DCSLAB

System Programming
Lab #10

2020-05-27

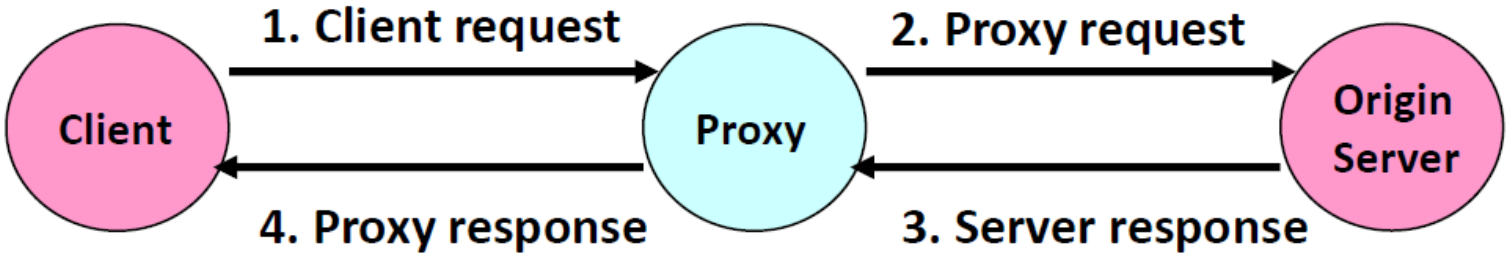
sp-tas

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) 2020-12345-proxylab.tar
  - Upload your files eTL
    - 압축파일 양식 : [학번]_[이름]_proxylab.tar (or .zip, etc)
    - Ex) 2020-12345_홍길동_proxylab.tar
  - A zip file should include
  - (1) a tarball of your implementation directory (2) report
    - tarball 양식 : [학번]-proxylab.tar eg) 2020-12345-proxylab.tar
    - Report 양식 : [학번]_[이름]_proxylab_report.pdf (or .doc, .txt etc)
- Please, READ the Hand-out and Lab material thoroughly!
- Assigned : May 27
- Deadline : June 17, 23:59:00 (NO Delay Allowed)

