Project 1 보고서

2017-11621 전기정보공학부 배병욱

1-1. Webserver

A. Bind & Listen

```
/* TODO : Now bind the host address using bind() call. 10% of score*/
if ( bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1 ){
  perror("bind error");
  exit(1);
}
  //it was mostly same as tutorial

/* TODO : listen on socket you created  10% of score*/
if ( listen(sockfd, 10) == -1 ){
  perror("listen error");
  exit(1);
}
```

same as tutorial

B. Authentication

Function, int authorization(int sock, char *token) returns 1 if authorization is complete and returns 0 if it doesn't. First check buffer whether there's authorization. If auth_key is empty, send HTTP1.1 401 message to require authorization. If auth key exists check whether message's token is same with server's token.

```
if(strcmp(auth_key, token)==0){
  shutdown(sock, SHUT_RDWR);
  close(sock);
  return 1;
}
//auth key is not valid, keeps sending 401 message
else{
  char message[] = "HTTP/1.1 401 Unauthorized\r\nWWW-Authenticate: Basic realm=\"Access to the staging site\"\r\n\r\n";
  sendMessage(sock, message);
}
```

If key is valid stop authorization phase and moves to respond phase.

C. Respond

First, get message from buffer. And get method, path, extension by parsing message.

As we must send different contents of message regarding different extension.

Use void checkExtension(char* extension, char* HTTP_CONTENT)

```
void checkExtension(char* extension, char* HTTP_CONTENT) {
  if(strcmp(extension, "html") == 0)
    strcpy(HTTP_CONTENT, "Content-Type: text/html;");
  else if(strcmp(extension, "png") == 0)
    strcpy(HTTP_CONTENT, "Content-Type: image/png;");
  else if(strcmp(extension, "css") == 0)
    strcpy(HTTP_CONTENT, "Content-Type: text/css;");
  else if(strcmp(extension, "js") == 0)
    strcpy(HTTP_CONTENT, "Content-Type: text/js;");
  else{
    printf("error\n");
    return;
  }
  printf("HTTP_CONTENT is %s\n",HTTP_CONTENT);
}
```

Check file_path and find out wheter file is present at server. And if file is not present send HTTP1.1 404 not found. If file exists, make file header with HTTP/1.1 200 OK and attach message using checkExtension. For contents of requested files, it can be bigger than buffer size. So cut them with block and send them sequentially until EOF. Then finish HTTP message with \r\n\r\n and shut down socket.

```
while(!feof(file)){
  int cur_size = fread(HTTP_BLOCK, 1, 9000, file);
  send(sock, HTTP_BLOCK, cur_size, 0);
  bzero(HTTP_BLOCK, 9000);
}
```

D. Result

Fig 1. Page before authentication.

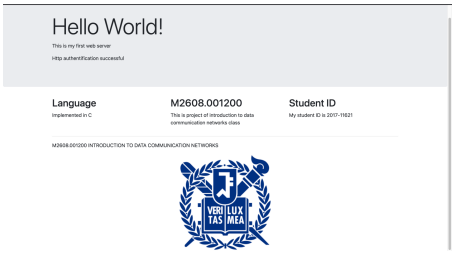

Fig 2. Page after authentication.

1-2. Torrent

A. Message protocol

For REQUEST_XXX messages. Just send message by connected socket and then close socket.

For PUSH_XXX messages. Same for sending messages but need additional step of sending binary messages that will send directly to socket. Size of binary messages can be calculated by checking structure of torrent_file.

```
// Message protocol: "PUSH_PEERS [MY_LISTEN_PORT] [MY_ID_HASH] [TORRENT_HASH] [TORRENT_NUM_PEERS]"[PEER_IPS][PEER_PORTS]
int push_peers_to_peer(char *peer, int port, torrent_file *torrent)
{
    if (silent_mode == 0)
        printf ("INFO - COMMAND PUSH_PEERS list to peer %s:%d, Torrent %x \n"
        , peer, port, torrent->hash);
    // TODO: Implement (5 Points)
    int sockfd = connect_socket(peer, port);
    if (sockfd < 0)
    {
        return -1;
    }
    char buf[STRING_LEN] = {0};
    torrent_info info;
    copy_torrent_to_info (torrent, &info);
    sprintf(buf, "PUSH_PEERS %d %x %x %d", listen_port, id_hash, info.hash, torrent->num_peers);
    send_socket(sockfd, buf, STRING_LEN);
    send_socket(sockfd, (char *)(torrent->peer_ip), sizeof(char)*MAX_PEER_NUM*STRING_LEN);
    send_socket(sockfd, (char *)(torrent->peer_port), sizeof(int)*MAX_PEER_NUM);
    close_socket(sockfd);
    return 0;
}
```

B. Server routine

First need parsing of messages. We can get cmd and other arguments by using strtok_r.

Then as there's possibility of sending message from server itself. Check peer_id_hash and server's id_hash. If they have same value just continue.

For handling command please check code itself.

