

System Programming Lab #10

2019-05-14

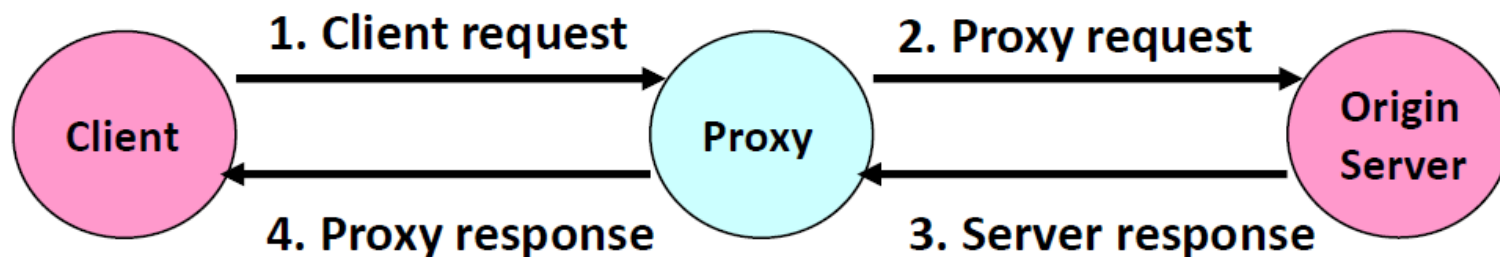
sp-tas

Lab Assignment #5 : Proxy Lab

- Download skeleton code & pdf from eTL
proxylab-handout.tar, proxylab-handout.pdf
- Hand In
 - First change **STUNO** to yours defined in Makefile
 - 'make handin' command will generate a tarball automatically
 - 구현 디렉토리 압축파일: 학번-proxylab.tar eg) 2019-12345-proxylab.tar
 - Upload your files eTL
 - 압축파일 양식 : [학번]_[이름]_proxylab.tar (or .zip, etc)
 - Ex) 2017-12345_홍길동_proxylab.tar
 - A zip file should include
 - (1) a tarball of your implementation directory (2) report
 - tarball 양식 : [학번]-proxylab.tar eg) 2019-12345-proxylab.tar
 - Report 양식 : [학번]_[이름]_proxylab_report.pdf (or .doc, .txt etc)
- Please, **READ** the Hand-out and Lab material thoroughly!
- Assigned : May 14
- Deadline : June 1, 23:59:59 (NO Delay Allowed)

