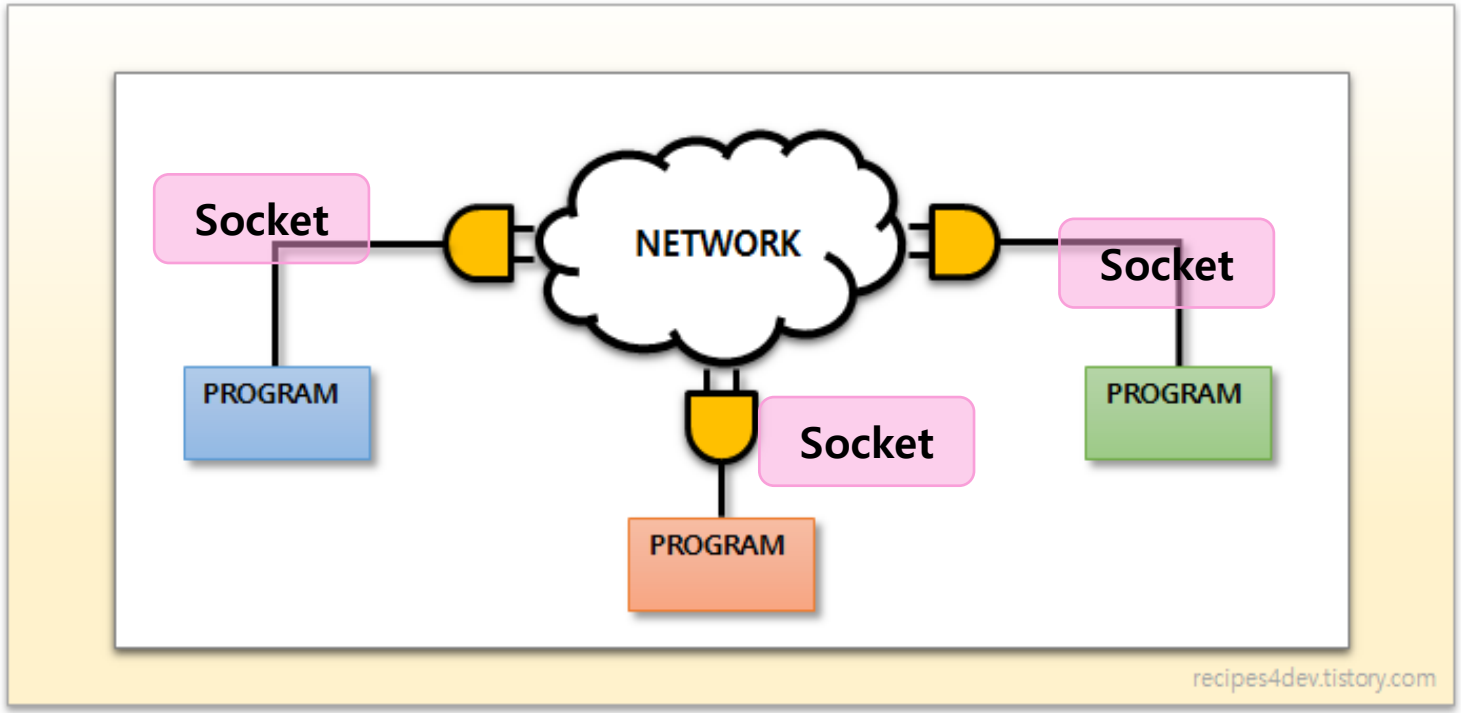
K-Digital Training 스마트 팩토리 3기

Socket

소켓(Socket)

- 네트워크를 경유해서 통신을 하기 위한 도구
- 소켓을 이용해서 컴퓨터 간 통신을 할 수 있음



소켓(Socket)

서버와 클라이언트를 연결해주는 도구!

- 서버 : 클라이언트에서 요청이 오면 소켓을 생성해 통신이 가능하도록 한다. 연결을 담당
- 클라이언트 : 실제로 데이터의 송·수신이 일어나는 곳!

