KWANGWOON UNIVERSITY

2022년 1학기 **시스템프로그래밍실습** 9주차

Construction Proxy Connection

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

2st Assignment's Descriptions

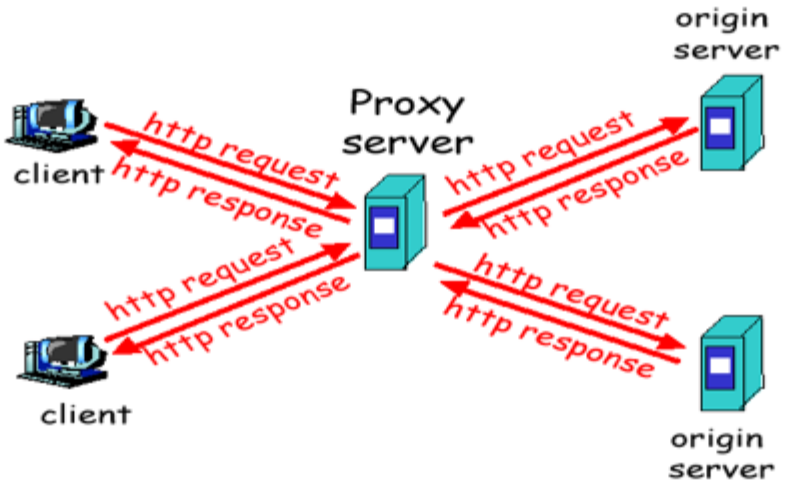
- Assigment 2-1
  - Implement server/client

- **Assignment 2-2**
  - Forward HTTP request to Web server & print the HTTP response

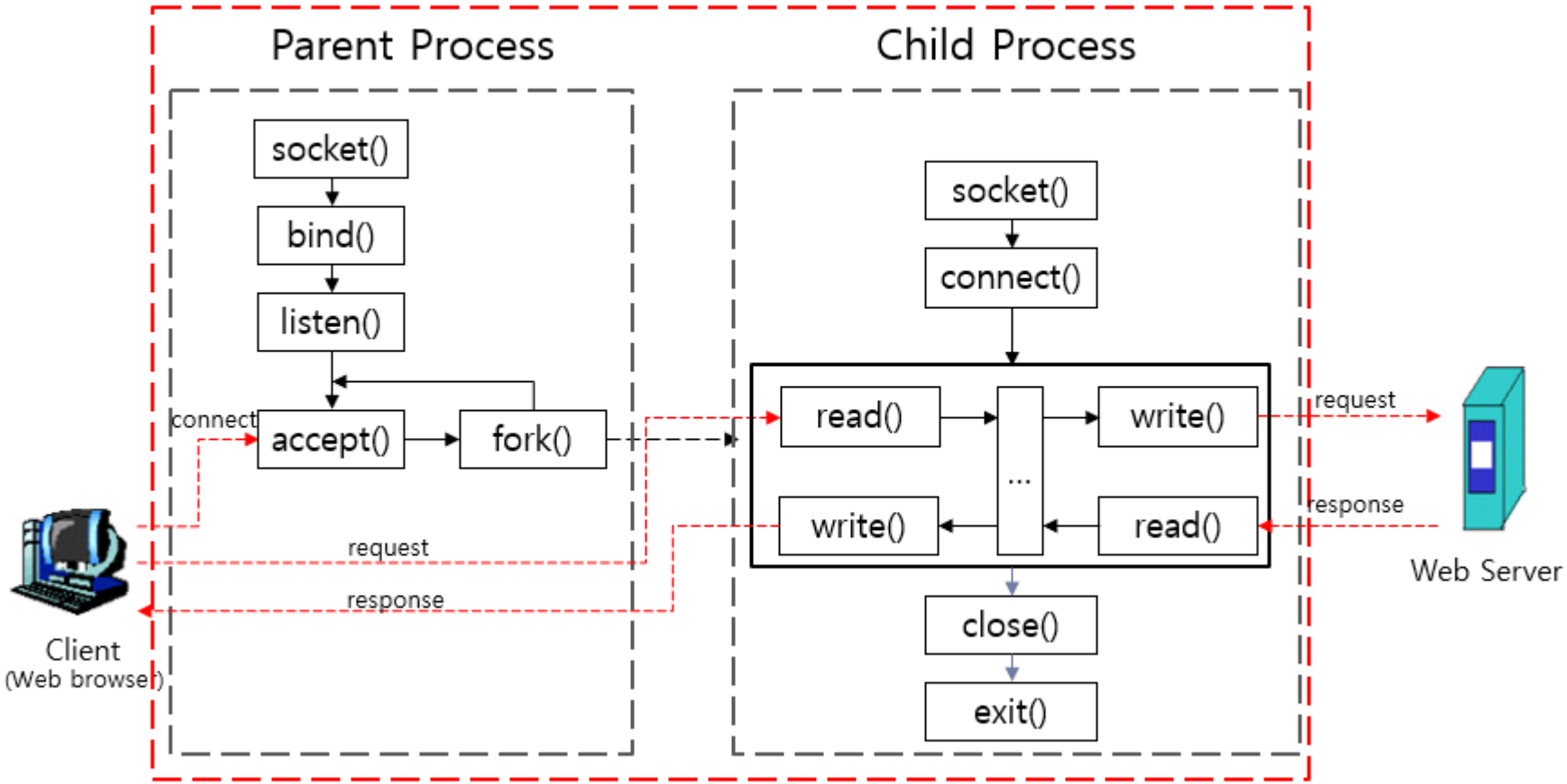
- Assignment 2-3 & 2-4
  - Integrate server side and client side into proxy server

Proxy Server의 동작 (1/2)

