Computer Network
Project 2
Web Proxy Server
-Performance Evaluation-
(230pts)

CSI4106-01

Fall, 2018

Prelim.

Before you do this homework, you must be fully aware of "**Project Policy Notice**"

Performance Evaluation Report

“average response time” with the four modes of your proxy server

Mode	Persistent Connection	Multithreaded
Naïve Proxy	X	X
PC only Proxy	O	X
MT only Proxy	X	O
MT+PC Proxy	O	O

Performance Evaluation Report

Option	Set	Description
Persistent Connection	**X**	All packets via your proxy are **forced** to use **non-persistent connections.**
Persistent Connection	**O**	All packets via your proxy are **forced** to use **persistent connections.**
Multithreaded	**X**	Your proxy is working in a naïve single-threading fashion. "**excluding select() function**"
Multithreaded	**O**	Your proxy is working in a **multithreading** fashion.

Goal (you are expected to)

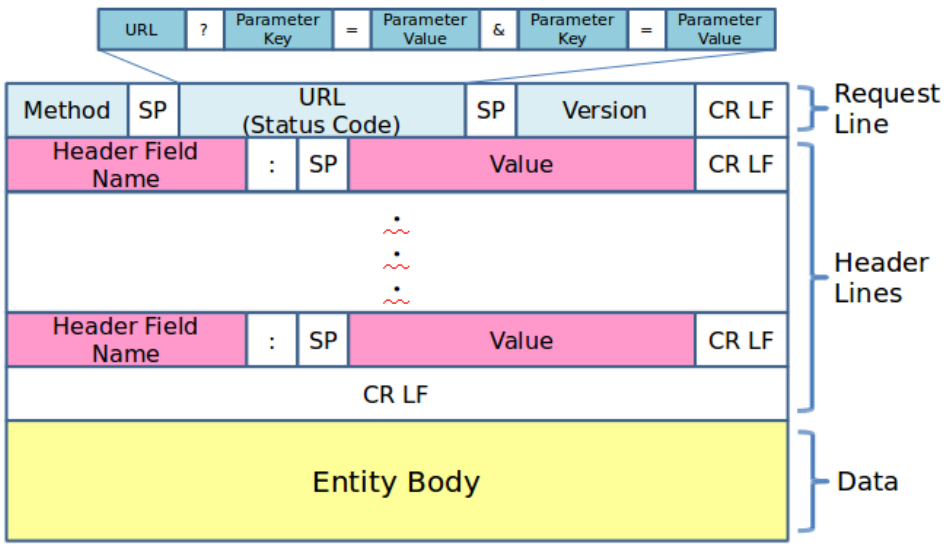
1. Learn how the **HTTP** works
   + the difference of **Persistent vs Non-persistent Connection**
2. Learn how a Proxy Server works
   + handling two sockets simultaneously.
   + implementing **Multithreaded Socket Programming**
3. **Write a simple web-proxy server…**