소켓 구성 요소

- IP주소, 포트번호, 프로토콜

소켓(Socket)

- 연결지향 TCP 소켓
  - TCP/IP
  - 각 소켓끼리 서로 연결
  - 연결된 상태에서 통신, 연결된 대상 외에 다른 대상과는 통신 불가능
  - 데이터를 잘 받았는지 중간중간 확인해서 안정적으로 데이터를 모두 보낼 수 있다.
  - 속도는 느리지만 안정성이 높다.
    - 데이터가 손실되면 안되는 경우 무조건 TCP 소켓 사용

소켓(Socket)

- 비연결지향 UDP 소켓
  - 연결되지 않은 상태에서 내가 원하는 주소에 데이터를 보낼 수 있는 통신 방법
  - 데이터를 보낸 후 확인작업이 없어서 데이터가 다 수신되었는지 확인 불가능
  - 속도가 빠르지만 데이터가 소실될 수 있다.
  - UDP 헤더의 체크섬 필드를 통해 최소한의 오류만을 검출
  - 동영상 스트리밍 서비스

소켓 서버 비교

TCP 서버

- 서버소켓은 연결만을 담당
- 서버와 클라이언트 1대1로 연결
- 스트림전송 → 전송 데이터의 크기가 무제한
- 패킷에 대한 응답으로 인한 시간지연, CPU 소모
- 스트리밍 서비스에 불리(손실될 경우 재전송 요청)

UDP 서버

- 1대1, 1대N, N대M 연결 가능
- 성능이 중요한 서비스에 사용

소켓 주소 체계

AF_UNSPEC	0	정의되지 않은 주소 영역
AF_INET	2	IPv4 주소 영역에서 사용
AF_IPX	6	IPX/SPX 주소 영역에서 사용
AF_APPLETALK	17	AppleTalk 에서 사용
AF_NETBIOS	17	NetBIOS 주소 영역에서 사용
AF_INET6	23	IPv6 주소 영역에서 사용
AF_IRDA	26	Infrared Data Associatino 주소 영역에서 사용
AF_BTH	32	bluetooth 주소 영역에서 사용

AF_UNIX 유닉스 주소체계 의미

소켓 주소 예시

```
struct sockaddr_un {
    sun_family;
    sun_path[-]; //파일시스템의 경로
}
//유닉스 주소 체계의 구조체
```

```
struct sockaddr_in {
    ADDRESS_FAMILY sin_family;
    USHORT sin_port; //port
    IN_ADDR sin_addr; //ip
}
//인터넷 주소 체계의 구조체
// ADDRESS_FAMILY => unsigned short
```

통합 주소 체계

- 여러 소켓 구조체를 통합해서 하나로 정의!
- 할당된 공간이 다른 주소 체계에서 필요한 공간보다 커야 함.

```
struct sockaddr{
        ADDRESS_FAMILY sa_family;
        CHAR sa_data[-];
}
```