1. **Client의 browser에서 proxy server를 설정**
2. Client의 browser는 모든 HTTP request를 proxy server에게 전송
3. HTTP request 정보 중 host 정보를 추출하여 HIT/MISS 판별
    1. MISS
        1. Client의 HTTP request를 web server에게 전송
        2. Server의 Response와 Host 정보를 이용한 Cache Directory 및 file 생성
        3. Response 정보를 client에 전송
    2. HIT
        1. Cache file의 Data를 client에 전송



Proxy Server의 동작 (2/2)

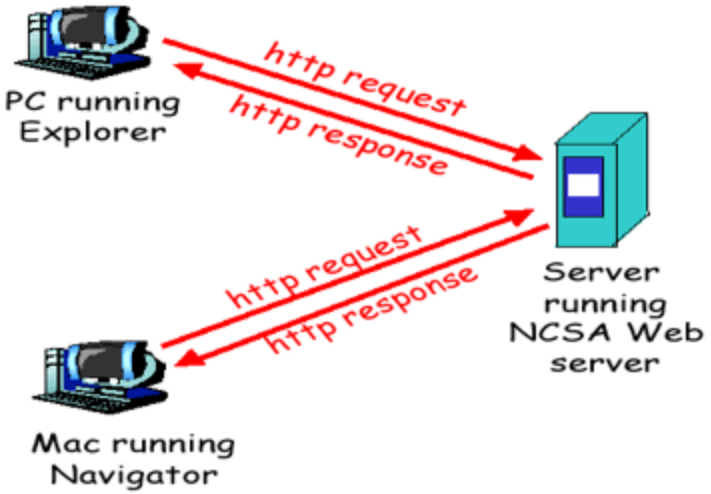


HTTP (1/3)

- **H**yper**T**ext **T**ransfer **P**rotocol
- The **standard** Web transfer protocol
- **WWW(W**orld **W**ide **W**eb)**'s application layer protocol**
- Using **on-demand** method
- **Client/Server model**
  - Client
    - **Browser** request, receives, display WWW objects
  - Server
    - WWW server sends objects in response to request

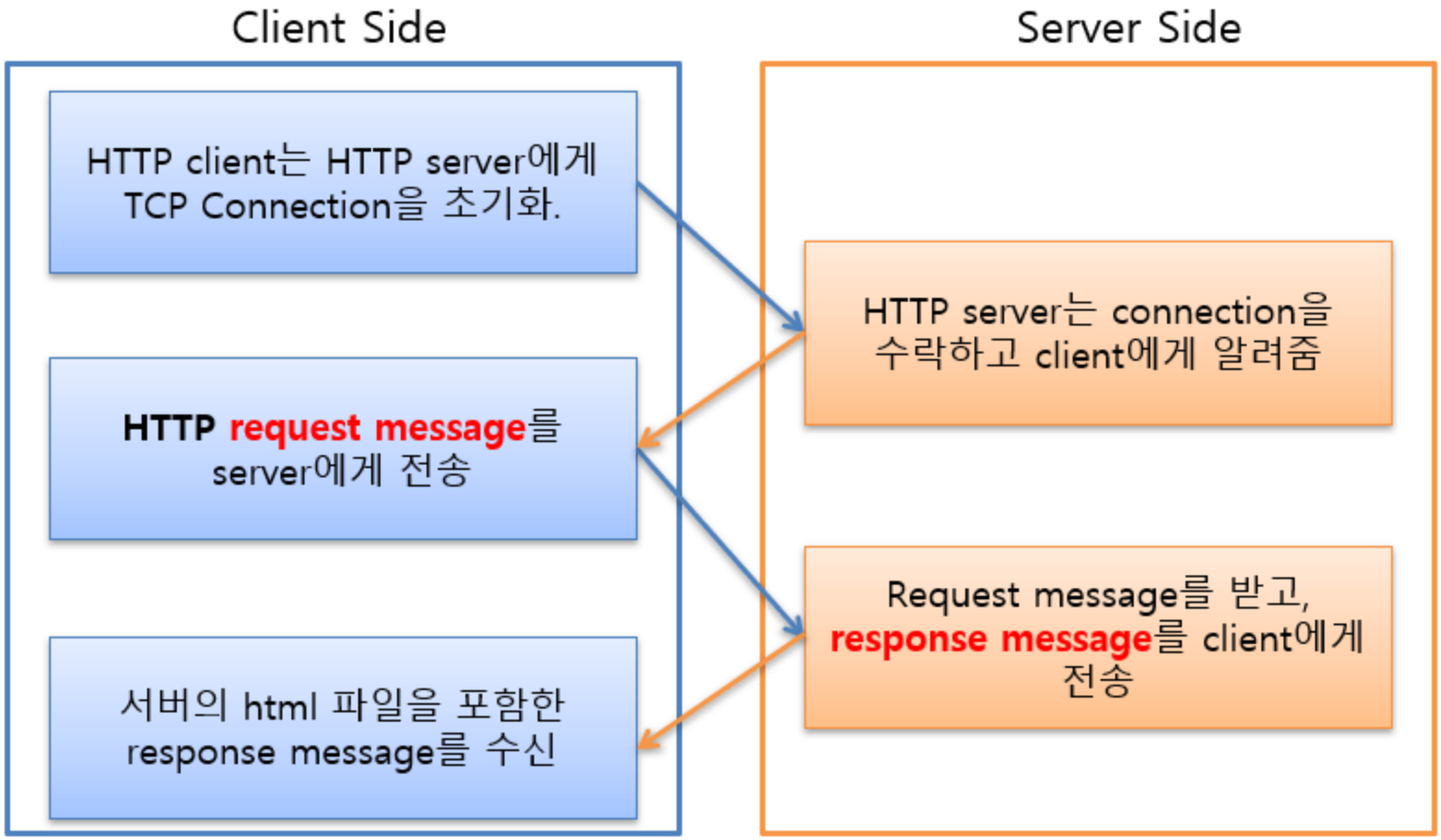
HTTP (2/3)