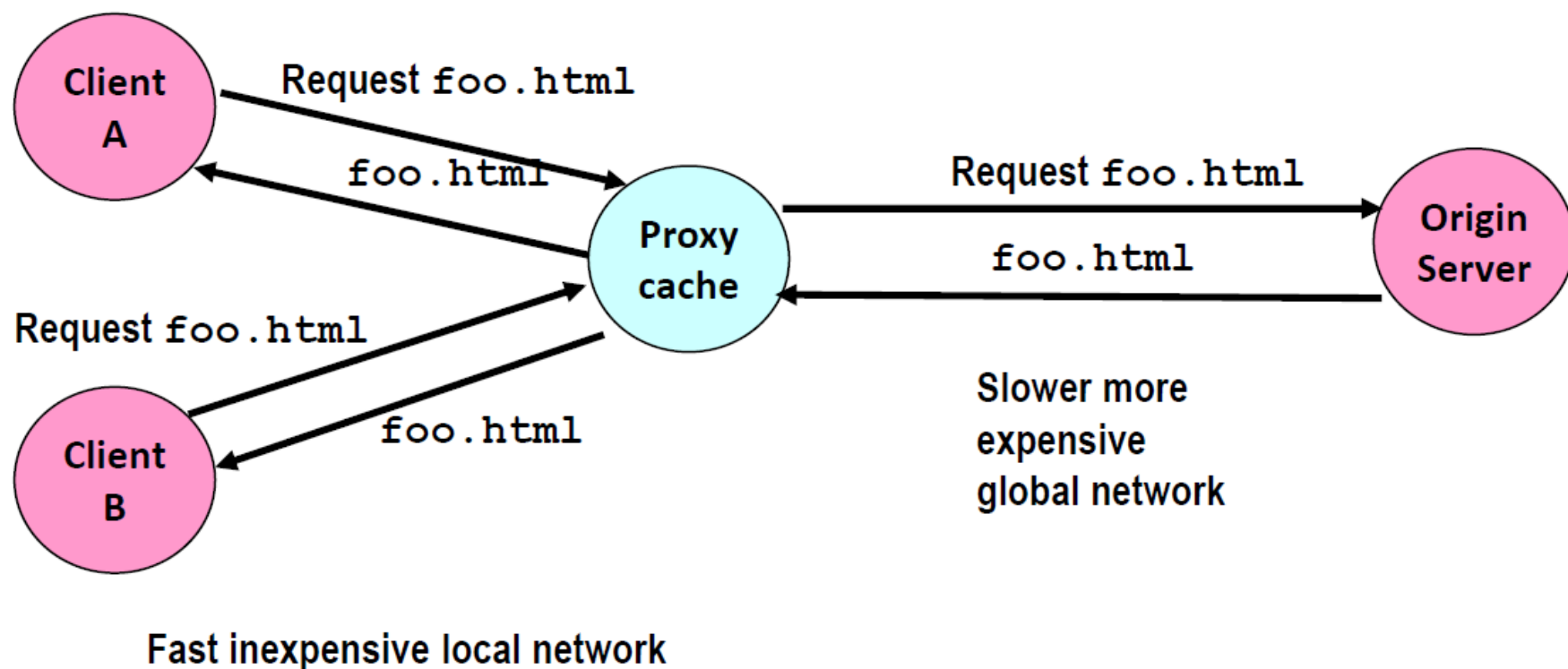
Proxies

- A **proxy** is an intermediary between a client and an **origin server**
 - To the client, the proxy acts like a server
 - To the server, the proxy acts like a client



