Despina Patronas
CSE 130

Write Up Document Assignment 3: httpproxy

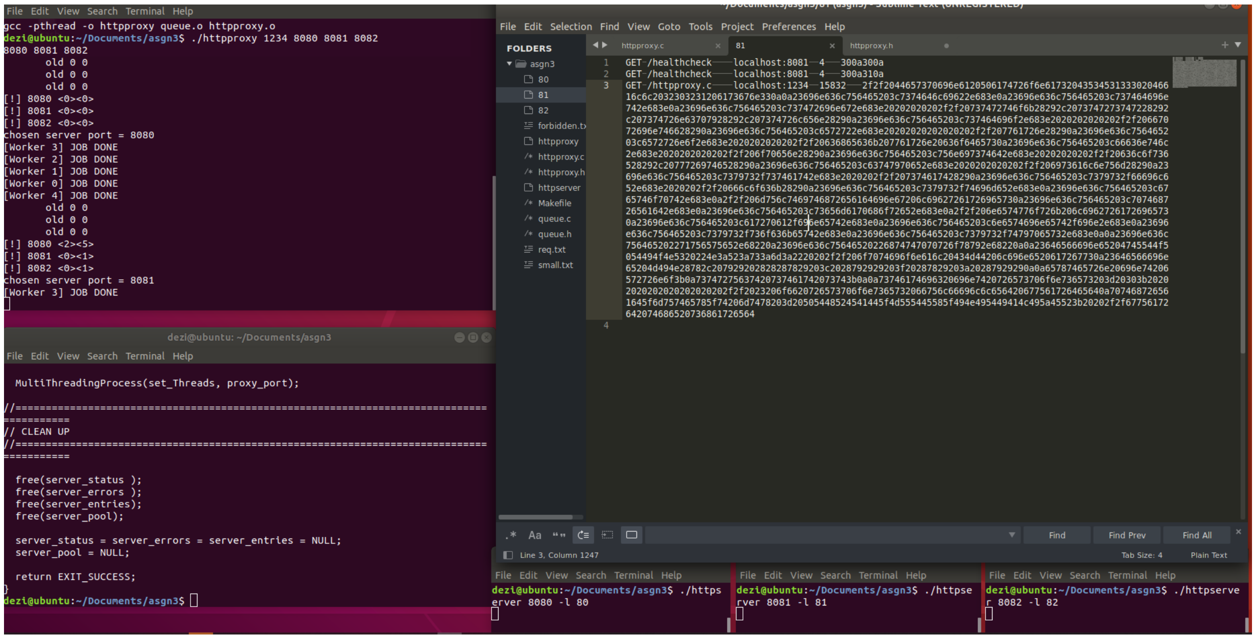
Testing

httpproxy, should be similar to an http protocol server.

i. When Client sends curl requests (HEAD) to *httpproxy*, *httpproxy* should perform the operation and respond back to client properly.

Testing Code for the spec:

Tests were done locally ..



```
gcc -pthread -o httpproxy queue.o httpproxy.o
dezi@ubuntu:~/Documents/asgn3$ ./httpproxy 1234 8080 8081 8082
8080 8081 8082
       old 0 0
       old 0 0
       old 0 0
[!] 8080 <0><0>
[!] 8081 <0><0>
[!] 8082 <0><0>
chosen server port = 8080
[Worker 3] JOB DONE
[Worker 2] JOB DONE
[Worker 1] JOB DONE
[Worker 0] JOB DONE
[Worker 4] JOB DONE
       old 0 0
       old 0 0
       old 0 0
[!] 8080 <2><5>
[!] 8081 <0><1>
[!] 8082 <0><1>
chosen server port = 8081
[Worker 3] JOB DONE
```

```
  MultiThreadingProcess(set_Threads, proxy_port);

//================================================================================
==========
// CLEAN UP
//================================================================================
==========

  free(server_status );
  free(server_errors );
  free(server_entries);
  free(server_pool);

  server_status = server_errors = server_entries = NULL;
  server_pool = NULL;

  return EXIT_SUCCESS;
}
dezi@ubuntu:~/Documents/asgn3$
```

```
GET /healthcheck    localhost:8081  4   300a300a
GET /healthcheck    localhost:8081  4   300a310a
GET /httpproxy.c    localhost:1234  15832   2f2f2044657370696e6120506174726f6e617320435345313330204661
6c6c2032303231206173676e330a0a23696e636c756465203c7374646c69622e683e0a23696e636c756465203c737464696e
742e683e0a23696e636c756465203c737472696e672e683e20202020202f2f20737472746f6b28292c20737472737472228292
c207374726e63707928292c207374726c656e28290a23696e636c756465203c737464696f2e683e202020202020202f2f206670
72696e746628290a23696e636c756465203c6572722e683e20202020202020202f2f207761726e28290a23696e636c7564652
03c6572726e6f2e683e202020202020202f2f20636865636b207761726e20636f6465730a23696e636c756465203c66636e746c
2e683e202020202020202f2f206f70656e28290a23696e636c756465203c756e697374642e683e20202020202f2f20636c6f736
528292c207772697465282290a23696e636c756465203c63747970652e683e202020202020202f2f206973616c6e756d28290a23
696e636c756465203c7379732f737461742e683e2020202f2f207374617428290a23696e636c756465203c7379732f66696c6
52e683e2020202f2f20666c6f636b28290a23696e636c756465203c7379732f74696d652e683e0a23696e636c756465203c67
65746f70742e683e0a2f2f206d756c74697468726561646696e67206c6962726172696572730a23696e636c756465203c7074687
26561642e683e0a23696e636c756465203c73656d6170686f72652e683e0a2f2f206e6574776f726b206c6962726172696573
0a23696e636c756465203c617270612f696e65742e683e0a23696e636c756465203c6e6574696e65742f696e2e683e0a23696
e636c756465203c7379732f736f636b65742e683e0a23696e636c756465203c7379732f74797065732e683e0a0a23696e636c
7564652022717565756522e68220a23696e636c75646520226874747070726f78792e68220a0a23646566696e65204745544f5
054494f4e5320224e3a523a733a6d3a2220202f2f206f7074696f6e616c20434d44206c696e6520617267730a23646566696e
65204d494e28782c2079292028287829203c2028792929203f20287829203a20287929290a0a65787465726e20696e74206
572726e6f3b0a737472756374207374617420737743b0a0a737461746963206e7420726573706f6e736573203d20303b2020
202020202020202020202f2f2023206f6620726573706f6e7365732066756c66696c6c6564206775617264656640a70746872656
1645f6d757465785f74206d7478203d2050544852454144f4d555445585f494e495449414c495a45523b20202f2f67756172
6420746865207368617265564
```

```
dezi@ubuntu:~/Documents/asgn3$ ./https
erver 8080 -l 80
```

```
dezi@ubuntu:~/Documents/asgn3$ ./httpse
rver 8081 -l 81
```

```
dezi@ubuntu:~/Documents/asgn3$ ./httpserve
r 8082 -l 82
```

Using various Server Terminals

Using a Proxy Terminal

Using a Client Terminal:

various curl commands for *httpproxy* which
try to fulfill requests for files in proxy / server directory:

<u>Proxy Responds</u>

1) request for !GET curl commands
2) requests which is invalid
3) requests when all servers are down

Proxy Handles:

4) requests where server healthcheck returns a bad status
5) requests when a server goes down which would have been the optimal server to port forward

