인터넷 프로토콜 5장

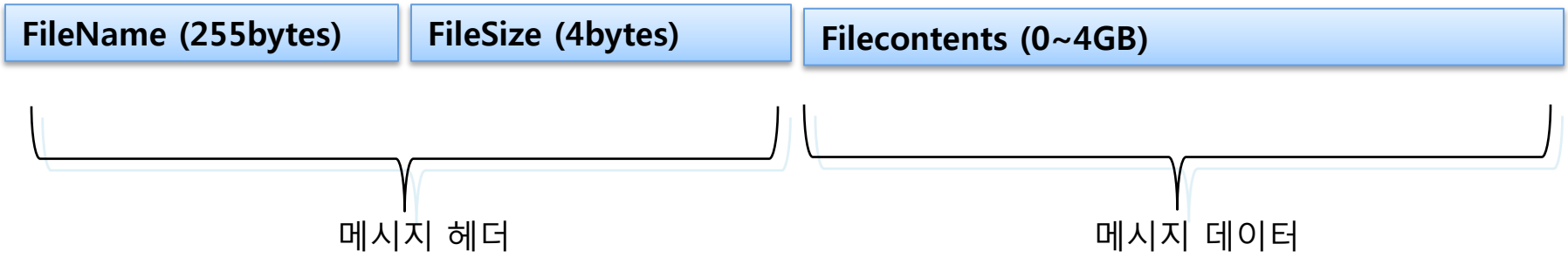
데이터 송수신 (3)

목포해양대 해양컴퓨터공학과

파일 전송 메시지 구성 예제 (고정 크기 메시지)

- 전송 방식: 고정 크기(바이너리 전송)
- 필요한 전송 정보
  - 파일 이름(최대 255자 => 255byte의 메모리 공간 필요)
  - 파일 크기(4byte의 경우 최대 4GB 크기의 파일 처리 가능)
  - 파일 내용(가변 길이, 0~4GB 크기)

- 메시지 구성




```
struct msghdr {
    char filename[255];
    unsigned int filesize;
}
```

파일 전송 메시지 구성 예제 (문자열 메시지)

- 전송 방식 : 가변 길이(문자열 전송 방식)
- 필요한 전송 정보
  - 구분자 (!$: 임의로 결정, 단 파일 이름이나 길이에 나오지 않아야함)
  - 파일 이름(크기 제한 없음, 필드의 크기는 파일 이름에 따라 가변)
  - 파일 크기(크기 제한 없음, 필드의 크기는 파일 크기에 따라 가변)
  - 파일 내용(가변 길이, 0~4GB 크기)
- 메시지 구성(255bytes의 helloworld.c를 전송할 경우)

파일 내용의 전송

- 파일 크기가 소켓 버퍼의 크기보다 크므로 아래와 같이 순차적 전송

```
#define BUFSIZE 1024

char fileBuf[BUFSIZE];

fp = fopen("test.txt", "r");
if(fp == NULL)
    DieWithError ("File open error");

while(1){
    len=fread(fileBuf, sizeof(char), BUFSIZE, fp);
    send(sock, fileBuf, len, 0);
    if(feof(fp))
        break;
}
```

파일 내용의 수신

```
#define BUFSIZE 1024

char fileBuf[BUFSIZE];
recvFileSize=0;

fp = fopen("test.txt", "w");

if(fp == NULL)

        DieWithError ("File open error");

while(origFileSize>recvFileSize) {

        if ((recvMsgSize = recv(clntSock,fileBuf,BUFSIZE, 0)) < 0)

                DieWithError("recv() failed");
        recvFileSize+=recvMsgSize;
        fwrite(fileBuf, sizeof(char), BUFSIZE, fp);

}
```

