Report: Project 2

CS425A Computer Networks 31st Aug 2016

Avinash Mohak, 13177 (amohak@iitk.ac.in)

Implemented Options

- HTTP Proxy
- GET method supported only

Design Choice

Dynamic-sized buffer (growing from zero to a maximum size of 8KB) was used to receive the request from the client using realloc.

Testing Results: (Browser: Firefox 48.0)

1. Telnet response (after connecting to localhost) for various sites. (Appendix A.1)
2. Response to bad requests sent via telnet (Appendix A.2)
3. Browser response for various links (2 links for example). (Appendix A.3)
4. 4 of 4 tests passed in modified *proxy_tester.py* (with *read_some*) (Appendix A.4)
5. 3 of 4 tests passed in given *proxy_tester.py* (with *read_all*) (Appendix A.5)
6. 13 of 13 tests passed in modified *proxy_tester_conc.py* (After converting *read_all* to *read_some)* (appendix A.6)
7. Maximum 12 of 13 tests passed in given *proxy_tester_conc.py* (Without converting *read_all* to *read_some)*. Many times 11 of 13 tests passed owing to the inconsistency related to cache. (Appendix A.6)

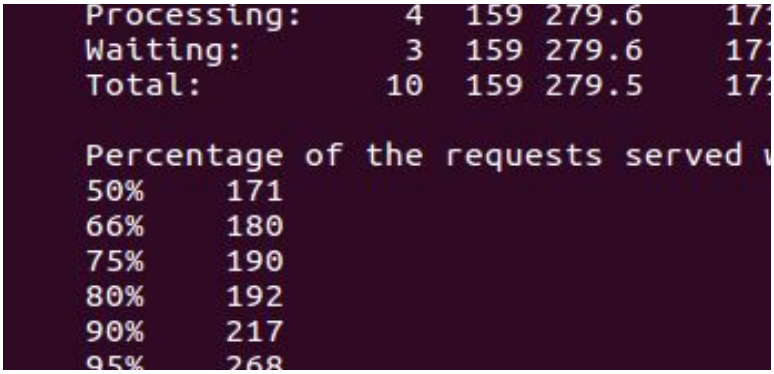
Summary

1. The proxy serves all the telnet requests and “http” sites via the browser properly to the best of my knowledge.
2. Secure sites (https sites) are not supported as they send “CONNECT” method, which has not been implemented.