6) Checking port forwarding is done correctly for load balancing scheme

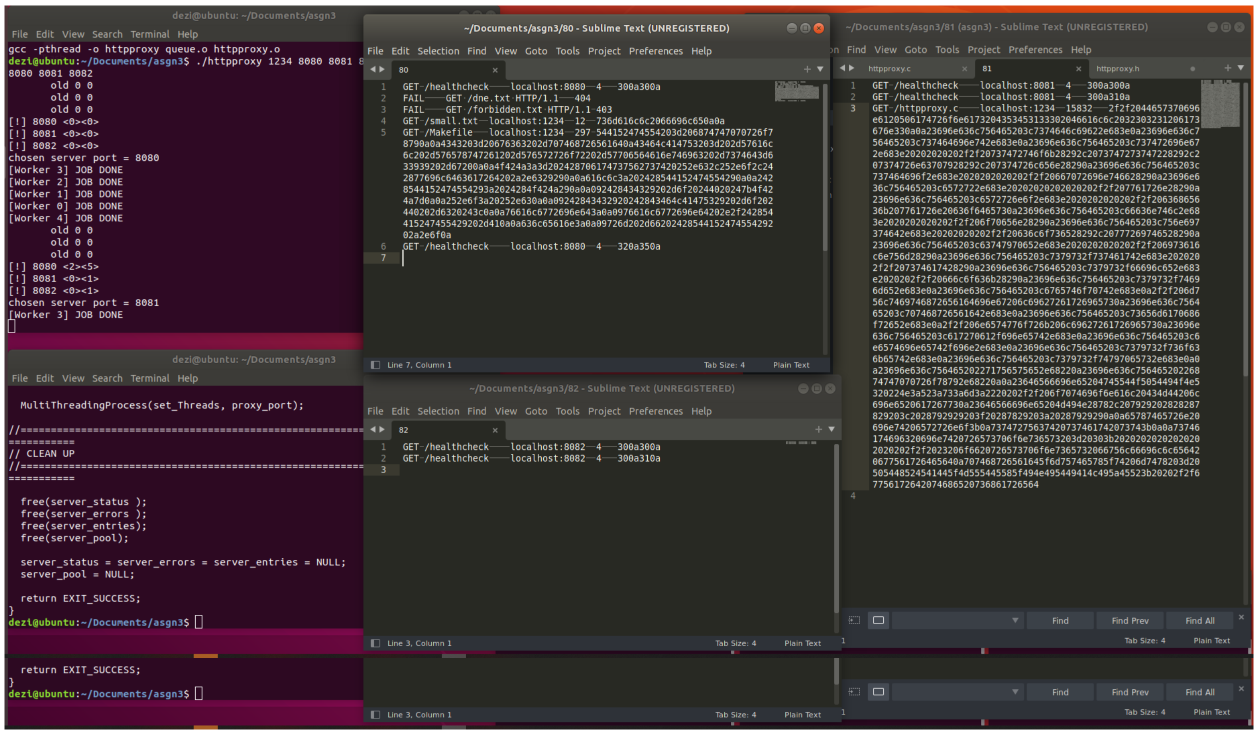
<u>Server Responds</u>

7) request for files that don't exist
8) request for files which are forbidden

Compares to expected outputs from *server*.
Each Request should run in independent threads on server
Healthchecks are done in dispatcher thread

Example Client Sends:

1. curl -T small.txt localhost:1234/small.txt
2. curl localhost:1234/-badrequest.txt
3. curl localhost:1234/dne.txt
4. curl localhost:1234/forbidden.txt
5. curl localhost:1234/small.txt
6. curl localhost:1234/Makefile


dezi@ubuntu: ~/Documents/asgn3
File Edit View Search Terminal Help
~/Documents/asgn3/80 - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
~/Documents/asgn3/81 (asgn3) - Sublime Text (UNREGISTERED)
httpproxy.c
81
httpproxy.h
~/Documents/asgn3/82 - Sublime Text (UNREGISTERED)
Line 7, Column 1
Line 3, Column 1
Tab Size: 4
Plain Text
Find
Find Prev
Find All

```
gcc -pthread -o httpproxy queue.o httpproxy.o
dezi@ubuntu:~/Documents/asgn3$ ./httpproxy 1234 8080 8081 8
8080 8081 8082
        old 0 0
        old 0 0
        old 0 0
[!] 8080 <0><0>
[!] 8081 <0><0>
[!] 8082 <0><0>
chosen server port = 8080
[Worker 3] JOB DONE
[Worker 2] JOB DONE
[Worker 1] JOB DONE
[Worker 0] JOB DONE
[Worker 4] JOB DONE
        old 0 0
        old 0 0
        old 0 0
[!] 8080 <2><5>
[!] 8081 <0><1>
[!] 8082 <0><1>
chosen server port = 8081
[Worker 3] JOB DONE
```

```
  MultiThreadingProcess(set_Threads, proxy_port);

//==============================================================
===========
// CLEAN UP
//==============================================================
===========

  free(server_status );
  free(server_errors );
  free(server_entries);
  free(server_pool);

  server_status = server_errors = server_entries = NULL;
  server_pool = NULL;

  return EXIT_SUCCESS;
}
dezi@ubuntu:~/Documents/asgn3$
```

```
  return EXIT_SUCCESS;
}
dezi@ubuntu:~/Documents/asgn3$
```

80

```
GET /healthcheck    localhost:8080  4   300a300a
FAIL    GET /dne.txt HTTP/1.1   404
FAIL    GET /forbidden.txt HTTP/1.1 403
GET /small.txt  localhost:1234  12  736d616c6c2066696c650a0a
GET /Makefile   localhost:1234  297 544152474554203d206874747070726f7
8790a0a4343203d20676363202d707468726561640a43464c414753203d202d57616c
6c202d57657874726120d576572726f72202d57706564616e746963202d7374643d6
33939202d67200a0a4f424a3a3d2024287061747375627374202522e632c252e6f2c24
2877696c6463617264202a2e6329290a0a616c6c3a2024285441524745542290a0a242
8544152474554293a2024284f424a290a0a0924284343292020d6f20244020247b4f42
4a7d0a0a252e6f3a20252e630a0a09242843432920242843464c41475329202d6f202
440202d6320243c0a0a76616c6772696e643a0a0976616c6772696e64202e2f242854
415247455429202d410a0a636c65616e3a0a09726d202d662024285441524745542922
02a2e6f0a
GET /healthcheck    localhost:8080  4   320a350a
```