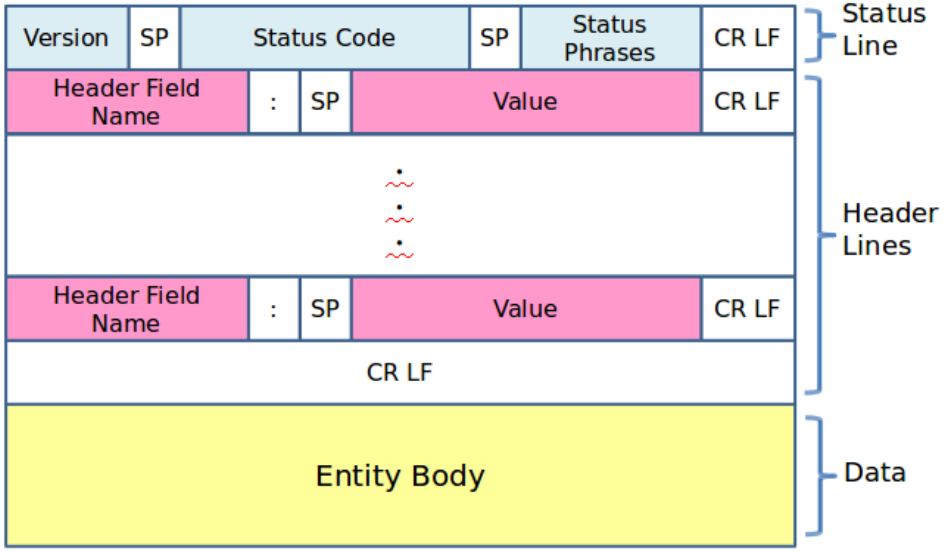
Steps to get this project done

1. Follow up by googling or reading your textbook
   - **We are providing the incomplete code**
   - **how a typical HTTP works**
   - **how a transparent proxy works**
2. ~~**Copy and Paste**~~ **Refactor some multithreaded socket codes online.**
3. **Complete the skeleton proxy code we provide.**

Background

HTTP Header
Request Response



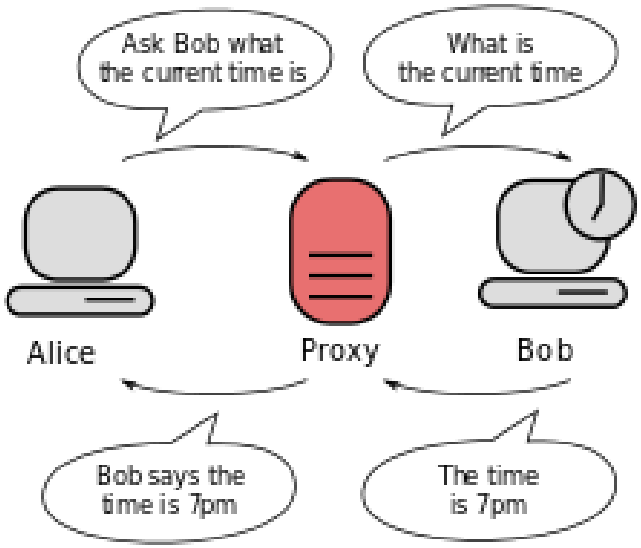



```
GET /css/overwrite.css HTTP/1.1\r\n
Host: mnet.yonsei.ac.kr\r\n
Connection: keep-alive\r\n
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win
Accept: text/css,*/*;q=0.1\r\n
Referer: http://mnet.yonsei.ac.kr/\r\n
Accept-Encoding: gzip, deflate, sdch\r\n
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,
```

```
HTTP/1.1 200 OK\r\n
Server: nginx/1.8.1\r\n
Date: Wed, 24 Aug 2016 05:39:54 GMT\r\n
Content-Type: text/css\r\n
Content-Length: 27466\r\n
Last-Modified: Tue, 24 May 2016 07:39:46 GM
Connection: keep-alive\r\n
ETag: "57440542-6b4a"\r\n
```

What is a proxy server?