- Client
  - Web server의 80 port에 TCP 요청
- Server
  - client로부터의 TCP connection을 수락
- Browser와 Web server간의 **HTTP message**들을 교환
  - HTTP Request
  - HTTP Response
- TCP connection 종료



HTTP (3/3)

- **Communication principle of HTTP Client/Server**



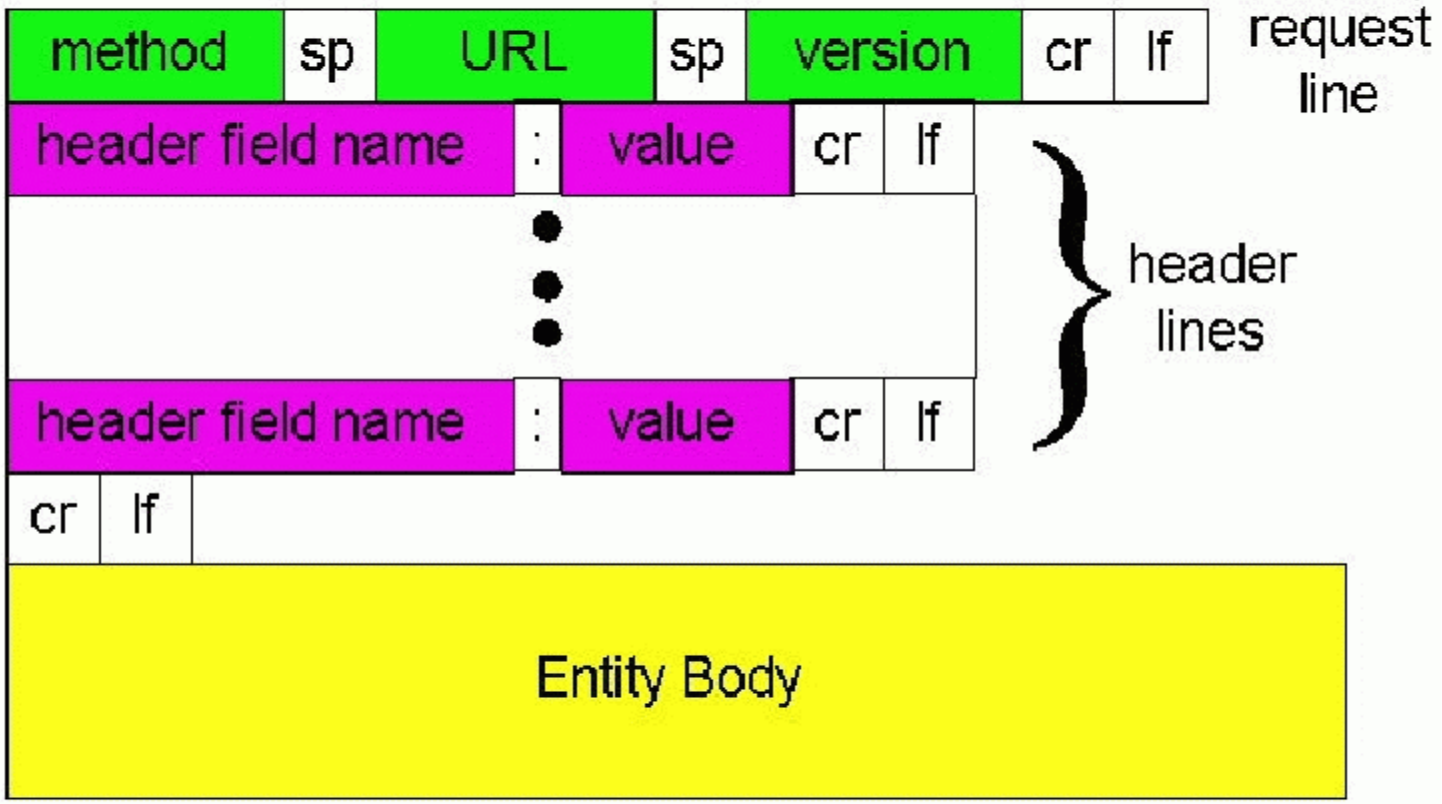
The format of an HTTP Message

- **Two types of HTTP message**
  - request, response

- **The format of an HTTP request**

```
GET http://sswlab.kw.ac.kr/test.html HTTP/1.0
Accept: */*
Accept-Language: ko
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; Net CLR 1.1.4322)
Host: sswlab.kw.ac.kr
Proxy-Connection: Keep-Alive
```

