

---

# Proxy Server

CS425A  
Computer Networks

## Implemented Features :-

- 1) Implemented the GET method
- 2) Received data from the client and forwarded that to the server
- 3) Send data back from the server to client
- 4) Implemented the header of Connection: close
- 5) Implemented handling of multiple clients
- 6) Implemented the upper bound on the number of forks that we can make
- 7) Server Port Initialised at start up.

## Running The code

```
$ make
$./proxy <port>
$make clean
```

Sai Kishan Pampana  
13458  
pkishan@iitk.ac.in

---

## Design Choices

- **Creating the header to send the sever**

After parsing the header that we get from the client, we need to send a header to the server so far this purpose instead of using the functions that are present in the parsing code I have manually made the GET request to the server. By manual I don't mean I hard coded but instead taking the parsed input into account I have created the header to send to the server.

- **Allow the port to be decided at the startup**

For the port that we need to use in the proxy, I have given an input in the beginning of the code as a system argument, so that we can keep changing the port if any of the port is busy.

- **Receiving the data from the client**

For reading the request from the client instead of reading the whole request in one go i have place a while loop on top of the receive command and came out of the loop when i see "\r\n\r\n".

- **Realloc**

Used Realloc function in the case that the header is too big for the assigned buffer to hold.

## IMPLEMENTATION OF THE PROJECT

## Browser Used:

The browser that was used for this project is Mozilla FireFox

## Test Procedure Used:

For the testing of the project this was done in different stages. Telnet, running the python scripts, Browser testing

```
• proxy2 (proxy2)
  => Project2 proxy2 8003
    Socket Established
    Binding Socket
    I have reached this point

The buffer containing the url:
GET http://www.google.com/ HTTP/1.0

This is the buffer that I am storing the request:
GET http://www.google.com/ HTTP/1.0

Well it just is that i dont have headerThe complete reques header has been made
GET / HTTP/1.0
Host: www.google.com

... add(Header2(rm))
  => Project2 telnet 127.0.0.1 8003
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET http://www.google.com/ HTTP/1.0

HTTP/1.0 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.co.in/?gfe_rd=cr&ei=HR0V4-d5cLl8epu70gCA
Content-Length: 260
Date: Wed, 31 Aug 2016 10:14:45 GMT

<!DOCTYPE html><meta http-equiv="content-type" content="text/html; charset=UTF-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>Moved</H1>
The document has moved
<A HREF="http://www.google.co.in/?gfe_rd=cr&ei=HR0V4-d5cLl8epu70gCA">here</A>
<P>You can also <A href="#">close</A> this window by pressing the close button on your browser's toolbar.

```

For the case of the telnet we first run the proxy on port and then use telnet to call the site.

As you can see in the image in the telnet we have called the [google.com](http://google.com) and as per that the proxy has returned the corresponding data.

Now in the python script for the we have two python scripts and in one we have 4 test and in another we have 13th test. In the second python script only 11 pass since the net is not fast enough.

```
(Connect: 0, Recvbyte: 0, Length: 7, Exceptions: 0)
Total transferred: 243164 bytes
HTML transferred: 245110 bytes
Requests per second: 1.67 [mean]
Time per request: 4971.656 [ms] (mean)
Time per request: 697.165 [ms] (mean, across all concurrent requests)
Transfer rate: 2.21 [Kbytes/sec] received

Connection times (ms)
min mean(±sd) median max
Connect: 0 0 0 0
Processing: 653 6629 2406.6 976 127408
Waiting: 0 2154 6893.1 746 63955
Total: 653 6628 2406.6 777 127408

Percentage of the requests served within a certain time (ms)
50% 777
66% 859
75% 1340
80% 1720
90% 3861
95% 15860
98% 20500
99% 127405
100% 127408 (longest request)
http://example.com/ with args -n 200 -c 10: [FAILED]

# Testing apache benchmark on args [n=1000 -c 50]
This is ApacheBench, Version 2.3 <Revision: 1528965 >
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to the Apache Software Foundation, http://www.apache.org/

Benchmarking example.com [throughput:17.27,conn:1.000000] (be patient)
apr socket_recv Connection reset by peer[1001]: [FAILED]
http://example.com/ with args n=1000 -c 50: [FAILED]

JMeter:
jmeter -t Type multi-threaded 11 of 13 tests passed.
```

## In the browser

The image displays two side-by-side screenshots of web browser windows.