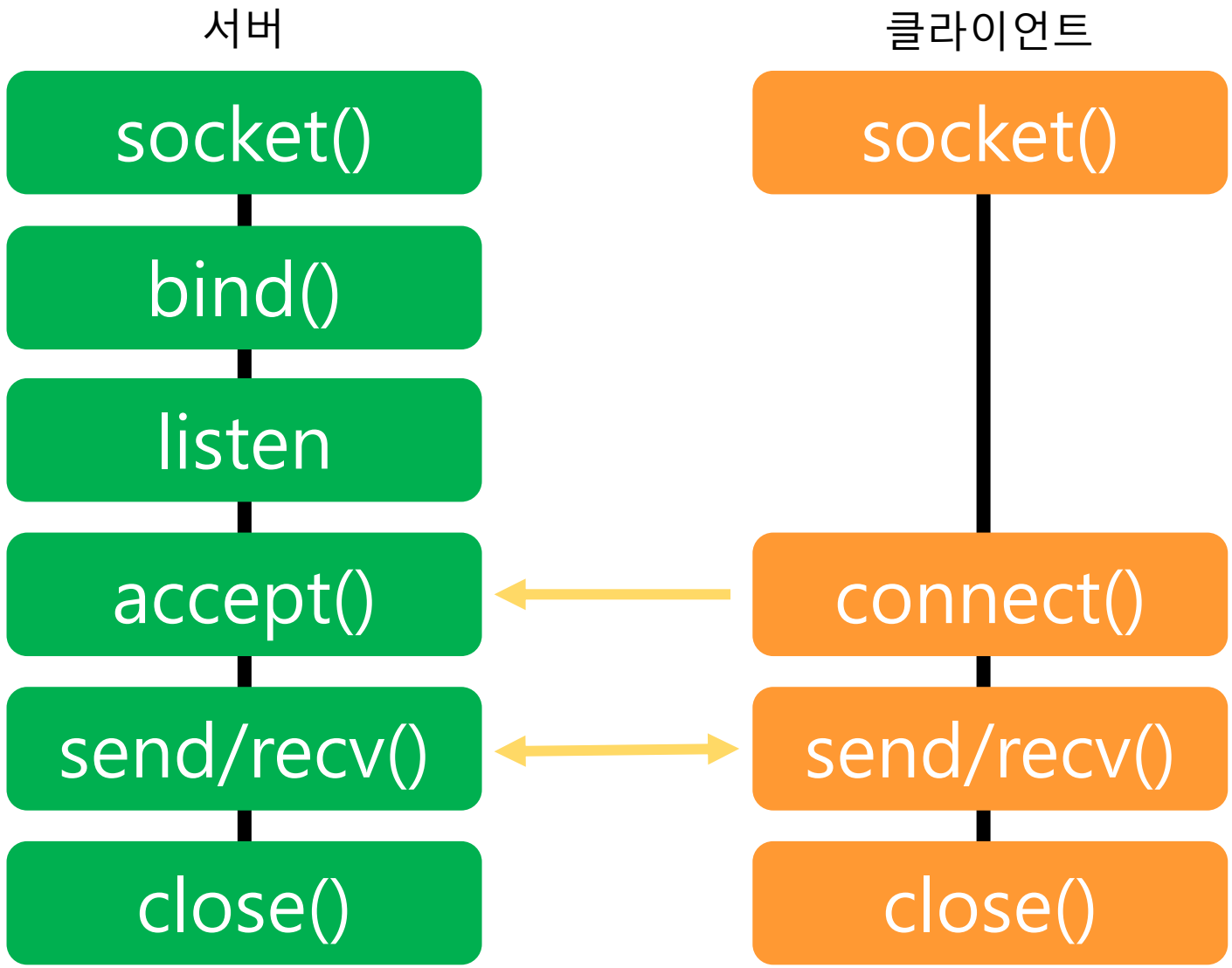
소켓 서비스

- 제공하는 서비스도 여러 개
  - 사람끼리 통신을 할 때 편지를 부치거나, 전화를 하는 것 처럼 말이에요!
- SOCK_STREAM
  - 연결형 서비스를 의미 (TCP 프로토콜에 대응)
- SOCK_DGRAM
  - 비연결형 서비스를 의미 (UDP 프로토콜에 대응)
- SOCK_RAW
  - IP 프로토콜을 직접 사용합니다. 실제로 자주 사용되지 않음.

소켓의 흐름 (TCP)

1. 서버와 클라이언트는 소켓을 각각 생성
2. 클라이언트가 서버에게 요청을 한다.
3. 서버는 클라이언트의 요청을 받는다.
4. 데이터를 주고 받는다.
5. 데이터를 주고 받는 동작이 끝나면 연결된 소켓을 닫는다!

소켓의 흐름 (TCP)


서버
클라이언트
socket()
bind()
listen
accept()
send/recv()
close()
socket()
connect()
send/recv()
close()

socket()

socket(주소 영역 지정, 서비스 타입, 프로토콜 지정)

- 소켓 생성
- socket() 성공적으로 실행되어 소켓이 만들어지면 해당 소켓의 디스크립터 반환

- 프로토콜

BTHPROTO_RFCOMM	3	Bluetooth Radio Frequency 통신을 위해 사용, SOCK_STREAM type과 함께 사용
IPPROTO_TCP	6	TCP를 사용, AF_INET 혹은 AF_INET6 af와 SOCK_STREAM type과 함께 사용
IPPROTO_UDP	17	UPD를 사용, AF_INET 혹은 AF_INET6 af와 SOCK_DGRAM type과 함께 사용

bind()

- 생성된 소켓에 주소 부여
- bind(소켓, 바인드될 소켓의 주소, 주소 크기)

listen()과 accept()