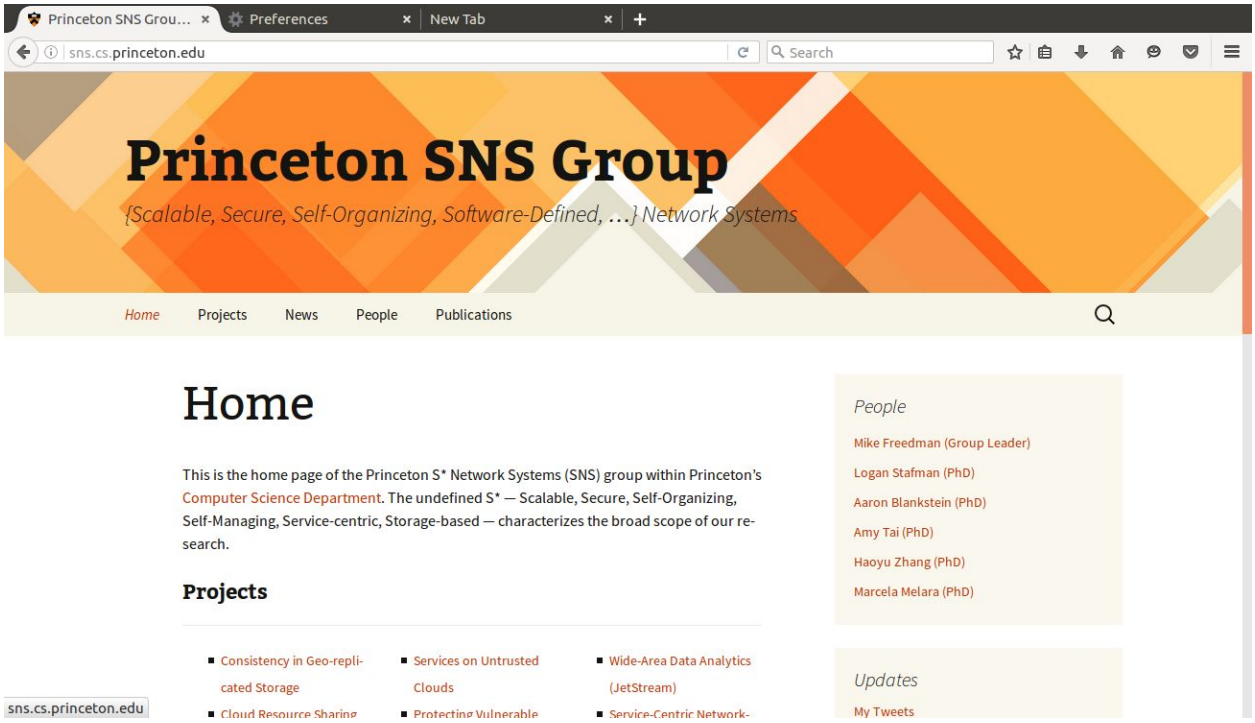
Appendix A

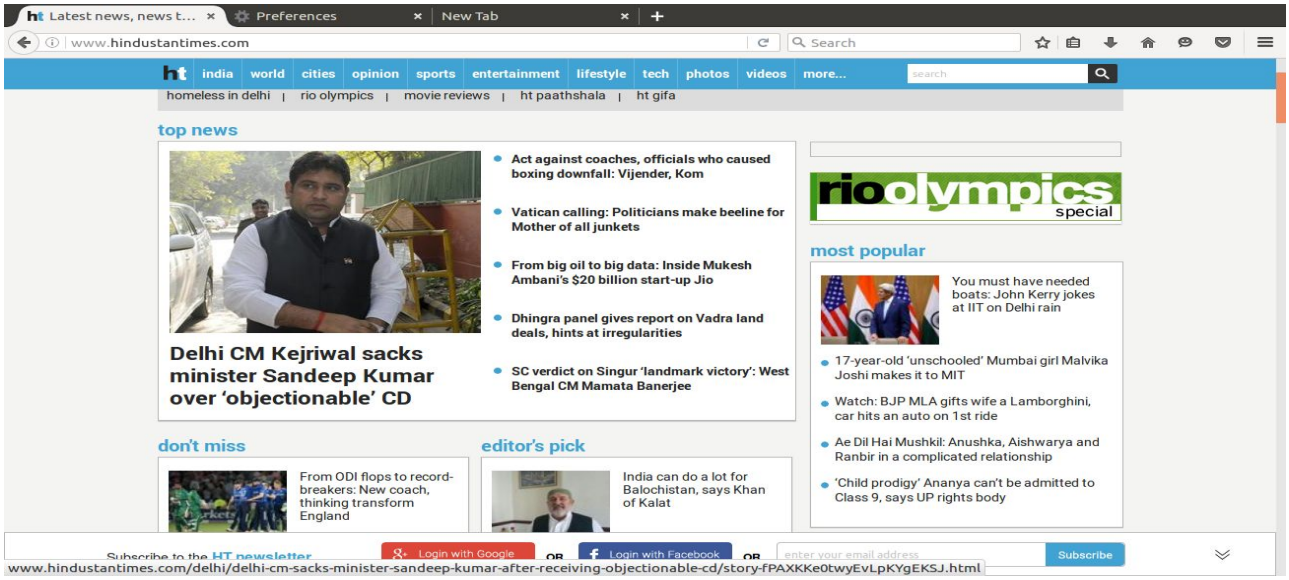
1. Telnet response

2. Bad request response



3. Firefox Response





4. *proxy_tester.py*

5. Output Log for modified *proxy_test_conc.py*

```
      Binary: ./proxy
Running on port 23598
### Testing: http://example.com/
http://example.com/: [PASSED]

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

### Testing: http://www.cs.princeton.edu/people/faculty
http://www.cs.princeton.edu/people/faculty: [PASSED]

### Testing:
http://www.cs.princeton.edu/sites/default/files/styles/gallery_full/public/
gallery-images/CS_building9.jpg?itok=meb0LzhS
http://www.cs.princeton.edu/sites/default/files/styles/gallery_full/public/
gallery-images/CS_building9.jpg?itok=meb0LzhS: [PASSED]

### Testing 2 concurrent connects to http://example.com/
Connect to http://example.com/, 2 concurrently: [PASSED]
```

```
### Testing 10 concurrent connects to http://example.com/
Connect to http://example.com/, 10 concurrently: [PASSED]

### Testing 2 concurrent fetches to http://example.com/
Fetch to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent fetches to http://example.com/
Fetch to http://example.com/, 10 concurrently: [PASSED]

### Testing 2 concurrent split fetches
Fetch to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent split fetches
Fetch to http://example.com/, 10 concurrently: [PASSED]

### Testing apache benchmark on args [-n 20 -c 1]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:23598] (be
patient).....done


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      1
    Time taken for tests:   0.104 seconds
    Complete requests:      20
    Failed requests:        0
    Total transferred:      34060 bytes
    HTML transferred:       25400 bytes
    Requests per second:    192.20 [#/sec] (mean)
    Time per request:       5.203 [ms] (mean)
    Time per request:       5.203 [ms] (mean, across all concurrent
requests)
    Transfer rate:          319.65 [Kbytes/sec] received
```

```
    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.1      0       0
    Processing:     4    5   1.3      5       8
    Waiting:        4    5   1.3      5       8
    Total:          4    5   1.3      5       8

    Percentage of the requests served within a certain time (ms)
    50%      5
    66%      5
    75%      6
    80%      7
    90%      8
    95%      8
    98%      8
    99%      8
    100%      8 (longest request)
http://example.com/ with args -n 20 -c 1: [PASSED]

