네트워크 프로그래밍

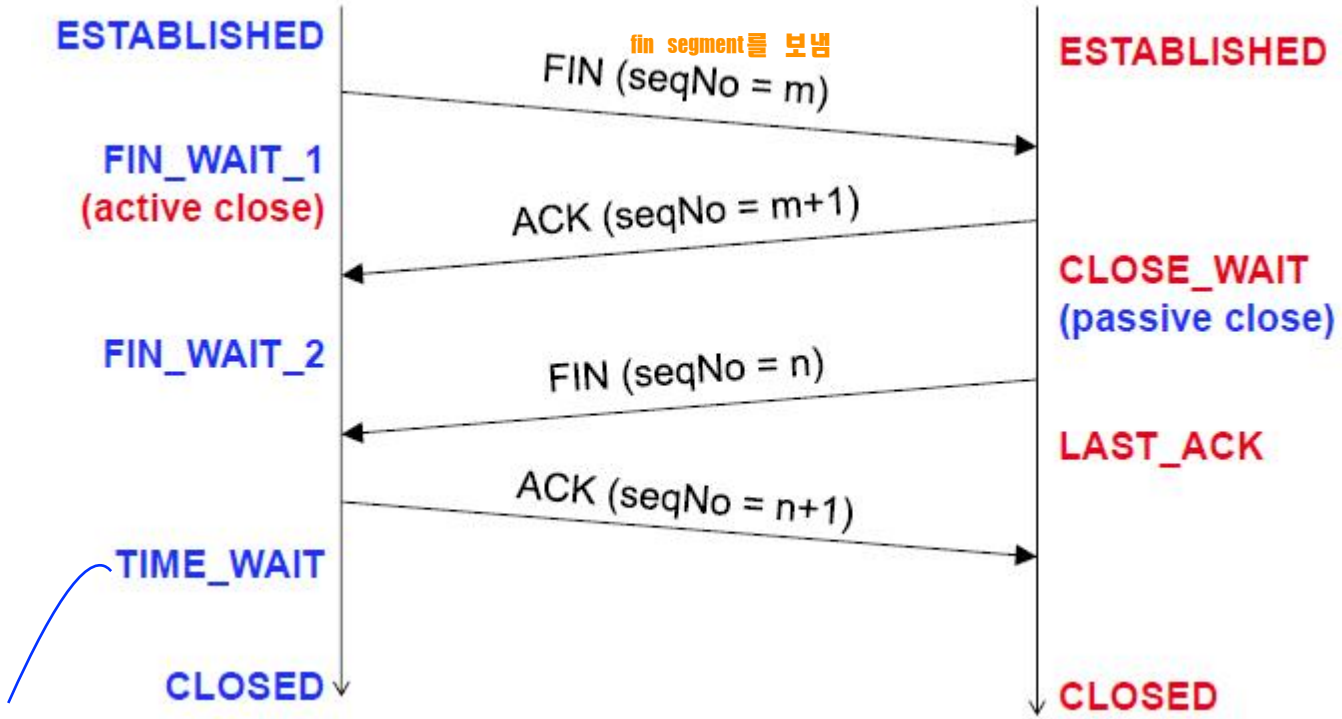
09. 소켓의 내부 동작 II - TCP 연결의 종료

TCP 연결의 종료

+ a closing
만약 동시에 fin을 날려서 서로fin을 받으면
closing상태로 전이된다.
이때 서로 ack를 주고받으면
time wait상태로 전이되고
일정시간이 지난후
연결이 종료 된다