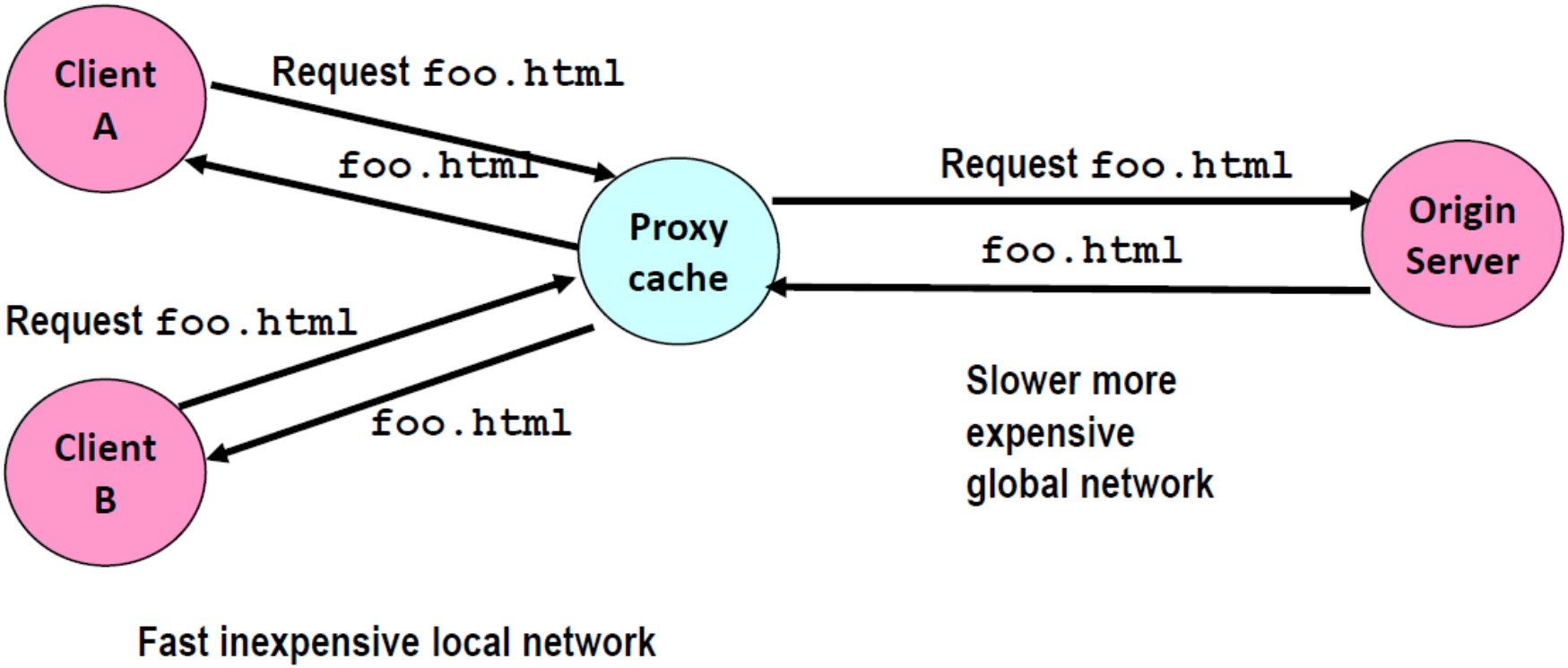
Proxies

- 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?

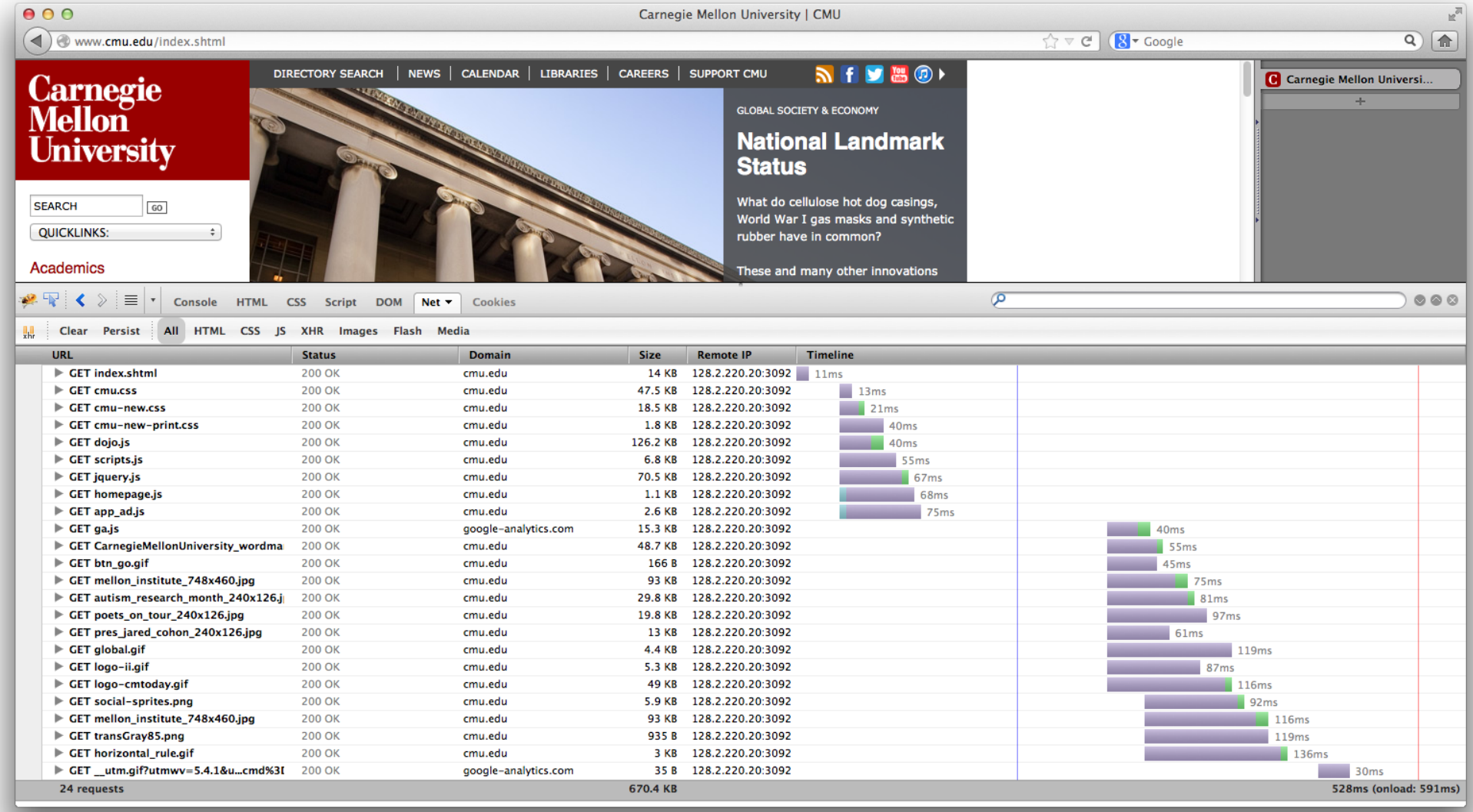
- **Can perform useful functions as requests and responses pass by**
  - Examples: Caching, logging, anonymization, filtering, transcoding