응용과제 2

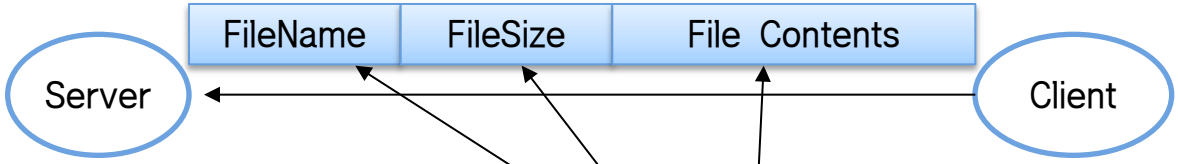
- 기존 에코 프로그램을 수정하여 다음의 기능을 가지는 프로그램을 작성하라
  - 클라이언트: 클라이언트 명령어의 두 번째 인자를 에코 문자열 대신 파일 이름으로 받아들여 파일을 서버에 전송하라
    - Ex) FileClient lily.mmu.ac.kr test.txt 5000
  - 서버: 해당 파일은 서버의 실행파일이 존재하는 디렉토리에 동일한 이름으로 저장되도록 한다.

좀 더 복잡한 메시지 구성

메시지 구성

- 파일 전송의 예

```
char FileName[256];
int FileSize;
char FileBuffer[1024];
```




```
recv(clntSock, FileName, 256, 0)
recv(clntSock, FileSize, 4, 0)
while(notRecvfullFile()) {
    recv(sock, FileBuffer, 1024,0)
}
```

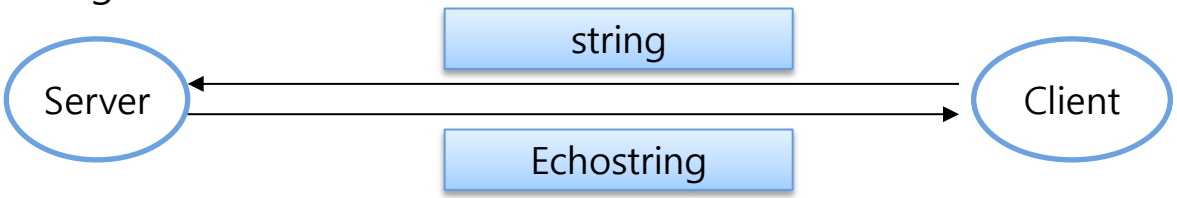
```
send(sock, FileName, 256, 0)
send(sock, FileSize, 4, 0)
while(feof(fd)) {
    Send(sock, FileBuffer, 1024,0)
}
```

에코와 파일 전송을 모두 지원하는 프로토콜

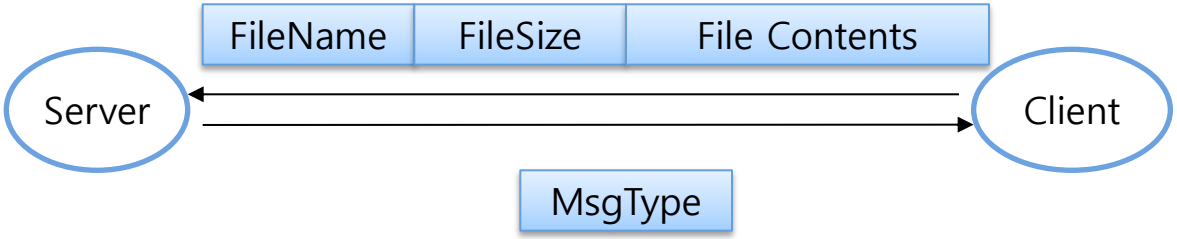
- 상황
  - 클라이언트는 서버에게 string 혹은 파일을 업로드 할 수 있다.
  - 서버는 string을 받을 경우 echo를 해주고 파일을 받을 경우, 디스크에 저장을 한 후 잘 받았다는 메시지를(acknowledge) 회신한다
  - 클라이언트는 echo메시지를 수신한 경우 ,echo 메시지를 출력하고, file ack를 받을 경우 file ack를 출력한다
- 예
  - client lily.mmu.ac.kr upload test.txt 5000 // 파일 전송
  - client lily.mmu.ac.kr echo hello 5000 // 에코 메시지