- listen() 과 accept() 는 서버 프로세스에서 실행

- listen()
  - 소켓 활성화
- accept(소켓, 주소, 주소의 길이)
  - 임의의 클라이언트의 연결 요구가 들어올 때까지 대기.
  - 연결 요청이 들어오면 둘 사이에 연결이 설정되고 서버에 새로운 소켓이 생성
  - 이후 데이터 송수신은 새로 생성된 소켓 이용

send() 와 recv()

- send()
  - 연결형 서비스를 제공하는 환경에서 데이터 전송

- recv()
  - 연결형 서비스를 제공하는 환경에서 데이터 수신

connect()

- connect( )
- 클라이언트 프로세스에서 사용
- 매개변수로 설정된 주소값이 가리키는 서버와 연결 설정

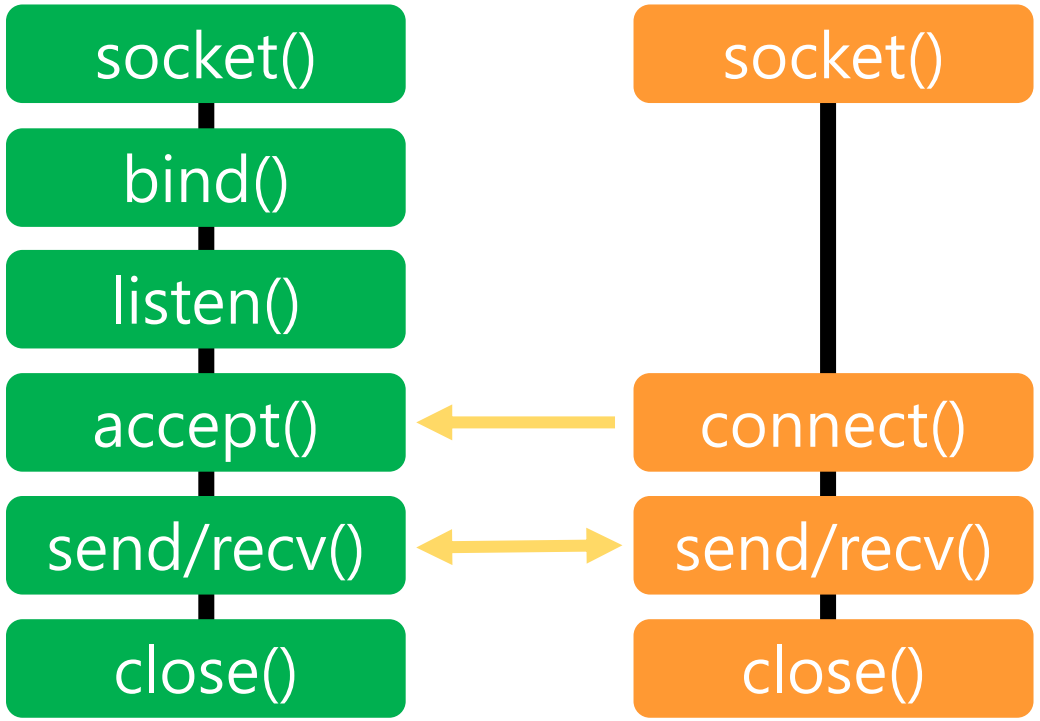
추가로 알면 좋은 함수와 상수

- htons() : 2바이트 이상의 변수에 대해 바이트 순서 체계를 바꿔 주는 것(short)
- htonl() (long int)

