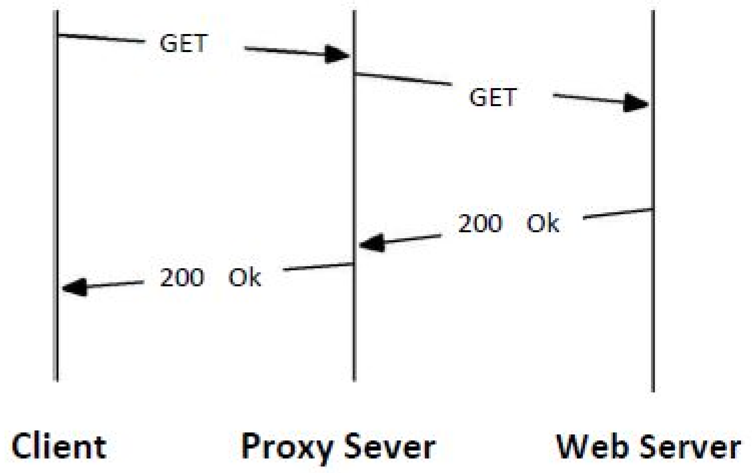
ECEN 602
Network Programming Assignment 4
Simple HTTP Proxy(Bonus Implemented)

TEAM : 1

SANKET AGARWAL : 329006490
DHIRAJ KUDVA : 829009538

Problem Description:

This assignment deals with the implementation of a simple HTTP proxy server and HTTP command-line client.



This Machine problem uses HTTP/1.0 and operates over the TCP from the transport layer. Since no time-critical information is being transmitted, it makes sense for using the TCP protocol. All data needs to be accurate even if there is a delay in receiving the file.

In order to save the response time, the proxy server implements an LRU cache that returns the page without accessing the webserver. This cache has a capacity of 10 pages. If the page is not present in the cache it will try to get that page from the web server and return it back to the client.

The page present in the cache need not be a valid page. I.e. it can be a stale page as well. This is known by the EXPIRES header of the HTTP/1.0 protocol. Sometimes, the EXPIRES header and the Last-modified header are missed from the protocol, and thus enquiring the staleness of a page becomes difficult.

Bonus Feature: In order to overcome the above difficulty of missing the date field, we have implemented the conditional GET command. This command asks the webserver if the page has changed or not. If a page has missing Expires and Modified GMT entries, then pages were not cached. However, with the use of this command, we have stored those pages even if they have missing fields since we will be verifying with the webserver about the staleness.

Team Contribution:
Client and functions was written by Sanket Agarwal whereas the Proxy server was implemented by Dhiraj Kudva. Test cases were implemented over a zoom call on weekends.

Directory Structure:

Client usage: ./client <IP_add> <Port_no>
<http://man7.org/linux/man-pages/dir_section_2.html>
Proxy usage: ./proxy <ip_add> <port>

1. Client.c: This contains the client code implementation for this machine problem. First, the connection gets established like previous MPs. Then we pass the command line argument 3 i.e. the HTML request page to the parser API to separate it out. After we get the separated HTML request name, we send it to the proxy server and wait for the server to return back with data. Once the message is received, we store a local copy of the received data by using fwrite function.

2. Functions.c: functions includes several APIs that are used by both the client and the proxy server. Most importantly it contains the parser API that is used to separate the passed URL into different names. Along with that it also contains the cache entry check, zombie handler, month converter, cache entry expiry check, and error message send APIs.

3. Server: This is the proxy server implementation. It first starts by listening to multiple clients like previous MPs. It uses the same SELECT API to listen to multiple clients as implemented in MP2. Once the HTTP request is received, it checks for the cache entry presence. If the return value is -1, that means the entry is not present in the cache. It prints that information and requests the webserver to get the complete data from there. Once the webserver replies, it updates the cache table and removes one entry if the cache is full based on the LRU policy. After updating the cache, it sends the data back to the client as it was initially requested by the client itself. If the cache entry is present and it is not a stale entry, it returns the data from its cache entry. As a part of the BONUS feature, it uses the GET command to verify the staleness of a page.

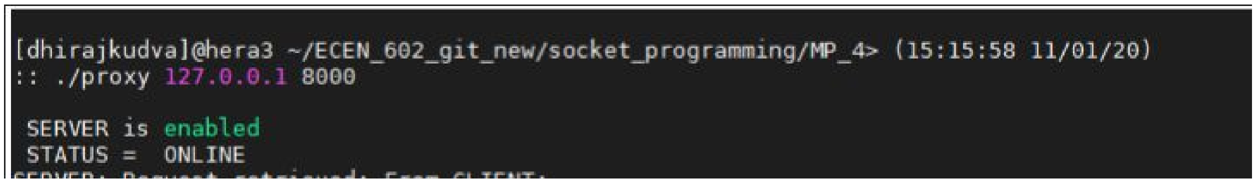
Usage:
The directory shared has the following necessary files:

1. client.C
2. proxy.c
3. Functions.c
4. Makefile

First step would be to generate the object files for client and server.
Once you are in this working directory, run “make” on the command line. This will result in two object files namely proxy(generated from proxy.c) and client (generated from client.c and stored in client_folder).

Note: For the multiple client case, you need to manually copy the client object file in the empty folder created by us and run different clients from all those folders.

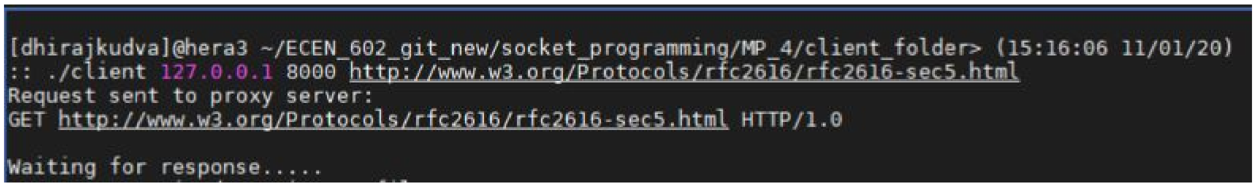
To setup client and socket environment, open another terminal and point to the same working directory where all the collaterals (necessary files) are generated.
The next step would be to run the proxy server and client. To run the proxy server, run the below command:
./proxy <ip_add> <port>
where ip_add corresponds to the local IP address of the host proxy server. And port can be any ephemeral port number. This same IP address and port number is needed to connect the client with the server.



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:15:58 11/01/20)
:: ./proxy 127.0.0.1 8000

 SERVER is enabled
 STATUS =  ONLINE
```

And to run the client from client_folder, run the following command:
./client <IP_add> <Port_no>   http://man7.org/linux/man-pages/dir_section_2.html

The third argument in the above command is the link of the HTML page that must be accessed and stored in cache.

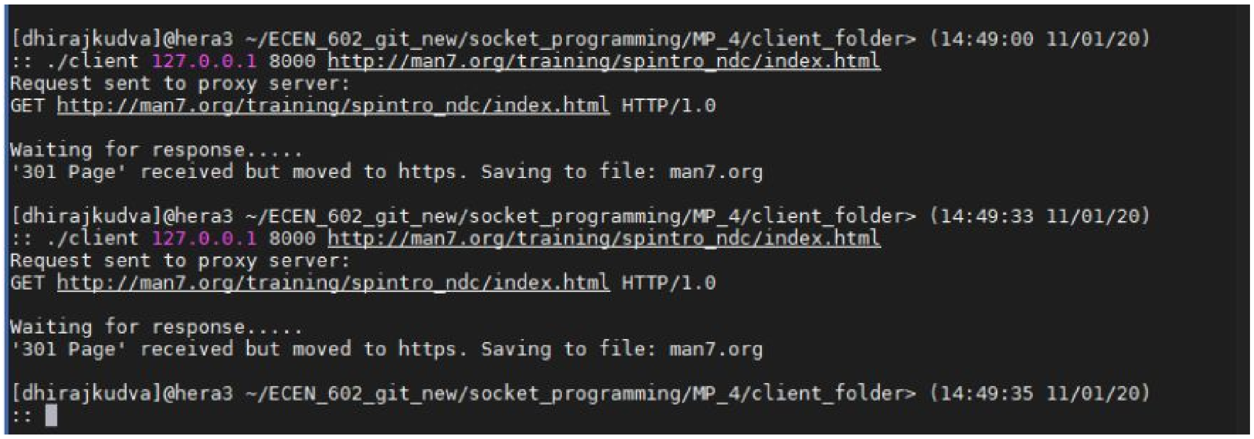


```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:16:06 11/01/20)
:: ./client 127.0.0.1 8000 http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
Request sent to proxy server:
GET http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html HTTP/1.0

Waiting for response.....
```

Once the client sends the request, the server then searches the particular file in the cache. If not present, then it makes connection with the web server and accesses that file and stores in the cache and also provides it to the client.

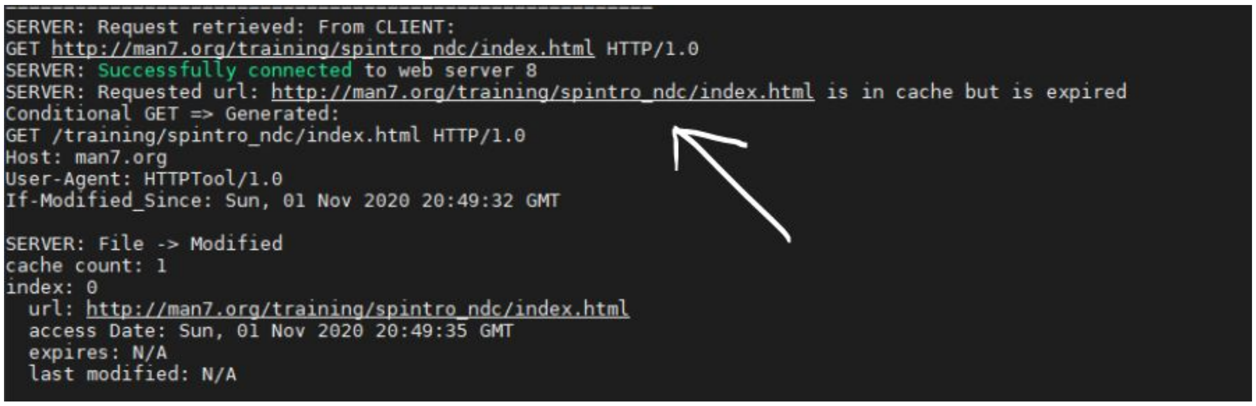
TESTCASE:

1. a cache hit returns the saved data to the requester;



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:00 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/training/spintro_ndc/index.html
Request sent to proxy server:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:33 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/training/spintro_ndc/index.html
Request sent to proxy server:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:35 11/01/20)
::
```