메시지(프로토콜) 설계

- 서버와 클라이언트는 동일 프로그램으로 두 개의 다른 상황을 모두 만족해야 함
  - EchoString



  - File Upload

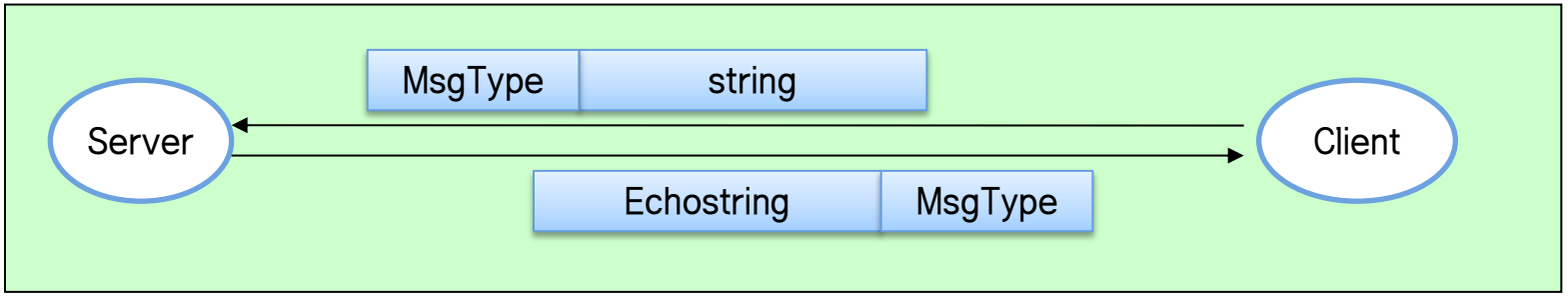


- 서버의 입장에서 클라이언트의 서비스 요청이 에코요청인지 파일업로드인지 구분할 수 있는 방법은?
  - 서비스 타입(에코 요청, 파일업로드) 필드를 준비하고 클라이언트를 이를 통해 서버에게 서비스 종류를 알림

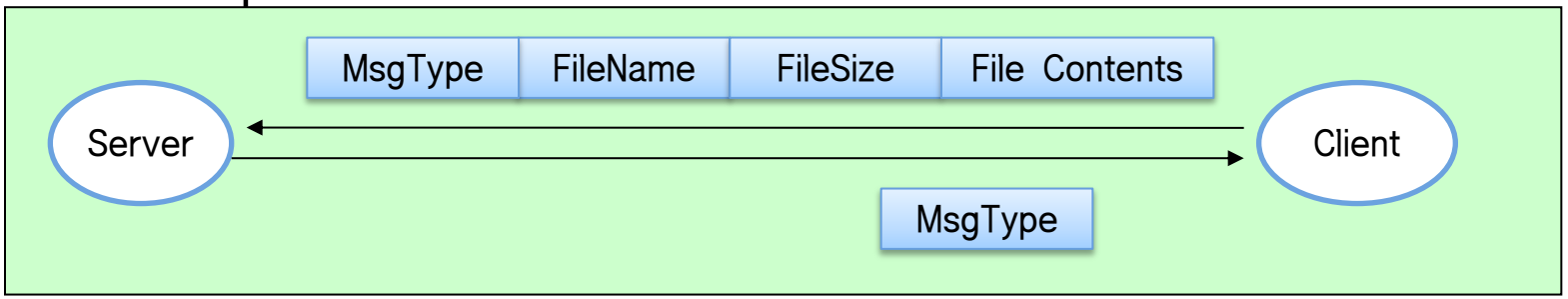
바이너리 프로토콜 설계

```
/* Message Type */
#define EchoReq      01
#define FileUpReq    02
#define EchoRep      11
#define FileAck      12
char MsgType;
```