### Testing apache benchmark on args [-n 200 -c 10]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:23598] (be patient)
    Completed 100 requests
    Completed 200 requests
    Finished 200 requests


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      10
    Time taken for tests:   0.300 seconds
    Complete requests:      200
    Failed requests:        0
```

```
    Total transferred:      340600 bytes
    HTML transferred:       254000 bytes
    Requests per second:    665.81 [#/sec] (mean)
    Time per request:       15.019 [ms] (mean)
    Time per request:       1.502 [ms] (mean, across all concurrent
requests)
    Transfer rate:          1107.30 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.1      0       0
    Processing:     4   14   5.4     14      29
    Waiting:        4   14   5.4     14      29
    Total:          4   14   5.4     14      29

    Percentage of the requests served within a certain time (ms)
    50%     14
    66%     16
    75%     18
    80%     19
    90%     22
    95%     24
    98%     26
    99%     26
    100%     29 (longest request)
http://example.com/ with args -n 200 -c 10: [PASSED]

### Testing apache benchmark on args [-n 1000 -c 50]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:23598] (be patient)
    Completed 100 requests
    Completed 200 requests
    Completed 300 requests
    Completed 400 requests
    Completed 500 requests
    Completed 600 requests
    Completed 700 requests
    Completed 800 requests
    Completed 900 requests
```

```
    Completed 1000 requests
    Finished 1000 requests


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      50
    Time taken for tests:   8.853 seconds
    Complete requests:      1000
    Failed requests:        0
    Total transferred:      1703000 bytes
    HTML transferred:       1270000 bytes
    Requests per second:    112.95 [#/sec] (mean)
    Time per request:       442.670 [ms] (mean)
    Time per request:       8.853 [ms] (mean, across all concurrent
requests)
    Transfer rate:          187.85 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.3      0       2
    Processing:     5  437 1091.9    193    5198
    Waiting:        5  437 1091.9    193    5198
    Total:          6  437 1091.9    193    5198

    Percentage of the requests served within a certain time (ms)
    50%    193
    66%    199
    75%    203
    80%    210
    90%    228
    95%   5177
    98%   5195
    99%   5197
    100%   5198 (longest request)
http://example.com/ with args -n 1000 -c 50: [PASSED]

Summary:
```

```
      Type multi-process: 13 of 13 tests passed.
```

6.