**Left Browser Window:** The title bar says "IIT Kanpur - Mozilla Firefox". The address bar shows "www.iitk.ac.in". The page content includes:

- Institute Overview:** Shows a building image.
- Latest News:**
  - Prof. S.N. Singh (EE) has been selected to receive the IEEE R10 (Asia-Pacific) Outstanding Volunteer Award (2016).
  - Dr. Arun Shukla of BISBE has been selected for the NASI-Young Scientist Platinum Jubilee Award in Biological Sciences for the year 2016.
- Announcements:**
  - Vacancies in Administrative and Technical Cadre
  - Online Form for posting your Research Queries
  - Courses
  - Post Doctoral Vacancies
  - Courses / Conferences / Workshop / Convention
  - 2016 International Conference on Law and Economics
- Featured Research:** An image of a DNA helix with text: "Structural basis of Human Complement Anaphylatoxin".
- HAPPENINGS AT IITK:** Shows a photo of people at an event.
- Media:**
  - Press Releases
  - Media Coverage
- International Relations:**

**Right Browser Window:** The title bar says "IANA - IANA-managed Reserved Domains - Mozilla Firefox". The address bar shows "www.iana.org/domains/reserved". The page content includes:

- IANA** Internet Assigned Numbers Authority
- DOMAINS NUMBERS PROTOCOLS ABOUT IANA**
- IANA-managed Reserved Domains**
- Certain domains are set aside, and normally registered to "IANA" for specific policy or technical purposes.
- Example domains**
- As described in RFC 3000 and RFC 6761, a number of domains such as `.example.com` and `.example.org` are maintained for documentation purposes. These domains may be used as illustrative examples in documents without prior coordination with us. They are not available for registration or transfer.
- Test IDN top-level domains**
- These domains were technically delegated by IANA for the ICANN Evaluation being conducted by ICANN.

IDN	Label	Language	Code
<code>.امن</code>	Secure	Arabic	Arabic
<code>.جنس</code>	Sexual	Arabic	Arabic
<code>.测试</code>	Test	Chinese	Han (Simplified variant)
<code>.测试</code>	Test	Chinese	Han (Traditional variant)
<code>.বিজ্ঞান</code>	Science	Bengali	Bengali
<code>.ভূগোল</code>	Geography	Bengali	Hindi
<code>.ഭൗഗോളിക്ക്</code>	Geographic	Devanagari (Nagari)	Greek
<code>.ഭൗഗോളിക്ക്</code>	Geographic	Devanagari (Modem)	Greek

---

## APPENDIX

```
#include "proxy_parse.h"
#include "proxy_parse.c"
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pthread.h>
#include <assert.h>
#include <time.h>
#include <netdb.h>
```

---

```
#include <sys/mman.h>
#include <sys/wait.h>

static int *count;

int main(int argc, char * argv[])
{
    int act_server_socket, actual_client_socket, act_clinet_socket;

    socklen_t addrlen;

    char buf[1024],head[1024];
    struct sockaddr_in client_address, server_address;
    struct hostent *name;

    char request[1024];

    if ((act_server_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) // Creating the server socket which will listen for client requests
    {
        printf("The socket was not created\n");
        exit(1);
    }

    printf("Socket Established \n");

    client_address.sin_family = AF_INET;
    client_address.sin_addr.s_addr = INADDR_ANY;
    client_address.sin_port = htons(atoi(argv[1])); //Setting up the port on which the server will be listening

    if (bind(act_server_socket, (struct sockaddr *) &client_address,
    sizeof(client_address)) == 0) //Binding the server sockets to the adress that was defined above
    {
        printf("Binding Socket\n");
    }
}
```