- EchoString

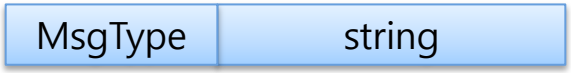


- File Upload



클라이언트 핵심 코드 (바이너리 프로토콜)

```
/* Message Type */
#define      EchoReq     01
#define      FileUpReq   02
#define      EchoRep     11
#define      FileAck     12
char MsgType;
char * operation;  //client 203.252.164.144 upload test.txt 5000
…
operation=argv[2]
…
If (!strcmp(operation,"upload"))
            MsgType=FileUpReq
            send(sock, &MsgType, 1,0)
            …
else if (!strcmp(operation,"echo"))
            MsgType=EchoReq
            send(sock, &MsgType,1,0)
            …
else {
    fprintf(stderr, "Usage: %s <Server IP> <operation><operand> <Echo Port>₩n", argv[0]);
    exit(1);
}
```



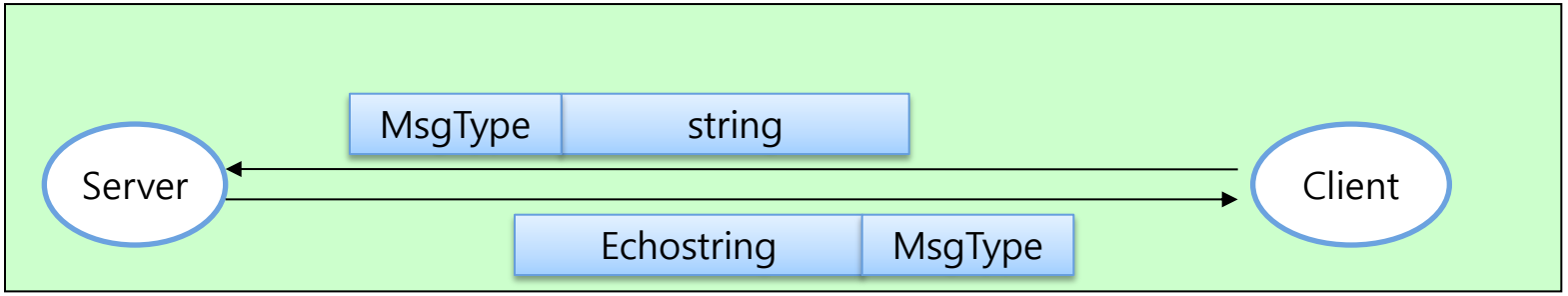
서버 핵심 코드 (바이너리 프로토콜)

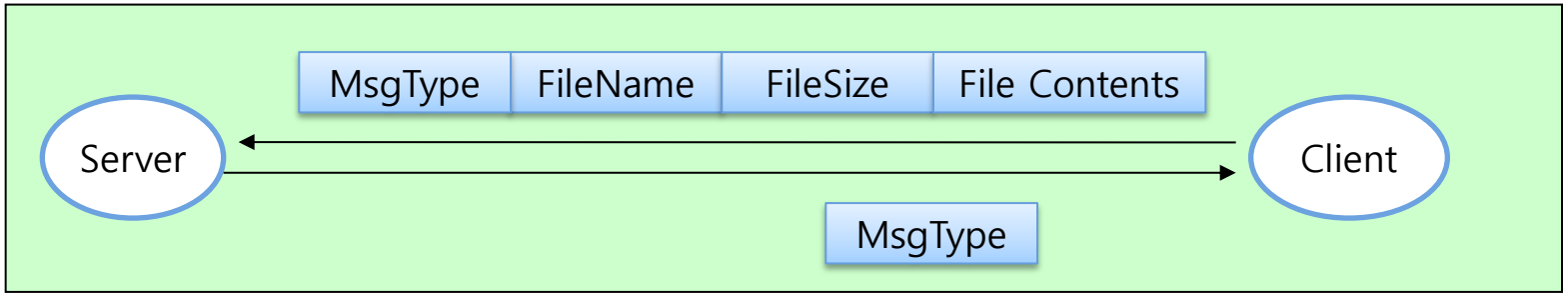
```
/* Message Type */
#define    EchoReq   01
#define    FileUpReq 02
#define    EchoRep   11
#define    FileAck   12
char MsgType;
…
recv(clntSock, &MsgType,1,0)

if (MsgType==FileUpReq) {
        recv(clntSock, FileName, …
        …
        MsgType=FileAck;
        send(clntSock, &MsgType, 1,0);
}
else if(MsgType==EchoReq) {
        recv(clntSock, EchoString, …
        MsgType=EchoRep;
        send(clntSock, EchoString,…);
}
else fprintf(stderr, "Bad request")
```