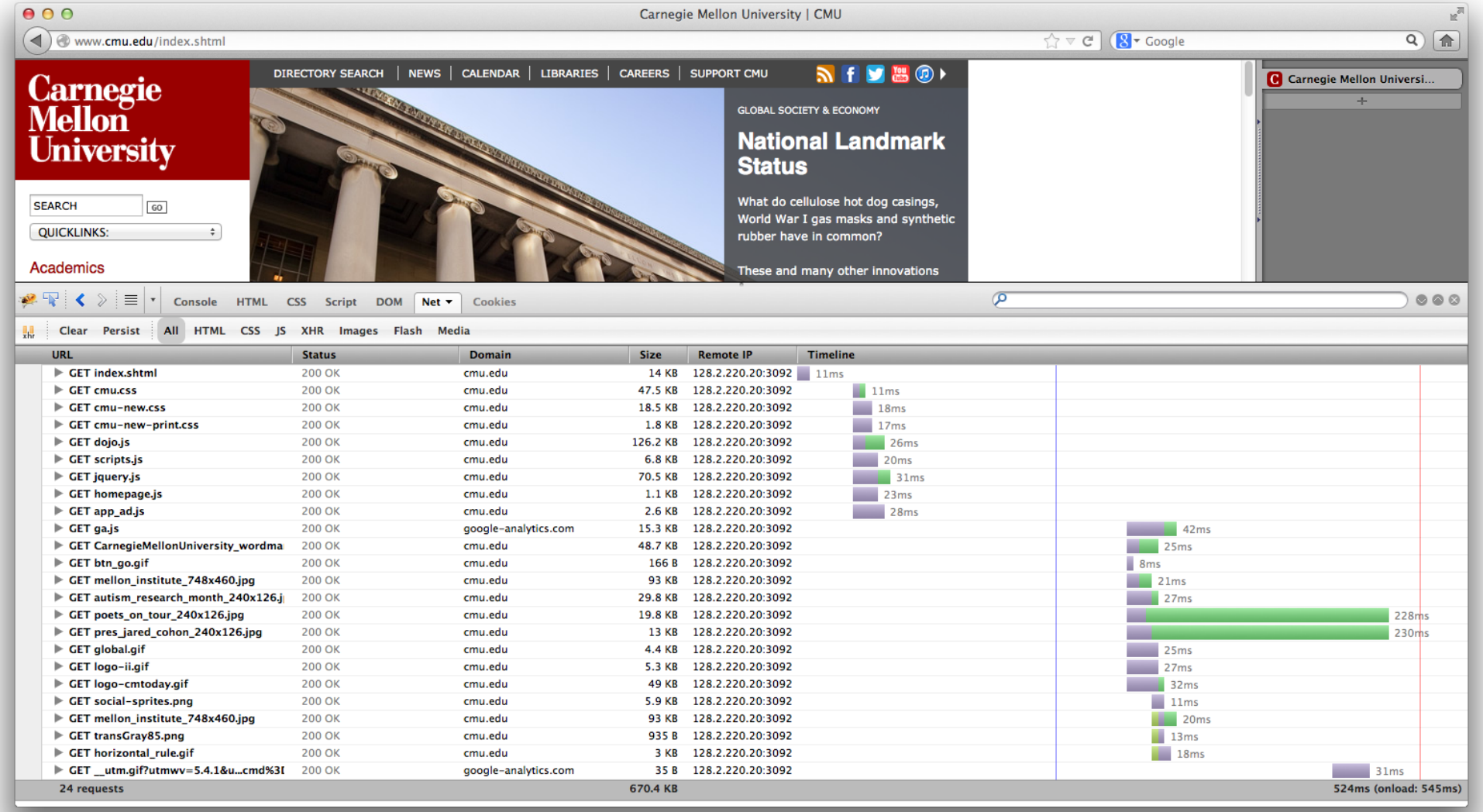
How the Web Really Works

- 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

```c
#include <stdio.h>

/* Recommended max cache and object sizes */
#define MAX_CACHE_SIZE 1049000
#define MAX_OBJECT_SIZE 102400

/* You won't lose style points for including this long line in your code */
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";

int main()
{
    printf("%s", user_agent_hdr);
    return 0;
}
```

Guide to start your implementation

- TAs implemented following structures and functions

```c
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);
```

```c
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);
```

```
/* 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);
```

```
/* 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)());
```

