네트워크프로그래밍 3주 화상강의

- 시간표
  - 9/29(수)
    - A반 11:00
    - 7반 15:00
    - M반 18:00
  - 9/30(목)
    - N반 11:00
    - 8반 16:00
    - O반 18:00

수업 시작하면
카메라 ON

네트워크프로그래밍-5주
화상강의 자료

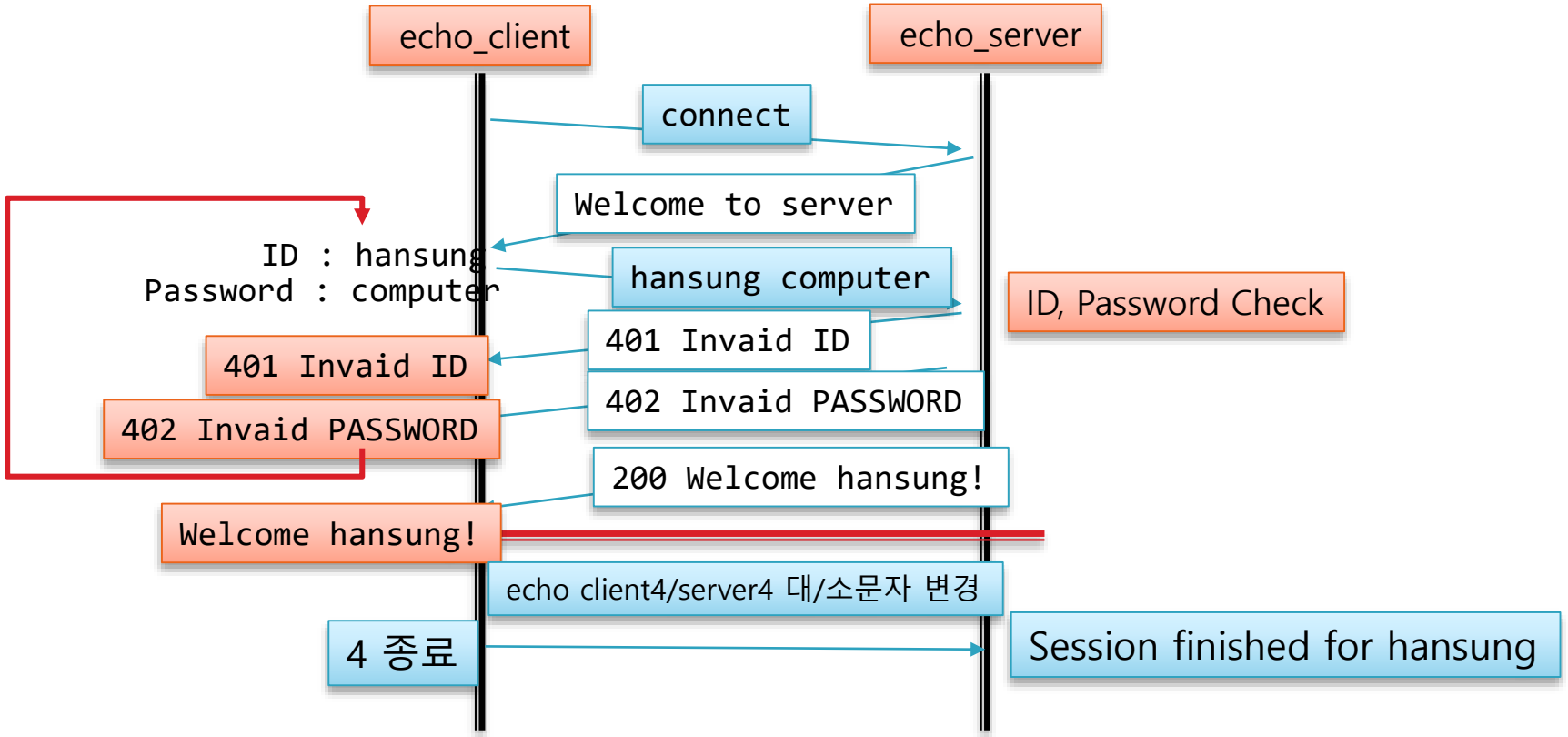
정인환교수

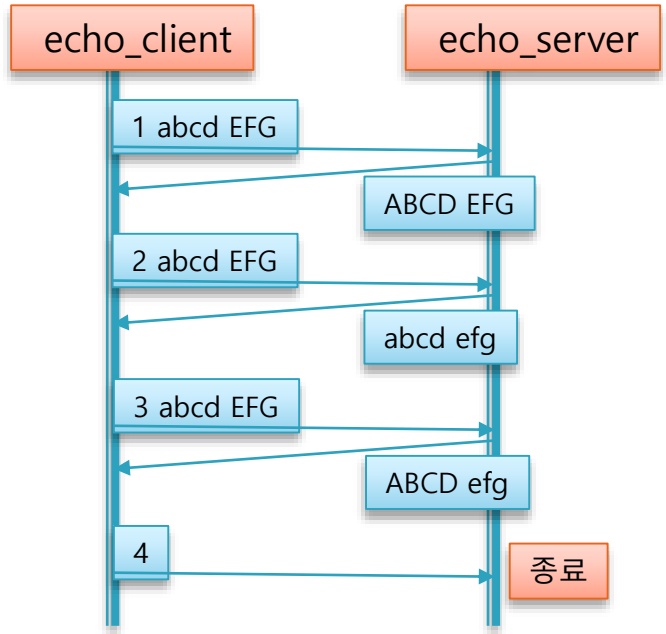
5주 화상강의 내용

- 과제 Review
  - echo_client/server 1, 2, 3, 4, 5
- 원격접속 Test
  - 학생 PC > 교수 PC Port Forwarding Windows 30000- Linux 40000

- 5주 강의 요약
  - file 전송
    - Windows > Windows
    - Windows > Linux
  - 과제
    - FTP 와 같이 양방향 file 전송 만들기 (get, put 기능 구현)

echo client/server 프로토콜





Q&A, 과제 Review

- 프로토콜 표준 준수
  - 표준을 따라야 서로 서로 다른 개발자 사이에 프로그램이 호환이 된다.
  - Client > Server 한번에 보내도록
    - “hansung computer”, “1 hello”, “2 Hi”, “3 Hansung Univ.”, “4”
  - 두번 보내는 경우 → 표준이 아님
    - send(s, id,..) send(s, pass, ..)
    - send(s, “1”, ..) send(s, buf, ..)
  - Login