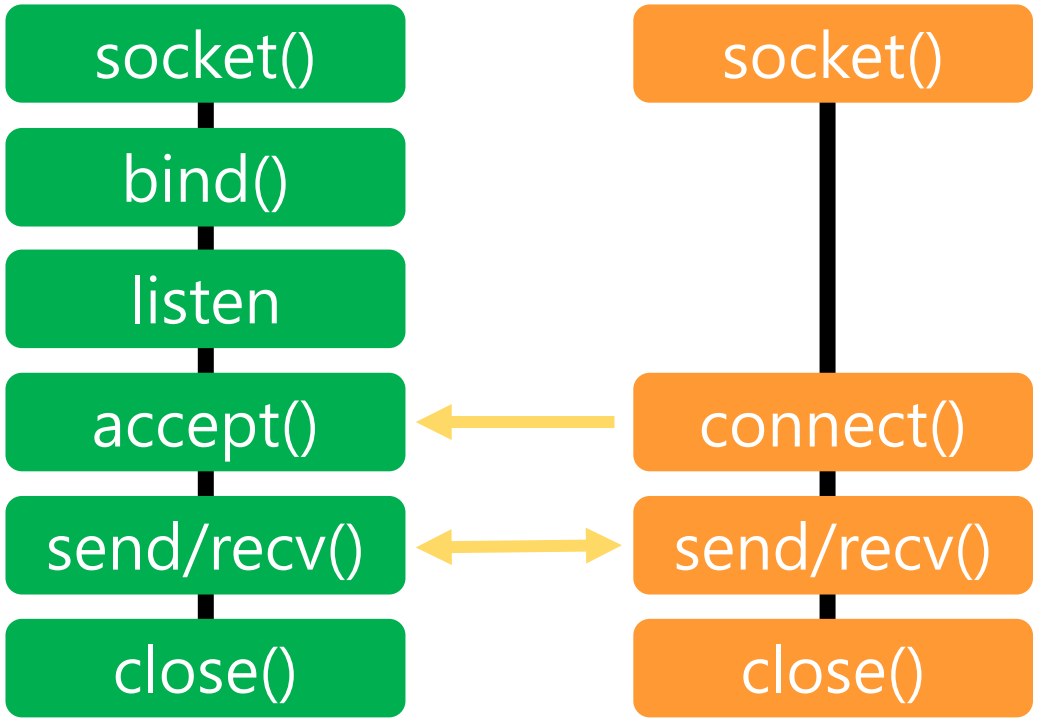
- INADDR_ANY : localhost 를 의미
- inet_addr() : 십진수 → 이진수
  - 점이 포함된 10진수로 표현된 주소를 IN_ADDR 구조체에 적합한 주소로 변환
- inet_ntoa() : 이진수 → 십진수

소켓의 흐름 (TCP)

- 서버 (Server)
  - socket() 으로 소켓 생성
  - bind() 로 주소와 소켓 묶음
  - listen() 소켓 활성화
  - accept() 대기하다가 요청 들어오면 연결
  - send()/recv() 데이터 송수신 (반복)
  - close() 데이터 송수신이 끝나면 종료