81

```
GET /healthcheck    localhost:8081  4   300a300a
GET /healthcheck    localhost:8081  4   300a310a
GET /httpproxy.c    localhost:1234  15832   2f2f2044657370696
e6120506174726f6e6173204353453133302046616c6c2032303231206173
676e330a0a23696e636c756465203c7374646c69622e683e0a23696e636c7
56465203c737464696e742e683e0a23696e636c756465203c737472696e67
2e683e2020202020202f2f20737472746f6b28292c2073747273747228292c2
07374726e63707928292c207374726c656e28290a23696e636c756465203c
737464696f2e683e202020202020202f2f20667072696e746628290a23696e6
36c756465203c6572722e683e20202020202020202f2f207761726e28290a
23696e636c756465203c6572726e6f2e683e2020202020202f2f206368656
36b207761726e20636f6465730a23696e636c756465203c66636e746c2e68
3e202020202020202f2f206f70656e28290a23696e636c756465203c756e697
374642e683e2020202020202f2f20636c6f736528292c20777269746528290a
23696e636c756465203c63747970652e683e2020202020202f2f206973616
c6e756d28290a23696e636c756465203c7379732f737461742e683e202020
2f2f207374617428290a23696e636c756465203c7379732f66696c652e683
e2020202f2f20666c6f636b28290a23696e636c756465203c7379732f7469
6d652e683e0a23696e636c756465203c6765746f70742e683e0a2f2f206d7
56c7469746872656164696e67206c69627261726965730a23696e636c7564
65203c707468726561642e683e0a23696e636c756465203c73656d6170686
f72652e683e0a2f2f206e6574776f726b206c69627261726965730a23696e
636c756465203c617270612f696e65742e683e0a23696e636c756465203c6
e6574696e65742f696e2e683e0a23696e636c756465203c7379732f736f63
6b65742e683e0a23696e636c756465203c7379732f74797065732e683e0a0
a23696e636c7564652022717565575652e68220a23696e636c756465202268
74747070726f78792e68220a0a23646566696e65204745544f5054494f4e5
320224e3a523a733a6d3a2220202f2f206f7074696f6e616c20434d44206c
696e6520617267730a2364656696e65204d494e28782c20792920282828287
829203c2028792929203f20287829203a20287929290a0a65787465726e20
696e74206572726e6f3b0a737472756374207374617420737743b0a0a73746
174696320696e7420726573706f6e736573203d20303b2020202020202020
2020202f2f2023206f6620726573706f6e73657320666756c66696c6c65642
0677561726465640a707468726561645f6d757465785f74206d7478203d20
505448524541445f4d555445585f494e495449414c495a45523b20202f2f6
775617264207468652073686172656564
```

82

```
GET /healthcheck    localhost:8082  4   300a300a
GET /healthcheck    localhost:8082  4   300a310a
```

For Assignment 3, please answer the following questions:

1) **For this assignment, your proxy distributed load based on number of requests the servers had already serviced, and how many failed.**
A more realistic implementation would consider performance attributes from the machine running the server. Why was this not used for this assignment?

The performance balancing was not used because we are not able to easily determine the performance of the server due to factors we can't control. Since the server may be receiving external requests, the server speed is variable as opposed to a server which is dedicated to our proxy server.

To implement a performance-based approach on a server which takes various requests from any connecting client or server, would require us to compute the performance / request and consider the type of request and the size of transactions in the log.

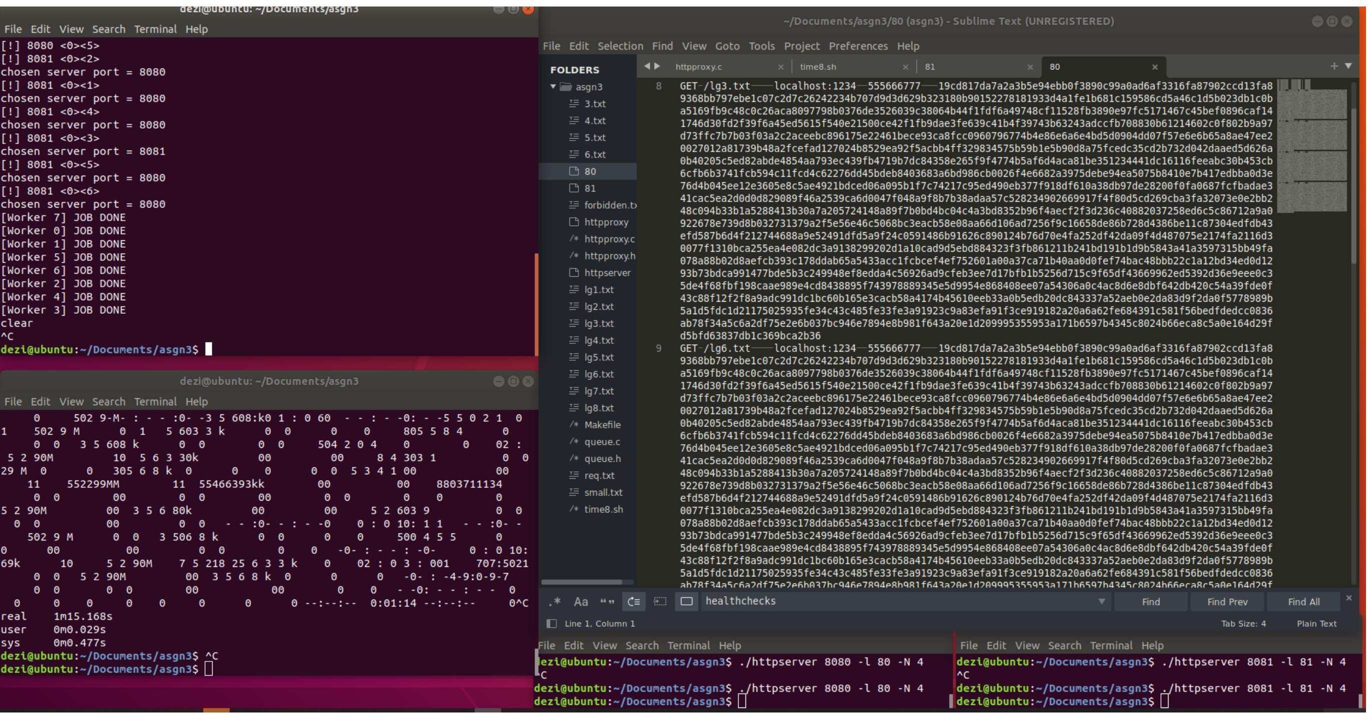
It is not straightforward and therefore, it has been omitted.

2) **Using the provided httpserver, do the following:**
– Place eight different large files in a single directory. The files should be around 400 MiB long, adjust according to your computer's capacity.
– Start two instances of your httpserver in that directory with four worker threads for each.
– Start your httpproxy with the port numbers of the two running servers and maximum of eight connections and disable the cache.
– Start eight separate instances of the client at the same time, one GETting each of the files and measure
(using time(1)) how long it takes to get the files. The best way to do this is to write a simple shell script (command file) that starts eight copies of the client program in the background, by using & at the end.

This took two attempts. One attempt tried to handle binary files, while the other handled text files…

Attempt 1: (8) Binary Files 400mb

My proxy doesn't crash, but the servers both end up crashing and curl looks corrupted.



```
[!] 8080 <0><5>
[!] 8081 <0><2>
chosen server port = 8080
[!] 8081 <0><1>
chosen server port = 8080
[!] 8081 <0><4>
chosen server port = 8080
[!] 8081 <0><3>
chosen server port = 8081
[!] 8081 <0><5>
chosen server port = 8080
[!] 8081 <0><6>
chosen server port = 8080
[Worker 7] JOB DONE
[Worker 0] JOB DONE
[Worker 1] JOB DONE
[Worker 5] JOB DONE
[Worker 6] JOB DONE
[Worker 2] JOB DONE
[Worker 4] JOB DONE
[Worker 3] JOB DONE
clear
^C
dezi@ubuntu:~/Documents/asgn3$
```

