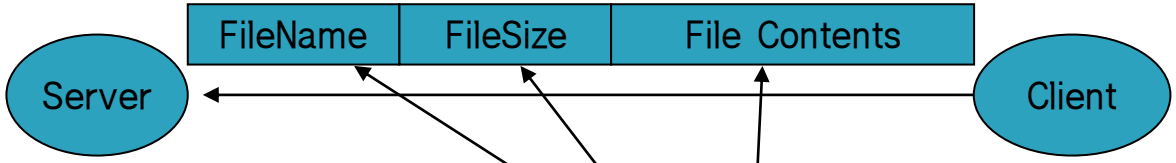
응용과제 3/4

네트워크 프로그램 설계 5장

고정된 길이 기반 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)
}
```

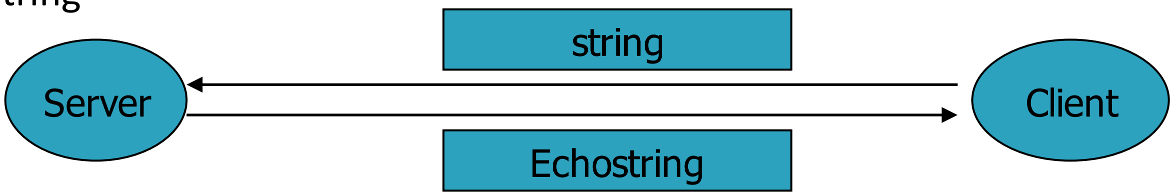
```
send(sock, FileName, 256, 0)
send(sock, FileSize, 4, 0)
while(feof(fd)) {
    send(sock, FileBuffer, 1024,0)
}
```

에코와 파일 전송을 모두 지원하는 프로토콜

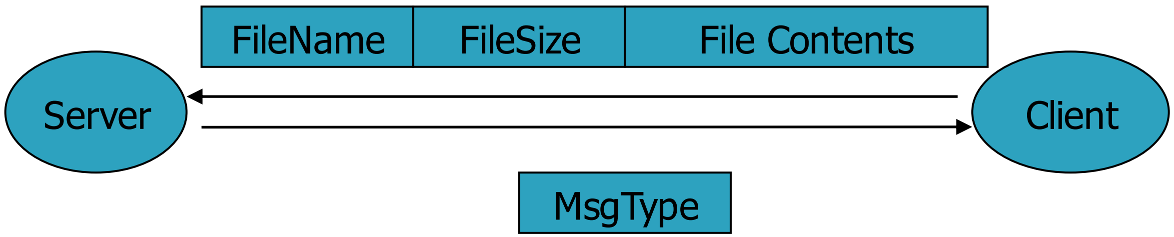
- 상황
  - 클라이언트는 서버에게 string 혹은 파일을 업로드 할 수 있다.
  - 서버는 string을 받을 경우 echo를 해주고 파일을 받을 경우, 디스크에 저장을 한 후 잘 받았다는 메시지를(Acknowledge) 회신한다
  - 클라이언트는 echo 메시지를 수신한 경우, echo 메시지를 출력하고, Ack를 받을 경우 file Ack를 출력한다
- 예
  - client lily.mmu.ac.kr upload test.txt 5000 // 파일 전송