HTTP Request Format

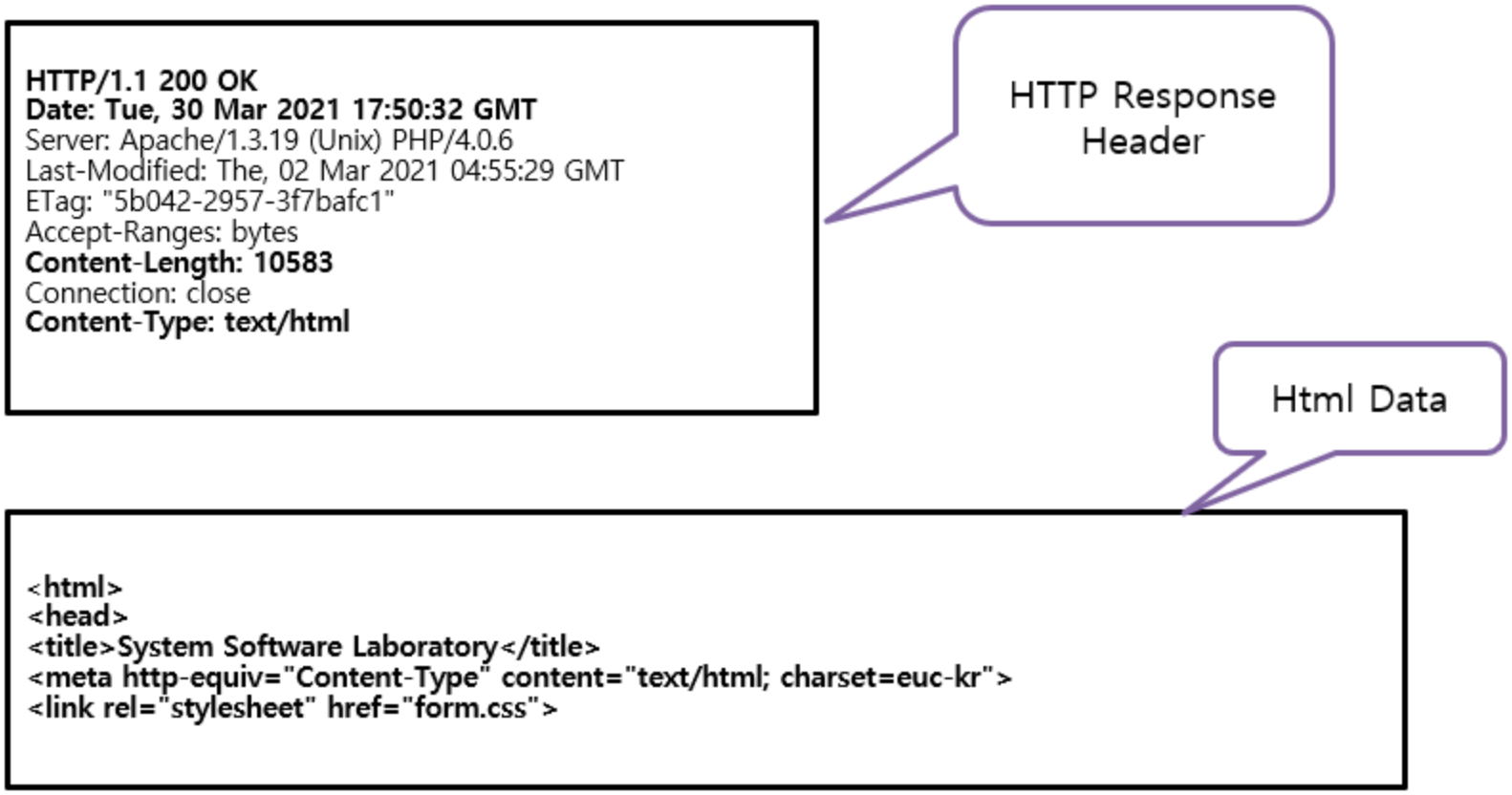


HTTP Request 의 구성

- **Method field**
  - Browser가 서버로 데이터를 전달하는 방법
  - GET, POST, UPDATE, DELETE
- **URL field**
  - Client(browser)가 request한 URL information
- **Accept field**
  - Client(browser)가 실행할 수 있는 application format
- **Accept-Language field**
  - Language
- **User-Agent field**
  - Client의 operation system과 browser의 정보
- **Host field**
  - Request를 요청 받은 host의 URL

HTTP Response

- The format of an HTTP response



HTTP Response 의 구성

- **HTTP/1.1 200 OK**
  - HTTP version과 server response code
- **Data field**
  - 서버가 reply하는 날짜와 시간
- **Last-modified field**
  - 서버가 reply하는 html page가 수정된 날짜와 시간
- **Accept-Ranges field**
  - 서버가 전송하는 데이터의 단위
- **Content-Length field**
  - 서버가 전송하는 데이터의 크기
- **Content-Type field**
  - 서버가 전송하는 데이터의 format

실습1. HTTP Request handling (1/4)

```c
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

#define BUFFSIZE        1024
#define PORTNO          40000

int main()
{
        struct sockaddr_in server_addr, client_addr;
        int socket_fd, client_fd;
        int len, len_out;

        if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
        {
                printf("Server : Can't open stream socket\n");
                return 0;
        }

        bzero((char*)&server_addr, sizeof(server_addr));
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        server_addr.sin_port = htons(PORTNO);

        if (bind(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0)
        {
                printf("Server : Can't bind local address\n");
                return 0;
        }

        listen(socket_fd, 5);
```

실습1. HTTP Request handling (2/4)

```
while (1)
{
        char buf[BUFFSIZE];
        char response_header[BUFFSIZE] = {0, };
        char response_message[BUFFSIZE] = {0, };

        struct in_addr inet_client_address;

        len = sizeof(client_addr);
        client_fd = accept(socket_fd, (struct sockaddr*)&client_addr, &len);
        if (client_fd < 0)
        {
                printf("Server : accept failed\n");
                return 0;
        }

        inet_client_address.s_addr = client_addr.sin_addr.s_addr;

        memset(response_header, 0, sizeof(response_header));
        memset(response_message, 0, sizeof(response_message));
        printf("[%s : %d] client was connected\n", inet_ntoa(inet_client_address), client_addr.sin_port);
        read(client_fd, buf, BUFFSIZE);

        puts("================================================");
        printf("Request from [%s : %d]\n", inet_ntoa(inet_client_address), client_addr.sin_port);
        puts(buf);
        puts("================================================\n");
```

실습1. HTTP Request handling (3/4)

```
            sprintf(response_message,
                    "<h1>RESPONSE</h1><br>"
                    "Hello %s:%d<br>", inet_ntoa(inet_client_address), client_addr.sin_port);

            sprintf(response_header,
                    "HTTP/1.0 200 OK\r\n"
                    "Server:2018 simple web server\r\n"
                    "Content-length:%lu\r\n"
                    "Content-type:text/html\r\n\r\n", strlen(response_message));

            write(client_fd, response_header, strlen(response_header));
            write(client_fd, response_message, strlen(response_message));

            printf("[%s : %d] client was disconnected\n", inet_ntoa(inet_client_address), client_addr.sin_port);
            close(client_fd);
        }
        close(socket_fd);
        return 0;
}
```