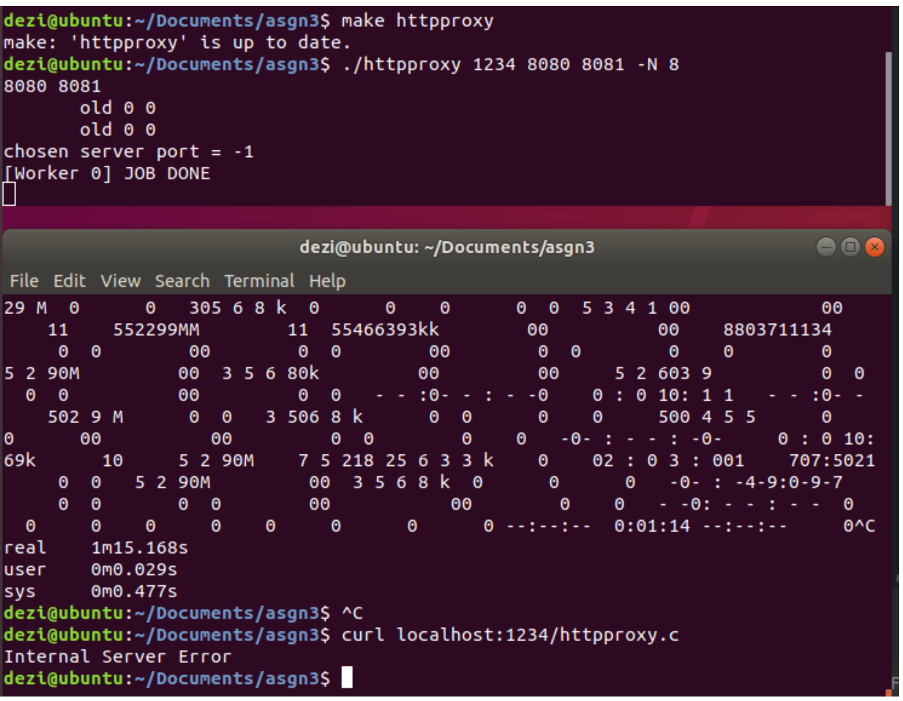
```
     0      502 9-M- : - - :0- -3 5 608:k0 1 : 0 60  - - : - -0: - -5 5 0 2 1  0
1     502 9 M       0  1    5 603 3 k       0  0       0     0      805 5 8 4       0
     0  0   3 5 608 k       0  0         0  0      504 2 0 4     0         0    02 :
 5 2 90M         10  5 6 3 30k         00          00      8 4 303 1          0  0
29 M  0      0   305 6 8 k  0      0     0      0  0  5 3 4 1 00            00
    11    552299MM        11  55466393kk        00           00    8803711134
     0  0       00        0  0         00         0  0         0    0         0
5 2 90M        00  3 5 6 80k         00          00      5 2 603 9          0  0
  0  0         00         0  0   - - :0- - : - -0    0 : 0 10: 1 1   - - :0- -
    502 9 M      0  0   3 506 8 k      0  0        0     0      500 4 5 5       0
0      00          00          0  0         0     0   -0- : - - : -0-      0 : 0 10:
69k       10      5 2 90M    7 5 218 25 6 3 3 k    0    02 : 0 3 : 001    707:5021
     0  0   5 2 90M         00  3 5 6 8 k  0      0      0   -0- : -4-9:0-9-7
     0  0       0  0         00           00         0     0   - -0: - - : - -  0
  0     0    0     0    0     0      0       0 --:--:--  0:01:14 --:--:--     0^C
real    1m15.168s
user    0m0.029s
sys     0m0.477s
dezi@ubuntu:~/Documents/asgn3$ ^C
dezi@ubuntu:~/Documents/asgn3$
```

```
GET /lg3.txt   localhost:1234  555666777   19cd817da7a2a3b5e94ebb0f3890c99a0ad6af3316fa87902ccd13fa8
9368bb797ebe1c07c2d7c26242234b707d9d3d629b323180b90152278181933d4a1fe1b681c159586cd5a46c1d5b023db1c0b
a5169fb9c48c0c26aca8097798b0376de3526039c38064b44f1fdf6a49748cf11528fb3890e97fc5171467c45bef0896caf14
1746d30fd2f39f6a45ed5615f540e21500ce42f1fb9dae3fe639c41b4f39743b63243adccfb708830b61214602c0f802b9a97
d73ffc7b7b03f03a2c2aceebc896175e22461bece93ca8fcc0960796774b4e86e6a6e4bd5d0904dd07f57e6e6b65a8ae47ee2
0027012a81739b48a2fcefad127024b8529ea92f5acbb4ff329834575b59b1e5b90d8a75fcedc35cd2b732d042daaed5d626a
0b40205c5ed82abde4854aa793ec439fb4719b7dc84358e265f9f4774b5af6d4aca81be351234441dc16116feeabc30b453cb
6cfb6b3741fcb594c11fcd4c62276dd45bdeb8403683a6bd986cb0026f4e6682a3975debe94ea5075b8410e7b417edbba0d3e
76d4b045ee12e3605e8c5ae4921bdced06a095b1f7c74217c95ed490eb377f918df610a38db97de28200f0fa0687fcfbadae3
41cac5ea2d0d0d829089f46a2539ca6d0047f048a9f8b7b38adaa57c528234902669917f4f80d5cd269cba3fa32073e0e2bb2
48c094b33b1a5288413b30a7a205724148a89f7b0bd4bc04c4a3bd8352b96f4aecf2f3d236c40882037258ed6c5c86712a9a0
922678e739d8b032731379a2f5e56e46c5068bc3eacb58e08aa66d106ad7256f9c16658de86b728d4386be11c87304edfdb43
efd587b6d4f212744688a9e52491dfd5a9f24c0591486b91626c890124b76d70e4fa252df42da09f4d487075e2174fa2116d3
0077f1310bca255ea4e082dc3a9138299202d1a10cad9d5ebd884323f3fb861211b241bd191b1d9b5843a41a3597315bb49fa
078a88b02d8aefcb393c178ddab65a5433acc1fcbcef4ef752601a00a37ca71b40aa0d0fef74bac48bbb22c1a12bd34ed0d12
93b73bdca991477bde5b3c249948ef8edda4c56926ad9cfeb3ee7d17bfb1b5256d715c9f65df43669962ed5392d36e9eee0c3
5de4f68fbf198caae989e4cd8438895f743978889345e5d9954e868408ee07a54306a0c4ac8d6e8dbf642db420c54a39fde0f
43c88f12f2f8a9adc991dc1bc60b165e3cacb58a4174b45610eeb33a0b5edb20dc843337a52aeb0e2da83d9f2da0f5778989b
5a1d5fdc1d21175025935fe34c43c485fe33fe3a91923c9a83efa91f3ce919182a20a6a62fe684391c581f56bedfdedcc0836
ab78f34a5c6a2df75e2e6b037bc946e7894e8b981f643a20e1d209995355953a171b6597b4345c8024b66eca8c5a0e164d29f
d5bfd63837db1c369bca2b36
GET /lg6.txt   localhost:1234  555666777   19cd817da7a2a3b5e94ebb0f3890c99a0ad6af3316fa87902ccd13fa8
9368bb797ebe1c07c2d7c26242234b707d9d3d629b323180b90152278181933d4a1fe1b681c159586cd5a46c1d5b023db1c0b
a5169fb9c48c0c26aca8097798b0376de3526039c38064b44f1fdf6a49748cf11528fb3890e97fc5171467c45bef0896caf14
1746d30fd2f39f6a45ed5615f540e21500ce42f1fb9dae3fe639c41b4f39743b63243adccfb708830b61214602c0f802b9a97
d73ffc7b7b03f03a2c2aceebc896175e22461bece93ca8fcc0960796774b4e86e6a6e4bd5d0904dd07f57e6e6b65a8ae47ee2
0027012a81739b48a2fcefad127024b8529ea92f5acbb4ff329834575b59b1e5b90d8a75fcedc35cd2b732d042daaed5d626a
0b40205c5ed82abde4854aa793ec439fb4719b7dc84358e265f9f4774b5af6d4aca81be351234441dc16116feeabc30b453cb
6cfb6b3741fcb594c11fcd4c62276dd45bdeb8403683a6bd986cb0026f4e6682a3975debe94ea5075b8410e7b417edbba0d3e
76d4b045ee12e3605e8c5ae4921bdced06a095b1f7c74217c95ed490eb377f918df610a38db97de28200f0fa0687fcfbadae3
41cac5ea2d0d0d829089f46a2539ca6d0047f048a9f8b7b38adaa57c528234902669917f4f80d5cd269cba3fa32073e0e2bb2
48c094b33b1a5288413b30a7a205724148a89f7b0bd4bc04c4a3bd8352b96f4aecf2f3d236c40882037258ed6c5c86712a9a0
922678e739d8b032731379a2f5e56e46c5068bc3eacb58e08aa66d106ad7256f9c16658de86b728d4386be11c87304edfdb43
efd587b6d4f212744688a9e52491dfd5a9f24c0591486b91626c890124b76d70e4fa252df42da09f4d487075e2174fa2116d3
0077f1310bca255ea4e082dc3a9138299202d1a10cad9d5ebd884323f3fb861211b241bd191b1d9b5843a41a3597315bb49fa
078a88b02d8aefcb393c178ddab65a5433acc1fcbcef4ef752601a00a37ca71b40aa0d0fef74bac48bbb22c1a12bd34ed0d12
93b73bdca991477bde5b3c249948ef8edda4c56926ad9cfeb3ee7d17bfb1b5256d715c9f65df43669962ed5392d36e9eee0c3
5de4f68fbf198caae989e4cd8438895f743978889345e5d9954e868408ee07a54306a0c4ac8d6e8dbf642db420c54a39fde0f
43c88f12f2f8a9adc991dc1bc60b165e3cacb58a4174b45610eeb33a0b5edb20dc843337a52aeb0e2da83d9f2da0f5778989b
5a1d5fdc1d21175025935fe34c43c485fe33fe3a91923c9a83efa91f3ce919182a20a6a62fe684391c581f56bedfdedcc0836
ab78f34a5c6a2df75e2e6b037bc946e7894e8b981f643a20e1d209995355953a171b6597b4345c8024b66eca8c5a0e164d29f
```

```
dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 4

dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 4
dezi@ubuntu:~/Documents/asgn3$
```

```
dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8081 -l 81 -N 4
^C
dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8081 -l 81 -N 4
dezi@ubuntu:~/Documents/asgn3$
```