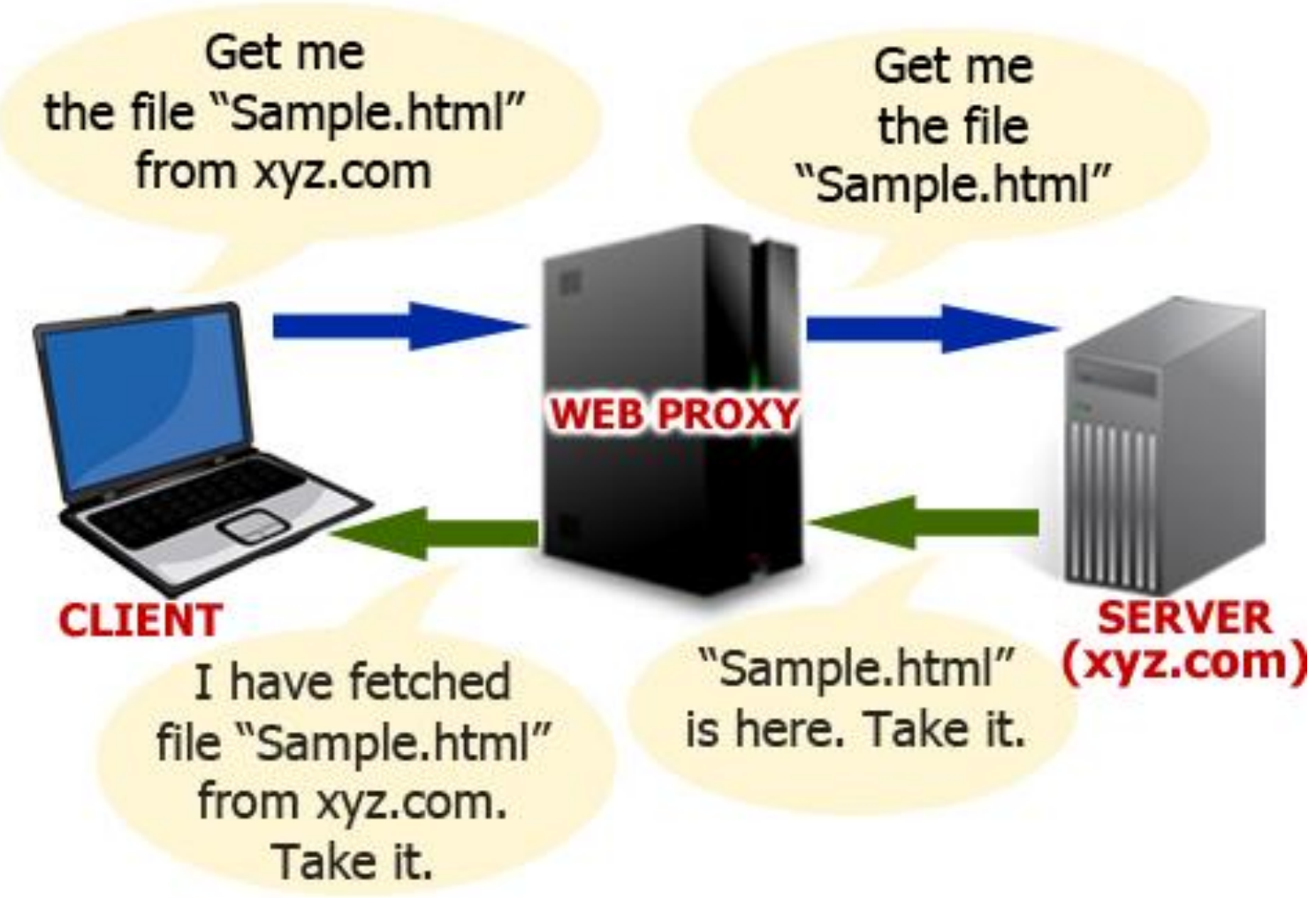
- An intermediary for clients(source) or servers(destination), which resides anywhere.
- It is deployed at a place close to clients(forward-proxy) or servers(reverse-proxy)



