WINDOW NETWORK -CHAPTER 2-

SOULSEEK

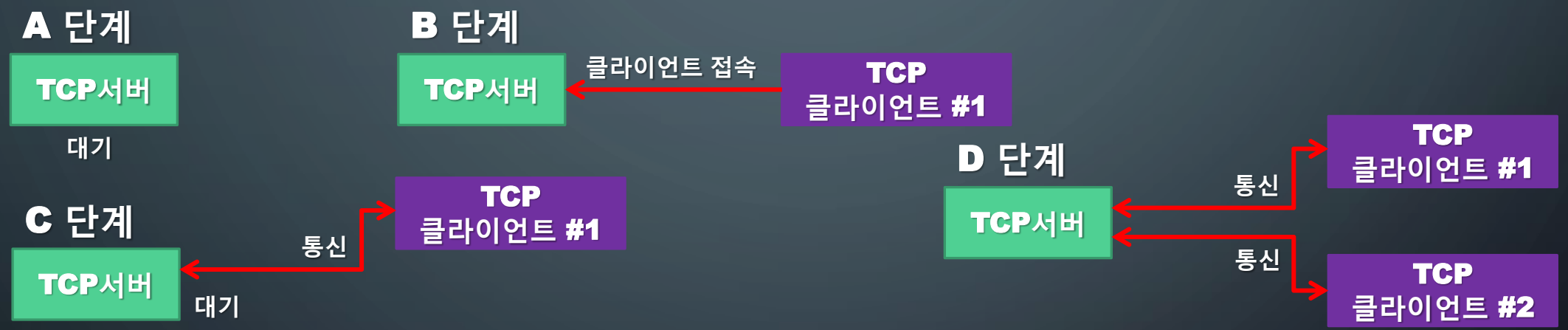
목차

1. TCP서버 – 클라이언트 개념

1. TCP서버 – 클라이언트 개념

TCP 서버 – 클라이언트 핵심 동작

1. 서버는 먼저 실행하여 클라이언트가 접속하기를 기다린다.
2. 클라이언트는 서버에 접속하여 데이터를 보낸다.
3. 서버는 클라이언트 접속을 수용하고 클라이언트가 보낸 데이터를 받아서 처리한다.
4. 서버는 처리한 데이터를 클라이언트에 보낸다.
5. 클라이언트는 서버가 보낸 데이터를 받아서 처리한다.
6. 데이터를 주고 받는 과정을 모두 마치면 접속을 끊는다.



동작 원리

A단계 **:** 서버는 소켓을 생성한 후 클라이언트가 접속하기를 기다린다**.** 이때 서버가 사용하는 소켓은 특정 포트번호와 결합되어 있어서 이 포트 번호로 접속하는 클라이언트만 수용 할 수 있다**.**

B단계 **:** 클라이언트가 서버에 접속 한다**.** 이때 **TCP** 프로토콜 수준에서 연결 설정을 위한 패킷 교환이 일어난다**.**

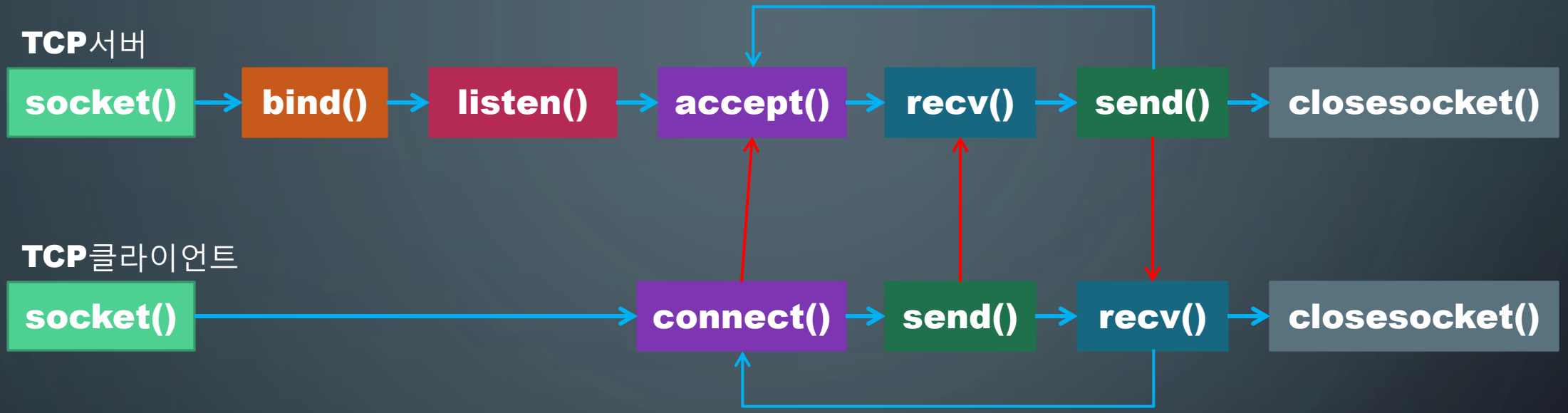
C단계 **: TCP** 프로토콜 수준의 연결 절차가 끝나면**,** 서버는 접속한 클라이언트와 통신할 수 있는 새로운 소켓을 생성한다**.** 이 소켓을 이용해 서버는 클라이언트와 데이터를 주고 받는다**.** 기존 소켓은 새로운 클라이언트 접속을 수용하는 용도로 계속 사용한다**.**