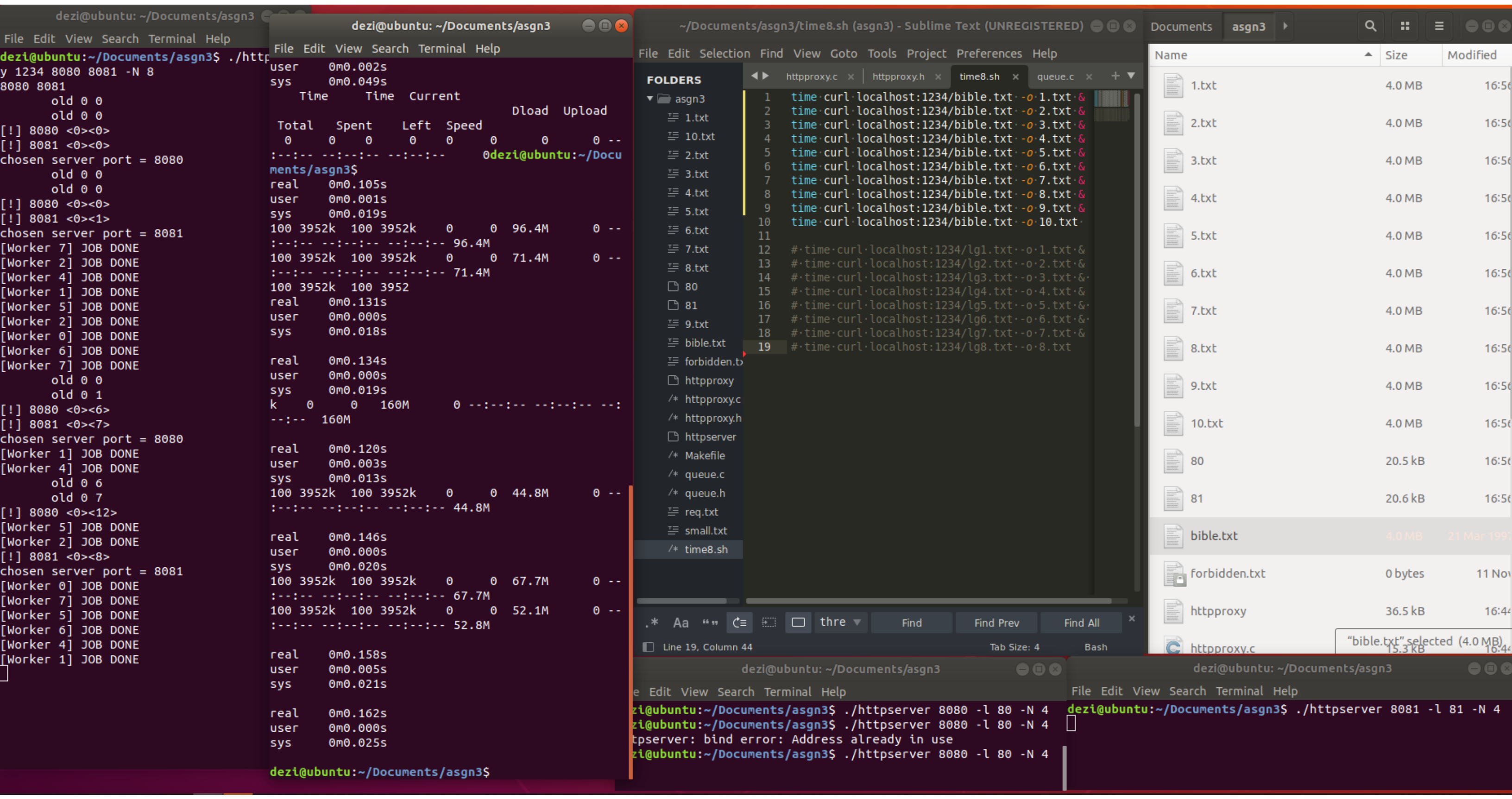
```
dezi@ubuntu:~/Documents/asgn3$ make httpproxy
make: 'httpproxy' is up to date.
dezi@ubuntu:~/Documents/asgn3$ ./httpproxy 1234 8080 8081 -N 8
8080 8081
        old 0 0
        old 0 0
chosen server port = -1
[Worker 0] JOB DONE
```

```
29 M  0      0   305 6 8 k  0      0     0      0  0  5 3 4 1 00            00
    11    552299MM        11  55466393kk        00           00    8803711134
     0  0       00        0  0         00         0  0         0    0         0
5 2 90M        00  3 5 6 80k         00          00      5 2 603 9          0  0
  0  0         00         0  0   - - :0- - : - -0    0 : 0 10: 1 1   - - :0- -
    502 9 M      0  0   3 506 8 k      0  0        0     0      500 4 5 5       0
0      00          00          0  0         0     0   -0- : - - : -0-      0 : 0 10:
69k       10      5 2 90M    7 5 218 25 6 3 3 k    0    02 : 0 3 : 001    707:5021
     0  0   5 2 90M         00  3 5 6 8 k  0      0      0   -0- : -4-9:0-9-7
     0  0       0  0         00           00         0     0   - -0: - - : - -  0
  0     0    0     0    0     0      0       0 --:--:--  0:01:14 --:--:--     0^C
real    1m15.168s
user    0m0.029s
sys     0m0.477s
dezi@ubuntu:~/Documents/asgn3$ ^C
dezi@ubuntu:~/Documents/asgn3$ curl localhost:1234/httpproxy.c
Internal Server Error
dezi@ubuntu:~/Documents/asgn3$
```

Appears some of the files were able to be processed while some were not. Both servers failed. Trying to process the binary files. Surprisingly my proxy was not able to send the internal server error response back like it should have when all servers are offline.