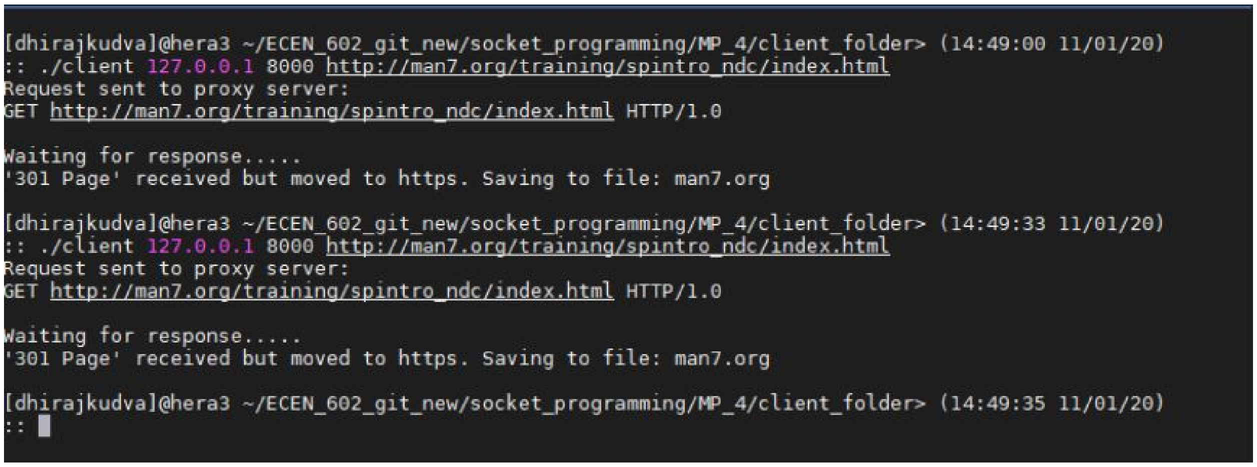
In this particular test case, it can be seen from the above snapshot that the client is requesting for an index.html page from the proxy server. This particular page is already in the server and will be accessed from its cache by the proxy server.
The below figure shows the message (highlighted in white box) that the html page is in cache and accessed from the server



```
SERVER: Request retrieved: From CLIENT:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0
SERVER: Successfully connected to web server 8
SERVER: Requested url: http://man7.org/training/spintro_ndc/index.html is in cache but is expired
Conditional GET => Generated:
GET /training/spintro_ndc/index.html HTTP/1.0
Host: man7.org
User-Agent: HTTPTool/1.0
If-Modified_Since: Sun, 01 Nov 2020 20:49:32 GMT

SERVER: File -> Modified
cache count: 1
index: 0
  url: http://man7.org/training/spintro_ndc/index.html
  access Date: Sun, 01 Nov 2020 20:49:35 GMT
  expires: N/A
  last modified: N/A
```

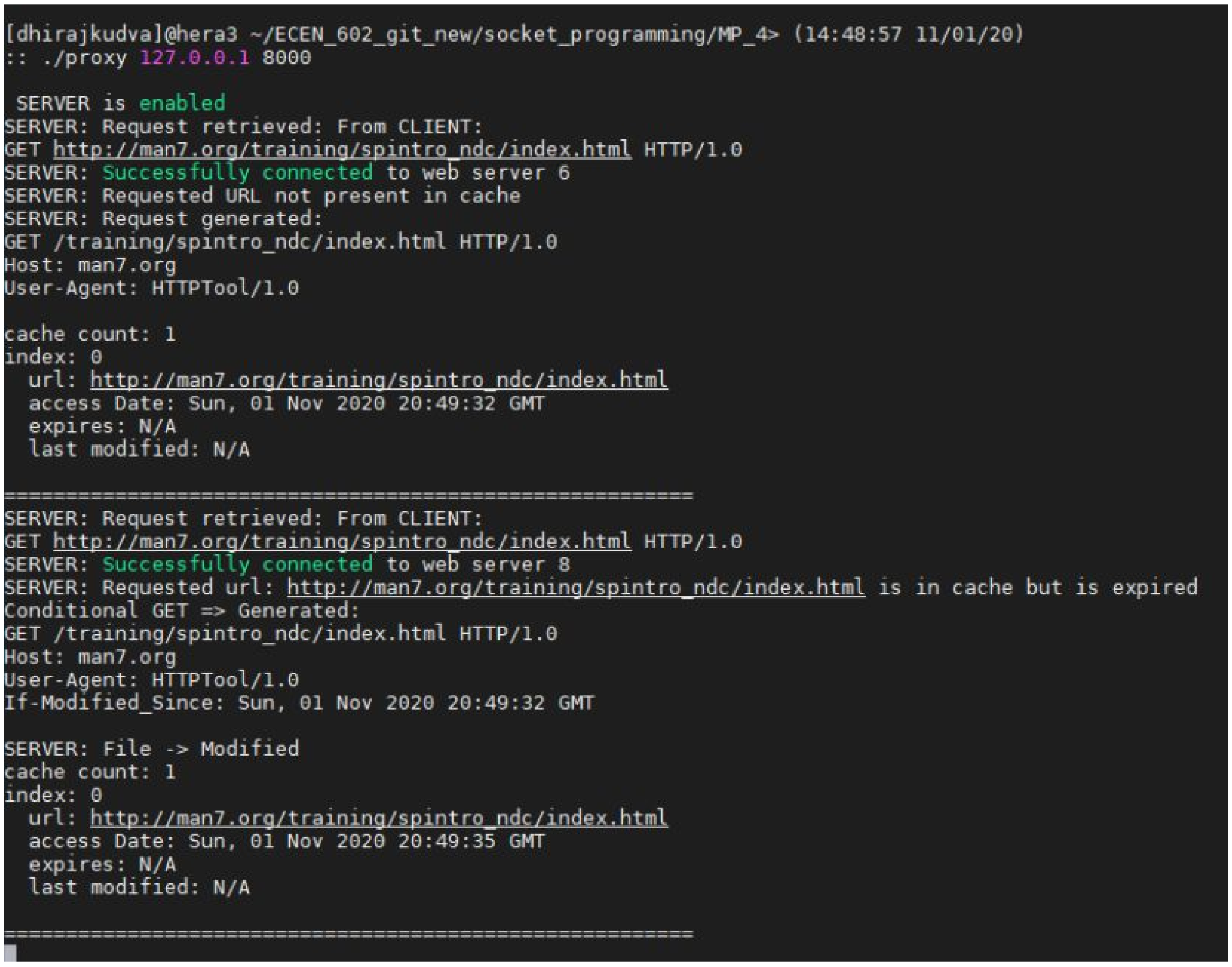
2. request that is not in the cache is proxied, saved in the cache, and returned to the requester

This is similar to the earlier testcase. But in this particular testcase, the file requested by the client would not be part of the proxy cache initially. So in this case it will result in a cache miss and the proxy server will then get access to the file from the web server and put it in its cache. Later when again the client requests, this time it will result in cache hit and server would itself share the requested file. It might update the file in cache if it is expired.
Client side:



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:00 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/training/spintro_ndc/index.html
Request sent to proxy server:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:33 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/training/spintro_ndc/index.html
Request sent to proxy server:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:35 11/01/20)
::
```

Server side:

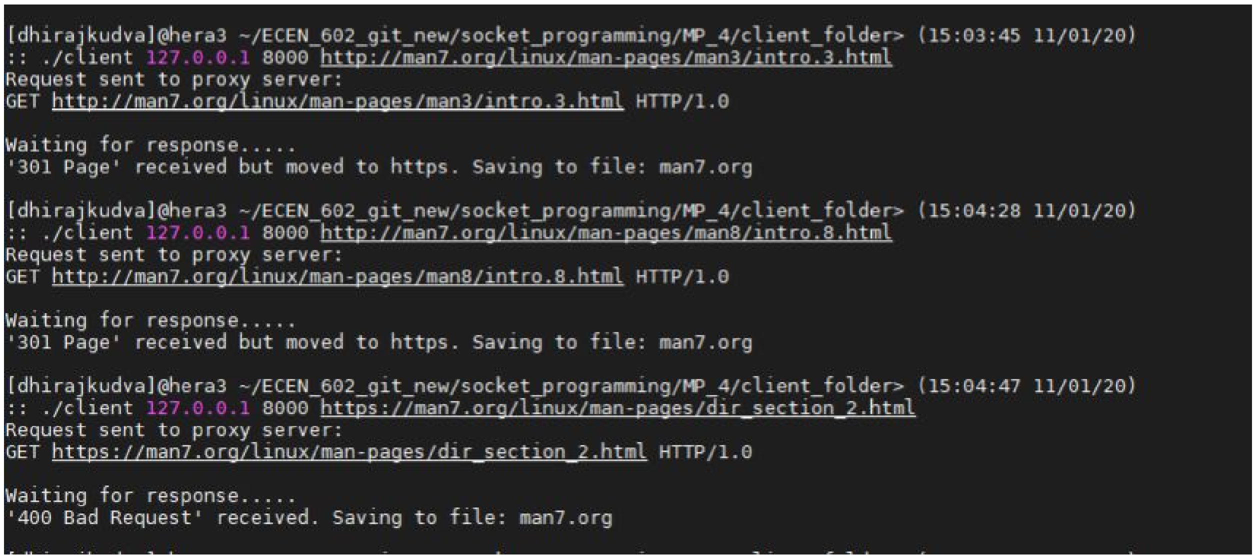


```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (14:48:57 11/01/20)
:: ./proxy 127.0.0.1 8000

 SERVER is enabled
SERVER: Request retrieved: From CLIENT:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0
SERVER: Successfully connected to web server 6
SERVER: Requested URL not present in cache
SERVER: Request generated:
GET /training/spintro_ndc/index.html HTTP/1.0
Host: man7.org
User-Agent: HTTPTool/1.0

cache count: 1
index: 0
  url: http://man7.org/training/spintro_ndc/index.html
  access Date: Sun, 01 Nov 2020 20:49:32 GMT
  expires: N/A
  last modified: N/A

========================================================
SERVER: Request retrieved: From CLIENT:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0
SERVER: Successfully connected to web server 8
SERVER: Requested url: http://man7.org/training/spintro_ndc/index.html is in cache but is expired
Conditional GET => Generated:
GET /training/spintro_ndc/index.html HTTP/1.0
Host: man7.org
User-Agent: HTTPTool/1.0
If-Modified_Since: Sun, 01 Nov 2020 20:49:32 GMT

SERVER: File -> Modified
cache count: 1
index: 0
  url: http://man7.org/training/spintro_ndc/index.html
  access Date: Sun, 01 Nov 2020 20:49:35 GMT
  expires: N/A
  last modified: N/A