실습1. HTTP Request handling (4/4)

- Command 창에서 server 실행
- Web browser가 server에 HTTP Request 전송
- Server는 Web browser로 Response Message (Response Header + Response Data) 전송


```
sslab@ubuntu:~$ ./server
[127.0.0.1 : 62649] client was connected
================================================
==
Request from [127.0.0.1 : 62649]
GET / HTTP/1.1
Host: 127.0.0.1:40000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0)
Gecko/20100101 Firefox/59.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

================================================
==

[127.0.0.1 : 62649] client was disconnected
^C
root @ubuntu:~$
```

실습2. Request URL Parsing (1/4)

```c
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

#define BUFFSIZE        1024
#define PORTNO          39999

int main()
{
        struct sockaddr_in server_addr, client_addr;
        int socket_fd, client_fd;
        int len, len_out;

        if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
        {
                printf("Server : Can't open stream socket\n");
                return 0;
        }

        bzero((char*)&server_addr, sizeof(server_addr));
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        server_addr.sin_port = htons(PORTNO);

        if (bind(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0)
        {
                printf("Server : Can't bind local address\n");
                return 0;
        }

        listen(socket_fd, 5);
```

실습2. Request URL Parsing (2/4)

```
while (1)
{
        struct in_addr inet_client_address;

        char buf[BUFFSIZE];

        char response_header[BUFFSIZE] = {0, };
        char response_message[BUFFSIZE] = {0, };

        char tmp[BUFFSIZE] = {0, };
        char method[20] = {0, };
        char url[BUFFSIZE] = {0, };

        char *tok = NULL;

        len = sizeof(client_addr);
        client_fd = accept(socket_fd, (struct sockaddr*)&client_addr, &len);
        if (client_fd < 0)
        {
                printf("Server : accept failed\n");
                return 0;
        }
        inet_client_address.s_addr = client_addr.sin_addr.s_addr;

        printf("[%s : %d] client was connected\n", inet_ntoa(inet_client_address), client_addr.sin_port);
        read(client_fd, buf, BUFFSIZE);
        strcpy(tmp, buf);
        puts("================================================");
        printf("Request from [%s : %d]\n", inet_ntoa(inet_client_address), client_addr.sin_port);
        puts(buf);
        puts("================================================\n");
```

실습2. Request URL Parsing (3/4)

```
                tok = strtok(tmp, " ");
                strcpy(method, tok);
                if(strcmp(method, "GET") == 0)
                {
                        tok = strtok(NULL, " ");
                        strcpy(url, tok);
                }
                sprintf(response_message,
                        "<h1>RESPONSE</h1><br>"
                        "Hello %s:%d<br>"
                        "%s", inet_ntoa(inet_client_address), client_addr.sin_port, url);

                sprintf(response_header,
                        "HTTP/1.0 200 OK\r\n"
                        "Server:2018 simple web server\r\n"
                        "Content-length:%lu\r\n"
                        "Content-type:text/html\r\n\r\n", strlen(response_message));

                write(client_fd, response_header, strlen(response_header));
                write(client_fd, response_message, strlen(response_message));

                printf("[%s : %d] client was disconnected\n", inet_ntoa(inet_client_address), client_addr.sin_port);
                close(client_fd);
        }
        close(socket_fd);
        return 0;
}
```

실습2. Request URL Parsing (4/4)

- Command 창에서 server 실행
- Web browser가 server에 HTTP Request 전송
- Server는 HTTP request로부터 host 정보(URL) 추출


```
sslab@ubuntu:~$ ./server
[127.0.0.1 : 62691] client was connected
================================================
==
Request from [127.0.0.1 : 62691]
GET /www.kw.ac.kr HTTP/1.1
Host: 127.0.0.1:39999
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0)
Gecko/20100101 Firefox/59.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

================================================
==

[127.0.0.1 : 62691] client was disconnected
^C
root@ubuntu:~$
```

inet_ntoa() Function

```
#include <arpa/inet.h>

char* inet_ntoa(struct in_addr in);
```

- **IPv4 기반 네트워크 주소를 dotted-decimal 형태의 문자열로 변환**
- **Returns**
  - Character pointer to a static buffer containing the text address in standard "." notation
  - If error, NULL
- **Parameter**
  - in
    - Represent an Internet host address

```
struct in_addr {
    // the IP address in network byte order
    in_addr_t s_addr;
}
```

setsockopt() Function (1/2)

```
#include <sys/types.h>
#include <sys/socket.h>

Int setsockopt(int s, int level, int optname, const void* optval, soklen_t optlen);
```

- **소켓의 옵션을 설정**
- **Returns**
  - On success, zero is returned. On error, -1 is returned
- **Parameters**
  - s : socket fd
  - Level : 프로토콜의 단계, 소켓-레벨은 **SOL_SOCKET** 사용
  - optname : option name
    - SO_KEEPALIVE : 주기적인 전송에서 접속 유지
    - SO_REUSEADDR : 포트가 busy상태일 지라도 그것을 계속해서 사용
  - optval, optlen : 옵션 값, optval 길이 -> 대부분 소켓-라벨 옵션은 integer사용

setsockopt() Function (2/2)

- **bind() error**
  - 프로그램을 종료 후에 다시 실행하면 bind()에서 error가 생기는 현상
  - TIME_WAIT state

- **setsockopt()를 이용해 bind()에 의해 생기는 TIME_WAIT 현상을 막을 수 있음**

```
int opt = 1;
server_fd = socket(PF_INET, SOCK_STREAM, 0);
Setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
```