This experiment did not go well, so I will try it with regular text files instead.

This test was successful.



```
dezi@ubuntu:~/Documents/asgn3$ ./http
y 1234 8080 8081 -N 8
8080 8081
       old 0 0
       old 0 0
[!] 8080 <0><0>
[!] 8081 <0><0>
chosen server port = 8080
       old 0 0
       old 0 0
[!] 8080 <0><0>
[!] 8081 <0><1>
chosen server port = 8081
[Worker 7] JOB DONE
[Worker 2] JOB DONE
[Worker 4] JOB DONE
[Worker 1] JOB DONE
[Worker 5] JOB DONE
[Worker 2] JOB DONE
[Worker 0] JOB DONE
[Worker 6] JOB DONE
[Worker 7] JOB DONE
       old 0 0
       old 0 1
[!] 8080 <0><6>
[!] 8081 <0><7>
chosen server port = 8080
[Worker 1] JOB DONE
[Worker 4] JOB DONE
       old 0 6
       old 0 7
[!] 8080 <0><12>
[Worker 5] JOB DONE
[Worker 2] JOB DONE
[!] 8081 <0><8>
chosen server port = 8081
[Worker 0] JOB DONE
[Worker 7] JOB DONE
[Worker 5] JOB DONE
[Worker 6] JOB DONE
[Worker 4] JOB DONE
[Worker 1] JOB DONE
```

```
user    0m0.002s
sys     0m0.049s
   Time     Time  Current
                                 Dload  Upload
 Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --
:--:-- --:--:-- --:--:--     0dezi@ubuntu:~/Docu
ments/asgn3$
real    0m0.105s
user    0m0.001s
sys     0m0.019s
100 3952k  100 3952k    0     0  96.4M      0 --
:--:-- --:--:-- --:--:-- 96.4M
100 3952k  100 3952k    0     0  71.4M      0 --
:--:-- --:--:-- --:--:-- 71.4M
100 3952k  100 3952
real    0m0.131s
user    0m0.000s
sys     0m0.018s

real    0m0.134s
user    0m0.000s
sys     0m0.019s
k    0     0   160M      0 --:--:-- --:--:-- --:
--:--  160M

real    0m0.120s
user    0m0.003s
sys     0m0.013s
100 3952k  100 3952k    0     0  44.8M      0 --
:--:-- --:--:-- --:--:-- 44.8M

real    0m0.146s
user    0m0.000s
sys     0m0.020s
100 3952k  100 3952k    0     0  67.7M      0 --
:--:-- --:--:-- --:--:-- 67.7M
100 3952k  100 3952k    0     0  52.1M      0 --
:--:-- --:--:-- --:--:-- 52.8M

real    0m0.158s
user    0m0.005s
sys     0m0.021s

real    0m0.162s
user    0m0.000s
sys     0m0.025s

dezi@ubuntu:~/Documents/asgn3$
```

```
time curl localhost:1234/bible.txt -o 1.txt &
time curl localhost:1234/bible.txt -o 2.txt &
time curl localhost:1234/bible.txt -o 3.txt &
time curl localhost:1234/bible.txt -o 4.txt &
time curl localhost:1234/bible.txt -o 5.txt &
time curl localhost:1234/bible.txt -o 6.txt &
time curl localhost:1234/bible.txt -o 7.txt &
time curl localhost:1234/bible.txt -o 8.txt &
time curl localhost:1234/bible.txt -o 9.txt &
time curl localhost:1234/bible.txt -o 10.txt

# time curl localhost:1234/lg1.txt -o 1.txt &
# time curl localhost:1234/lg2.txt -o 2.txt &
# time curl localhost:1234/lg3.txt -o 3.txt &
# time curl localhost:1234/lg4.txt -o 4.txt &
# time curl localhost:1234/lg5.txt -o 5.txt &
# time curl localhost:1234/lg6.txt -o 6.txt &
# time curl localhost:1234/lg7.txt -o 7.txt &
# time curl localhost:1234/lg8.txt -o 8.txt
```

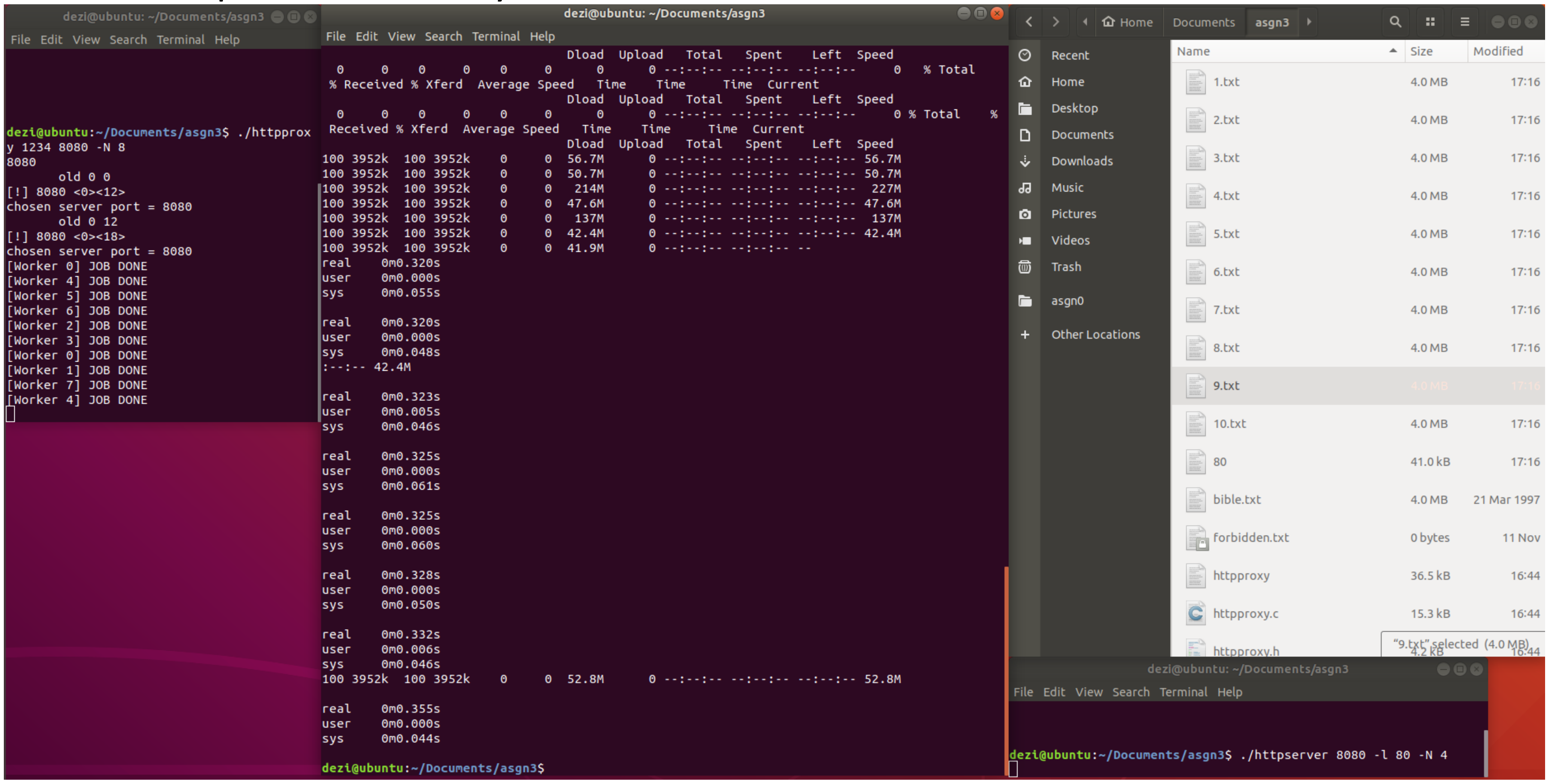
```
zi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 4
zi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 4
tpserver: bind error: Address already in use
zi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 4
```

```
dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8081 -l 81 -N 4
```