TCP 연결의 종료

fin은 close가 호출 될 때 전송된다



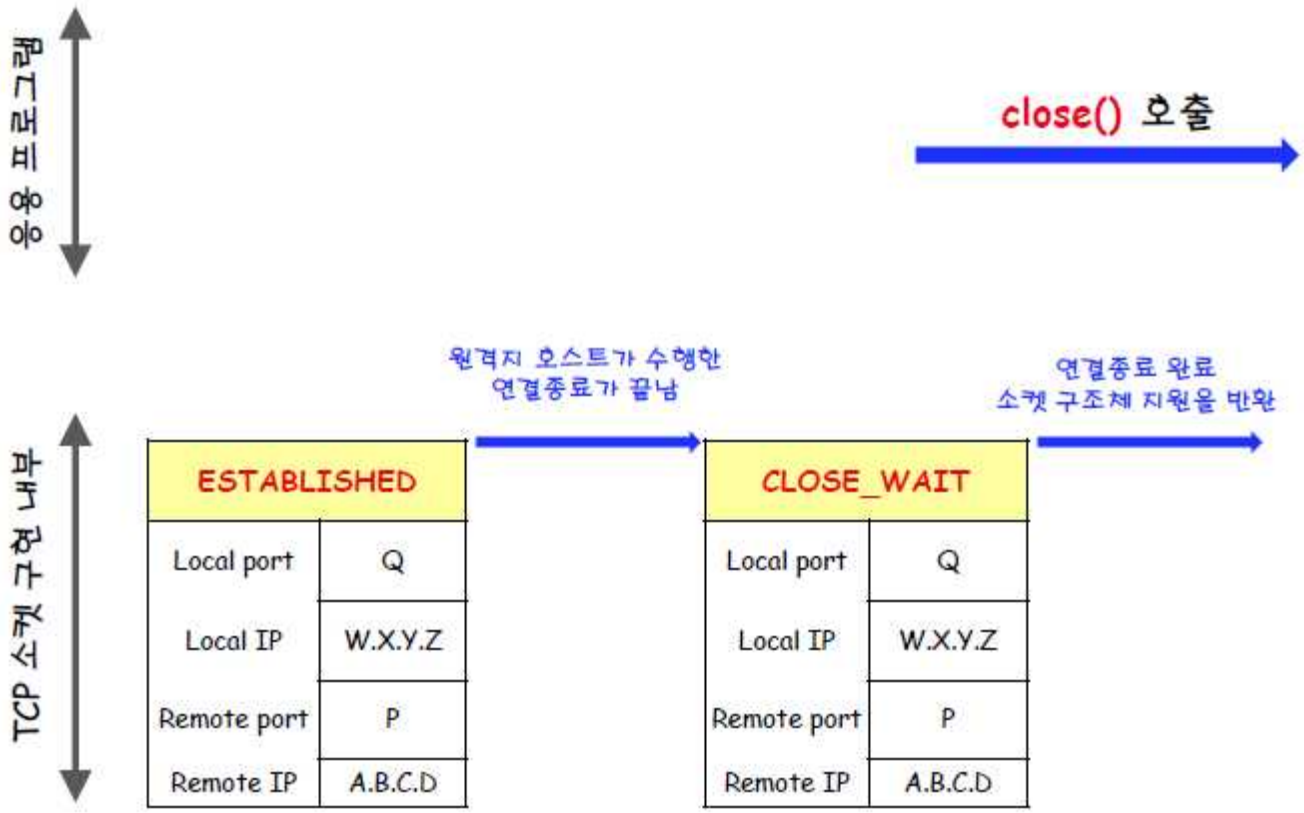
fin에대한 ack가 손실되었을때
재전송을 위해 일정시간 기다림

실제 close를 호출한다고 하더라도
close시 수신,송신자간 시간차가
존재하므로 바로 소켓에서 바인딩이
해제되거나 하지 않는다

연결 종료를 먼저 시도한 경우의 TCP 연결 종료과정

close를 호출한다고 fin이 즉석으로 날아가는 것이 아니라
send Q 에 있는게 모두 비워 질 때쯤,
맨 마지막 세그먼트에 F 플래그를 1로 만들어서 보냄.