---

```
if (listen(act_server_socket, 10) < 0)      //Make the server to start listening
to the requests of the clients
{
    perror("server: listen");
    exit(1);
}

count = mmap(NULL, sizeof *count, PROT_READ | PROT_WRITE,
            MAP_SHARED | MAP_ANONYMOUS, -1, 0);
*count = 0;

while(1)
{
    if((actual_client_socket = accept(act_server_socket, (struct sockaddr *)
&client_address, &addrlen)) < 0)
    {
        perror("Could not create new socket");
        break;
    }
    pid_t pid;

    if(*count < 20)
    {
        pid = fork();
    }
    else
    {
        wait(NULL);
        pid = fork();
    }
    if(pid != 0)
    {
        close(actual_client_socket);
        continue;
    }

    *count = *count + 1;

    char tmp_buf[1024];
```

---

```

bzero((char *)buf, sizeof(buf));
while(1)
{
    bzero((char *)tmp_buf, sizeof(tmp_buf));
    if(recv(actual_client_socket, tmp_buf, 1024, 0)<= 0 )
    {
        printf("No message is received form the client");
        *count = *count -1;
        exit(1);
    }
    strcat(buf, tmp_buf);
    if(strstr(tmp_buf, "\r\n\r\n") != NULL)
        break;
}

printf("I have reached this point\n");

printf("\nThe buffer containing the url:\n%s\n\n", buf);

struct ParsedRequest *pr;
pr = ParsedRequest_create();      //Here we are just making an struct

int ret = ParsedRequest_parse(pr, buf, 1024);

if(ret != 0)
{
    char resp[1024];
    bzero((char*) resp, sizeof(resp));
    printf("Something went wrong while parsing\n");
    strcat(resp, "HTTP/1.1 500 Internal Server Error\r\n");
    strcat(resp, "Content-Length: 155");
    strcat(resp, "\r\n");
    strcat(resp, "Content-Type: text/html");
    strcat(resp, "\r\n");
    strcat(resp, "Connection: close");
    strcat(resp, "\r\n");
    strcat(resp, "\r\n");
}

```

---

```
    strcat(resp, "<html><head><TITLE>500 Internal Server Error</TITLE></head> \r\n<body><H1>500 Internal Server Error</H1>\r\nThere is some error on this server.\r\n</body></html>");
```

```
        write(actual_client_socket,resp,strlen(resp));
        *count = *count - 1;
        exit(1);
    }

bzero((char *)request, sizeof(request));

strcat(request, pr->method);
strcat(request, " ");
strcat(request, pr->path);
strcat(request, " ");
strcat(request, pr->version);
strcat(request, "\r\n");
strcat(request, "Host: ");
strcat(request, pr->host);
strcat(request, "\r\n");
strcat(request, "Connection: close\r\n");

if(pr->headers[0].key == NULL)
{
    printf("Well it just is that i dont have header");
}

else
{
    printf("Well what do you know i have headers\n");
    int list = 0;
    while(1)
    {
        if(pr->headers[list].key == NULL)
            break;
        if((strcmp(pr->headers[list].key, "Connection")!=0))
        {
            strcat(request, pr->headers[list].key);
            strcat(request, ": ");
            strcat(request, pr->headers[list].value );
        }
    }
}
```

```

        strcat(request, "\r\n");
    }
    list = list + 1;
}
}
strcat(request, "\r\n");
printf("The complete reques header has been made\n");
printf("%s\n", request);

act_clinet_socket=socket(AF_INET,SOCK_STREAM,0);
server_address.sin_family = AF_INET;
server_address.sin_port = htons(80);

name = gethostbyname(pr->host);

bcopy((char *)name->h_addr, (char *)&server_address.sin_addr.s_addr,
name->h_length);

if(connect(act_clinet_socket,(struct
sockaddr*)&server_address,sizeof(struct sockaddr)))
{
    printf("Error : Connection Failed");
    *count = *count -1;
    exit(1);
}
int b;
int a;
int c;
char buff[4096];

c = write(act_clinet_socket,request,1024);
while(1)
{
    bzero((char *)buff, sizeof(buff));
    b = recv(act_clinet_socket, buff, 4096, 0);

    if( b<=0)

```

```
{  
    close(act_clinet_socket);  
    break;  
}  
a = write(actual_client_socket,buff,b);  
}  
  
printf("I have finished the whole thing\n");  
  
close(actual_client_socket);  
*count = *count -1;  
exit(0);  
}  
return 0;  
}
```