Processing 8 text files 4.0 MiB Concurrently
On Proxy with 8 Threads
Using 2 Servers with 4 Threads each

Fastest time:		**Longest time:**	
real	0m0.071s	real	0m0.162s
user	0m0.000s	user	0m0.000s
sys	0m0.014s	sys	0m0.025s

3) Repeat the same experiment, but turn off one of the servers. Is there any difference in performance? What do you observe?



```
dezi@ubuntu:~/Documents/asgn3$ ./httpprox
y 1234 8080 -N 8
8080
        old 0 0
[!] 8080 <0><12>
chosen server port = 8080
        old 0 12
[!] 8080 <0><18>
chosen server port = 8080
[Worker 0] JOB DONE
[Worker 4] JOB DONE
[Worker 5] JOB DONE
[Worker 6] JOB DONE
[Worker 2] JOB DONE
[Worker 3] JOB DONE
[Worker 0] JOB DONE
[Worker 1] JOB DONE
[Worker 7] JOB DONE
[Worker 4] JOB DONE
```

```
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0   % Total
 % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0 % Total    %
 Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0  56.7M      0 --:--:-- --:--:-- --:--:-- 56.7M
100 3952k  100 3952k    0     0  50.7M      0 --:--:-- --:--:-- --:--:-- 50.7M
100 3952k  100 3952k    0     0   214M      0 --:--:-- --:--:-- --:--:--  227M
100 3952k  100 3952k    0     0  47.6M      0 --:--:-- --:--:-- --:--:-- 47.6M
100 3952k  100 3952k    0     0   137M      0 --:--:-- --:--:-- --:--:--  137M
100 3952k  100 3952k    0     0  42.4M      0 --:--:-- --:--:-- --:--:-- 42.4M
100 3952k  100 3952k    0     0  41.9M      0 --:--:-- --:--:-- --
real    0m0.320s
user    0m0.000s
sys     0m0.055s

real    0m0.320s
user    0m0.000s
sys     0m0.048s
:--:-- 42.4M

real    0m0.323s
user    0m0.005s
sys     0m0.046s

real    0m0.325s
user    0m0.000s
sys     0m0.061s

real    0m0.325s
user    0m0.000s
sys     0m0.060s

real    0m0.328s
user    0m0.000s
sys     0m0.050s

real    0m0.332s
user    0m0.006s
sys     0m0.046s
100 3952k  100 3952k    0     0  52.8M      0 --:--:-- --:--:-- --:--:-- 52.8M

real    0m0.355s
user    0m0.000s
sys     0m0.044s
dezi@ubuntu:~/Documents/asgn3$
```

Name	Size	Modified
1.txt	4.0 MB	17:16
2.txt	4.0 MB	17:16
3.txt	4.0 MB	17:16
4.txt	4.0 MB	17:16
5.txt	4.0 MB	17:16
6.txt	4.0 MB	17:16
7.txt	4.0 MB	17:16
8.txt	4.0 MB	17:16
9.txt	4.0 MB	17:16
10.txt	4.0 MB	17:16
80	41.0 kB	17:16
bible.txt	4.0 MB	21 Mar 1997
forbidden.txt	0 bytes	11 Nov
httpproxy	36.5 kB	16:44
httpproxy.c	15.3 kB	16:44
httpproxy.h	4.2 kB	16:44

```
dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 4
```

The single server, gave times that are about 2x slower than the previous experiment. This makes sense since only one server can be replying to the messages sent by the proxy.

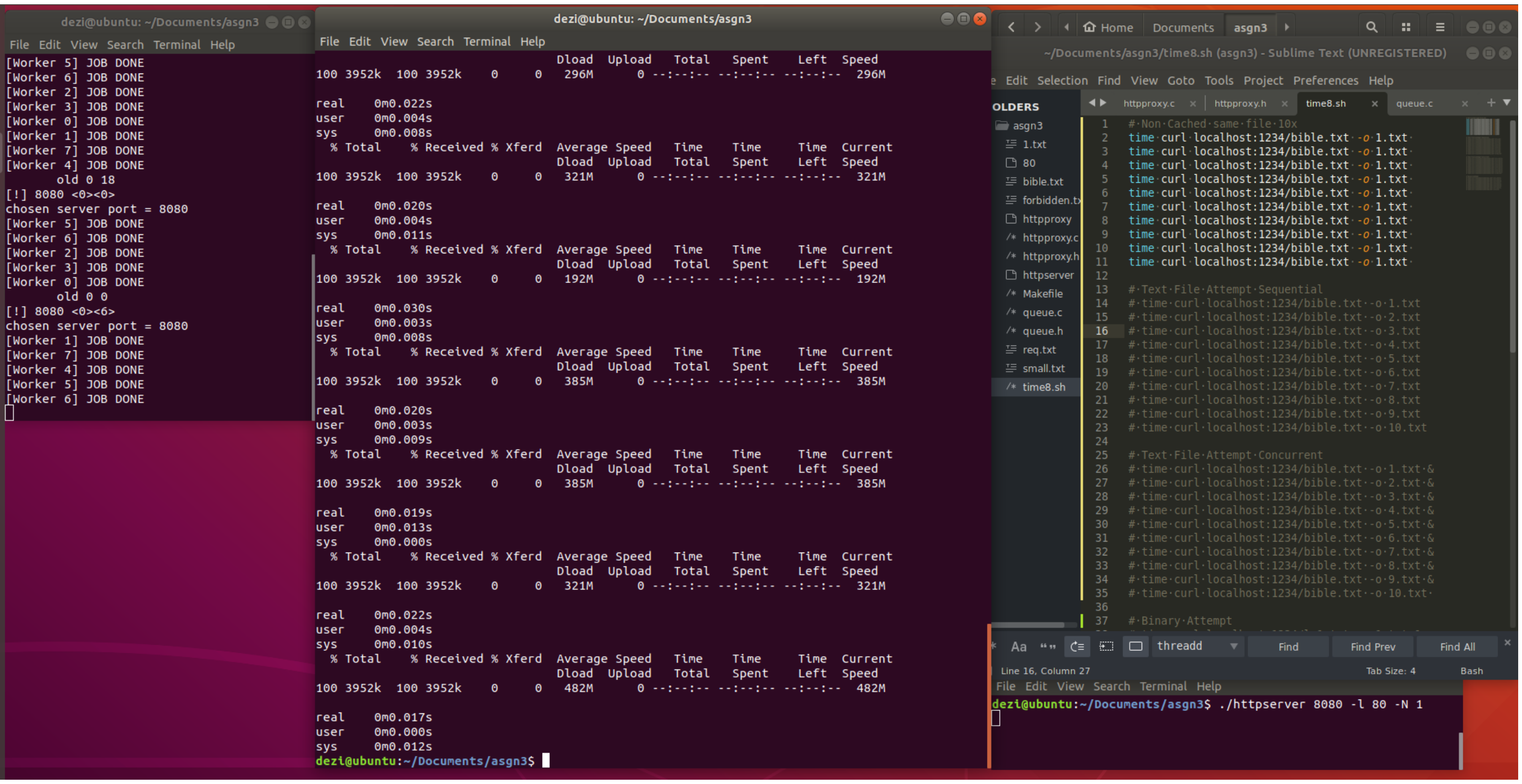
Processing 8 text files 4.0 MiB Concurrently
on Proxy with 8 Threads
Using 1 Servers with 4 Threads