========================================================
```

3. a cache miss with 10 items already in the cache is proxied, saved in the LRU location in cache, and the data is returned to the requester;

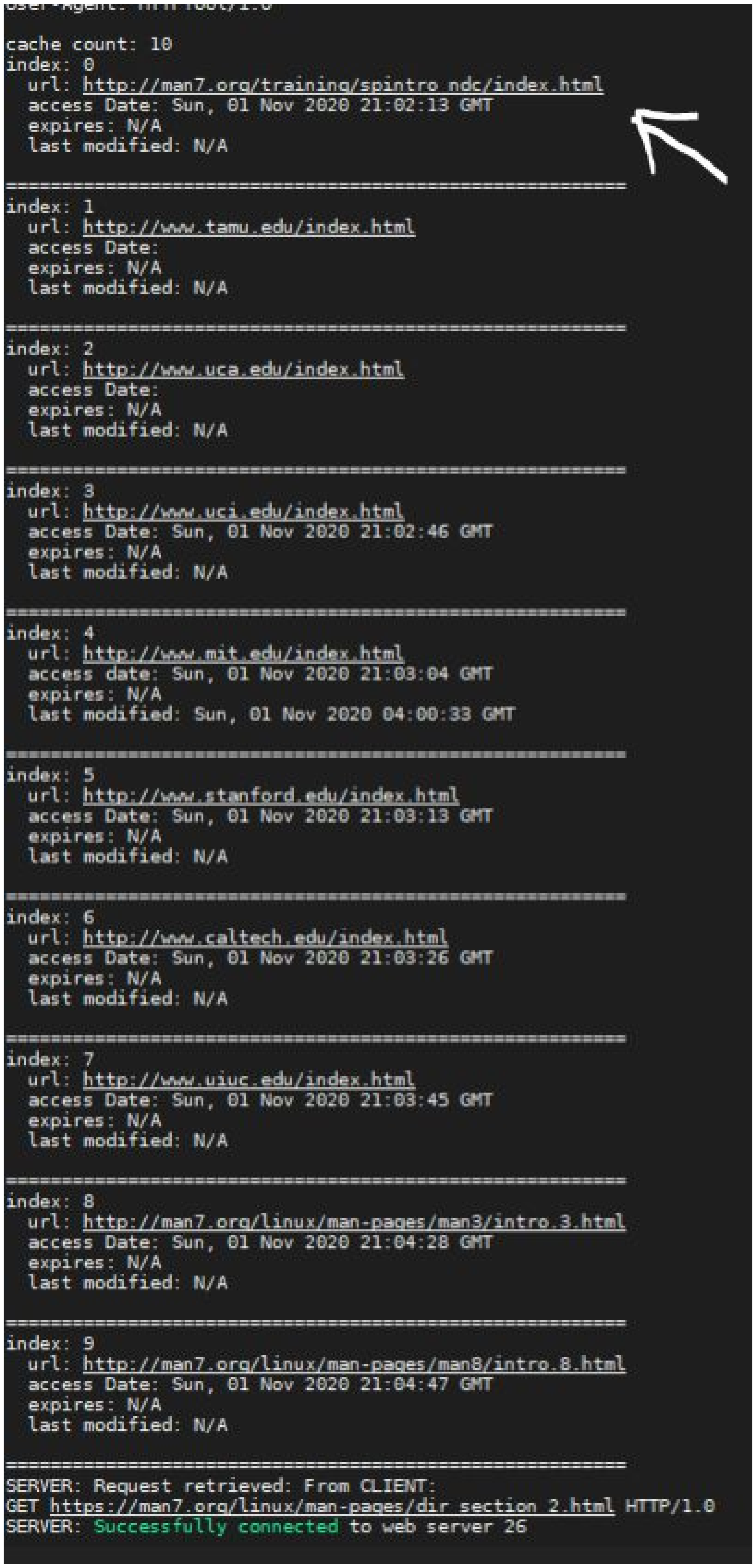
In this testcase, the client requests for more than 10 items from the proxy server. The first 10 are stored in the server proxy cache. Any new entries requested by the client (cache miss will occur), will replace one of the previous entries in cache of proxy server and then will be provided by the server.

Client side :



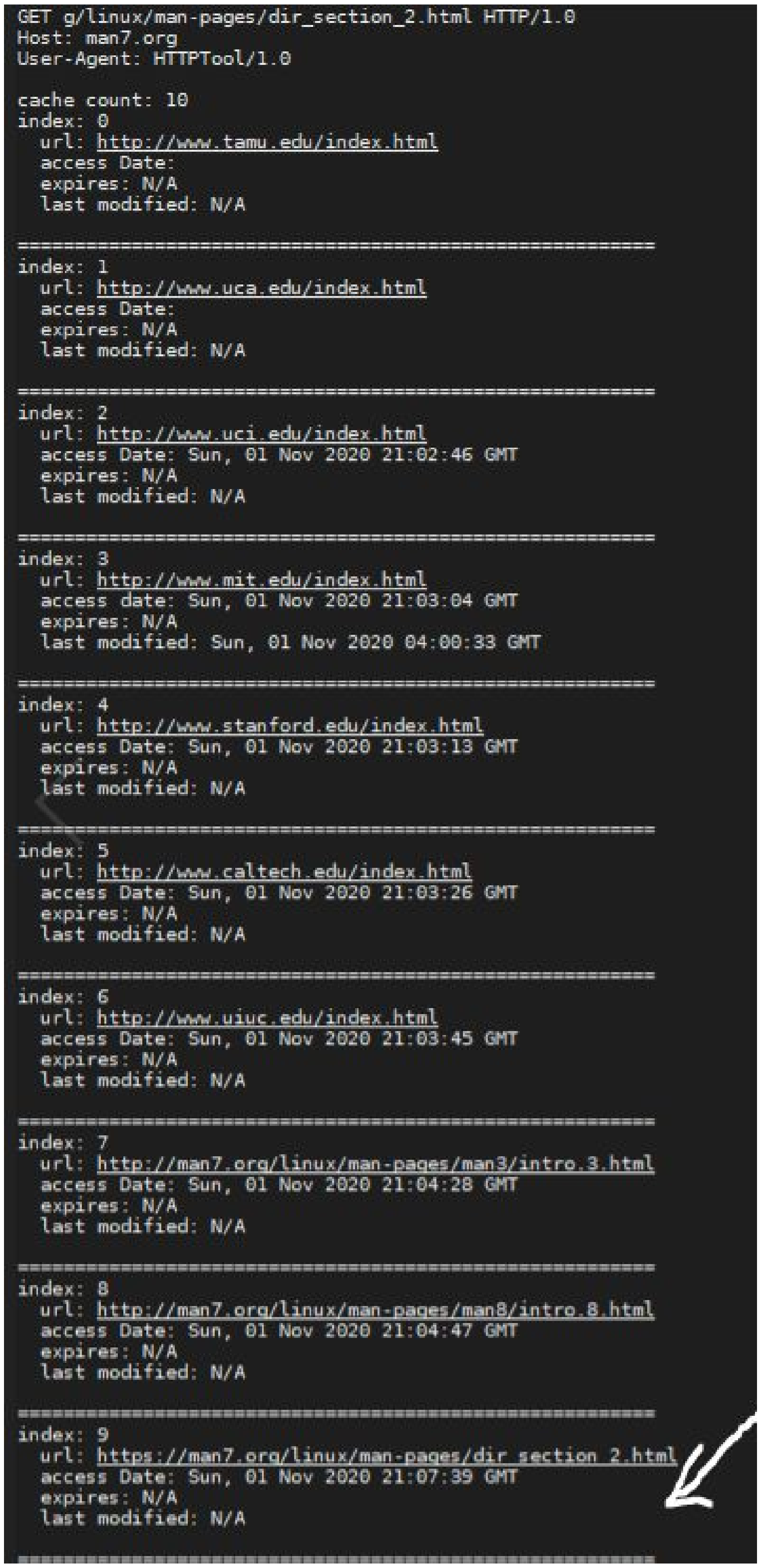
```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:03:45 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/linux/man-pages/man3/intro.3.html
Request sent to proxy server:
GET http://man7.org/linux/man-pages/man3/intro.3.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:04:28 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/linux/man-pages/man8/intro.8.html
Request sent to proxy server:
GET http://man7.org/linux/man-pages/man8/intro.8.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:04:47 11/01/20)
:: ./client 127.0.0.1 8000 https://man7.org/linux/man-pages/dir_section_2.html
Request sent to proxy server:
GET https://man7.org/linux/man-pages/dir_section_2.html HTTP/1.0

Waiting for response.....
'400 Bad Request' received. Saving to file: man7.org
```

Server side (Pre replacement):



```
cache count: 10
index: 0
  url: http://man7.org/training/spintro_ndc/index.html
  access Date: Sun, 01 Nov 2020 21:02:13 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 1
  url: http://www.tamu.edu/index.html
  access Date:
  expires: N/A
  last modified: N/A

========================================================
index: 2
  url: http://www.uca.edu/index.html
  access Date:
  expires: N/A
  last modified: N/A

========================================================
index: 3
  url: http://www.uci.edu/index.html
  access Date: Sun, 01 Nov 2020 21:02:46 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 4
  url: http://www.mit.edu/index.html
  access date: Sun, 01 Nov 2020 21:03:04 GMT
  expires: N/A
  last modified: Sun, 01 Nov 2020 04:00:33 GMT

========================================================
index: 5
  url: http://www.stanford.edu/index.html
  access Date: Sun, 01 Nov 2020 21:03:13 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 6
  url: http://www.caltech.edu/index.html
  access Date: Sun, 01 Nov 2020 21:03:26 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 7
  url: http://www.uiuc.edu/index.html
  access Date: Sun, 01 Nov 2020 21:03:45 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 8
  url: http://man7.org/linux/man-pages/man3/intro.3.html
  access Date: Sun, 01 Nov 2020 21:04:28 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 9
  url: http://man7.org/linux/man-pages/man8/intro.8.html
  access Date: Sun, 01 Nov 2020 21:04:47 GMT
  expires: N/A
  last modified: N/A

========================================================
SERVER: Request retrieved: From CLIENT:
GET https://man7.org/linux/man-pages/dir_section_2.html HTTP/1.0
SERVER: Successfully connected to web server 26
```

Server side : After LRU



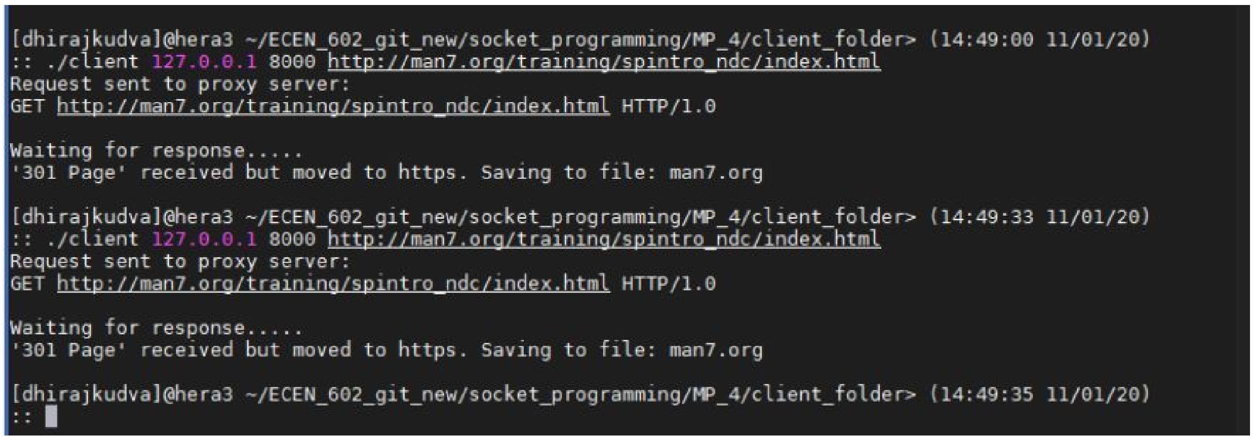
```
GET g/linux/man-pages/dir_section_2.html HTTP/1.0
Host: man7.org
User-Agent: HTTPTool/1.0

cache count: 10
index: 0
  url: http://www.tamu.edu/index.html
  access Date:
  expires: N/A
  last modified: N/A