Main functionalities of proxy

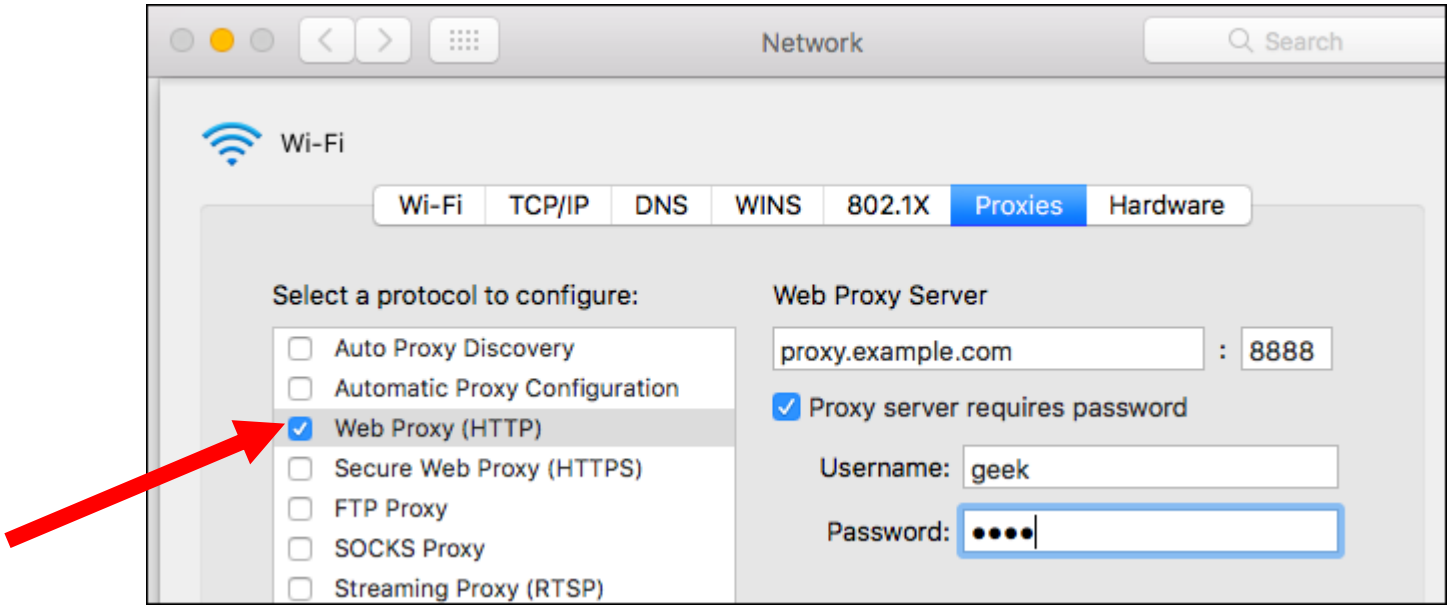
- It **builds** a cache to reduce a response time.
- It acts as a **load-balancer** by redirecting requests to other nodes.
- It **blocks** unauthorized requests/responses.
- It **eavesdrops** on the data-flow between clients and the web.
- and so on…

How a http proxy server works?



How to configure a HTTP proxy?

- Go to google.
- Type “how to set up proxy server on windows / mac / linux”



Hint (1) Managing Two Sockets!

- **Question: Why does it need two sockets??**
  - One socket for receiving HTTP Request from a client.
  - Another socket for fetching HTTP Response from the server (destination).
- **(Hint)** Then you need to forward Requests and Responses to….

Where should they be forwarded?

Hint (2) Persistent Connection

- You must have learned this in the mid-term.
- Most of web-servers (HTTP/1.1) support and use this feature in a default option.
- You have to modify HTTP header to communicate using the persistent connection for HTTP/1.0 and 1.1.
- **(Hint)** On a HTTP request header, you may encounter

```
Connection: keep-alive
Connection: close
Proxy-Connection: (empty)
```

Hint (3) Multithreading Feature

- (Hint) The skeleton code
- (Hint) Google
- (Hint) Your friend’s code
- (Hint) The course book of “Operating System”

`proxy.py` (incomplete)

project.py

```
1   from socket import *
2   from urllib.parse import urlparse
3   import threading
4   import sys
5
6   BUFSIZE = 2048
7   TIMEOUT = 5
8   CRLF = '\r\n'
9
10  # Dissect HTTP header into line(first line), header(second line to end), body
11  def parseHTTP(data):...
14
15
16  # Receive HTTP packet with socket
17  # It support seperated packet receive
18  def recvData(conn):...
59
60
61  # HTTP packet class
62  # Manage packet data and provide related functions
63  class HTTPPacket:...
97
98
99  # Proxy handler thread class
100 class ProxyThread(threading.Thread):
101     def __init__(self, conn, addr):...
105
106     # Thread Routine
107     def run(self):...
134
135 def main():...
```