- Output log for given *proxy_tester_conc.py* (With 12 passed tests)

```
      Binary: ./proxy
Running on port 3901
### Testing: http://example.com/
http://example.com/: [PASSED]

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

### Testing: http://www.cs.princeton.edu/people/faculty
http://www.cs.princeton.edu/people/faculty: [PASSED]

### Testing:
http://www.cs.princeton.edu/sites/default/files/styles/gallery_full/public/
gallery-images/CS_building9.jpg?itok=meb0LzhS
!!!! Socket error while attempting to talk to proxy!
http://www.cs.princeton.edu/sites/default/files/styles/gallery_full/public/
gallery-images/CS_building9.jpg?itok=meb0LzhS: [FAILED]

### Testing 2 concurrent connects to http://example.com/
Connect to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent connects to http://example.com/
Connect to http://example.com/, 10 concurrently: [PASSED]

### Testing 2 concurrent fetches to http://example.com/
Fetch to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent fetches to http://example.com/
Fetch to http://example.com/, 10 concurrently: [PASSED]

### Testing 2 concurrent split fetches
Fetch to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent split fetches
Fetch to http://example.com/, 10 concurrently: [PASSED]

### Testing apache benchmark on args [-n 20 -c 1]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
```

```
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:3901] (be patient).....done


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      1
    Time taken for tests:   0.081 seconds
    Complete requests:      20
    Failed requests:        0
    Total transferred:      34080 bytes
    HTML transferred:       25400 bytes
    Requests per second:    246.67 [#/sec] (mean)
    Time per request:       4.054 [ms] (mean)
    Time per request:       4.054 [ms] (mean, across all concurrent
requests)
    Transfer rate:          410.47 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.0      0       0
    Processing:     3    4   0.2      4       4
    Waiting:        3    4   0.2      4       4
    Total:          4    4   0.2      4       4

    Percentage of the requests served within a certain time (ms)
    50%      4
    66%      4
    75%      4
    80%      4
    90%      4
    95%      4
    98%      4
    99%      4
    100%      4 (longest request)
```

```
http://example.com/ with args -n 20 -c 1: [PASSED]

### Testing apache benchmark on args [-n 200 -c 10]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:3901] (be patient)
    Completed 100 requests
    Completed 200 requests
    Finished 200 requests


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      10
    Time taken for tests:   0.149 seconds
    Complete requests:      200
    Failed requests:        0
    Total transferred:      340800 bytes
    HTML transferred:       254000 bytes
    Requests per second:    1344.46 [#/sec] (mean)
    Time per request:       7.438 [ms] (mean)
    Time per request:       0.744 [ms] (mean, across all concurrent
requests)
    Transfer rate:          2237.26 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.1      0       1
    Processing:     4    7   1.5      7      17
    Waiting:        4    7   1.5      7      17
    Total:          4    7   1.5      7      17

    Percentage of the requests served within a certain time (ms)
    50%      7
    66%      8
```

```
    75%      8
    80%      8
    90%      9
    95%     10
    98%     11
    99%     13
    100%     17 (longest request)
http://example.com/ with args -n 200 -c 10: [PASSED]

### Testing apache benchmark on args [-n 1000 -c 50]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:3901] (be patient)
    Completed 100 requests
    Completed 200 requests
    Completed 300 requests
    Completed 400 requests
    Completed 500 requests
    Completed 600 requests
    Completed 700 requests
    Completed 800 requests
    Completed 900 requests
    Completed 1000 requests
    Finished 1000 requests


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      50
    Time taken for tests:   8.127 seconds
    Complete requests:      1000
    Failed requests:        0
    Total transferred:      1704000 bytes
    HTML transferred:       1270000 bytes
    Requests per second:    123.05 [#/sec] (mean)
```

```
    Time per request:       406.354 [ms] (mean)
    Time per request:       8.127 [ms] (mean, across all concurrent
requests)
    Transfer rate:          204.76 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.2      0       1
    Processing:     5  402 1116.2    112    5291
    Waiting:        5  402 1116.2    112    5291
    Total:          6  402 1116.2    112    5291

    Percentage of the requests served within a certain time (ms)
    50%    112
    66%    192
    75%    214
    80%    255
    90%    302
    95%   5197
    98%   5269
    99%   5283
    100%   5291 (longest request)
http://example.com/ with args -n 1000 -c 50: [PASSED]

Summary:
      Type multi-process: 12 of 13 tests passed.
```