========================================================
index: 1
  url: http://www.uca.edu/index.html
  access Date:
  expires: N/A
  last modified: N/A

========================================================
index: 2
  url: http://www.uci.edu/index.html
  access Date: Sun, 01 Nov 2020 21:02:46 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 3
  url: http://www.mit.edu/index.html
  access date: Sun, 01 Nov 2020 21:03:04 GMT
  expires: N/A
  last modified: Sun, 01 Nov 2020 04:00:33 GMT

========================================================
index: 4
  url: http://www.stanford.edu/index.html
  access Date: Sun, 01 Nov 2020 21:03:13 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 5
  url: http://www.caltech.edu/index.html
  access Date: Sun, 01 Nov 2020 21:03:26 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 6
  url: http://www.uiuc.edu/index.html
  access Date: Sun, 01 Nov 2020 21:03:45 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 7
  url: http://man7.org/linux/man-pages/man3/intro.3.html
  access Date: Sun, 01 Nov 2020 21:04:28 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 8
  url: http://man7.org/linux/man-pages/man8/intro.8.html
  access Date: Sun, 01 Nov 2020 21:04:47 GMT
  expires: N/A
  last modified: N/A

========================================================
index: 9
  url: https://man7.org/linux/man-pages/dir_section_2.html
  access Date: Sun, 01 Nov 2020 21:07:39 GMT
  expires: N/A
  last modified: N/A

========================================================
```

It can be seen that man7.org at index 0 was removed from server cache and the place was taken by file dir_section_2.html (marked as index 9)

4. a stale Expires header in the cache is accessed, the cache entry is replaced with a fresh copy, and the fresh data is delivered to the requester

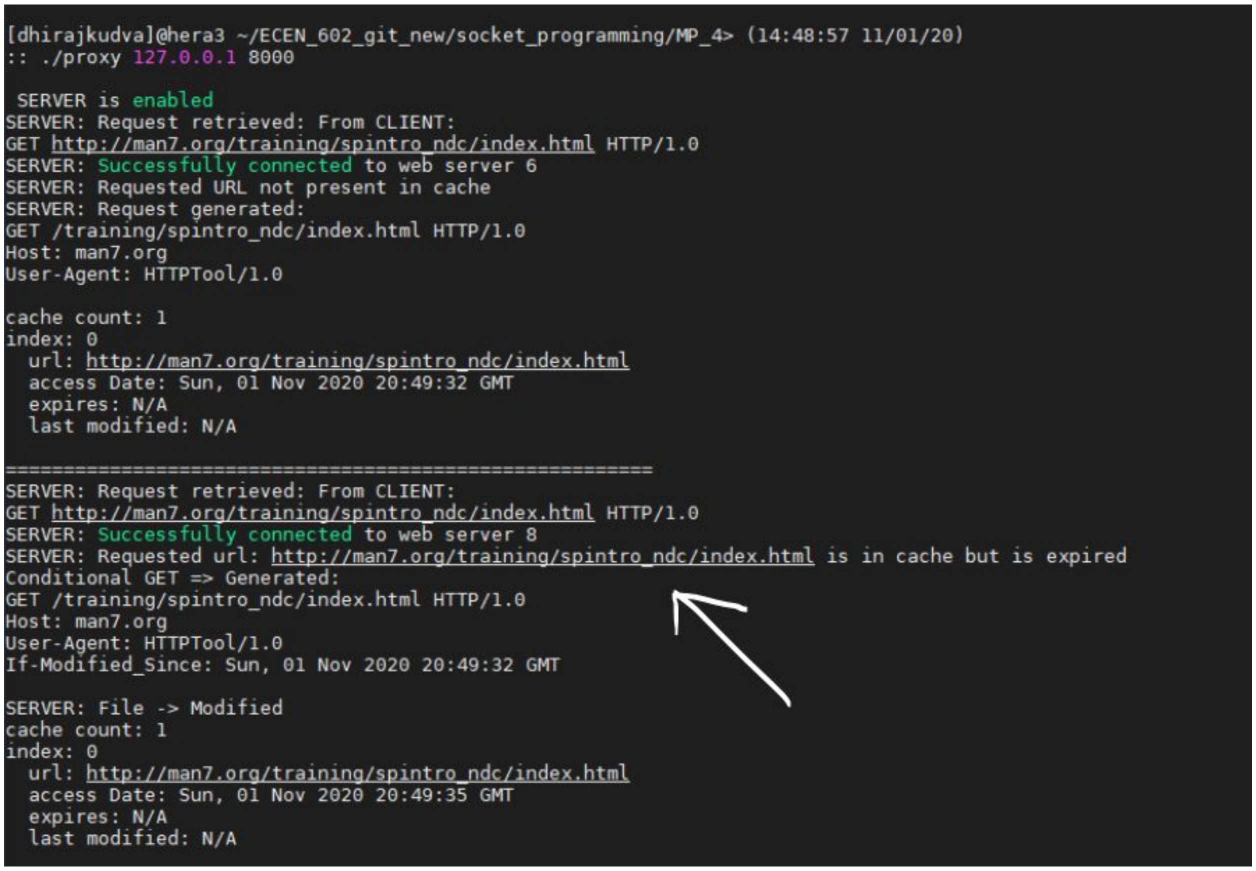
In this testcase, an expired entry in cache is updated by replacing it with updated copy from the web server. This can be clearly seen from the below snapshot. It can be seen that as there is no Expires header, it is considered as a stale and retransmitted with a fresh copy of data.

Client output:



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:00 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/training/spintro_ndc/index.html
Request sent to proxy server:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:33 11/01/20)
:: ./client 127.0.0.1 8000 http://man7.org/training/spintro_ndc/index.html
Request sent to proxy server:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: man7.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (14:49:35 11/01/20)
:: 
```

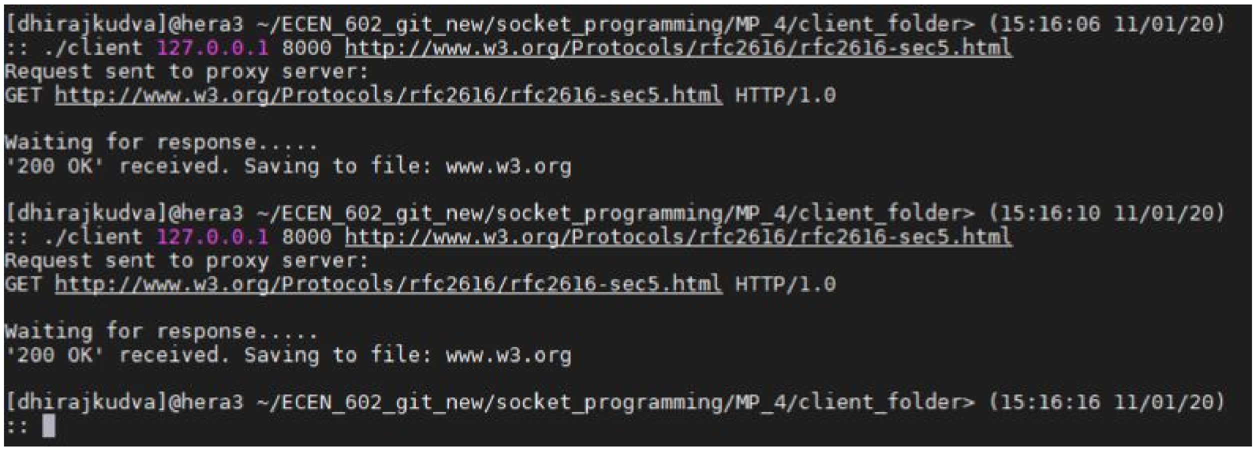
Server output:



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (14:48:57 11/01/20)
:: ./proxy 127.0.0.1 8000

 SERVER is enabled
SERVER: Request retrieved: From CLIENT:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0
SERVER: Successfully connected to web server 6
SERVER: Requested URL not present in cache
SERVER: Request generated:
GET /training/spintro_ndc/index.html HTTP/1.0
Host: man7.org
User-Agent: HTTPTool/1.0

cache count: 1
index: 0
  url: http://man7.org/training/spintro_ndc/index.html
  access Date: Sun, 01 Nov 2020 20:49:32 GMT
  expires: N/A
  last modified: N/A

========================================================
SERVER: Request retrieved: From CLIENT:
GET http://man7.org/training/spintro_ndc/index.html HTTP/1.0
SERVER: Successfully connected to web server 8
SERVER: Requested url: http://man7.org/training/spintro_ndc/index.html is in cache but is expired
Conditional GET => Generated:
GET /training/spintro_ndc/index.html HTTP/1.0
Host: man7.org
User-Agent: HTTPTool/1.0
If-Modified_Since: Sun, 01 Nov 2020 20:49:32 GMT