Why Proxies?

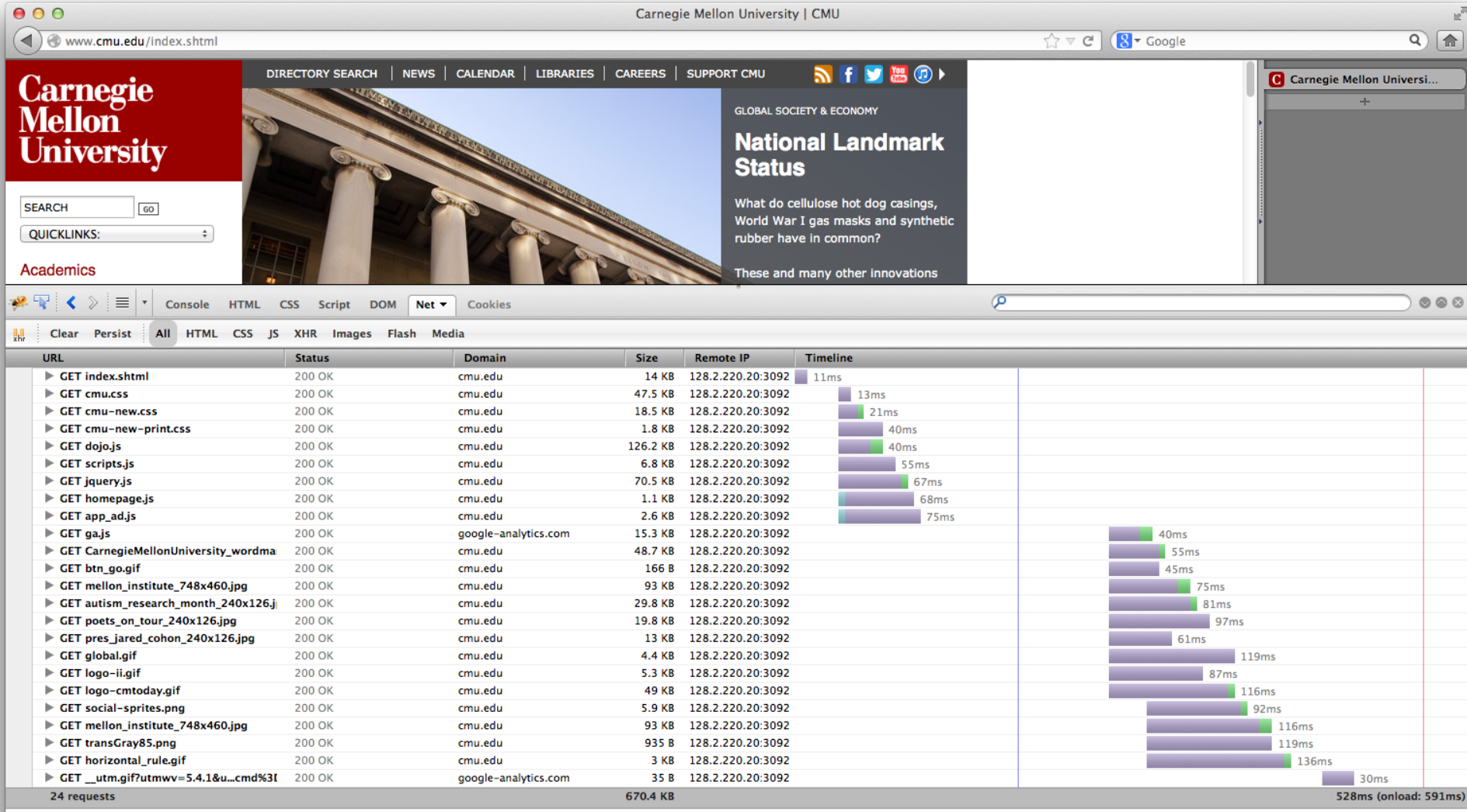
- Can perform useful functions as requests and responses pass by
 - Examples: Caching, logging, anonymization, filtering, transcoding