D단계 **:** 두 클라이언트가 접속한 후의 상태를 나타낸 것이다**.** 서버에는 소켓이 총 세 개 존재하며**,** 이 중 두 소켓을 접속한 클라이언트와 통신하는 용도로 사용한다**.**

2. TCP서버 – 클라이언트 구현

2. TCP서버 – 클라이언트 구현

Winsock 응용 프로그램의 공통 구조



TCP 서버 함수

1. **socket()** 함수로 소켓을 생성함으로써 사용할 프로토콜을 결정한다.
2. **bind()** 함수로 지역**IP** 주소와 지역 포트 번호를 결정한다.
3. **listen()** 함수로 **TCP**를 **LISTENING** 상태로 변경한다.
4. **accept()** 함수로 자신에게 접속한 클라이언트와 통신할 수 있는 새로운 소켓을 생성한다, 이때 원격 **IP** 주소와 원격 포트 번화가 결정된다.
5. **send(), recv()**함수로 클라이언트와 통신을 수행한 후, **closesocket()** 함수로 소켓을 닫는다.
6. 새로운 클라이언트 접속이 들어올 때마다 **4~5** 과정을 반복한다.

2. TCP서버 – 클라이언트 구현

bind() 함수 : 소켓과 소켓정보를 만들어 준다.

int bind(SOCKET s, const struct sockaddr* name, int namelen);

- **첫 번째 인자 :** 클라이언트 접속을 수용할 목적으로 만든 소켓으로, 지역 **IP** 주소와 지역 포트 번호가 아직 결정되지 않은 상태.
- **두 번째 인자 :** 소켓 주소 구조체(**TCP/IP**의 경우 **SOCKADDR_IN**)를 지역**IP**주소와 지역 포트 번호로 초기화하여 전달한다.
- **세 번째 인자 :** 소켓 주소 구조체의 길이(바이트 단위)다.

listen() 함수 : 클라이언트를 접속 받을 준비를 한다.

Int listen(SOCKET s, int backlog);

- **첫 번째 인자 :** 클라이언트 접속을 수용할 목적으로 만든 소켓으로, **bind()** 함수로 지역 **IP**주소와 지역 **PORT**번호를 설정한 상태.
- **두 번째 인자 :** 서버가 당장 처리하지 않더라도 접속 가능한 클라이언트의 개수.

accept() 함수 : 클라이언트의 접속요청을 기다렸다가 클라이언트 접속을 승인 시 클라이언트 소켓 정보로 소켓을 만들어 리턴 한다.

SOCKET accept(SOCKET s, struct sockaddr* addr, int* addrlen);

- **첫 번째 인자 :** 클라이언트 접속을 수용할 목적으로 만든 소켓, **bind()**함수로 지역**IP**주소와 지역 **PORT**번호를 설정하고 **listen()**함수로 **TCP**포트 상태를 **LISTENING**으로 변경한 상태이다.
- **두 번째 인자 :** 소켓 주소 구조체를 전달하면 접속한 클라이언트의 주소체계 정보로 채워진다.
- **세 번째 인자 :** 소켓 주소 구조체의 크기

2. TCP서버 – 클라이언트 구현

TCP 클라이언트 함수

- **socket()** 함수의 소켓을 생성함으로써 사용할 프로토콜을 결정한다.
- **connect()** 함수로 서버에 접속한다. 이때 원격 **IP** 주소와 원격 포트 번호는 물론, 지역 **IP**주소와 지역 포트 번호도 결정된다.
- **send(), recv()**등의 데이터 전송 함수로 서버와 통신한 후, **closesocket()** 함수로 소켓을 닫는다.

connect()함수

int connect(SOCKET s, const struct sockaddr *name, int namelen);

- **첫 번째 인자 :** 서버와 통신할 목적으로 만든 소켓.
- **두 번째 인자 :** 소켓 주소 구조체를 서버 주소로 초기화하여 전달한다.
- **세 번째 인자 :** 소켓 주소 구조체의 길이다.