SERVER: File -> Modified
cache count: 1
index: 0
  url: http://man7.org/training/spintro_ndc/index.html
  access Date: Sun, 01 Nov 2020 20:49:35 GMT
  expires: N/A
  last modified: N/A
```

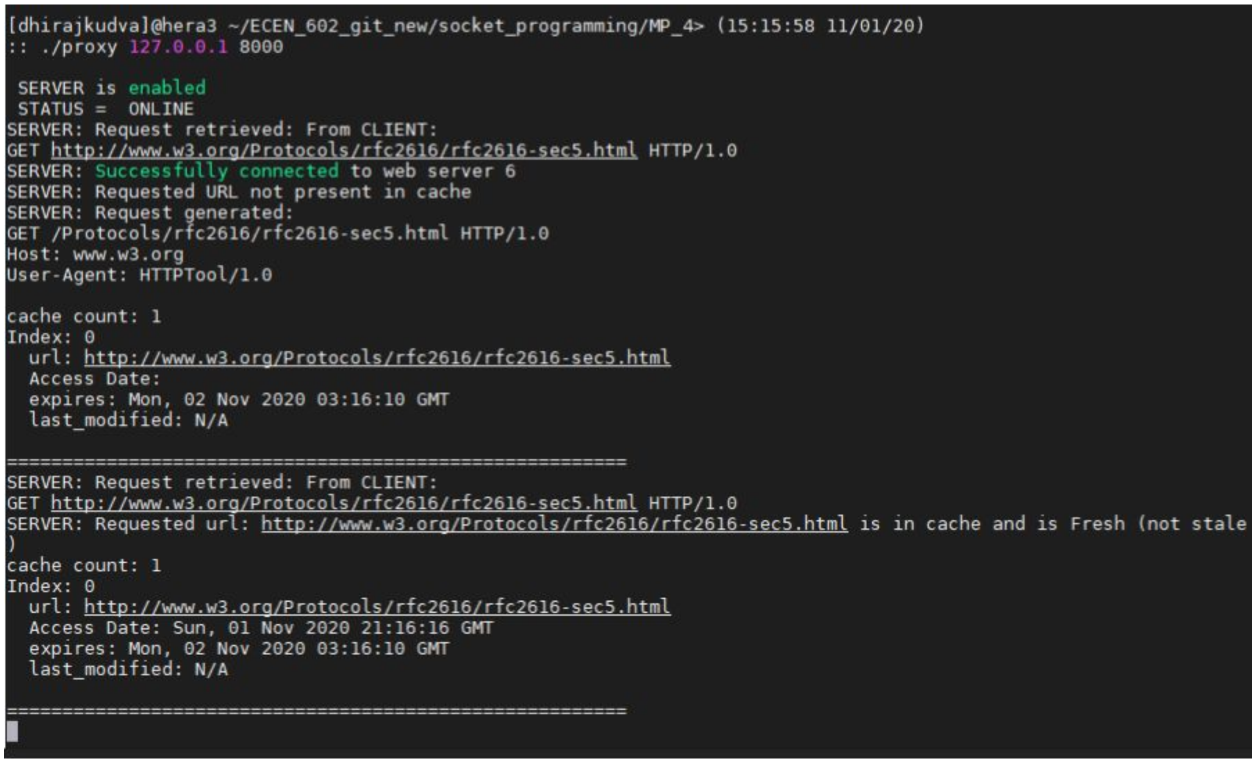
If a data which has a future expiry header (that means it is fresh) is requested by client, then the server will send it from its own cache without any update from the web server.

Client side:



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:16:06 11/01/20)
:: ./client 127.0.0.1 8000 http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
Request sent to proxy server:
GET http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html HTTP/1.0

Waiting for response.....
'200 OK' received. Saving to file: www.w3.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:16:10 11/01/20)
:: ./client 127.0.0.1 8000 http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
Request sent to proxy server:
GET http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html HTTP/1.0

Waiting for response.....
'200 OK' received. Saving to file: www.w3.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:16:16 11/01/20)
::
```

Server side:

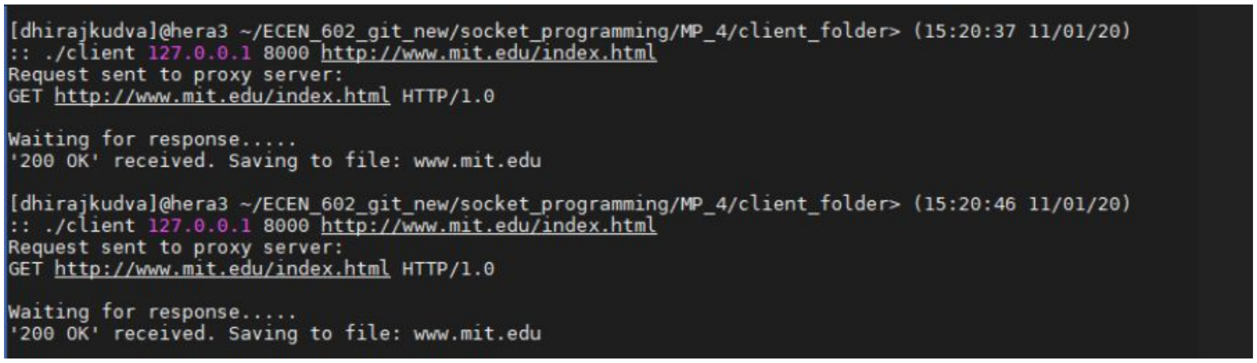


```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:15:58 11/01/20)
:: ./proxy 127.0.0.1 8000

 SERVER is enabled
 STATUS =  ONLINE
SERVER: Request retrieved: From CLIENT:
GET http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html HTTP/1.0
SERVER: Successfully connected to web server 6
SERVER: Requested URL not present in cache
SERVER: Request generated:
GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.0
Host: www.w3.org
User-Agent: HTTPTool/1.0

cache count: 1
Index: 0
  url: http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
  Access Date:
  expires: Mon, 02 Nov 2020 03:16:10 GMT
  last_modified: N/A

========================================================
SERVER: Request retrieved: From CLIENT:
GET http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html HTTP/1.0
SERVER: Requested url: http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html is in cache and is Fresh (not stale
)
cache count: 1
Index: 0
  url: http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
  Access Date: Sun, 01 Nov 2020 21:16:16 GMT
  expires: Mon, 02 Nov 2020 03:16:10 GMT
  last_modified: N/A

========================================================
```

5. a stale entry in the cache without an Expires header is determined based on the last Web server access time and last modification time, the stale cache entry is replaced with fresh data, and the fresh data is delivered to the requester ->[BONUS IMPLEMENTED]

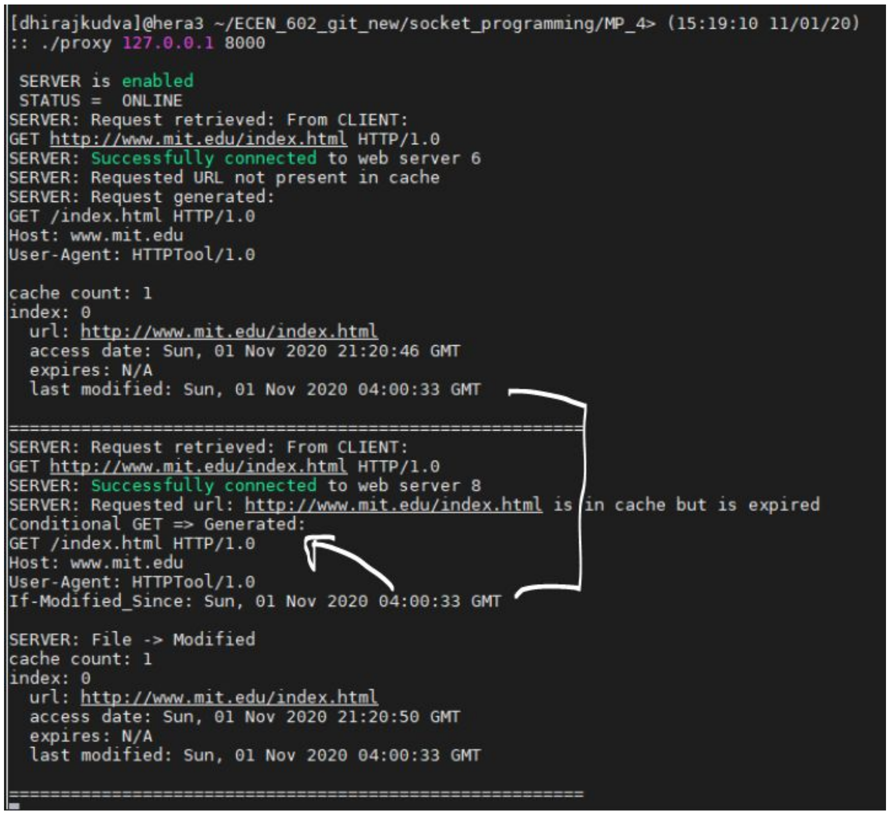
In this testcase, the entry in the cache would be without an expiry header and hence will be considered as stale based on the last access time. This entry is then replaced again with a new entry.
BONUS: The proxy server issues a conditional GET to the webserver using if modified since. As there is no expiry date, “if modified since” will take the header from the last modified.

Client :



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:20:37 11/01/20)
:: ./client 127.0.0.1 8000 http://www.mit.edu/index.html
Request sent to proxy server:
GET http://www.mit.edu/index.html HTTP/1.0

Waiting for response.....
'200 OK' received. Saving to file: www.mit.edu

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:20:46 11/01/20)
:: ./client 127.0.0.1 8000 http://www.mit.edu/index.html
Request sent to proxy server:
GET http://www.mit.edu/index.html HTTP/1.0

Waiting for response.....
'200 OK' received. Saving to file: www.mit.edu
```

Server:

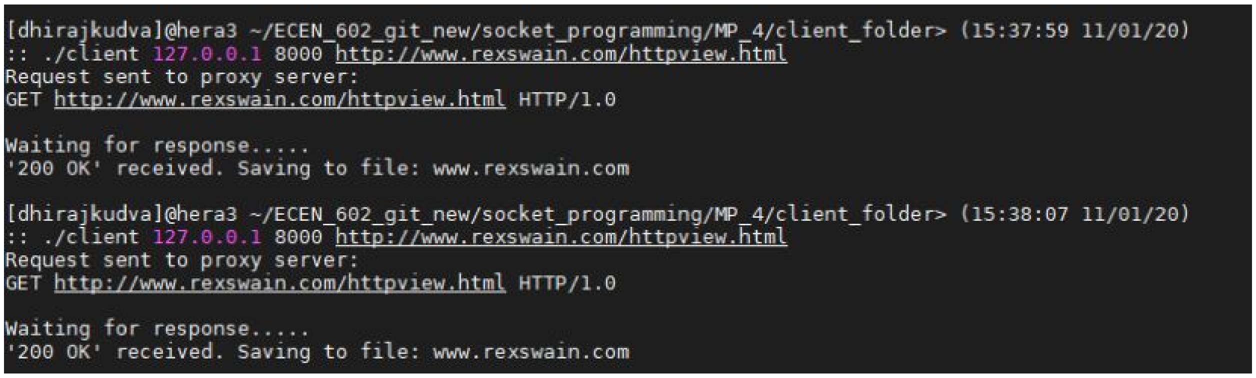


```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:19:10 11/01/20)
:: ./proxy 127.0.0.1 8000

 SERVER is enabled
 STATUS =  ONLINE
SERVER: Request retrieved: From CLIENT:
GET http://www.mit.edu/index.html HTTP/1.0
SERVER: Successfully connected to web server 6
SERVER: Requested URL not present in cache
SERVER: Request generated:
GET /index.html HTTP/1.0
Host: www.mit.edu
User-Agent: HTTPTool/1.0

cache count: 1
index: 0
  url: http://www.mit.edu/index.html
  access date: Sun, 01 Nov 2020 21:20:46 GMT
  expires: N/A
  last modified: Sun, 01 Nov 2020 04:00:33 GMT

=========================================================
SERVER: Request retrieved: From CLIENT:
GET http://www.mit.edu/index.html HTTP/1.0
SERVER: Successfully connected to web server 8
SERVER: Requested url: http://www.mit.edu/index.html is in cache but is expired
Conditional GET => Generated:
GET /index.html HTTP/1.0
Host: www.mit.edu
User-Agent: HTTPTool/1.0
If-Modified_Since: Sun, 01 Nov 2020 04:00:33 GMT

SERVER: File -> Modified
cache count: 1
index: 0
  url: http://www.mit.edu/index.html
  access date: Sun, 01 Nov 2020 21:20:50 GMT
  expires: N/A
  last modified: Sun, 01 Nov 2020 04:00:33 GMT