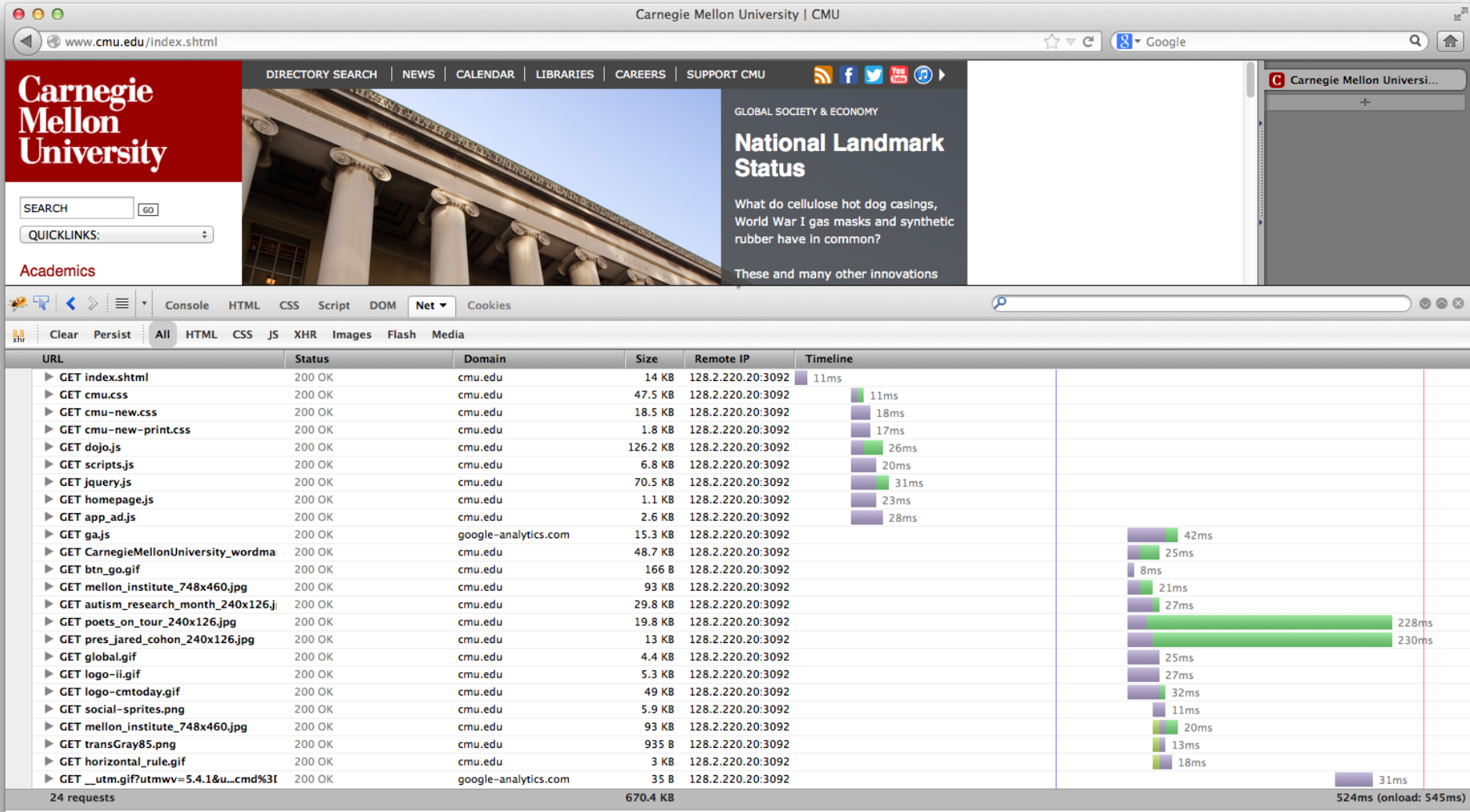
How the Web Really Works

- In reality, a single HTML page today may depend on 10s or 100s of support files (images, stylesheets, scripts, etc.)
- Builds a good argument for concurrent servers
 - Just to load a single modern webpage, the client would have to wait for 10s of back-to-back request
 - I/O is likely slower than processing, so back
- Caching is simpler if done in pieces rather than whole page
 - If only part of the page changes, no need to fetch old parts again
 - Each object (image, stylesheet, script) already has a unique URL that can be used as a key

Sequential Proxy



Concurrent Proxy



You will implement

- Write a simple HTTP proxy that caches web objects
- Part 1: Implementing a sequential web Proxy
 - **Basic HTTP operation & socket programming**
 - set up the proxy to accept incoming connections
 - read and parse requests
 - forward requests to web servers
 - read the servers' responses
 - forward those responses to the corresponding clients
- Part 2: Dealing with multiple concurrent requests
 - upgrade your proxy to deal with multiple **concurrent** connections
 - multi-threading
- Part 3: Caching web objects
 - add caching to your proxy using a simple main memory cache of recently accessed web content
 - cache individual objects, not the whole page
 - **Use an LRU eviction policy**
 - your caching system must allow for concurrent reads while maintaining consistency

Guide to start your implementation

- `int main(int argc, char *argv[])`
 - initialize everything such as data structure
 - checking port number
 - establish listening requests
 - when a client connects, spawn a new thread to handle it

```
1  #include <stdio.h>
2
3  /* Recommended max cache and object sizes */
4  #define MAX_CACHE_SIZE 1049000
5  #define MAX_OBJECT_SIZE 102400
6
7  /* You won't lose style points for including this long line in your code */
8  static const char *user_agent_hdr = "User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:10.0.3) Gecko/20120305 Firefox/10.0.3\r\n";
9
10 int main()
11 {
12     printf("%s", user_agent_hdr);
13     return 0;
14 }
15
```

Guide to start your implementation

- TAs implemented following structures and functions

```
typedef struct {  
    } Request;  
  
void *handle_client(void *vargp);  
void initialize_struct(Request *req);  
void parse_request(char request[MAXLINE], Request *req);  
void parse_absolute(Request *req);  
void parse_relative(Request *req);  
void parse_header(char header[MAXLINE], Request *req);  
void assemble_request(Request *req, char *request);  
int get_from_cache(Request *req, int clientfd);  
void get_from_server(Request *req, char request[MAXLINE], int clientfd, rio_t rio_to_client);  
void close_wrapper(int fd);  
void print_full(char *string);  
void print_struct(Request *req);  
  
typedef struct CachedItem CachedItem;  
  
struct CachedItem {  
};  
  
typedef struct {  
} CacheList;  
  
extern void cache_init(CacheList *list);  
extern void cache_URL(char *URL, void *item, size_t size, CacheList *list);  
extern void evict(CacheList *list);  
extern CachedItem *find(char *URL, CacheList *list);  
extern void move_to_front(char *URL, CacheList *list);  
extern void print_URLs(CacheList *list);  
extern void cache_destruct(CacheList *list);
```