- Output log for *proxy_tester_conc.py* (with 11 tests passed)

```
      Binary: ./proxy
Running on port 33267
### Testing: http://example.com/
http://example.com/: [PASSED]

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

### Testing: http://www.cs.princeton.edu/people/faculty
compare_url failed on http://www.cs.princeton.edu/people/faculty
Proxy:  Date: Wed, 31 Aug 2016 17:32:11 GMT
Direct: Server: Apache/2.2.15 (Red Hat)
http://www.cs.princeton.edu/people/faculty: [FAILED]
```

```
### Testing:
http://www.cs.princeton.edu/sites/default/files/styles/gallery_full/public/
gallery-images/CS_building9.jpg?itok=meb0LzhS
!!!! Socket error while attempting to talk to proxy!
http://www.cs.princeton.edu/sites/default/files/styles/gallery_full/public/
gallery-images/CS_building9.jpg?itok=meb0LzhS: [FAILED]

### Testing 2 concurrent connects to http://example.com/
Connect to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent connects to http://example.com/
Connect to http://example.com/, 10 concurrently: [PASSED]

### Testing 2 concurrent fetches to http://example.com/
Fetch to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent fetches to http://example.com/
Fetch to http://example.com/, 10 concurrently: [PASSED]

### Testing 2 concurrent split fetches
Fetch to http://example.com/, 2 concurrently: [PASSED]

### Testing 10 concurrent split fetches
Fetch to http://example.com/, 10 concurrently: [PASSED]

### Testing apache benchmark on args [-n 20 -c 1]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:33267] (be
patient).....done


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      1
```

```
    Time taken for tests:   0.080 seconds
    Complete requests:      20
    Failed requests:        0
    Total transferred:      34060 bytes
    HTML transferred:       25400 bytes
    Requests per second:    250.45 [#/sec] (mean)
    Time per request:       3.993 [ms] (mean)
    Time per request:       3.993 [ms] (mean, across all concurrent
requests)
    Transfer rate:          416.53 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.0      0       0
    Processing:     3    4   0.3      4       5
    Waiting:        3    4   0.3      4       5
    Total:          4    4   0.3      4       5

    Percentage of the requests served within a certain time (ms)
    50%      4
    66%      4
    75%      4
    80%      4
    90%      4
    95%      5
    98%      5
    99%      5
    100%      5 (longest request)
http://example.com/ with args -n 20 -c 1: [PASSED]

### Testing apache benchmark on args [-n 200 -c 10]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/

    Benchmarking example.com [through 127.0.0.1:33267] (be patient)
    Completed 100 requests
    Completed 200 requests
    Finished 200 requests


    Server Software:        ECS
```

```
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      10
    Time taken for tests:   0.197 seconds
    Complete requests:      200
    Failed requests:        0
    Total transferred:      340600 bytes
    HTML transferred:       254000 bytes
    Requests per second:    1016.50 [#/sec] (mean)
    Time per request:       9.838 [ms] (mean)
    Time per request:       0.984 [ms] (mean, across all concurrent
requests)
    Transfer rate:          1690.53 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.1      0       0
    Processing:     4    9   2.5     10      17
    Waiting:        4    9   2.5     10      17
    Total:          4    9   2.5     10      18

    Percentage of the requests served within a certain time (ms)
    50%     10
    66%     11
    75%     11
    80%     11
    90%     12
    95%     13
    98%     15
    99%     16
    100%     18 (longest request)
http://example.com/ with args -n 200 -c 10: [PASSED]

### Testing apache benchmark on args [-n 1000 -c 50]
    This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
    Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
    Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
    Benchmarking example.com [through 127.0.0.1:33267] (be patient)
    Completed 100 requests
    Completed 200 requests
    Completed 300 requests
    Completed 400 requests
    Completed 500 requests
    Completed 600 requests
    Completed 700 requests
    Completed 800 requests
    Completed 900 requests
    Completed 1000 requests
    Finished 1000 requests


    Server Software:        ECS
    Server Hostname:        example.com
    Server Port:            80

    Document Path:          /
    Document Length:        1270 bytes

    Concurrency Level:      50
    Time taken for tests:   3.393 seconds
    Complete requests:      1000
    Failed requests:        0
    Total transferred:      1703000 bytes
    HTML transferred:       1270000 bytes
    Requests per second:    294.68 [#/sec] (mean)
    Time per request:       169.675 [ms] (mean)
    Time per request:       3.393 [ms] (mean, across all concurrent
requests)
    Transfer rate:          490.08 [Kbytes/sec] received

    Connection Times (ms)
    min  mean[+/-sd] median   max
    Connect:        0    0   0.3      0       2
    Processing:     4  165  67.4    193     272
    Waiting:        4  165  67.3    193     270
    Total:          6  165  67.2    193     272

    Percentage of the requests served within a certain time (ms)
    50%    193
    66%    202
```

```
    75%    217
    80%    226
    90%    240
    95%    251
    98%    263
    99%    265
    100%    272 (longest request)
http://example.com/ with args -n 1000 -c 50: [PASSED]

Summary:
      Type multi-process: 11 of 13 tests passed.
```