=========================================================
```

6. a cache entry without an Expires header that has been previously accessed from the Web server in the last 24 hours and was last modified more than one month ago is returned to the requester;

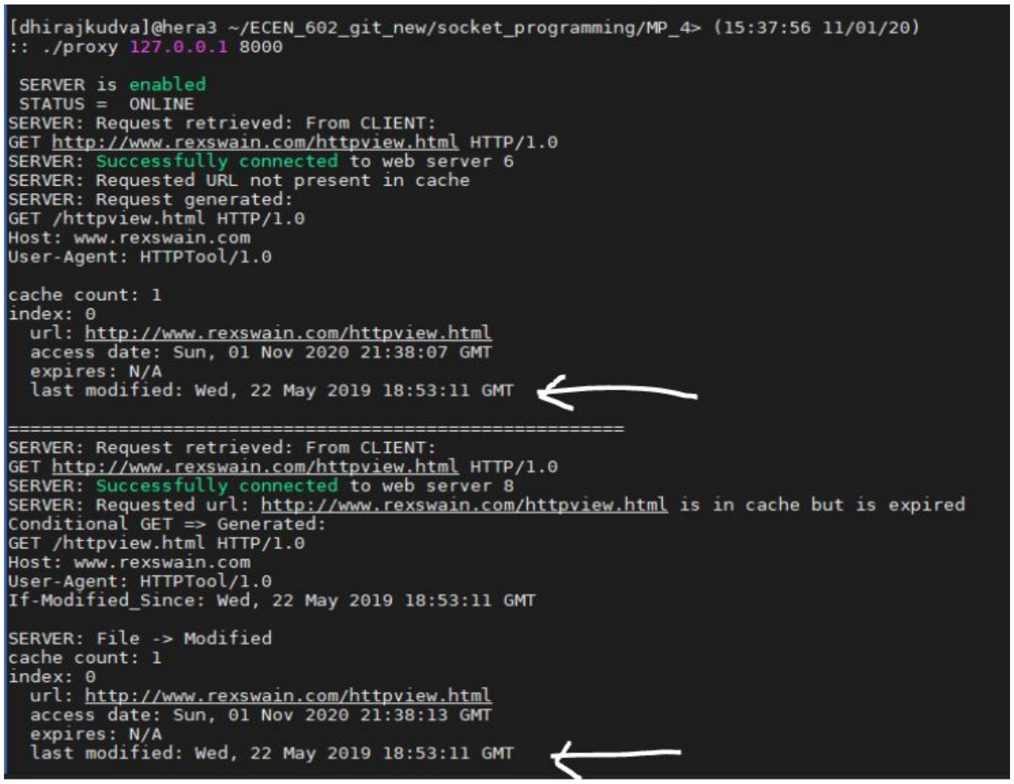
This is similar to previous case, just that the requested page was modified more than a month ago.
BONUS: Implemented the conditional GET and if-modified-since in this testcase.

Client :



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:37:59 11/01/20)
:: ./client 127.0.0.1 8000 http://www.rexswain.com/httpview.html
Request sent to proxy server:
GET http://www.rexswain.com/httpview.html HTTP/1.0

Waiting for response.....
'200 OK' received. Saving to file: www.rexswain.com

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client_folder> (15:38:07 11/01/20)
:: ./client 127.0.0.1 8000 http://www.rexswain.com/httpview.html
Request sent to proxy server:
GET http://www.rexswain.com/httpview.html HTTP/1.0

Waiting for response.....
'200 OK' received. Saving to file: www.rexswain.com
```

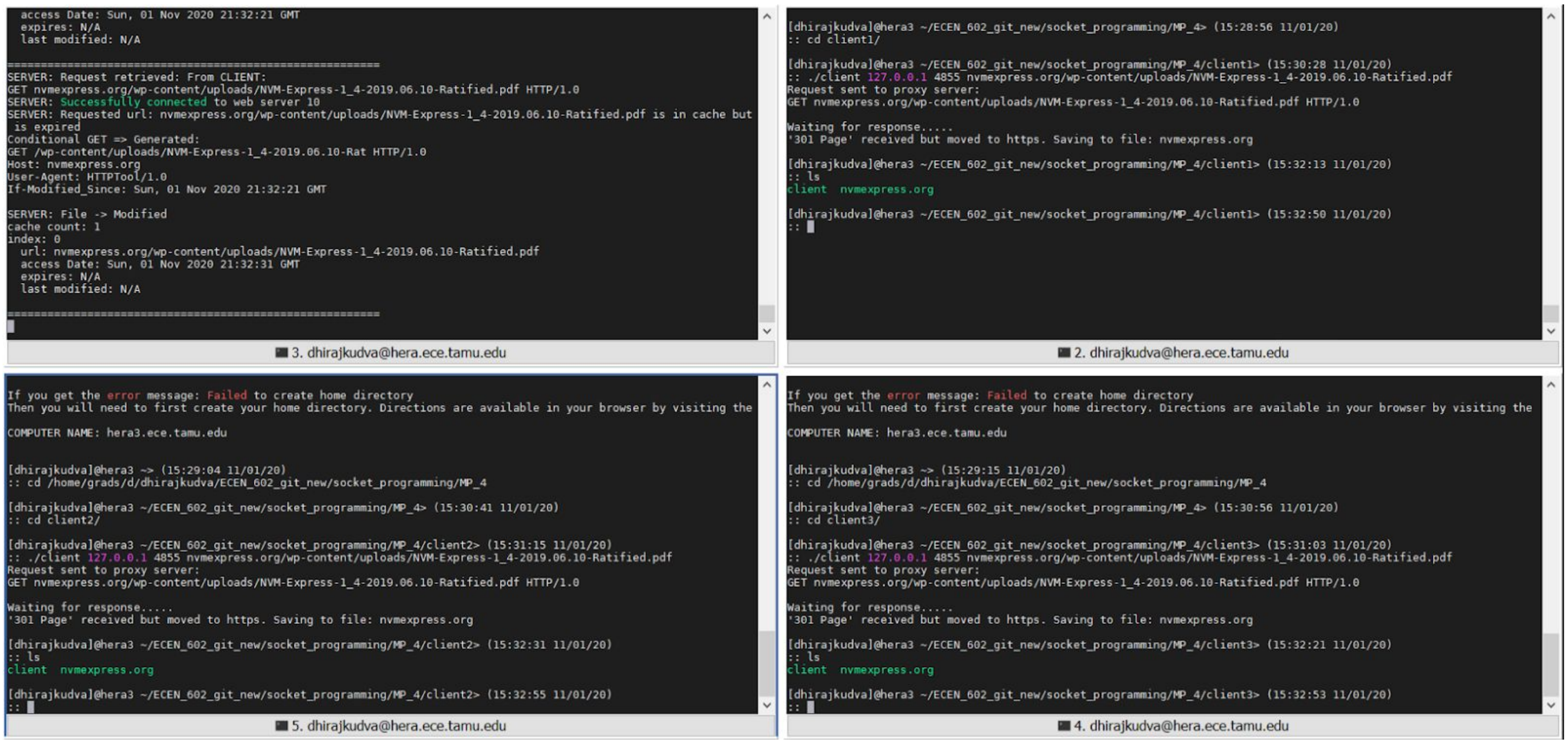
Server side:
The below snapshot highlights that the accessed page was modified more than one year ago.



```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:37:56 11/01/20)
:: ./proxy 127.0.0.1 8000

 SERVER is enabled
 STATUS =  ONLINE
SERVER: Request retrieved: From CLIENT:
GET http://www.rexswain.com/httpview.html HTTP/1.0
SERVER: Successfully connected to web server 6
SERVER: Requested URL not present in cache
SERVER: Request generated:
GET /httpview.html HTTP/1.0
Host: www.rexswain.com
User-Agent: HTTPTool/1.0

cache count: 1
index: 0
  url: http://www.rexswain.com/httpview.html
  access date: Sun, 01 Nov 2020 21:38:07 GMT
  expires: N/A
  last modified: Wed, 22 May 2019 18:53:11 GMT
=========================================================
SERVER: Request retrieved: From CLIENT:
GET http://www.rexswain.com/httpview.html HTTP/1.0
SERVER: Successfully connected to web server 8
SERVER: Requested url: http://www.rexswain.com/httpview.html is in cache but is expired
Conditional GET => Generated:
GET /httpview.html HTTP/1.0
Host: www.rexswain.com
User-Agent: HTTPTool/1.0
If-Modified_Since: Wed, 22 May 2019 18:53:11 GMT

SERVER: File -> Modified
cache count: 1
index: 0
  url: http://www.rexswain.com/httpview.html
  access date: Sun, 01 Nov 2020 21:38:13 GMT
  expires: N/A
  last modified: Wed, 22 May 2019 18:53:11 GMT
```

7. three clients can simultaneously access the proxy server and get the correct data (you will need to access large documents to show this)

In this testcase, three clients simultaneously accesses the proxy server and then receive the required file from the server. In the below snapshot, the top left one is the server and the rest three are the clients that request for their respective files from the proxy server. The proxy server sends the data to each of the clients based on their request. The clients are called from different subdirectories like client_1, client_2 and client_3. **<u>For this, the client object file needs to be manually copied in all three empty folders.</u>** The success in transmission can also be seen by checking the requested file in these subdirectories (folders).



```
  access Date: Sun, 01 Nov 2020 21:32:21 GMT
  expires: N/A
  last modified: N/A

========================================================
SERVER: Request retrieved: From CLIENT:
GET nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf HTTP/1.0
SERVER: Successfully connected to web server 10
SERVER: Requested url: nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf is in cache but
 is expired
Conditional GET => Generated:
GET /wp-content/uploads/NVM-Express-1_4-2019.06.10-Rat HTTP/1.0
Host: nvmexpress.org
User-Agent: HTTPTool/1.0
If-Modified_Since: Sun, 01 Nov 2020 21:32:21 GMT

SERVER: File -> Modified
cache count: 1
index: 0
  url: nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf
  access Date: Sun, 01 Nov 2020 21:32:31 GMT
  expires: N/A
  last modified: N/A

========================================================
```

3. dhirajkudva@hera.ece.tamu.edu

```
[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:28:56 11/01/20)
:: cd client1/

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client1> (15:30:28 11/01/20)
:: ./client 127.0.0.1 4855 nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf
Request sent to proxy server:
GET nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: nvmexpress.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client1> (15:32:13 11/01/20)
:: ls
client  nvmexpress.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client1> (15:32:50 11/01/20)
::
```

2. dhirajkudva@hera.ece.tamu.edu

```
If you get the error message: Failed to create home directory
Then you will need to first create your home directory. Directions are available in your browser by visiting the

COMPUTER NAME: hera3.ece.tamu.edu


[dhirajkudva]@hera3 ~> (15:29:04 11/01/20)
:: cd /home/grads/d/dhirajkudva/ECEN_602_git_new/socket_programming/MP_4

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:30:41 11/01/20)
:: cd client2/

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client2> (15:31:15 11/01/20)
:: ./client 127.0.0.1 4855 nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf
Request sent to proxy server:
GET nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: nvmexpress.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client2> (15:32:31 11/01/20)
:: ls
client  nvmexpress.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client2> (15:32:55 11/01/20)
::
```