문자열 프로토콜 설계

```
/* Message Type */
#define EchoReq      "EchoReq"
#define FileUpReq    "FileUpReq"
#define EchoRep      "EchoRep"
#define FileAck      "FileAck"
char MsgType[10];
```

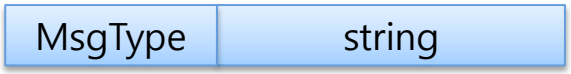




클라이언트 핵심 코드 (문자열 프로토콜)

```
/* Message Type */
#define EchoReq             "EchoReq|"
#define FileUpReq           "FileUpReq|"
#define EchoRep             "EchoRep|"
#define FileAck             "FileAck|"
#define Delimeter           '|'

char msgType[10];
char * operation;  //client 203.252.164.144 upload test.txt 5000
...
operation=argv[2]
...
If (!strcmp(operation,"upload"))
                strcpy(msgType, FileUpReq);
                send(sock, msgType, strlen(msgType),0)
                ...
else if (!strcmp(operation,"echo"))
                strcpy(msgType, EchoReq);
                send(sock, msgType,strlen(msgType,0)
                ...
else {
    fprintf(stderr, "Usage: %s <Server IP> <operation><operand> <Echo Port>₩n", argv[0]);
    exit(1);
}
```

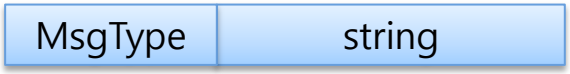


서버 핵심 코드 (문자열 프로토콜)

```
/* Message Type */
#define EchoReq                 "EchoReq|"
#define FileUpReq               "FileUpReq|"
#define EchoRep                 "EchoRep|"
#define FileAck                 "FileAck|"
#define Delimeter               '|'

char msgType[10], recvChar; int i=0;
...
while(i<10) {
                recvChar=recv(clntSock, msgType[i],sizeof(char),0);
                if (msgType[i]==Delimeter) break;
                ++i;
}

if (!(strcmp(msgType,FileReq)) {
                recv(clntSock, FileName, ...
                ...
                strcpy(msgType,FileAck);
                send(clntSock, msgType, strlen(msgType),0);
}
else if(!(strcmp(msgType, EchoReq)) {
                recv(clntSock, EchoString, ...
                strcpy(msgtype,EchoRep);;
                send(clntSock, msgType, strlen(msgTytpe),0);
                ...
}
else fprintf(stderr, "Bad request")
```



응용과제 3

- 에코와 파일 전송을 모두 지원하는 프로토콜을 정의하고, 서버와 클라이언트를 작성하여 동작을 확인하시오.
  - 전송 프로토콜 (tcp/udp) - 학번 끝자리 (짝수 / 홀수)
  - 인코딩/디코딩 (text / binary) - 학번 끝에서 2번째 자리 (짝수/홀수)
  - TCP의 경우 length 프레이밍 이용

응용과제 4

- 파일 전송/수신 클라이언트-서버 구현
  - 기능 :
    - get (파일 가져오기)
    - put (파일 보내기)
    - ls (서버측 파일 보여주기)
    - cd (서버측 디렉토리 변경)
    - quit(종료)
  - 2학년 UNIX 프로그래밍 참고