Source Code

```
#include <bits/stdc++.h>
#include <sys/types.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/sendfile.h>
#include <unistd.h>
#include <netinet/in.h>
#include <signal.h>
#include <netdb.h>
#include <fcntl.h>

#include "proxy_parse.h"
#include "book_keeping.h"

using namespace std;

int childCount = 0;
void fun(int x)
{
        childCount--;
}

#define MAX_PENDING 100
#define MSGRESLEN 8192
#define MAX_BUFFER_SIZE 8192
#define MAX_CHILD 20
```

```
int main(int argc, char * argv[])
{
	int s,SERVER_PORT;
	if (argc==2)
		SERVER_PORT = atoi(argv[1]);
	else
	{
		fprintf(stderr, "usage: outfile server_port\n");
		exit(1);
	}

	struct sockaddr_in sin;
	unsigned int len;
	memset((char *)&sin, 0, sizeof(sin));
	sin.sin_family = AF_INET;
	sin.sin_addr.s_addr = htonl(INADDR_ANY);
	sin.sin_port = htons(SERVER_PORT);

	if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0)
	{
		perror("Cannot create the socket");
		exit(1);
	}

	if ((bind(s, (struct sockaddr *)&sin, sizeof(sin))) < 0)
	{
		perror("Unable to bind");
		exit(1);
	}

	listen(s, MAX_PENDING);
	struct sigaction sigchld_action;
	sigchld_action.sa_handler = fun;
	sigchld_action.sa_flags = SA_NOCLDWAIT;

	sigaction(SIGCHLD, &sigchld_action, NULL);		//handles death of child so
that childCount is decremented after every child completes

	while(1)
	{
```

```
		/* For the first part, the proxy acts as the server accepting requests from the client */

		int new_s;
		len = 10;
		if ((new_s = accept(s, (struct sockaddr *)&sin, &len)) < 0) {
			continue;
		}


		childCount++ ;
		while(childCount > MAX_CHILD)					//If number of processes is greater than 20, the process waits
			wait();

		// cout << "No. of children now: " << childCount << "\n";

		/* As soon as a client is connected and number of threads alive is less than 20, create a thread to serve the client */
		if(fork() == 0)
		{
			char *requestmessage;				// contains the request message recieved by the proxy
			char temp_request[MSGRESLEN];
			char request_to_server[MSGRESLEN];		//contains the request message to be sent to the server by the proxy
			int l;
			char headers[MSGRESLEN];				// contains the headers sent by the client
			int temp_flag = 0;
			int requestmessage_length = 0;

			/* The loop makes sure split fetches work properly */
			while(1)
			{
				l = recv(new_s,temp_request,MSGRESLEN,0);
				if(l<=0)
				{
					close(new_s);
					exit(1);
				}

				requestmessage_length+=l;
```

```
                                requestmessage = (char
*)realloc(requestmessage,min(requestmessage_length, MAX_BUFFER_SIZE));
                                if(!temp_flag)
                                {
                                        strcpy(requestmessage,temp_request);
                                        temp_flag = 1;
                                }

                                else
                                {
                                        strcat(requestmessage,temp_request);
                                }

                                if(requestmessage[requestmessage_length-4]=='\r' &&
requestmessage[requestmessage_length-3]=='\n' &&
requestmessage[requestmessage_length-2]=='\r' &&
requestmessage[requestmessage_length-1]=='\n') break;
                        }

                        ParsedRequest *req = ParsedRequest_create();

                        /* The following "if" condition handles the bad parse error */

                        if (ParsedRequest_parse(req, requestmessage,
requestmessage_length) < 0) {
                                char *buf = (char *)malloc(100);