KWANGWOON UNIVERSITY

2022년 1학기 **시스템프로그래밍실습**

Proxy #2-2

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Proxy #2-2 (1/8)

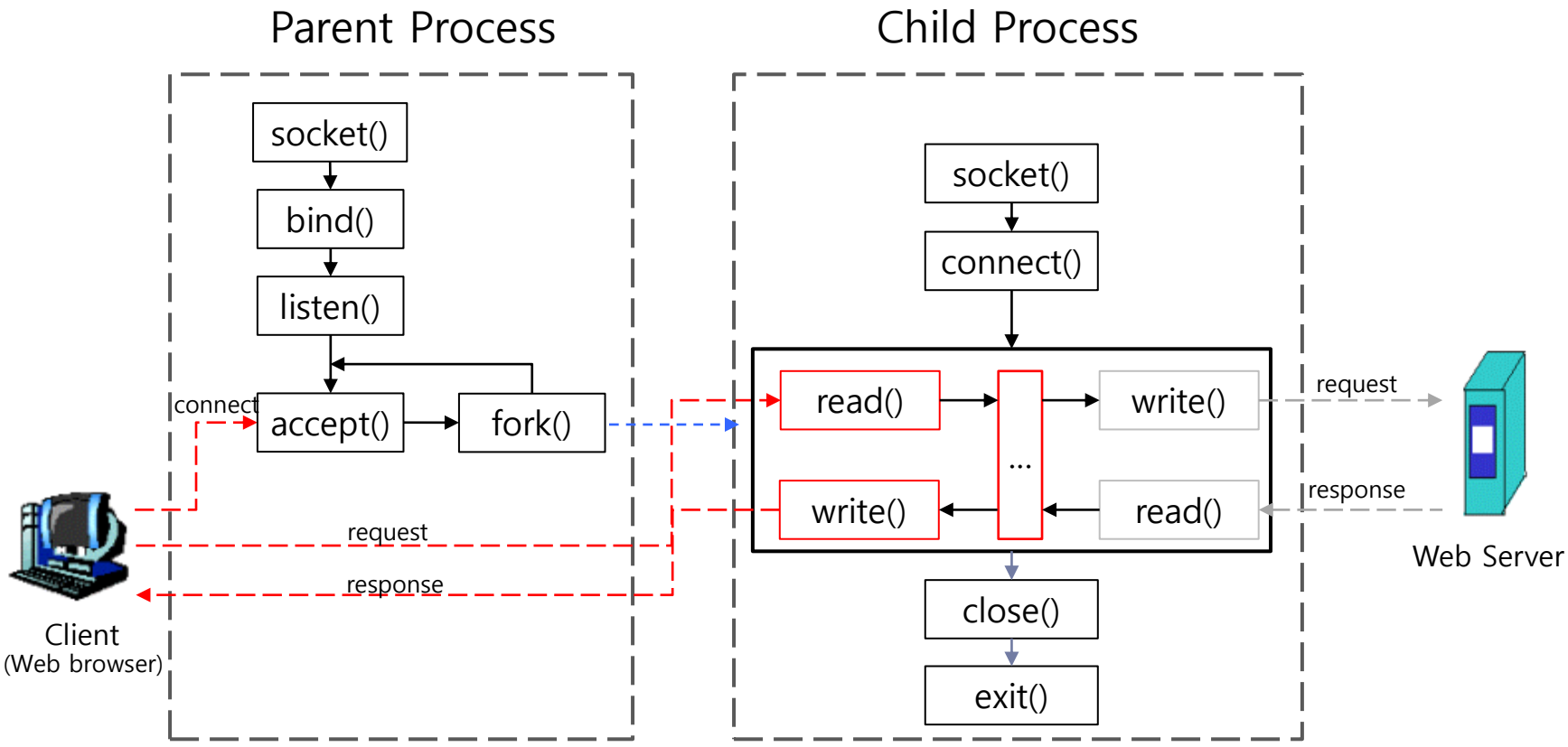
- **Requirements**
  - Proxy server는 Web browser가 URL을 입력할 경우 HTTP request를 받음
  - Child process는 Web browser 의 요청에 응답 후 종료
  - HTTP request header로부터 host 정보(URL) 추출
  - 추출한 host 정보를 Hashed URL로 변환 후 Cache Directory에서 HIT/MISS 판별(Assignment1-2)
    - **HIT**
      - Response header 생성
      - Response Message (Response Header + Response Data) 전송
    - **MISS**
      - Make cache directory and files
      - Response header 생성
      - Response Message (Response Header + Response Data) 전송
    - **Response Data 양식**
      - **HIT**