상대방이 연결 종료를 요청한 경우의 TCP연결 종료과정



TCP 기반의 Half-close

일방적인 연결 종료의 문제점

- close 함수의 기능
  - 소켓의 완전 소멸을 의미한다.
  - 소켓이 소멸되므로 더 이상의 입출력은 불가능하다.
  - 상대방의 상태에 상관 없이 일방적인 종료의 형태를 띤다.
  - 때문에 상대 호스트의 데이터 송수신이 아직 완료되지 않은 상황이라면, 문제가 발생할 수 있다.
  - 이러한 문제의 대안으로 Half-close 기법이 존재한다.

소켓의 Half-close

- Half-close
  - 2개의 입출력 스트림 중 하나만 종료하는 것
  - 우아한 종료(graceful close)라고도 부름

우아한 종료를 위한 shutdown()


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

int shutdown(int sock, int howto);
```

➡ 성공 시 0, 실패 시 -1 반환

- sock — 종료할 소켓의 파일 디스크립터 전달.
- howto — 종료방법에 대한 정보 전달.
  - SHUT_RD — 입력 스트림 종료
  - SHUT_WR — 출력 스트림 종료
  - SHUT_RDWR — 입출력 스트림 종료

close대신에 호출 할 경우
보낼 것만 닫으니까
받을 건 다 받고 종료 할 수 있음

- close 함수가 호출되면 **상대 호스트(소켓)으로 EOF가 전달**된다. 이는 **모든 데이터의 전송이 끝났다는 신호의 의미**를 갖는다.
- **출력 스트림만 종료를 해도 EOF가 전달**이 되니, **close 함수의 호출을 대체**하고도, 상대 호스트의 종료를 기다릴 수 있다. 상대방이 연결을 종료했다는 사실은 read()의 반환값이 0임을 확인함으로써 알 수 있음.

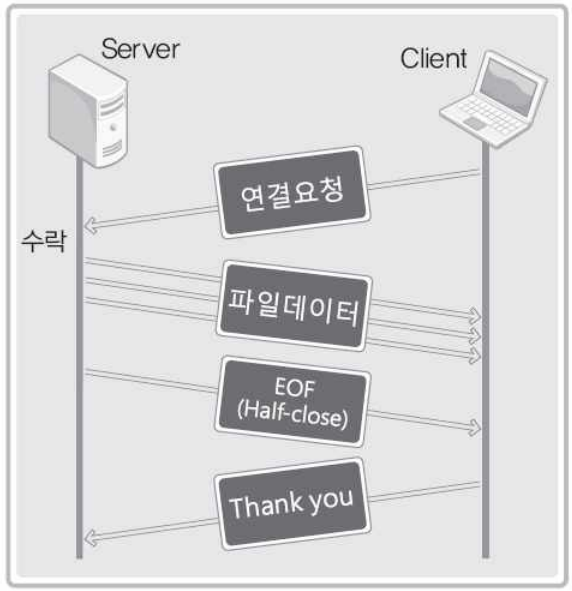
Half-close 기반 파일 전송 프로그램

file_server.c의 일부

```
while(1)
{
    read_cnt=fread((void*)buf, 1, BUF_SIZE, fp);
    if(read_cnt<BUF_SIZE)
    {
        write(clnt_sd, buf, read_cnt);
        break;
    }
    write(clnt_sd, buf, BUF_SIZE);
}

shutdown(clnt_sd, SHUT_WR);
read(clnt_sd, buf, BUF_SIZE);
printf("Message from client: %s \n", buf);

fclose(fp);
close(clnt_sd); close(serv_sd);
```

file_client.c의 일부

```
while((read_cnt=read(sd, buf, BUF_SIZE ))!=0)
    fwrite((void*)buf, 1, read_cnt, fp);

puts("Received file data");
write(sd, "Thank you", 10);
fclose(fp);
close(sd);
```



Half-close가 필요한 상황의 연출

shutdown() with SHUT_WR