    0    0   0.0      0    0
Total:      1    1   0.7      1   17

Percentage of the requests served within a certain time (ms)
50%    1
66%    1
75%    1
80%    1
90%    2
95%    2
98%    2
99%    3
100%   17 (longest request)
http://netaccess.iitm.ac.in/ with args -n 1000 -c 50: [PASSED]

Summary:
Type multi-process: 11 of 11 tests passed.
hk3@hk3:~/cs3205/Assignment3/CS17B103$
```

Also ran using telnet, output attached below:

```
hk3@hk3: ~/cs3205/Assignment3/CS17B103
File Edit View Search Terminal Help

hk3@hk3:~/cs3205/Assignment3/CS17B103$ telnet localhost 10000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET http://www.cse.iitm.ac.in/ HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 09 Mar 2020 15:23:56 GMT
Server: Apache/2.4.10 (Debian)
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="EN" lang="EN" dir="ltr">
<head profile="http://gmpg.org/xfn/11">
<title>Department of Computer Science & Engineering</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="imagetoolbar" content="no" />
<link rel="stylesheet" href="styles/layout.css" type="text/css" />
<link rel="stylesheet" href="styles/maintab.css" type="text/css" /><!-- <link re
l="stylesheet" href="css/timeline.css" type="text/css" /> -->
<!-- 3 Column Stylesheet Added To The Page And Not To The Layout.css -->
<link rel="stylesheet" href="styles/3_column.css" type="text/css" />
<!-- Featured Slider Elements -->
<script type="text/javascript" src="scripts/jquery-1.4.1.min.js"></script>
<script type="text/javascript" src="scripts/jquery-s3slider.js"></script>
<script type="text/javascript" src="scripts/jquery-s3slider.setup.js"></script>
<!-- End Featured Slider Elements -->
<!-- JQuery drop down menu-->
    <link type="text/css" href="styles/menu.css" rel="stylesheet" />
    <script type="text/javascript" src="scripts/jqry.js"></script>

<!-- JQuery drop down menu--<script type="text/javascript" src="jqry/menu.js"></
script-->

<!-- calendar -->
<link rel="stylesheet" href="./calendar/jquery-ui.css" />
<script src="./calendar/jquery-1.9.1.js"></script>
<script src="./calendar/jquery-ui.js"></script>
<!-- calendar -->

<!-- sliding images -->
```

Inferences:

- 1) Many of the websites of the testing script had gone invalid (being very old), hence changed to known websites.
- 2) When proxy starts, it establishes a socket connection that it listens for incoming connections. The proxy listens on the port specified from the command line and waits for incoming client connections.
- 3) For every client, we fork a new process, which has file descriptors for the TCP stream, whereas parent process keeps waiting for further requests.
- 4) if parsing is successful, we open a connection to hostname mentioned in request, and pass-on the received data to the client, now the connection is closed and the forked child exits.
- 5) The == operator has higher precedence than = operator, not using proper brackets can cause impossible to find bugs in the program.
- 6) Since the data sent by sites is massive, we use a while loop to keep of reading the stream and forwarding the data to client.

Summary:

With this experiment we learn the use of TCP sockets in C, using file descriptors to have I/O in forked process. Also challenging in terms of concurrency and debugging is tough since we cannot use debugger to track forked process.

This Assignment was not the most difficult one, but certainly very informative, practical and well-structured. I did this task individually and found it very interesting.