But it looks like there's major bug in handling PUSH_PEERS

```
else if (strcmp(cmd, "PUSH_PEERS") == 0)
{
    // A peer sends a list of peers!
    // Peer's Message: "PUSH_PEERS [MY_LISTEN_PORT] [MY_ID_HASH] [TORRENT_HASH] [TORRENT_NUM_PEERS]"[PEER_IPS][PEER_PORTS]
    // Hint: You might want to use get_torrent(), or add_peer_to_torrent().
    //       Dont forget to add the peer who sent the list(), if not already added.
    unsigned int torrent_hash;
    int rcv_peer_num;
    sscanf(arg[2], "%x", &torrent_hash);
    sscanf(arg[3], "%d", &rcv_peer_num);
    torrent_file *torrent = get_torrent(torrent_hash);

    if (torrent != NULL)
    {
        char rcv_peer_ip[MAX_PEER_NUM][STRING_LEN] = {0};
        int rcv_peer_port[MAX_PEER_NUM] = {0};

        recv_socket(newsockfd, (char *)&rcv_peer_ip, sizeof(sizeof(char)*MAX_PEER_NUM*STRING_LEN));
        recv_socket(newsockfd, (char *)&rcv_peer_port, sizeof(sizeof(int)*MAX_PEER_NUM));

        for(int i=0; i< torrent->num_peers; i++){
            int rcv_peer_idx = get_peer_idx(torrent, rcv_peer_ip[i], rcv_peer_port[i]);
            if (rcv_peer_idx < 0){
                add_peer_to_torrent(torrent, rcv_peer_ip[i], rcv_peer_port[i], NULL);
            }
        }
        if (get_peer_idx(torrent, peer, peer_port) < 0)
        {
            add_peer_to_torrent(torrent, peer, peer_port, NULL);
        }
        torrent->peer_req_num[get_peer_idx(torrent, peer, peer_port)] = 0;
    }
    close_socket(newsockfd);
    // TODO: Implement (5 Points)
}
```

By debugging, it turns out there's no data(NULL) in received rcv_peer_ip or rcv_peer_port. This leads to another bug afterward which will be discussed at D(result)

C. Client routine

For block request, as torrent's peer_req_num is char, we need to type cast to integer to compare with PEER_EVICTION_REQ_NUM and add peer_req_num.

For block info update. Used for loop which iterate through torrent's peer_list.

For peer list request, as we need to choose random peer, I used rand().

```
if(torrent->num_peers == 0) return 0; //peers are not loaded, wait!
int random_peer_idx = rand()%(torrent->num_peers); //get random index from 0~num of peers
```

D.Result

D-1. my server_routine (for my code set test_mode to 1)
At early stage, it seems like working well. But as time goes by, push_peers result such error due to previously explained error.
(INFO - SERVER: Received command PUSH_PEERS 12781 6b3ffbe5 1f77b213 1 from peer 127.0.0.1:12781
ERROR - Peer port is out of range – 49)
This leads to continuous call of INFO - CLIENT: Adding peer 127.0.0.1:12781 to the block request list
Consequently, downloading of heavy file(pdf) stopped and only download of light files(png, txt) succeeded.

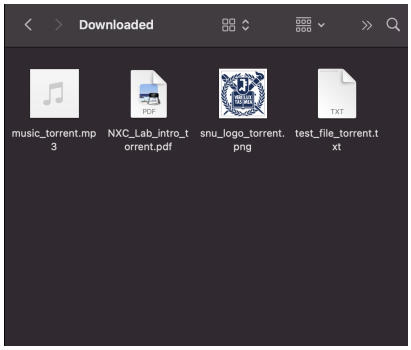

Fig 3. Result of D-1 .

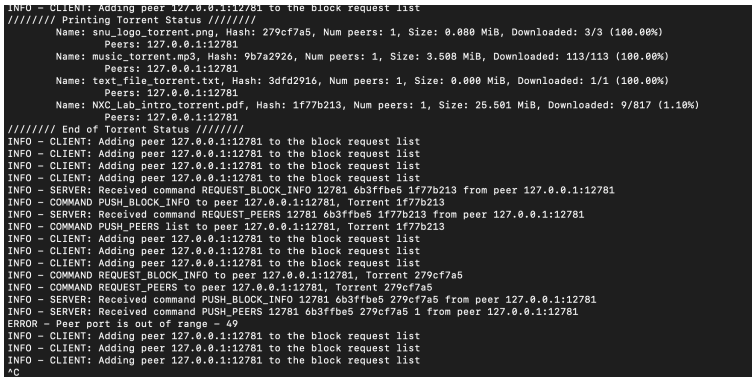

Fig 4. Terminal of D-1

D-2. my client_routine (for my code set test_mode to 2)
It seems like download of every files has succeeded.
D-3. my client_routine & my server_routine
Same result as D-1.

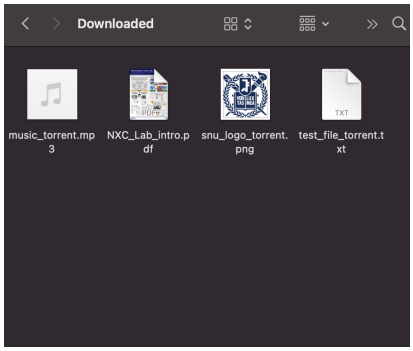

Fig 5. Result of D-2.

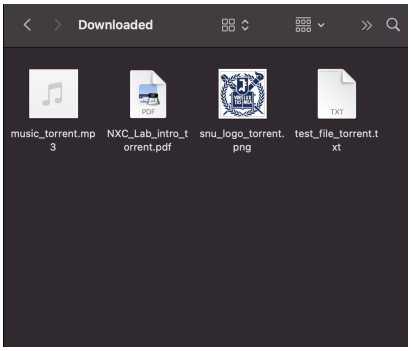

Fig 6. Result of D-3.