`proxy.py`: The skeleton includes

- Support for Chunked-Encoding
- Basic form of Multithreaded Programming
- Only Comments for proxy-handling codes
- HTTP Receiving codes but no parsing function
- HTTP Packing codes

Guidelines

Your program must take the three parameters

- **Port Number, MT(multithread) option, PC(persistent connection) option**
- `python proxy.py 5555 –mt –pc`
  - **Both MT and PC are enabled**
- `python proxy.py 5555 –mt`
- `python proxy.py 5555 –pc`
  - **Either MT or PC is enabled**
- `python proxy.py 5555`
  - **Nothing is enabled**

Guidelines

Proxy Server (case 1)

```
[root@localhost p2]# python proxy.py 8888
Proxy Server started on port 8888 at 29/Nov/2018 16:33:22.004
* Multithreading – [OFF]
* Persistent Connection – [OFF]

[1] 29/Nov/2018 16:33:25.500
[1] > Connection from 211.143.100.33:3322
[1] > GET http://yonsei.ac.kr/sc/index.jsp HTTP/1.1
[1] < HTTP/1.1 200 OK
[1] < text/html; charset=UTF-8 137142bytes
[1] 29/Nov/2018 16:33:28.800 : (this) 3300ms (average) 3300ms

[2] 29/Nov/2018 16:33:31.350
[2] > Connection from 211.143.100.33:3325
[2] > GET http://yonsei.ac.kr/sc/image-hello-world.png HTTP/1.1
[2] < HTTP/1.1 404 Not Found
[2] < text/html; charset=UTF-8 2035bytes
[2] 29/Nov/2018 16:33:37.850 : (this) 6500ms (average) 4900ms

KeyboardInterrupt
[root@localhost p2]#
```

port number

start time

show function is enabled or not

[conn No.] start time

[] Client IP:port

[] Request Header First line

[] Response Header First line

[] Response Information

[] end time, this response time, average response time

Second loop..

Terminated with Ctrl+C

Proxy Server (case 2)

```
[root@localhost p2]# python proxy.py 8888 -mt -pc
Proxy Server started on port 8888 at 29/Nov/2018 16:33:22.004
* Multithreading – [ON]
* Persistent Connection – [ON]

[1] 29/Nov/2018 16:33:25.500
[1] > Connection from 211.143.100.33:3322
[1] > GET http://yonsei.ac.kr/sc/index.jsp HTTP/1.1
[1] < HTTP/1.1 200 OK
[1] < text/html; charset=UTF-8 137142bytes
[1] 29/Nov/2018 16:33:25.800 : (this) 300ms (average) 300ms

[2] 29/Nov/2018 16:33:26.350
[2] > Connection from 211.143.100.33:3325
[2] > GET http://yonsei.ac.kr/sc/image-hello-world.png HTTP/1.1
[2] < HTTP/1.1 404 Not Found
[2] < text/html; charset=UTF-8 2035bytes
[2] 29/Nov/2018 16:33:25.850 : (this) 500ms (average) 400ms

KeyboardInterrupt
[root@localhost p2]#
```

Guidelines

You are not asked to build a perfect proxy server.