Use csapp.[ch] functions

- Also, csapp.[ch] codes are included! yeah!

```
/* Sockets interface wrappers */
int Socket(int domain, int type, int protocol);
void Setsockopt(int s, int level, int optname, const void *optval, int optlen);
void Bind(int sockfd, struct sockaddr *my_addr, int addrlen);
void Listen(int s, int backlog);
int Accept(int s, struct sockaddr *addr, socklen_t *addrlen);
void Connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

```
/* Protocol independent wrappers */
void Getaddrinfo(const char *node, const char *service,
                 const struct addrinfo *hints, struct addrinfo **res);
void Getnameinfo(const struct sockaddr *sa, socklen_t salen, char *host,
                 size_t hostlen, char *serv, size_t servlen, int flags);
void Freeaddrinfo(struct addrinfo *res);
void Inet_ntop(int af, const void *src, char *dst, socklen_t size);
void Inet_pton(int af, const char *src, void *dst);
```

```
/* DNS wrappers */
struct hostent *Gethostbyname(const char *name);
struct hostent *Gethostbyaddr(const char *addr, int len, int type);
```

```
/* Pthreads thread control wrappers */
void Pthread_create(pthread_t *tidp, pthread_attr_t *attrp,
                   void * (*routine)(void *), void *argp);
void Pthread_join(pthread_t tid, void **thread_return);
void Pthread_cancel(pthread_t tid);
void Pthread_detach(pthread_t tid);
void Pthread_exit(void *retval);
pthread_t Pthread_self(void);
void Pthread_once(pthread_once_t *once_control, void (*init_function)());
```

```
/* Rio (Robust I/O) package */
ssize_t rio_readn(int fd, void *usrbuf, size_t n);
ssize_t rio_writen(int fd, void *usrbuf, size_t n);
void rio_readinitb(rio_t *rp, int fd);
ssize_t rio_readnb(rio_t *rp, void *usrbuf, size_t n);
ssize_t rio_readlineb(rio_t *rp, void *usrbuf, size_t maxlen);
```

```
/* Wrappers for Rio package */
ssize_t Rio_readn(int fd, void *usrbuf, size_t n);
void Rio_writen(int fd, void *usrbuf, size_t n);
void Rio_readinitb(rio_t *rp, int fd);
ssize_t Rio_readnb(rio_t *rp, void *usrbuf, size_t n);
ssize_t Rio_readlineb(rio_t *rp, void *usrbuf, size_t maxlen);
```

```
/* Reentrant protocol-independent client/server helpers */
int open_clientfd(char *hostname, char *port);
int open_listenfd(char *port);
```

```
/* Wrappers for reentrant protocol-independent client/server helpers */
int Open_clientfd(char *hostname, char *port);
int Open_listenfd(char *port);
```

Checking Your Work

- Auto grader
 - ./driver.sh will run the tests:
 - Ability to pull basic web pages from a server
 - Handle a (concurrent) request while another request is still pending
 - Fetch a web page again from your cache after the server has been stopped
 - This should help answer the question “is this what my proxy is supposed to do?”
 - Please don’t use this grader to definitively test your proxy; there are many things not tested here

Checking Your Work

- Test your proxy liberally
 - The web is full of special cases that want to break your proxy
 - Generate a port for yourself with `./port-for-user.pl [sp ID]`
 - Generate more ports for web servers and such with `./free-port.sh`
- Create a handin file with *make handin*
 - First you should change STUNO defined in Makefile to your student number
 - Will create a tar file for you with the contents of your proxylab-handin folder

Telnet/cURL Demo

- Telnet

- Interactive remote shell – like ssh without security
- Must build HTTP request manually
 - This can be useful if you want to test response to malformed headers

```
ta@sp1:~$ telnet snu.ac.kr 80
```

```
Trying 147.46.10.58...
```

```
Connected to snu.ac.kr.
```

```
Escape character is '^]'.  
GET /index.html HTTP/1.0
```

```
GET /index.html HTTP/1.0
```

```
HTTP/1.1 200 OK
```

```
Date: Tue, 22 May 2018 08:43:50 GMT
```

```
Set-Cookie: PHPSESSID=rlribnfbe9qj2n4if54scm2dr2; path=/; domain=.snu.ac.kr
```