5. dhirajkudva@hera.ece.tamu.edu

```
If you get the error message: Failed to create home directory
Then you will need to first create your home directory. Directions are available in your browser by visiting the

COMPUTER NAME: hera3.ece.tamu.edu


[dhirajkudva]@hera3 ~> (15:29:15 11/01/20)
:: cd /home/grads/d/dhirajkudva/ECEN_602_git_new/socket_programming/MP_4

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4> (15:30:56 11/01/20)
:: cd client3/

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client3> (15:31:03 11/01/20)
:: ./client 127.0.0.1 4855 nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf
Request sent to proxy server:
GET nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf HTTP/1.0

Waiting for response.....
'301 Page' received but moved to https. Saving to file: nvmexpress.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client3> (15:32:21 11/01/20)
:: ls
client  nvmexpress.org

[dhirajkudva]@hera3 ~/ECEN_602_git_new/socket_programming/MP_4/client3> (15:32:53 11/01/20)
::
```

4. dhirajkudva@hera.ece.tamu.edu

8. BONUS FEATURE : Implemented in testcase 5 and 6 and highlighted in the snapshot as well as in the description.

CODE:
Proxy.c

```
/********* HTTP proxy code***************/
/*Authors: Sanket Agarwal and Dhiraj Kudva
Organization: Texas A&M University
Description: Proxy server. Manages the url from the client.*/

//including the library files. these are standard library files.
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
#include <dirent.h>
#include <pthread.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ELEMENT_NOT_IN_CACHE -1
#define MAX_CACHE_ENTRY 10
#define MAX_LEN 1024
#define STALE_ENTRY 1

struct cache_content {
	char url[256];
	char last_modified[50];
	char access_date[50];
	char expires[50];
	char *body;
};

static const struct cache_content reset_cache_data;
int cache_entries = 0;

struct cache_content http_cache_proxy[MAX_CACHE_ENTRY];

void cache_table_update(char *url, char *buf, int flag, int x)
{
	int j;

	// New Entry
	if (flag == 1) {
		if (cache_entries==MAX_CACHE_ENTRY) {
			http_cache_proxy[0] = reset_cache_data;
			for (j = 0; j < MAX_CACHE_ENTRY; j++) {
				if (j+1 != MAX_CACHE_ENTRY)
					http_cache_proxy[j] = http_cache_proxy[j+1];
```

```
                                        else {
                                                // printf("checkpoint #1\n");
                                                bzero(&http_cache_proxy[j], sizeof(struct
cache_content));
                                                memcpy(http_cache_proxy[j].url, url, 256);
                                                http_cache_proxy[j].body = (char *)malloc(strlen(buf)
* sizeof(char));
                                                memcpy(http_cache_proxy[j].body, buf, strlen(buf));
                                                parser_server("expires:", buf,
http_cache_proxy[j].expires);
                                                parser_server("Last-Modified:", buf,
http_cache_proxy[j].last_modified);
                                                parser_server("Date:", buf,
http_cache_proxy[j].access_date);
                                        }
                                }
                        } else {
                                // printf("checkpoint #2\n");
                                http_cache_proxy[cache_entries] = reset_cache_data;
                                memcpy(http_cache_proxy[cache_entries].url, url, 256);
                                parser_server("expires:", buf,
http_cache_proxy[cache_entries].expires);
                                parser_server("Last-Modified:", buf,
http_cache_proxy[cache_entries].last_modified);
                                parser_server("Date:", buf,
http_cache_proxy[cache_entries].access_date);
                                http_cache_proxy[cache_entries].body = (char *)malloc(strlen(buf)
* sizeof(char));
                                memcpy(http_cache_proxy[cache_entries].body, buf, strlen(buf));
                                cache_entries++;
                        }
                } else {
                        struct cache_content tmp;
                        struct tm tmp_t;
                        time_t nw = time(NULL);
                        const char* op_tmp = "%a, %d %b %Y %H:%M:%S GMT";
                        bzero(&tmp, sizeof(struct cache_content));
                        tmp = http_cache_proxy[x];
                        for (j = x; j < cache_entries; j++) {
                                if (j == cache_entries - 1)
                                        break;
                                http_cache_proxy[j] = http_cache_proxy[j+1];
                        }
                        http_cache_proxy[cache_entries -1] = tmp;
                        tmp_t = *gmtime(&nw);
                        strftime(http_cache_proxy[cache_entries - 1].access_date, 50, op_tmp,
&tmp_t);
                }
}

void display_cache_entries()
{
        int t;

        if (cache_entries == 0)
```

```
			fprintf(stdout, "CACHE IS EMPTY \n");
		else {
			fprintf(stdout, "cache count: %d\n", cache_entries);
			for (t = 0; t < cache_entries; t++) {
				if (strcmp(http_cache_proxy[t].expires, "") != 0 &&
				    strcmp(http_cache_proxy[t].last_modified, "") != 0)
					fprintf(stdout, "index: %d  \n  url: %s  \n  access date:
%s  \n  expires: %s  \n  last modified: %s\n\n",
						t, http_cache_proxy[t].url,
http_cache_proxy[t].access_date, http_cache_proxy[t].expires,
						http_cache_proxy[t].last_modified);
				else if (strcmp(http_cache_proxy[t].expires, "") == 0 &&
					 strcmp(http_cache_proxy[t].last_modified, "") == 0)
					fprintf(stdout, "index: %d  \n  url: %s  \n  access Date:
%s  \n  expires: N/A  \n  last modified: N/A\n\n",
						t, http_cache_proxy[t].url,
http_cache_proxy[t].access_date);
				else if (strcmp(http_cache_proxy[t].expires, "") == 0)
					fprintf(stdout, "index: %d  \n  url: %s  \n  access date:
%s  \n  expires: N/A  \n  last modified: %s\n\n",
						t, http_cache_proxy[t].url,
http_cache_proxy[t].access_date, http_cache_proxy[t].last_modified);
				else if (strcmp(http_cache_proxy[t].last_modified, "") == 0)
					fprintf(stdout, "Index: %d  \n  url: %s  \n  Access Date:
%s  \n  expires: %s  \n  last_modified: N/A\n\n",
						t, http_cache_proxy[t].url,
http_cache_proxy[t].access_date, http_cache_proxy[t].expires);
				fprintf(stdout,
"=========================================================\n" );
			}
		}
}


int is_it_fresh(int cache_pointer)
{
		struct tm tmp_t, expires;
		time_t nw  = time(NULL);
		tmp_t = *gmtime(&nw);

		if (strcmp(http_cache_proxy[cache_pointer].expires, "") != 0) {
			strptime(http_cache_proxy[cache_pointer].expires, "%a, %d %b %Y %H:%M:%S
%Z", &expires);
			time_t EXP = mktime(&expires);
			time_t NOW = mktime(&tmp_t);
			if (difftime (NOW, EXP) < 0)
				return STALE_ENTRY;
			else
				return -1;
		} else
			return -1;
}

int element_of_cache_table(char *url)
{
	int b;

	for (b = 0; b < MAX_CACHE_ENTRY; b++) {
		if (strcmp(http_cache_proxy[b].url, url)==0)
			return b;
	}

	return ELEMENT_NOT_IN_CACHE;
}

int socket_web(char *host)
{
	struct addrinfo dynamic_address, *ai, *p;
	int ret_val, server_socket_descriptor;

	bzero(&dynamic_address, sizeof(dynamic_address));
	dynamic_address.ai_family = AF_INET;
	dynamic_address.ai_socktype = SOCK_STREAM;
	ret_val = getaddrinfo(host, "http", &dynamic_address, &ai);
	if (ret_val != 0) {
		fprintf(stderr, "SERVER: %s\n", gai_strerror(ret_val));
		exit(1);
	}
	for (p = ai; p != NULL; p = p->ai_next) {
		server_socket_descriptor = socket(p->ai_family, p->ai_socktype,
p->ai_protocol);
		if (server_socket_descriptor >= 0 && (connect(server_socket_descriptor,
p->ai_addr, p->ai_addrlen) >= 0))
			break;
	}
	if (p == NULL)
		server_socket_descriptor = -1;

	freeaddrinfo(ai);
	return server_socket_descriptor;
}


int http_proxy_server(int client_fd)
{
	// printf("First checkpoint");
	int server_socket_descriptor, return_read, cache_table_element,
extract_read_return, port = 80;
	char *message_pointer;
	char client_message[MAX_LEN] = {0};
	char *read_extractor, *forward_client;
	char name_of_host[64], path_name[256], url[256], conditional_get_message[256],
url_parse[256];
	char http_method[8];
	char http_protocol[16];

	message_pointer = (char *)malloc(MAX_LEN * sizeof(char));
	return_read = read(client_fd, message_pointer, MAX_LEN);
	printf("SERVER: Request retrieved: From CLIENT: \n%s", message_pointer);
	if (return_read < 0)
```

```
              system_error ("SERVER: ERROR:  In extracting message request from
client");
       sscanf(message_pointer, "%s %s %s", http_method, url, http_protocol);
       if ((cache_table_element = element_of_cache_table (url)) !=
ELEMENT_NOT_IN_CACHE
            && (is_it_fresh(cache_table_element) == STALE_ENTRY)) {
              fprintf (stdout, "SERVER: Requested url: %s is in cache and is Fresh (not
stale)\n", url);
              cache_table_update(url, NULL, 0, cache_table_element);
              forward_client = (char
*)malloc(strlen(http_cache_proxy[cache_table_element].body) * sizeof(char));
              memcpy(forward_client, http_cache_proxy[cache_table_element].body,
strlen(http_cache_proxy[cache_table_element].body));
       } else { // In cache, but stale
              bzero(name_of_host, 64);
              bzero(path_name, 256);
              memcpy(&url_parse[0], &url[0], 256);
              parse_client(url_parse, name_of_host, &port, path_name);
              if ((server_socket_descriptor = socket_web (name_of_host)) == -1)
                     system_error ("SERVER: ERROR: In  connecting with web server \n");
              fprintf(stdout, "SERVER: Successfully connected to web server %d\n",
server_socket_descriptor);
              if (cache_table_element != ELEMENT_NOT_IN_CACHE) { // cache entry expired
                     fprintf(stdout, "SERVER: Requested url: %s is in cache but is
expired\n", url);
                     if (strcmp(http_cache_proxy[cache_table_element].expires, "") != 0
&&
                         strcmp(http_cache_proxy[cache_table_element].last_modified,
"") != 0)
                            snprintf(conditional_get_message, MAX_LEN,
                                   "%s %s %s\r\nHost: %s\r\nUser-Agent:
HTTPTool/1.0\r\nIf-Modified_Since: %s\r\n\r\n",
                                  http_method, path_name, http_protocol, name_of_host,
http_cache_proxy[cache_table_element].expires);
                     else if (strcmp(http_cache_proxy[cache_table_element].expires, "")
== 0 &&

strcmp(http_cache_proxy[cache_table_element].last_modified, "") == 0)
                            snprintf(conditional_get_message, MAX_LEN,
                                   "%s %s %s\r\nHost: %s\r\nUser-Agent:
HTTPTool/1.0\r\nIf-Modified_Since: %s\r\n\r\n",
                                   http_method, path_name, http_protocol, name_of_host,
http_cache_proxy[cache_table_element].access_date);
                     else if (strcmp(http_cache_proxy[cache_table_element].expires, "")
== 0)
                            snprintf(conditional_get_message, MAX_LEN,
                                   "%s %s %s\r\nHost: %s\r\nUser-Agent:
HTTPTool/1.0\r\nIf-Modified_Since: %s\r\n\r\n",
                                   http_method, path_name, http_protocol, name_of_host,
http_cache_proxy[cache_table_element].last_modified);
                     else if
(strcmp(http_cache_proxy[cache_table_element].last_modified, "") == 0)
                            snprintf(conditional_get_message, MAX_LEN,
                                   "%s %s %s\r\nHost: %s\r\nUser-Agent:
HTTPTool/1.0\r\nIf-Modified_Since: %s\r\n\r\n",
                                  http_method, path_name, http_protocol, name_of_host,
http_cache_proxy[cache_table_element].expires);
                    fprintf(stdout, "Conditional GET => Generated: \n%s",
conditional_get_message);
                    write(server_socket_descriptor, conditional_get_message, MAX_LEN);
                    read_extractor = (char *)malloc(100000 * sizeof(char));
                    extract_read_return =
extract_read_from_main(server_socket_descriptor, read_extractor);
                    forward_client = (char *)malloc(strlen(read_extractor) *
sizeof(char));
                    if (strstr(read_extractor, "304 : Not Modified") != NULL) {
                          fprintf(stdout, "'304 :Not Modified' received. Sending file
in cache\n");
                          memcpy(forward_client,
http_cache_proxy[cache_table_element].body,
strlen(http_cache_proxy[cache_table_element].body));
                          cache_table_update(url, NULL, 0, cache_table_element);
                    } else {
                          fprintf(stdout, "SERVER: File -> Modified\n");
                          memcpy(forward_client, read_extractor,
strlen(read_extractor));
                          cache_table_update(url, NULL, 0, cache_table_element);
                          http_cache_proxy[--cache_entries] = reset_cache_data;
                          cache_table_update(url, read_extractor, 1, 0);
                    }
              } else { // document is not cached
                    fprintf(stdout, "SERVER: Requested URL not present in cache\n");
                    bzero(client_message, MAX_LEN);
                    snprintf(client_message, MAX_LEN,
                           "%s %s %s\r\nHost: %s\r\nUser-Agent:
HTTPTool/1.0\r\n\r\n",
                           http_method, path_name, http_protocol, name_of_host);
                    fprintf(stdout, "SERVER: Request generated: \n%s",
client_message);
                    write(server_socket_descriptor, client_message, MAX_LEN);
                    read_extractor = (char *)malloc(100000 * sizeof(char));
                    extract_read_return =
extract_read_from_main(server_socket_descriptor, read_extractor);
                    forward_client = (char *)malloc(strlen(read_extractor) *
sizeof(char));
                    memcpy(forward_client, read_extractor, strlen(read_extractor));
                    cache_table_update(url, read_extractor, 1, 0);
              }
       }
       display_cache_entries();
       write(client_fd, forward_client, strlen(forward_client) + 1);
}

int extract_read_from_main(int file_descriptor, char *ptr)
{
       int total_count = 0, cnt = 1;
       char buffer[MAX_LEN];

       while (cnt > 0) {
             bzero(buffer, sizeof(buffer));
```

```
		cnt = read(file_descriptor, buffer, MAX_LEN);
		if (cnt == 0)
			break;
		strcat(ptr, buffer);
		total_count += cnt;
		if (buffer[cnt - 1] == EOF) {
			strncpy(ptr,ptr,(strlen(ptr)-1));
			total_count--;
			break;
		}
	}

	return total_count;
}


int main(int argc, char *argv[])
{
	int socket_file_descriptor, connect_file_descriptor;
	int port_number;
	struct sockaddr_storage remote_address;
	struct sockaddr_in client_address;
	socklen_t length;
	pthread_t thread;

	if (argc != 3) {
		system_error ("USAGE: ./proxy <Server IP Address> <Port_Number>");
		return 0;
	}

	port_number = atoi(argv[2]);
	socket_file_descriptor = socket(AF_INET, SOCK_STREAM, 0);
	if (socket_file_descriptor < 0)
		system_error ("ERROR: Socket Error");
	bzero( &client_address, sizeof(client_address));
	client_address.sin_family = AF_INET;
	client_address.sin_addr.s_addr = inet_addr(argv[1]);
	client_address.sin_port = htons(port_number);
	if (bind(socket_file_descriptor, (struct sockaddr *)&client_address, 
sizeof(client_address)) < 0)
		system_error ("ERROR: Bind Error");
	if (listen(socket_file_descriptor, 10) < 0)
		system_error ("ERROR: Listen Error");
	bzero(http_cache_proxy, MAX_CACHE_ENTRY * sizeof(struct cache_content));
	length = sizeof(remote_address);
	fprintf(stdout, "\n SERVER is enabled\n STATUS =  ONLINE\n");
	while (1) {
		connect_file_descriptor = accept(socket_file_descriptor, (struct 
sockaddr*)&remote_address,&length);
		pthread_create(&thread, NULL, (void *)(&http_proxy_server), (void 
*)(intptr_t)connect_file_descriptor);
	}
}
```