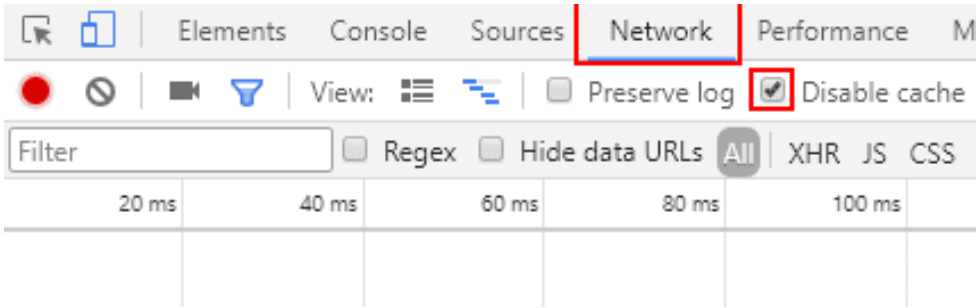
- **Do not consider "HTTPS" connections.**
  - You may encounter CONNECT method.
  - But ignore them or detour them.
- **Your Proxy server should listen to 0.0.0.0**
  - So we can test yours from an external device which does not have an IP address of 127.0.0.1

TA **You** **Website**

Guidelines

How to measure the response time correctly?

- **You should disable "cache" on your web-browser** to measure the correct performance of your proxy server.
- **For chrome browsers, it can be turned off on Developer Tools (F12 key)**



Score Policy

Report (120pts)

- **Introduction/Reference** (10pts)
  - Language, Experiment Setup, Measurement Method
- **Flow chart or Diagram** (15pts)
  - Must show the logic of your program
  - Focus on describing how your client and server work.
- **Snapshots of at least 3 results of different websites**, which prove your codes are working well. (15pts)

Score Policy

Report (120pts)

- **Logical explanations block by block in detail**. (20pts)
- **Comprehensive Analysis of the performance comparison** using the four modes in the objectives slide. (40pts)
  - You need to spend at least 1 page with charts.
  - 5+ repeated measurements are required for each mode.
  - Reasoning, Conclusion…
- **Study of Forward/Reverse Proxy** (20pts)
  - Description, Pros/Cons, When is it needed?

Score Policy

(on **Ubuntu with Python or C**) Code (110pts)

- Your program can
  - Run with **custom Port** (5pts)
  - Handle **external** connections (5pts)
  - Feature **socket-reuse** (**port reuse**) (15pts)
  - Work perfectly **with no error** (20pts)
  - Be **terminated** by only Ctrl+C (5pts)
  - Run with **PC-enabled** only Mode (10pts)
  - Run with **Multithreaded** only Mode (10pts)
  - Run with **MT+PC-enabled** Mode (20pts)
  - **Close the sockets** (by **`netstat`**) (20pts)
    - You observe CLOSE_WAIT? ➔ You are doing wrong.

Score Policy

You will get 0 points if you…

- Copy your friend's codes
  - + Change a little bit of them.
  - + Wish that TAs don't catch that.
- Use a 3rd-party API or codes.
  - **Only except for multithreading**
- **Make your program a liar.**
  - Your report or your program may say a different thing for the same experiment.

Score Policy

Max. 230pts

1	Not submitted / not working / missing files	**0 pts**
2	Overdue ➔ Delay	**-33% pts/day**
3	**The rules or directions whose scores are not specified are not followed**	-10 pts/rule
4	**Any 3rd party framework is used**	**0 pts**
5	**Plagiarizing / Over-implementation (Any kinds of Suspicion of Code-copy)**	**0 pts**
6	**Impolite Report / Lack of Comments**	**0 pts / -50% pts**

Deliverable

- **Only one zip file of "YourID_p2.zip"**
  - **If your ID is 2018147123, 2018147123_p2.zip should be your deliverable file name.**

- **In the zip file only the three files must be included without any folder.**
  - **report.pdf**
  - **proxy.py or proxy.c**
    - **if you use C language, include compile.sh as well**

- **DUE DATE**

  15/Nov/2018 23:59:59 KST

  No exception for exceeding deadline

- **Delay Policy**

-33%pts for ~16/Nov 23:59:59

-66%pts for ~17/Nov 23:59:59

-100%pts for 18/Nov 00:00:00~