```
Expires: Thu, 19 Nov 1981 08:52:00 GMT
```

```
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
```

```
Pragma: no-cache
```

```
Connection: close
```

```
Content-Type: text/html
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" .....
```

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="ko" xml:lang="ko">
```

```
<head>
```

```
.....
```

Telnet/cURL Demo

- cURL

- “URL transfer library” with a command line program
- Builds valid HTTP requests for you!

```
ta@sp1:~$ curl http://snu.ac.kr/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" .....
<html xmlns="http://www.w3.org/1999/xhtml" lang="ko" xml:lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<title>서울대학교</title>
<meta name="author" content="SEOUL NATIONAL UNIVERSITY" />
<meta name="robots" content="all" />
```

.....

- Can also be used to generate HTTP proxy requests:

```
ta@sp1:~$ curl --proxy localhost:15214 http://snu.ac.kr/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" .....
<html xmlns="http://www.w3.org/1999/xhtml" lang="ko" xml:lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<title>서울대학교</title>
<meta name="author" content="SEOUL NATIONAL UNIVERSITY" />
<meta name="robots" content="all" />
```

.....

Testing with Web Browser (chrome, REST Client)

The image displays two REST client interfaces side-by-side. The left interface is the REST Client web application, and the right is the Postman desktop application. Both show a GET request to the URL `http://sysprog.snucse.org:15213/home.html`.

REST Client Interface (Left):

- Method: GET
- Request URL: `http://sysprog.snucse.org:15213/home.html`
- Status: 200 OK
- Time: 29 ms
- Response Body (HTML):

```
<html>
  <head>
    <title>test</title>
  </head>
  <body>
    
    Dave O'Hallaron
  </body>
</html>
```

Postman Interface (Right):

- Method: GET
- Request URL: `http://sysprog.snucse.org:15213/home.html`
- Status: 200 OK
- Time: 8.90 ms
- Response Body (HTML):

```
<html>
<head><title>test</title></head>
<body>

Dave O'Hallaron
</body>
</html>
```

Both interfaces also show a list of recent requests in the history pane on the left, with the current request highlighted.

Testing with Web Browser (chrome, set proxy)

testing with
<http://snu.ac.kr>

SwitchyOmega

SETTINGS

- Interface
- General
- Import/Export

PROFILES

- proxy
- auto switch
- + New profile...

ACTIONS

- Apply changes
- Discard changes

Profile :: proxy

Proxy servers

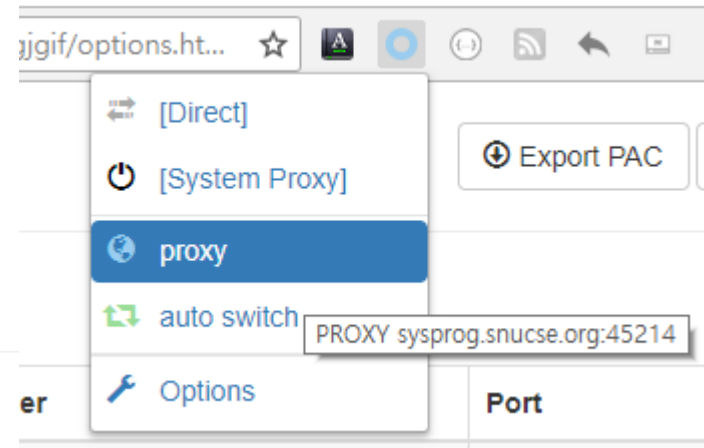
Scheme	Protocol	Server	Port	
(default)	HTTP	sysprog.snucse.org	45214	
Show Advanced				

Bypass List

Servers for which you do not want to use any proxy: (One server on each line.)