        HIT (heading)
        ip: port
        kw+본인 학번
      - **MISS**

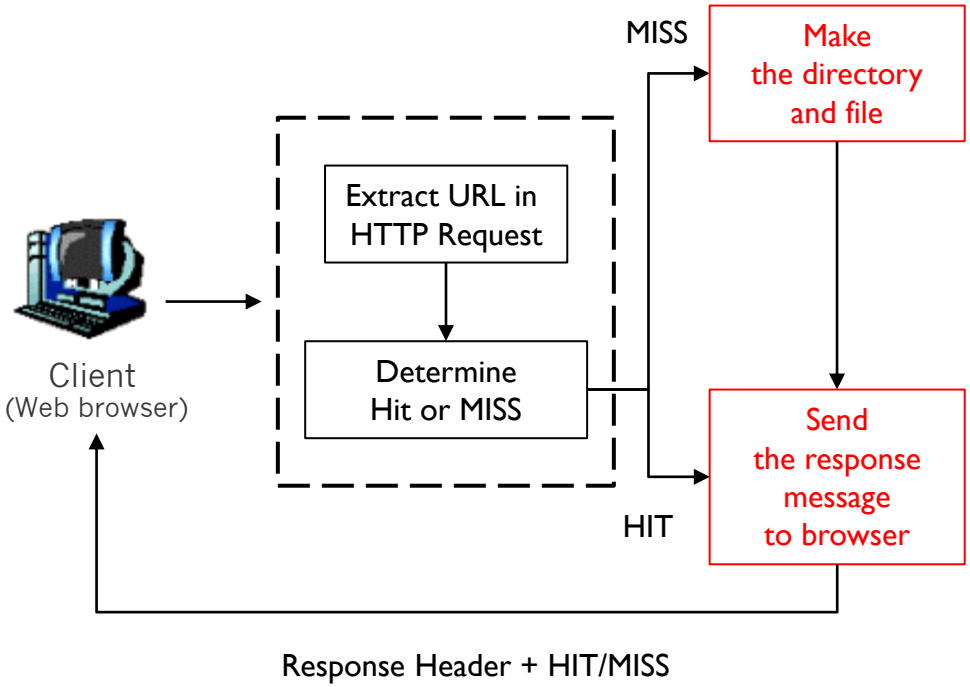
        Miss (heading)
        ip: port
        kw+본인 학번

- **소스 코드 명 : *.h, *.c (자유롭게 구성 가능)**

Proxy #2-2 (2/8)

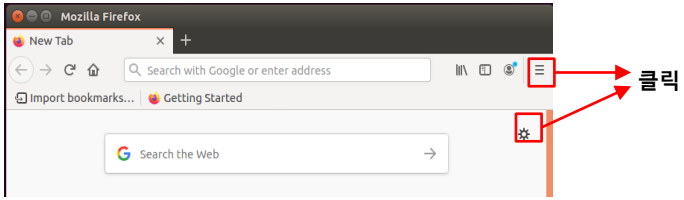

Parent Process
Child Process
socket()
bind()
listen()
connect
accept()
fork()
socket()
connect()
read()
write()
...
write()
read()
request
response
close()
exit()
request
response
Client
(Web browser)
Web Server

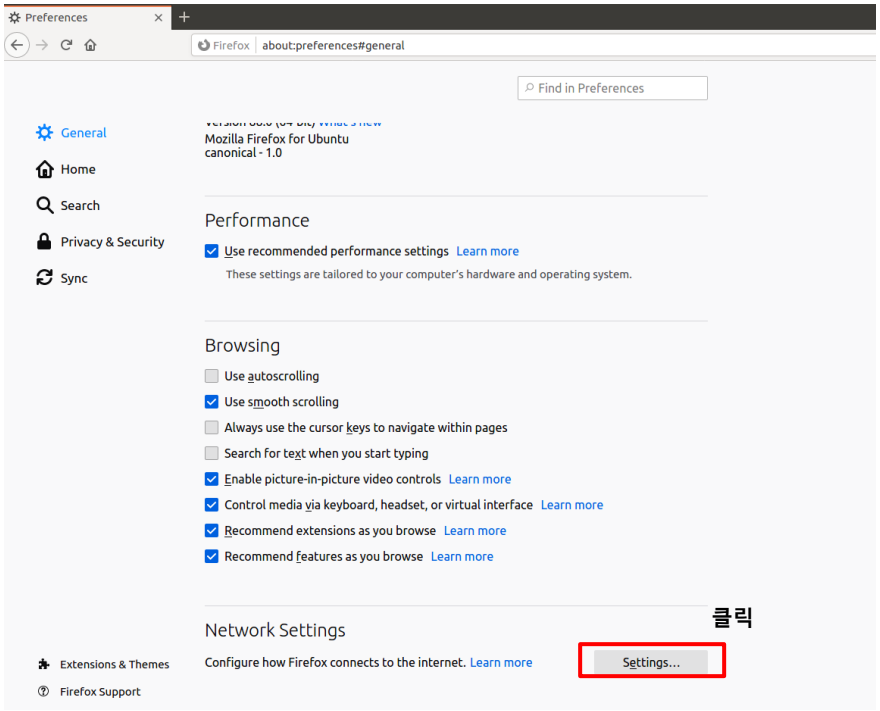
Proxy #2-2 (3/8)


MISS
Make
the directory
and file
Extract URL in
HTTP Request
Determine
Hit or MISS
Client
(Web browser)
HIT
Send
the response
message
to browser
Response Header + HIT/MISS

Proxy #2-2 (4/8)

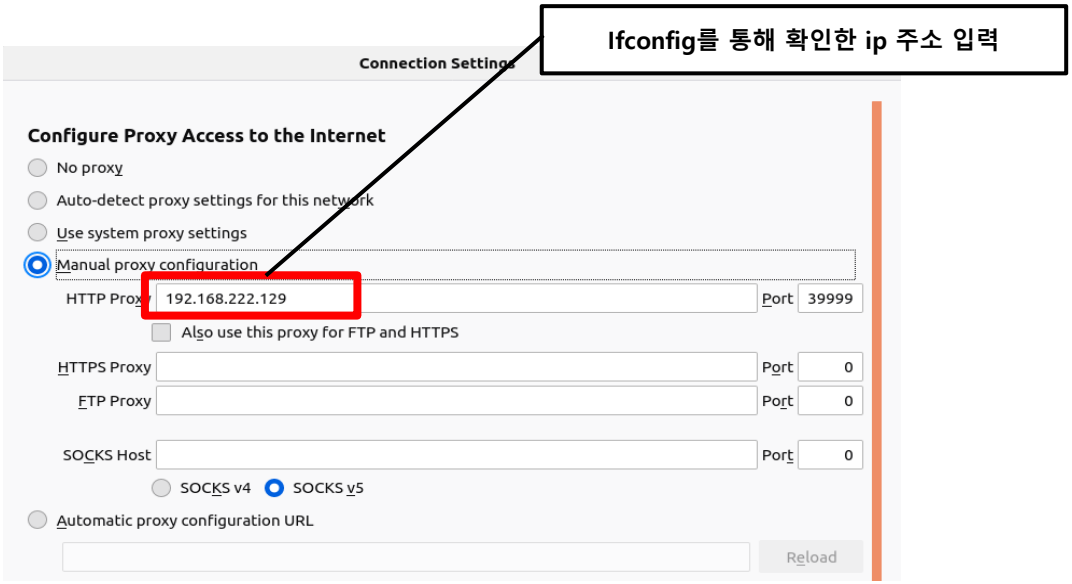
- **프로그램 실행 전 준비 사항**
  - Proxy 설정
    - Firefox로 예시
      - "Preferences" 이동