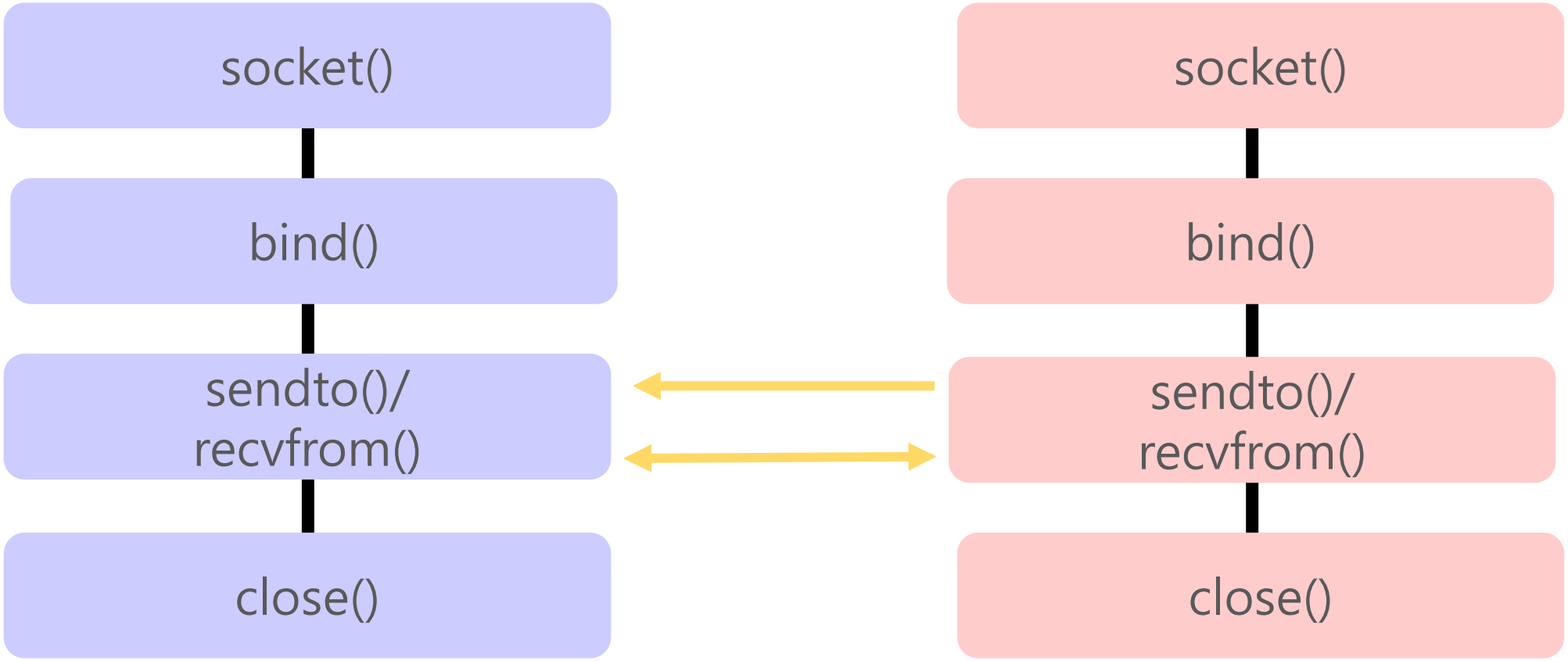


소켓의 흐름 (TCP)



- 클라이언트 (Client)
  - socket() 생성
  - connect() 서버에 연결 요청
  - recv() & send() 반복하며 통신
  - close()

소켓의 흐름 (UDP)


socket()
bind()
sendto()/
recvfrom()
close()
socket()
bind()
sendto()/
recvfrom()
close()

소켓의 흐름 (UDP)

- 비연결형 서비스에서는 전송 데이터마다 수신자의 소켓 주소를 함께 전송해야 함.
- 클라이언트 측에서도 bind() 실행
- 연결절차(listen 과 accept)가 생략됨.

Socket 함수

SOCKET socket(int af, int type, int protocol);

- 소켓 생성
- socket() 성공적으로 실행되어 소켓이 만들어지면 해당 소켓의 디스크립터 반환

- Af : 주소 영역 지정
- Type : 통신 타입 지정
- Protocol : 호스트간 통신에 사용할 프로토콜 지정

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

- 생성된 소켓에 주소 부여

- s : 클라이언트의 연결을 기다리는 소켓 객체. **socket()**으로 생성된 소켓 객체
- Name : 소켓과 연결할 주소 정보(type, port 등)를 담고 있는 구조체
- Namelen : name의 크기

int listen(SOCKET s, int backlog)

- 소켓 활성화

- S:클라이언트의 연결을 기다리는 소켓 객체. **socket()**으로 생성된 소켓 객체
- Backlog : 보류 중인 연결 대기열의 최대 길이

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

- 임의의 클라이언트의 연결 요구가 들어올 때까지 대기.
- 연결 요청이 들어오면 둘 사이에 연결이 설정되고 서버에 새로운 소켓이 생성
- 이후 데이터 송수신은 새로 생성된 소켓 이용

- S: 클라이언트의 연결을 기다리는 소켓 객체. **socket()**으로 생성된 소켓 객체
- Addr : client 연결을 가져오면, 이 매개 변수에 client 주소 정보를 저장
- Addrlen : addr의 크기

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

- 클라이언트 프로세스에서 사용
- 매개변수로 설정된 주솟값이 가리키는 서버와 연결 설정

- S:서버에 연결하기 위해 만들어 둔 소켓 객체
- Name: 연결할 서버 정보(host, port, type 등)가 담긴 구조체
- Namelen: name 크기

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

- 연결형 서비스를 제공하는 환경에서 데이터 전송

- S: 소켓 객체
  - Client: 서버에 연결된 소켓. Connect 이후
  - Server : 대기 상태인 소켓. Accept 성공 이후.
- Buf : 전송할 데이터.
- Len : buf 의 길이.
- Flags : 호출이 이루어지는 방식을 지정

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

- 연결형 서비스를 제공하는 환경에서 데이터 수신

- S: 소켓 객체
- Buf : 들어오는 데이터를 받을 버퍼
- Len : buf 의 길이
- Flags : 함수의 동작에 영향을 미치는 플래그 집합.