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

Double-Enter →

```
ubuntu@sp:~$ telnet apache.org 80
Trying 40.79.78.1...
Connected to apache.org.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 27 May 2020 06:59:34 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Wed, 27 May 2020 06:10:30 GMT
ETag: "14a3e-5a69b11aa7bb0"
Accept-Ranges: bytes
Content-Length: 84542
Vary: Accept-Encoding
Cache-Control: max-age=3600
Expires: Wed, 27 May 2020 07:59:34 GMT
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
```

Telnet/cURL Demo

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

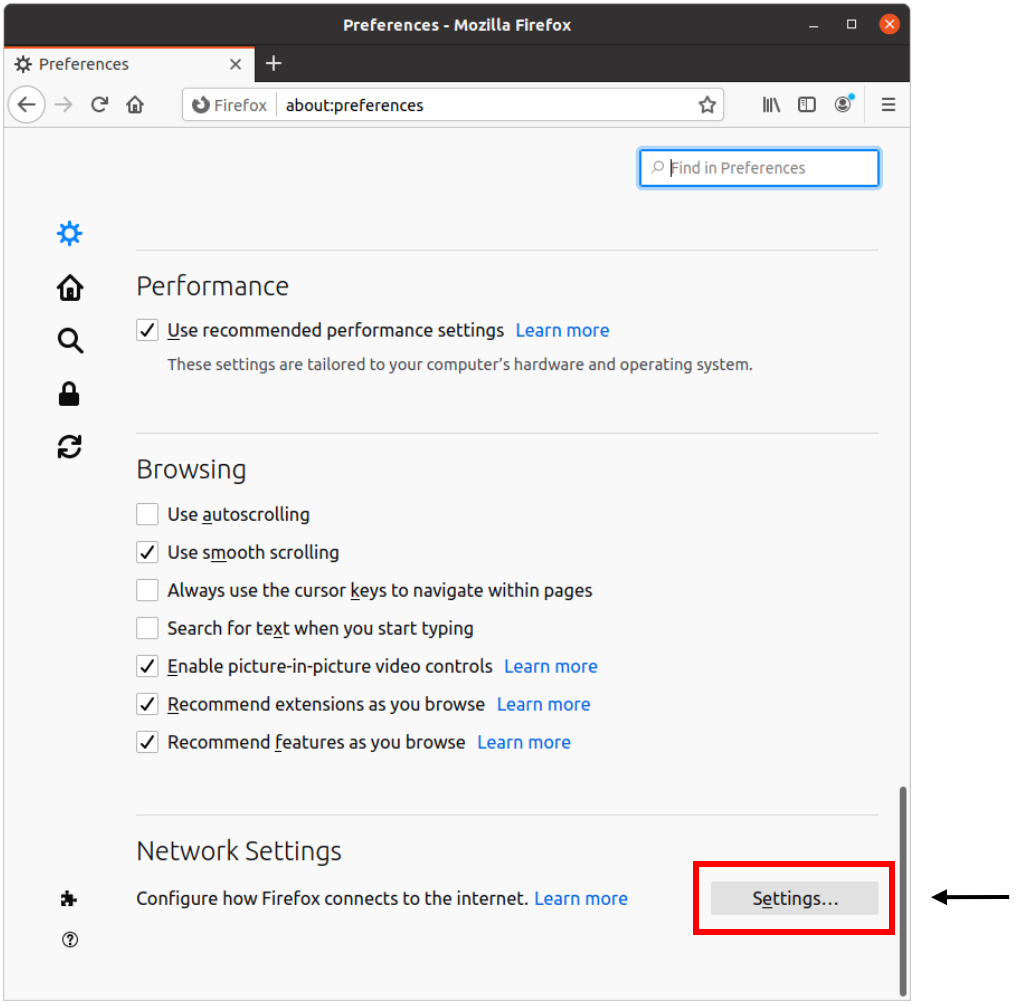
```
ubuntu@sp:~$ curl http://apache.org
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="Home page of The Apache Software Foundation">

  <link rel="apple-touch-icon" sizes="57x57" href="/favicons/apple-touch-icon-57x57.png">
  <link rel="apple-touch-icon" sizes="60x60" href="/favicons/apple-touch-icon-60x60.png">
  <link rel="apple-touch-icon" sizes="72x72" href="/favicons/apple-touch-icon-72x72.png">
```

  - Can also be used to generate HTTP proxy requests:

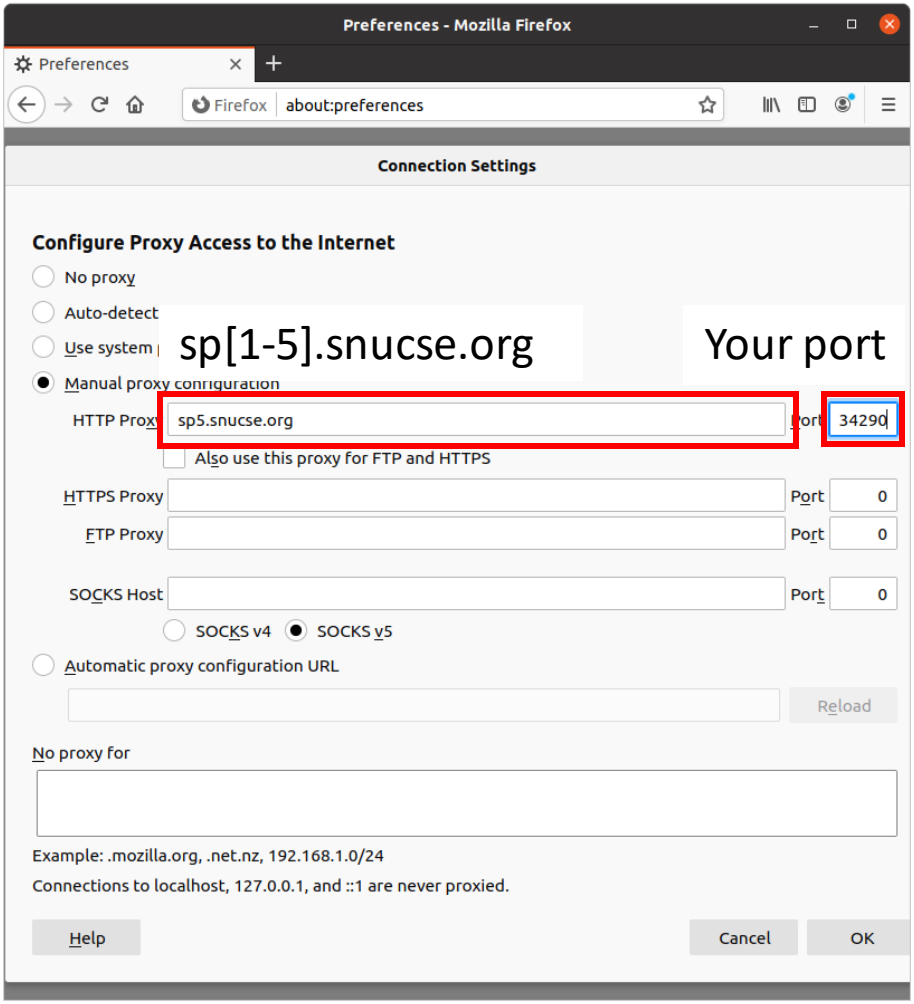
```
ubuntu@sp:~$ curl --proxy localhost:52184 http://apache.org > curl.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16962  100 16962    0     0  21.9M      0 --:--:-- --:--:-- --:--:-- 16.1M
ubuntu@sp:~$ head -n 3 curl.txt
◆n◆
I◆6◆).mR◆ڳGĂDU(◆PU◆◆$◆"f^>b>3%3+
                                ◆A◆h}◆-◆◆◆|◆◆◆◆ _N◆◆◆^EE+◆@I◆◆◆&*◆S◆v◆Y)◆◆◆◆◆?w"◆Q◆"x◆◆/◆◆zP◆I◆◆◆}u
x◆K_:"◆◆<◆a7◆^oK◆$]<Q"◆#%◆a~)S%◆◆=b◆a◆◆T◆◆Wd◆◆◆◆Ď7v◆◆◆◆◆◆~◆"Σ◆^◆◆◆f◆◆z~◆~◆l}◆◆◆\Vz◆z%z◆v7◆◆獏'W}s◆
◆Qm_◆4◆&W◆X◆fz◆◆f3damT◆◆◆*◆◆aSy7#C%[2D◆◆◆5e⅗C9◆◆◆]◆◆l◆◆w◆◆◆Pec◆◆◆q◆gy*◆5x◆◆{=6◆T>U◆◆K◆+◆◆hD◆KO◆◆◆2◆◆
```

Gzipped (encoded) data from proxy

Testing with Web Browser (Firefox)



Testing with Web Browser (Firefox)



Test manually using curl

- Manually testing following real pages
  - http://neverssl.com
  - http://example.com
  - http://apache.org
  - http://gnu.org
- You should always use **./port-for-user.pl *username*** when testing your proxy manually

Evaluation

- Total Score: 80 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)
- 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