TCP 데이터 전송 함수

- **send/recv()**가 가장 기본형태이며 다양한 함수가 존재한다.
- 소켓 버퍼로의 접근을 가능하게 만들어준다.
- **소켓 버퍼 :** 송신버퍼와 수신버퍼를 통칭해서 이르는 말이다.
- **송신 버퍼**는 데이터를 전송하기 전까지 임시로 저장해두는 영역
- **수신 버퍼**는 데이터를 받아서 처리하기 전까지 임시로 저장해두는 영역
- **TCP**는 데이터의 경계가 구분되지 않기 때문에 서로 데이터를 구분하는 약속이 필요하다.

2. TCP서버 – 클라이언트 구현

send()함수

int send(SOCKET s, const char* buf, int len, int flags);

- **첫 번째 인자 :** 통신할 대상과 연결된 소켓
- **두 번째 인자 :** 보낼 데이터를 담고 있는 버퍼의 주소.
- **세 번째 인자 :** 보낼 데이터의 크기
- **네 번째 인자 : send()**함수의 동작을 바꾸는 옵션, 대부분 **0**을 사용

send()에 사용하는 소켓의 특성

- **블로킹 소켓**
  - 앞선 예제들에서 작성한 모든 소켓.
  - 소켓을 대상으로 **send()**함수를 호출하면, 송신 버퍼의 여유 공간이 **send()**함수의 세 번째 인자인 **len**보다 작을 경우 해당 프로세스는 대기 상태가 된다.
  - 송신 버퍼에 충분한 공간이 생기면 프로세스는 깨어나고 **len** 크기만큼 데이터 복사가 일러난 후 **send()**함수가 리턴한다. 리턴값은 **len**과 같다.

- **넌블로킹 소켓**
  - **IoctIsocket()**함수를 이용하면 블로킹 소켓을 넌블로킹으로 바꿀 수 있다.
  - 소켓을 대상으로 **send()**함수를 호출하면, 송신버퍼의 여유 공간만큼 데이터를 복사한 후 실제 복사한 바이트 수를 리턴한다. 이 경우 **send()**함수의 리턴 값은 최소**1**, 최대 **len**이다.

2. TCP서버 – 클라이언트 구현

recv()함수

Int recv(SOCKET s, char* buf, int len, int flags);
첫 번째 인자 **:** 통신할 대상과 연결된 소켓이다**.**
두 번째 인자 **:** 받은 데이터를 저장할 버퍼의 주소이다**.**
세 번째 인자 **: OS**의 수신 버퍼로 부터 복사할 최대 데이터 크기다**,** 이 값은 **buf**가 가리키는 응용 프로그램 버퍼보다 크지 않아야 한다**.**
네 번째 인자 **: recv()**함수의 동작을 바꾸는 옵션으로**,** 대부분 **0**을 사용하면 된다**.**

recv()함수가 성공적인 리턴을 할 수 있는 상황.

수신 버퍼에 데이터가 도달한 경우

- **recv()**함수의 세 번째 인자인 **len**보다 크지 않은 범위에서 가능하면 많은 데이터를 응용 프로그램 버퍼에 복사한 후 실제 복사한 바이트 수를 리턴 한다**.** 이 경우 **recv()**함수의 리턴 값은 최소 **1,** 최대 **len**이다**.**

접속이 정상 종료 한 경우

- 상대편 응용 프로그램이 **closesocket()** 함수를 호출해 접속을 종료하면**, TCP** 프로토콜 수준에서 접속종료를 위한 패킷 교환 절차가 일어난다**.** 이 경우 **recv()**함수는 **0**을 리턴한다**. recv()** 함수의 리턴 값이 **0**인 경우를 정상종료라 부른다**.**

TCP Server, TCP Client 프로젝트를 참고하자**.**