    - Client : sprintf(buf2, “%s %s”, id, pass); send(s, buf2, 128, 0);
    - Server : recv(client_fd, buf, 128,0); sscanf(buf, “%s %s”, id, pass);
  - 대/소문자 변환
    - Client :

```
if (menu!=4)
     fgets(buf, 128, stdin);
     sprintf(buf2, “%d %s”, menu, buf); send(s, buf2, 128, 0);
else
     sprintf(buf2, “%d”, menu); send(s, buf2, 128, 0);
```

    - Server :

```
recv(cliend_fd, buf, 128,0); menu = buf[0] – ‘0’;
if (menu!=4)
     strncpy(buf2, buf+2, msg_size – 2);
char *s = buf2;
// s 로 변환
send(client_fd, buf2, 128, 0);
```

Q&A, 과제 Review

- 문자열 선언시 memory 확인
  - char *welcome = “Welcome server.”;
  - send(clien_fd, welcome, 128, 0); // welcome은 128 byte가 아님
- strtok() 사용 주의
  - recv(client_fd, buf, 128, 0); // “hansung computer” 라면
  - id, password 를 strtok()로 분리하는 경우 Linux 쪽에서 오류 발생.
  - sscanf(buf, “%s %s”, id, pass); // 이런 식으로 해결
- 기타