                                strcat(buf, "HTTP/1.0 500 Internal Error\r\n\r\n Internal Error
Occured\n");
                                send_new(new_s,buf,strlen(buf));
                                close(new_s);
                                exit(1);
                        }

                        if (ParsedHeader_set(req, "Connection", "close") < 0){
// Setting "Connection" header to CLOSE
                                perror("set header key not working\n");
                                exit(1);
                        }

                        char host_address[1000];
                        strcat(host_address,req->host);

                        if(req->port != NULL)
```

```
                        {
                                strcat(host_address,":");
                                strcat(host_address,req->port);
                        }
                        else
                        {
                                req->port = (char *)malloc(3);
                                strcpy(req->port,"80");
                        }

                        if (ParsedHeader_set(req, "Host",host_address) < 0){                //
Setting "Host Address"
                                perror("set header key not working\n");
                                exit(1);
                        }

                        ParsedRequest_unparse_headers(req, headers, sizeof(headers));


prepare_request(req->method,req->path,req->version,headers,request_to_server);   //
concatenated request line and the headers sent by the client

                        // cout << request_to_server << "\n";

                        /* Now the proxy acts as the client and will be sending the request to
the server */

                        struct hostent *hp;
                        struct sockaddr_in sin;

                        hp = gethostbyname(req->host);
                        memset((char *)&sin, 0, sizeof(sin));
                        sin.sin_family = AF_INET;
                        memcpy((char *)&sin.sin_addr, hp->h_addr, hp->h_length);
                        sin.sin_port = htons(atoi(req->port));

                        int client_socket;
                        if ((client_socket = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
                                perror("error in creating socket");
                                exit(1);
                        }

                        if (connect(client_socket, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
```

```
                        perror("error in socket connection");
                        close(client_socket);
                        exit(1);
                }

                send_new(client_socket, request_to_server,
strlen(request_to_server));  // sending the request to the server

                char buf[MAX_BUFFER_SIZE];
                memset(buf,'\0',sizeof(buf));

                int recv_len = 0;

                /* Whatever is received from the server is simply sent to the client */

                while((recv_len = recv(client_socket, buf, MAX_BUFFER_SIZE - 1, 0))>0)
                {
                        send_new(new_s,buf,recv_len);
                        memset(buf,'\0',sizeof(buf));
                }

                if(recv_len < 0)
                {
                        perror("Error in receiving buffer");
                        exit(1);
                }

                // Closing the connection
                close(client_socket);
                close(new_s);
                exit(0);
        }

        else
        {
                close(new_s);
        }
    }

    return 0;
}
```