  - client lily.mmu.ac.kr echo hello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

메시지(프로토콜) 설계

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



  - File Upload

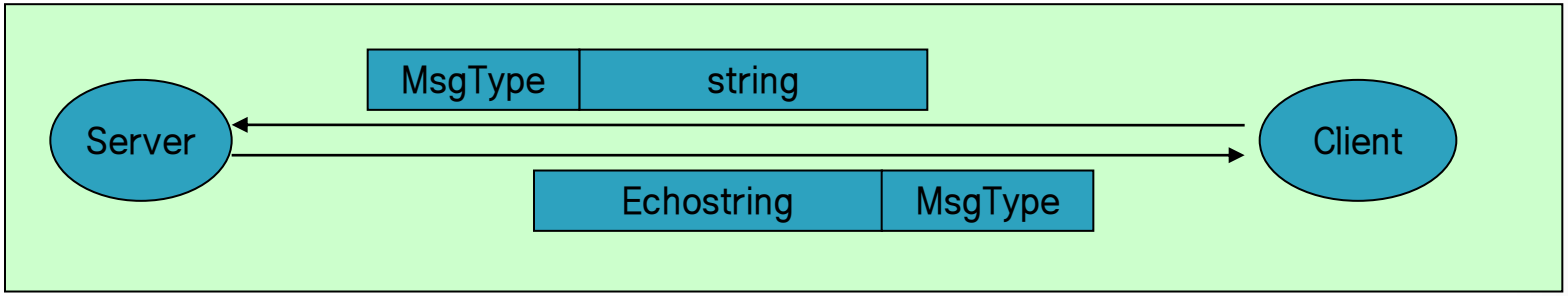


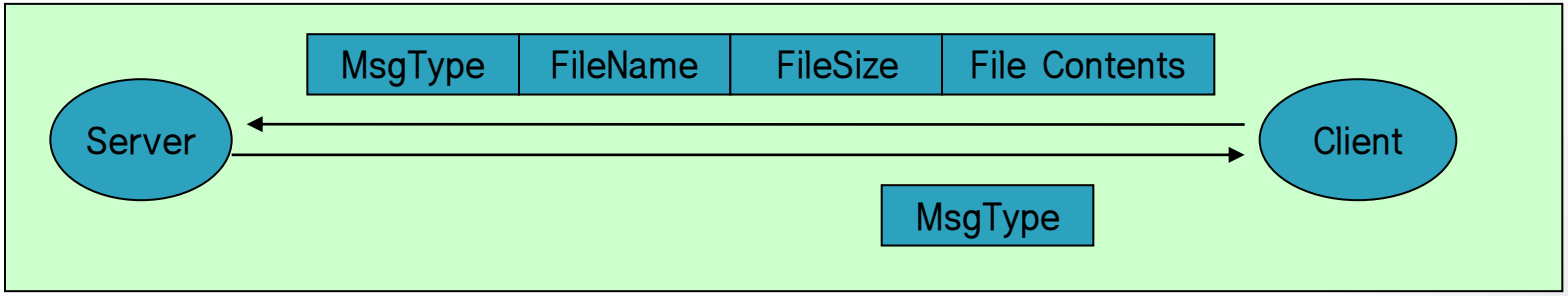
- 서버의 입장에서 클라이언트의 서비스 요청이 에코요청인지 파일업로드인지 구분할 수 있는 방법은?
  - 서비스 타입(에코 요청, 파일업로드) 필드를 준비하고 클라이언트를 이를 통해 서버에게 서비스 종류를 알림

바이너리 프로토콜 설계

- EchoString

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

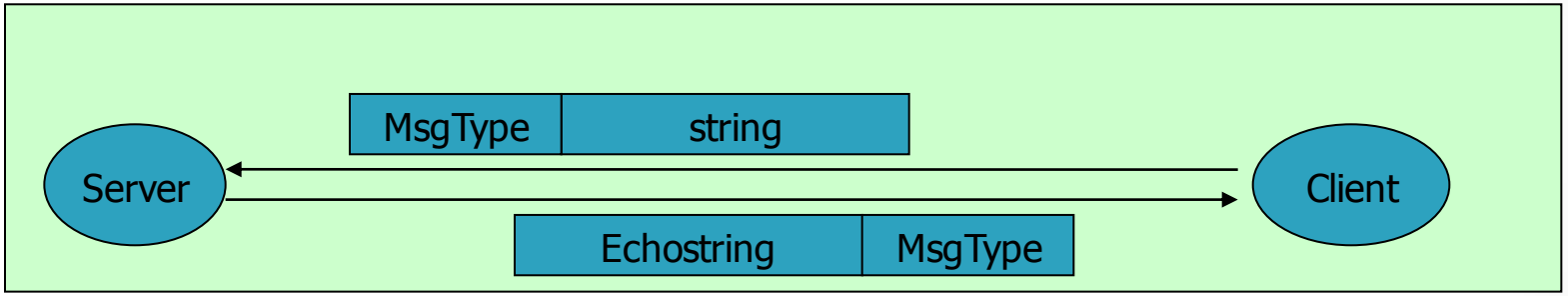


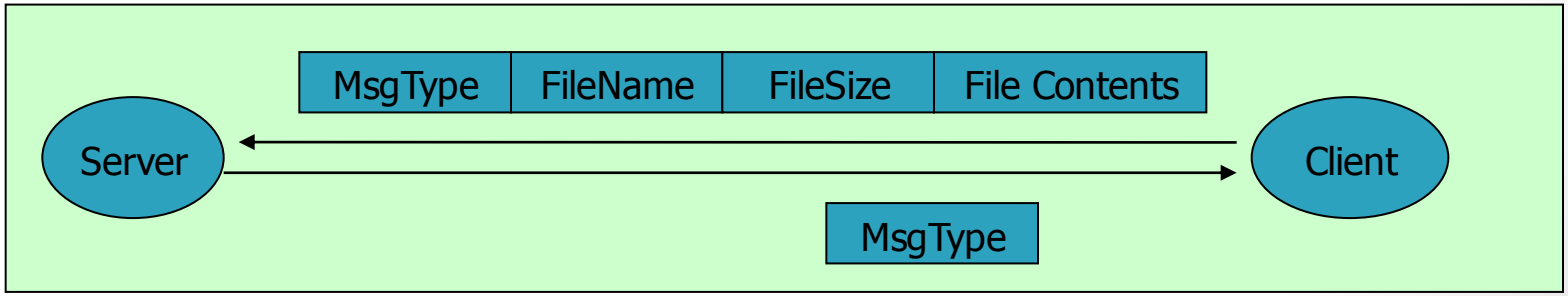


문자열 프로토콜 설계

- EchoString

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





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