4주 과제 Review

- echo_client4

```c
int menu_scr()
{
    int menu;
    do {
        printf("*** 대/소문자 변환 메뉴입니다. ***\n");
        printf(" (1) 모두 대문자 변환\n");
        printf(" (2) 모두 소문자 변환\n");
        printf(" (3) 대>소 소>대 변환\n");
        printf(" (4) 종료\n");
        printf("선택하세요 : ");
        scanf("%d", &menu); getchar(); // read \n
    } while (menu < 1 || menu>4);
    return menu;
}
```

```c
while (1) {
    int menu;
    /* 키보드 입력을 받음 */
    menu = menu_scr();
    if (menu != 4) {
        memset(buf, '\0', BUF_LEN);
        printf("Input string : ");
        if (fgets(buf, BUF_LEN, stdin)) {
            len_out = BUF_LEN;
        }
        else {
            printf("fgets error\n");
            exit(0);
        }
        sprintf(buf2, "%d %s", menu, buf);
    }
    else
        sprintf(buf2, "%d\n", menu);
    /* echo 서버로 메시지 송신 */
    if (send(s, buf2, BUF_LEN, 0) < 0) {
        printf("write error\n");
        exit(0);
    }
    // 4번 선택하면 종료
    if (menu == 4) {
        break;
    }
    if ((n = recv(s, buf, BUF_LEN, 0)) < 0) {
        printf("read error\n");
        exit(0);
    }
    printf("Received %d bytes : %s\n", n, buf);
}
closesocket(s);
```

4주 과제 Review

- echo_server4/5 대/소문자 변환

```
// 대/소문자 변환 처리
while (1) {
    msg_size = recv(client_fd, buf, BUF_LEN, 0);
    if (msg_size <= 0) {
        printf("recv error\n");
        break;
    }
    printf("Received len=%d : %s\n", msg_size, buf);
    memset(buf2, 0, BUF_LEN);
    strncpy(buf2, buf + 2, msg_size - 2);
    char* s = buf2;
    if (buf[0] == '1') {// 모두 대문자 변경
        while (*s) {
            *s = toupper(*s);
            s++;
        }
    }
    else if (buf[0] == '2') { // 모두 소문자 변경
        while (*s) {
            *s = tolower(*s);
            s++;
        }
    }
    else if (buf[0] == '3') { // 대->소 소->대 변경
        while (*s) {
            if (islower(*s))
                *s = toupper(*s);
            else
                *s = tolower(*s);
            s++;
        }
    }
    else if (buf[0] == '4') { //exit 대신 4를 종료 처리
        printf("Session finished for %s.\n", id);
        break;
    }
    printf("Sending len=%d : %s\n", BUF_LEN, buf2);
    msg_size = send(client_fd, buf2, BUF_LEN, 0);
    if (msg_size <= 0) {
        printf("send error\n");
        break;
    }
}
```

echo_client5, echo_server5 로그인 기능

```c
void input_id_pass(char *id, char *pass) {
    printf("ID : ");
    scanf("%s", id);
    printf("Password : ");
    scanf("%s", pass);
    getchar(); // read '\n'
}
```