Fastest time:		**Longest time:**	
real	0m0.273s	real	0m0.355s
user	0m0.007s	user	0m0.000s
sys	0m0.047s	sys	0m0.044s

4) Using one of the files that you created for the previous question, start the provided server with only one thread, then do the following:

– Start your proxy without caching.

– Request the file ten times. How long does it take?



```
[Worker 5] JOB DONE
[Worker 6] JOB DONE
[Worker 2] JOB DONE
[Worker 3] JOB DONE
[Worker 0] JOB DONE
[Worker 1] JOB DONE
[Worker 7] JOB DONE
[Worker 4] JOB DONE
      old 0 18
[!] 8080 <0><0>
chosen server port = 8080
[Worker 5] JOB DONE
[Worker 6] JOB DONE
[Worker 2] JOB DONE
[Worker 3] JOB DONE
[Worker 0] JOB DONE
      old 0 0
[!] 8080 <0><6>
chosen server port = 8080
[Worker 1] JOB DONE
[Worker 7] JOB DONE
[Worker 4] JOB DONE
[Worker 5] JOB DONE
[Worker 6] JOB DONE
```

```
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   296M      0 --:--:-- --:--:-- --:--:--  296M

real    0m0.022s
user    0m0.004s
sys     0m0.008s
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   321M      0 --:--:-- --:--:-- --:--:--  321M

real    0m0.020s
user    0m0.004s
sys     0m0.011s
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   192M      0 --:--:-- --:--:-- --:--:--  192M

real    0m0.030s
user    0m0.003s
sys     0m0.008s
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   385M      0 --:--:-- --:--:-- --:--:--  385M

real    0m0.020s
user    0m0.003s
sys     0m0.009s
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   385M      0 --:--:-- --:--:-- --:--:--  385M

real    0m0.019s
user    0m0.013s
sys     0m0.000s
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   321M      0 --:--:-- --:--:-- --:--:--  321M

real    0m0.022s
user    0m0.004s
sys     0m0.010s
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3952k  100 3952k    0     0   482M      0 --:--:-- --:--:-- --:--:--  482M

real    0m0.017s
user    0m0.000s
sys     0m0.012s
dezi@ubuntu:~/Documents/asgn3$
```

```
# Non Cached same file 10x
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt
time curl localhost:1234/bible.txt -o 1.txt

# Text File Attempt Sequential
# time curl localhost:1234/bible.txt -o 1.txt
# time curl localhost:1234/bible.txt -o 2.txt
# time curl localhost:1234/bible.txt -o 3.txt
# time curl localhost:1234/bible.txt -o 4.txt
# time curl localhost:1234/bible.txt -o 5.txt
# time curl localhost:1234/bible.txt -o 6.txt
# time curl localhost:1234/bible.txt -o 7.txt
# time curl localhost:1234/bible.txt -o 8.txt
# time curl localhost:1234/bible.txt -o 9.txt
# time curl localhost:1234/bible.txt -o 10.txt

# Text File Attempt Concurrent
# time curl localhost:1234/bible.txt -o 1.txt &
# time curl localhost:1234/bible.txt -o 2.txt &
# time curl localhost:1234/bible.txt -o 3.txt &
# time curl localhost:1234/bible.txt -o 4.txt &
# time curl localhost:1234/bible.txt -o 5.txt &
# time curl localhost:1234/bible.txt -o 6.txt &
# time curl localhost:1234/bible.txt -o 7.txt &
# time curl localhost:1234/bible.txt -o 8.txt &
# time curl localhost:1234/bible.txt -o 9.txt &
# time curl localhost:1234/bible.txt -o 10.txt

# Binary Attempt
```

```
dezi@ubuntu:~/Documents/asgn3$ ./httpserver 8080 -l 80 -N 1
```

Processing 8 text files 4.0 MiB Sequentially
on Proxy with 5 Threads
Using 1 Servers with 1 Threads

Fastest time:		**Longest time:**	
real	0m0.016s	real	0m0.020s
user	0m0.004s	user	0m0.003s
sys	0m0.008s	sys	0m0.008s

5) Now stop your proxy and start again, this time with caching configured so it can store the selected file.

I have not done caching yet…

6) Request the same file ten times again. How long does it take now?

I have not done caching yet…

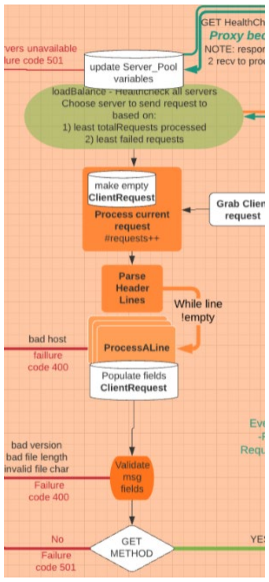
7) What did you learn about system design from this class? In particular, describe how each of the basic techniques (abstraction, layering, hierarchy, and modularity) helped you by simplifying your design, making it more efficient, or making it easier to design.

Abstraction –

Having very generic functions help increase abstraction. Tolerant of inputs and strict on outputs ie: making a file descriptor be defined as anything from a client to a server, or even a file or stdout. They are all treated similarly. The result is that an operation will take place on that file descriptor no matter what it points to. Very powerful technique to reducing repetition and extraneous bugs caused by unnecessary code that has the same functionality but used in a different part of the code. All buffer transfer logic remained consistent throughout projects and it was interesting to see how a function could be used in both instances of communication server<->client or GET<->PUT. Very similar functionality but with roles being reversed.

Layering –

Learned to make a design more linear by layering modules in order to reduce the interactions. This kind of control flow makes it easy to detect when a portion breaks and cannot progress in the line of modules. This kind of module organization makes the program easier to trace as well.



Hierarchy –

Using structures and enumeration to produce hierarchical organization. Threads are organized In structures which contain information about the threads.
Using structures nested with enumerated or defined structures make it easy to store, and contain information which can be defined in a fault tolerant way.

Modularity –

learned to group operations better into functions and make functions more GENERIC to produce functionality in broad terms. When there is a logical jump in flow, it is best to put that code in a function to isolate it from other operations. This way, when a block of code is misbehaving, one can ascertain the issue lies in the parameters passed in or the logic within the code which gives an incorrect output or outcome. It's easier to narrow down bugs using modularity since it provides a indication to the piece of logic that has failed based on the function that performed it.