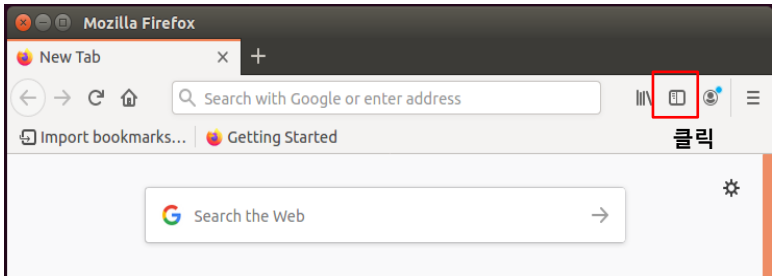
Proxy #2-2 (5/8)

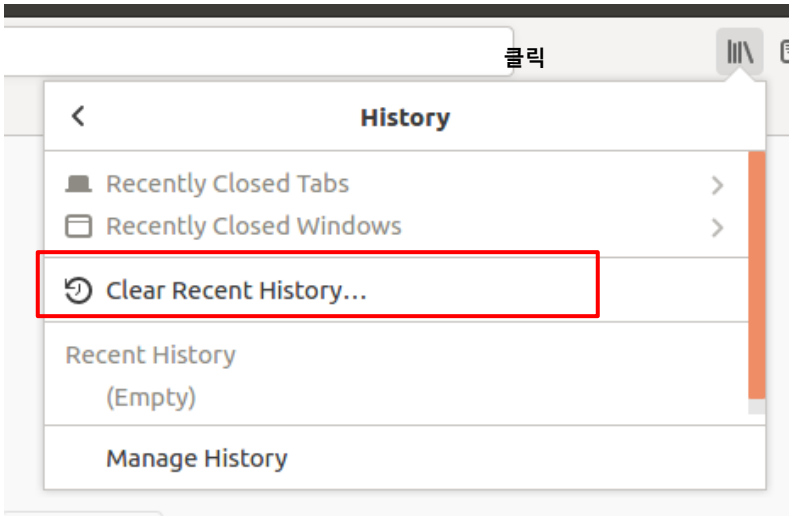
- **프로그램 실행 전 준비 사항**
  - Proxy 설정
    - Command에 ifconfig 명령어를 입력하여 IP 주소 확인
    - 포트 번호: 39999
    - 프록시 설정 시 다음 아래와 같이 설정



Proxy #2-2 (6/8)

- **프로그램 실행 전 준비 사항**
  - 인터넷 기록 지우기





Proxy #2-2 (7/8)

- **실행 예제**

1) Command에 proxy server를 실행

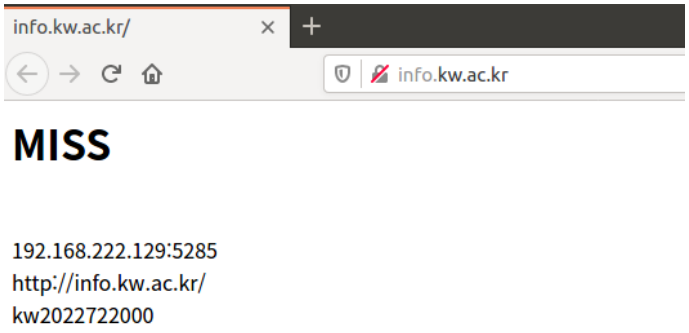
```
$ ./proxy_cache
```

2) proxy server는 accept에서 client(web browser)의 connect를 대기

Proxy #2-2 (8/8)

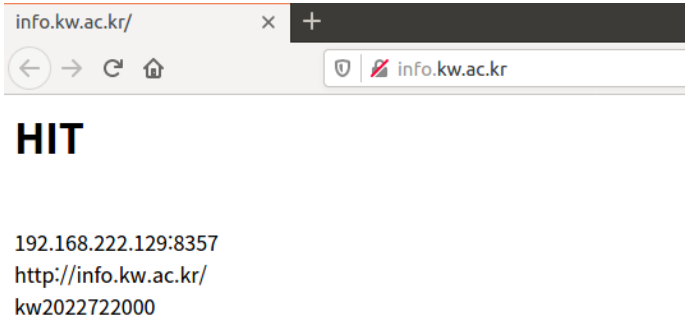
- **실행 예제**

3) Web browser의 요청이 들어오면 이에 응답




```
[192.168.222.129 : 5285] client was connected
================================================
Request from [192.168.222.129 : 5285]
GET http://info.kw.ac.kr/ HTTP/1.1
Host: info.kw.ac.kr
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

================================================
[192.168.222.129 : 5285] client was disconnected
```




```
[192.168.222.129 : 8357] client was connected
================================================
Request from [192.168.222.129 : 8357]
GET http://info.kw.ac.kr/ HTTP/1.1
Host: info.kw.ac.kr
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

================================================
[192.168.222.129 : 8357] client was disconnected
```