echo_client5.c

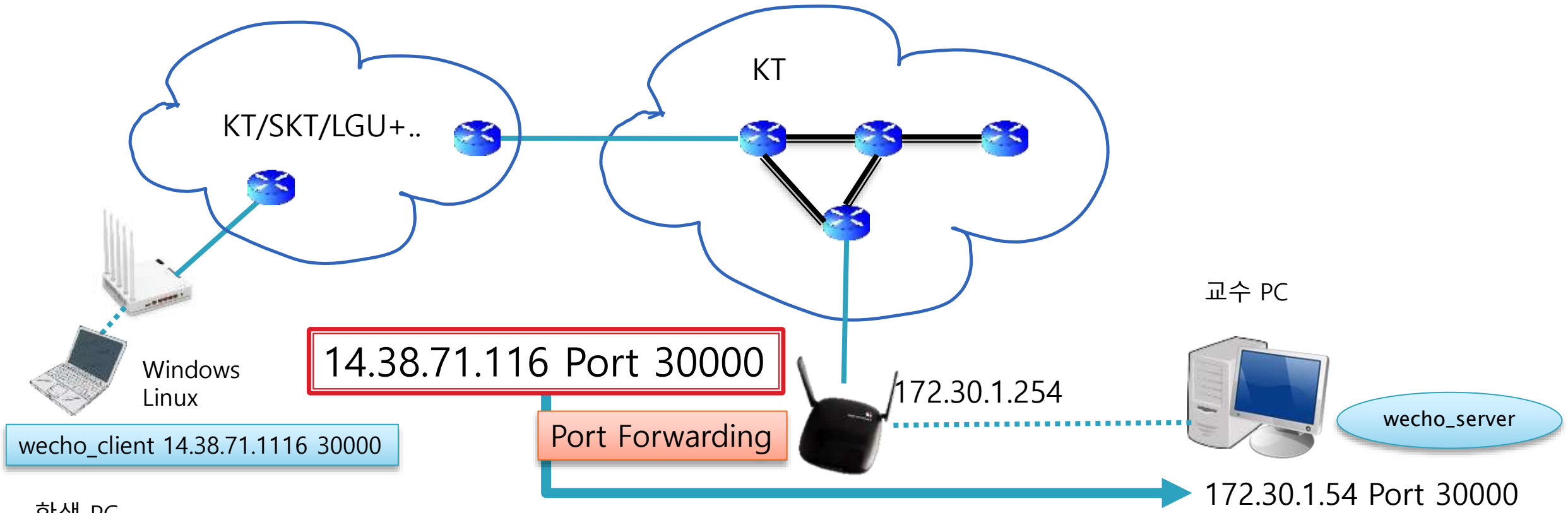
```c
/* 연결요청 */
if (connect(s, (struct sockaddr*)&server_addr,
    sizeof(server_addr)) < 0) {
    printf("can't connect.\n");
    exit(0);
}
// Welcome message 수신
if ((n = recv(s, buf, BUF_LEN, 0)) <= 0) {
    printf("read error\n");
    exit(0);
}
printf("Received %d bytes : %s\n", n, buf);
while (1) {
    char id[BUF_LEN], pass[BUF_LEN];
    input_id_pass(id, pass);
    sprintf(buf2, "%s %s", id, pass);
    /* echo 서버로 메시지 송신 */
    if (send(s, buf2, BUF_LEN, 0) <= 0) {
        printf("send error\n");
        exit(0);
    }
    if ((n = recv(s, buf, BUF_LEN, 0)) <= 0) {
        printf("recv error\n");
        exit(0);
    }
    if (strncmp(buf, LOGIN_OK, 3) == 0) { // ID, Password 정상
        printf("%s\n", &buf[4]); // Welcome Message 만 출력
        break;
    }
    else {
        printf("%s\n", buf);
    }
```