(Wildcards and more available...)

```
127.0.0.1
::1
localhost
```



Test manually using curl

- Manually testing following real pages
 - <http://www.snu.ac.kr/index.html>
 - <http://csapp.cs.cmu.edu>
 - <http://www.sk.co.kr>
 - <http://www.culture.go.kr>
- You should always use **./port-for-user.pl *username*** and **./free-port.sh *port*** when testing your proxy manually

```
root@sp3:/home/ta/hkim/proxylab/src# make
gcc -g -Wall -c proxy.c
gcc -g -Wall -c cache.c
gcc -g -Wall -c csapp.c
gcc -g -Wall proxy.o cache.o csapp.o -o proxy -lpthread
root@sp3:/home/ta/hkim/proxylab/src# ./port-for-user.pl ta
ta: 48232
root@sp3:/home/ta/hkim/proxylab/src# ./proxy 48232
^C
root@sp3:/home/ta/hkim/proxylab/src# ./free-port.sh 48232
4500
root@sp3:/home/ta/hkim/proxylab/src#
```

```
root@sp3:/home/ta#
root@sp3:/home/ta#
root@sp3:/home/ta#
root@sp3:/home/ta# clear
root@sp3:/home/ta# curl --proxy localhost:48232 http://snu.ac.kr/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ko" xml:lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<title>서울대학교</title>
<meta name="author" content="SEOUL NATIONAL UNIVERSITY" />
<meta name="robots" content="all" />

<link rel="icon" href="/favicon.ico" type="image/x-icon" />
<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />

<link rel="apple-touch-icon-precomposed" href="/mobileicon.png" />

<link rel="stylesheet" type="text/css" href="/_skin/default/css/default.css?ver=2017-05-23" media="all" />
<link rel="stylesheet" type="text/css" href="/_skin/default/css/layout_0720.css?ver=2018-02-12xx" media="all" />
<link rel="stylesheet" type="text/css" href="/_skin/default/css/common.css?ver=2017-06-01" media="all" />

<link rel="stylesheet" type="text/css" href="/_skin/default/css/about.css?ver=2019-02-13" media="all" />
<link rel="stylesheet" type="text/css" href="/_skin/default/css/education.css?ver=2018-08-07x" media="all" />
<link rel="stylesheet" type="text/css" href="/_skin/default/css/research.css?ver=2017-04-21" media="all" />
<link rel="stylesheet" type="text/css" href="/_skin/default/css/withsnu.css?ver=2018-04-30" media="all" />
<link rel="stylesheet" type="text/css" href="/_skin/default/css/utility.css?ver=2017-04-21" media="all" />
```

Evaluation

- Total Score: 100 points
- Basic Correctness (40 points)
 - basic proxy operation (auto graded)
- Concurrency (15 points)
 - handling concurrent requests (auto graded)
- Cache (15 points)
 - working cache (auto graded)
- Real Pages (20 points)
 - correctly serving the real pages (5 points each)
- Report (10 points)
 - describes the goal of proxy lab and how to implement for each part
 - what you learn in this lab
 - what was difficult
 - what was surprising and so on

Announcement - security issue

- sp2.snucse.org attacked from malicious users
- So we cleaned accounts who didn't change default password
 - stuXXX@sp1.snucse.org
 - stuXXX@sp2.snucse.org
- Default password changed
 - will be announced in lab session
 - some people lost their contents saved in the server

Kernel Lab Evaluation

- Total Score: 120 points
- Delay policy: Same as before
- Part 0 (No delay) (10 points)
- ptree evaluation (50 points)
 - (10 points) successful compilation (make/make clean) (no compile error)
 - (10 points) (init) create dir/files in debugfs + (exit) remove dir/files in debugfs
 - (15 points) test case 1 (init process pid) (result of 'cat ptree')
 - (15 points) test case 2 (random child process pid) (result of 'cat ptree')
- paddr evaluation (50 points)
 - (10 points) successful compilation (make/make clean) (no compile error)
 - (10 points) (init) create dir/files in debugfs + (exit) remove dir/files in debugfs
 - (10 points) test case 0 (0x42341000 or 0x234512000) and (0x20000)
 - (15 points) test case 1 (0xXXXXX000 or 0xXXXXXX000)
 - (15 points) test case 2 (0xXX000)
- report (10 points)

Schedule

- 5/21 중간고사 2 (실습 내용 포함)
- 5/21 No Lab Session
- 5/23 (수업시간) ProxyLab Q&A Session
- 5/28 Lab Assignment #6
 - final assignment (6:30~)
 - ProxyLab Q&A Session (7:30~)
 - *중간고사2 Claim*
- 6/1 ProxyLab Due (23:59:59)
 - No Delay Allowed
- *6/4 Lab Assignment #6 Q&A Session*
- 6/13 기말고사 (실습 내용 포함)
- 6/17 Lab Assignment #6 Due (23:59:59)
 - No Delay Allowed