Client.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <errno.h>
#include <dirent.h>
#include <pthread.h>
#include <signal.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>



int main(int argc,char *argv[])
{
	FILE *file_pointer;
	int socket_descriptor, n, port_request = 80;
	char host_name[64], request[100], path_name[256], buffer_holder[100000];
	unsigned int pt_no;
	char *p, *ptr;
	struct sockaddr_in server_address_client;

	if (argc != 4) {
		system_error ("USAGE: ./client <Server_IP_Address> <Port_Number> <URL>");
		exit(1);
	}

	pt_no = atoi(argv[2]);
	bzero(&server_address_client,sizeof server_address_client);
	server_address_client.sin_family=AF_INET;
	server_address_client.sin_port=htons(pt_no);
	if (inet_aton(argv[1], (struct in_addr
*)&server_address_client.sin_addr.s_addr) < 0)
		system_error ("ERROR: inet_aton error");

	socket_descriptor=socket(AF_INET,SOCK_STREAM,0);
	if (socket_descriptor < 0)
		system_error ("ERROR: Socket Error");

	if (connect(socket_descriptor,(struct sockaddr
*)&server_address_client,sizeof(server_address_client)) < 0)
		system_error ("ERROR: Connect Error");
	bzero(request, sizeof(request));
	sprintf(request, "GET %s HTTP/1.0\r\n", argv[3]);
	fprintf(stdout, "Request sent to proxy server: \n%s\n", request);
	if (send(socket_descriptor, request, strlen(request), 0) < 0) {
		system_error("CLIENT: Send");
		exit(2);
	}
	bzero(buffer_holder, sizeof(buffer_holder));
	parse_client(argv[3], host_name, &port_request, path_name);
	file_pointer=fopen(host_name, "w");
	fprintf(stdout, "Waiting for response.....\n");
	if (recv(socket_descriptor, buffer_holder, 100000, 0) < 0) {
		system_error("CLIENT: In receiving");
		fclose(file_pointer);
		close(socket_descriptor);
		return 1;
	}
	if((strstr(buffer_holder, "200")) != NULL)
		fprintf(stdout, "'200 OK' received. Saving to file: %s\n", host_name);
	else if ((strstr(buffer_holder, "400") != NULL))
		fprintf(stdout, "'400 Bad Request' received. Saving to file: %s\n",
host_name);
	else if ((strstr(buffer_holder, "404") != NULL))
		fprintf(stdout, "'404 Page Not Found' received. Saving to file: %s\n",
host_name);
	else if ((strstr(buffer_holder, "301") != NULL))
		fprintf(stdout, "'301 Page' received but moved to https. Saving to file:
%s\n", host_name);
	ptr = strstr(buffer_holder, "\r\n\r\n");
	fwrite(ptr + 4, 1, strlen(ptr) - 4, file_pointer);
	fclose(file_pointer);
	close(socket_descriptor);

	return 0;
}
```

Functions.c

```
/*********Common fucntions code****************/
/*Authors: Sanket Agarwal and Dhiraj Kudva
Organization: Texas A&M University
Description: functions for parsing the URL and generating system error messages.*/

//including the library files. these are standard library files.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ELEMENT_NOT_IN_CACHE -1
#define MAX_CACHE_ENTRY 10
#define MAX_LEN 1024
#define STALE 1

struct cache_content {
	char url[256];
	char last_modified[50];
	char access_date[50];
	char expires[50];
	char *body;
};
```

```
int system_error(const char* x);
void parse_client(char* url_address, char *hostname_addr, int *port_address, char
*path_address);
void parser_server(const char* header, char* buffer, char* output_file);


void parser_server(const char* header, char* buffer, char* output_file)
{
	// printf("Inside parser server\n");
	char *start_string = strstr(buffer, header);
	if (!start_string)
		return;
	char *end_string = strstr(start_string, "\r\n");
	start_string += strlen(header);
	while(*start_string == ' ')
		++start_string;
	while(*(end_string - 1) == ' ')
		--end_string;
	strncpy(output_file, start_string, end_string - start_string);
	output_file[end_string - start_string] = '\0';
}

void parse_client(char* url_address, char *hostname_addr, int *port_address, char
*path_address) {
	// printf("Inside parse client\n");
	char *token_no, *temp_host_holder, *temp_path_address, *tmp1_holder,
*tmp2_holder;
	int counter = 0;
	char string_holder[16];

	if (strstr(url_address,"http") != NULL) {
		token_no = strtok(url_address, ":");
		tmp1_holder = token_no + 7;
	} else
		tmp1_holder = url_address;
	tmp2_holder = (char *)malloc(64 * sizeof(char));
	memcpy(tmp2_holder, tmp1_holder, 64);
	if(strstr(tmp1_holder, ":") != NULL) {
		temp_host_holder = strtok(tmp1_holder, ":");
		*port_address = atoi(tmp1_holder + strlen(temp_host_holder) + 1);
		sprintf(string_holder, "%d", *port_address);
		temp_path_address = tmp1_holder + strlen(temp_host_holder) +
strlen(string_holder) + 1;
	} else {
		temp_host_holder = strtok(tmp1_holder, "/");
		*port_address = 80;
		temp_path_address = tmp2_holder + strlen(temp_host_holder);
	}
	if (strcmp(temp_path_address, "") == 0)
		strcpy(temp_path_address, "/");
	memcpy(hostname_addr, temp_host_holder, 64);
	memcpy(path_address, temp_path_address, 256);
}

int system_error(const char* x)	// Error display source code
```

```
{
	perror(x);
	fprintf(stderr, "\n");
	exit(1);
}
```

MAKEFILE

```
RM = rm -rf

all: proxy client

proxy: proxy.c functions.c
	gcc -I . -pthread proxy.c functions.c -o proxy

client: client.c functions.c
	mkdir client_folder
	gcc -I . client.c functions.c -o client_folder/client

clean:
	$(RM) client proxy
```