echo_server4.c

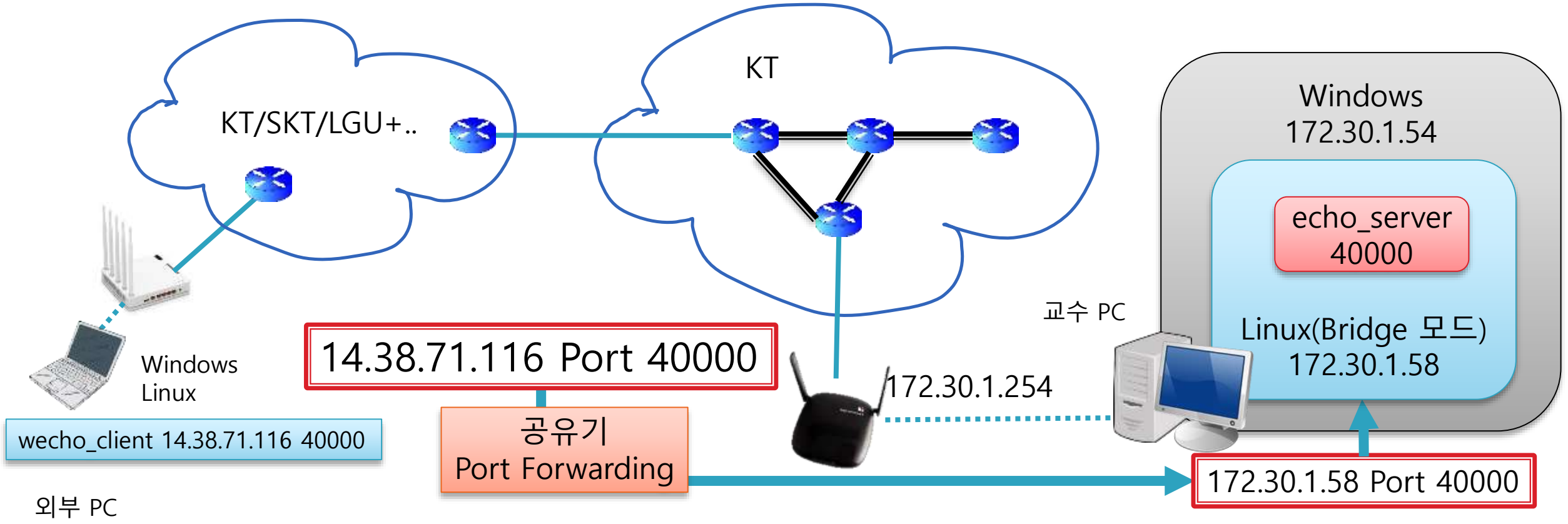
```c
printf("Client connected from %s:%d\n", inet_ntoa(client_addr.sin_a
printf("client_fd = %d\n", client_fd);

sprintf(buf, "Welcome to Server!!");
printf("Sending len=%d : %s\n", BUF_LEN, buf);
msg_size = send(client_fd, buf, BUF_LEN, 0);
if (msg_size <= 0) {
    printf("send error\n");
    break;
}
// Login 처리
while (1) {
    msg_size = recv(client_fd, buf, BUF_LEN, 0);
    if (msg_size <= 0) {
        printf("recv error\n");
        goto CLOSE_CLIENT;
    }
    sscanf(buf, "%s %s", id, pass);
    printf("Received id=%s pass=%s\n", id, pass);
    memset(buf2, 0, BUF_LEN);
    if (strcmp(id, LOGIN_ID) != 0) {
        sprintf(buf2, "%s", LOGIN_INVALID_ID);
        printf("Sending len=%d : %s\n", BUF_LEN, buf2);
        send(client_fd, buf2, BUF_LEN, 0);
    }
    else if (strcmp(pass, LOGIN_PASS) != 0) {
        sprintf(buf2, "%s", LOGIN_INVALID_PASS);
        printf("Sending len=%d : %s\n", BUF_LEN, buf2);
        send(client_fd, buf2, BUF_LEN, 0);
    }
    else { // id, password OK
        sprintf(buf2, "%s Welcome %s!!", LOGIN_OK, id);
        printf("Sending len=%d : %s\n", BUF_LEN, buf2);
        send(client_fd, buf2, BUF_LEN, 0);
        break;
    }
}
// 대/소문자 변환 처리
```

학생 PC echo_client <WiFi> <ISP> 교수 Windows echo_server

14.38.71.116 Port 30000



외부 PC echo_client <WiFi> <ISP> 교수 Linux echo_server



Port Forwarding 설정 (KT 공유기)

5주 강의 요약

- 파일 전송
- file_client/server 기본 + 응용
- 과제
  - 5주 강의 복습
  - ftp 명령어와 같이 구현
    - file_client/server 양방향으로 구현
    - get/put/dir/quit/ldir/!cmd 기능 구현