[TCP/IP 소켓 프로그래밍의 이해]

홍지우

1. 소켓(Socket)

소켓 (Socket)
“사전적인 의미: 구멍, 연결, 콘센트”

네트워크 소켓(Network Socket)
“네트워크 환경에 연결할 수 있게 만들어진 연결부.”

TCP/IP 소켓
TCP/IP 프로토콜 스택에서 IP계층과 TCP계층에서 동작하는 소켓 (또는 OSI 3계층과 4계층)

2. TCP/IP 소켓 프로그래밍

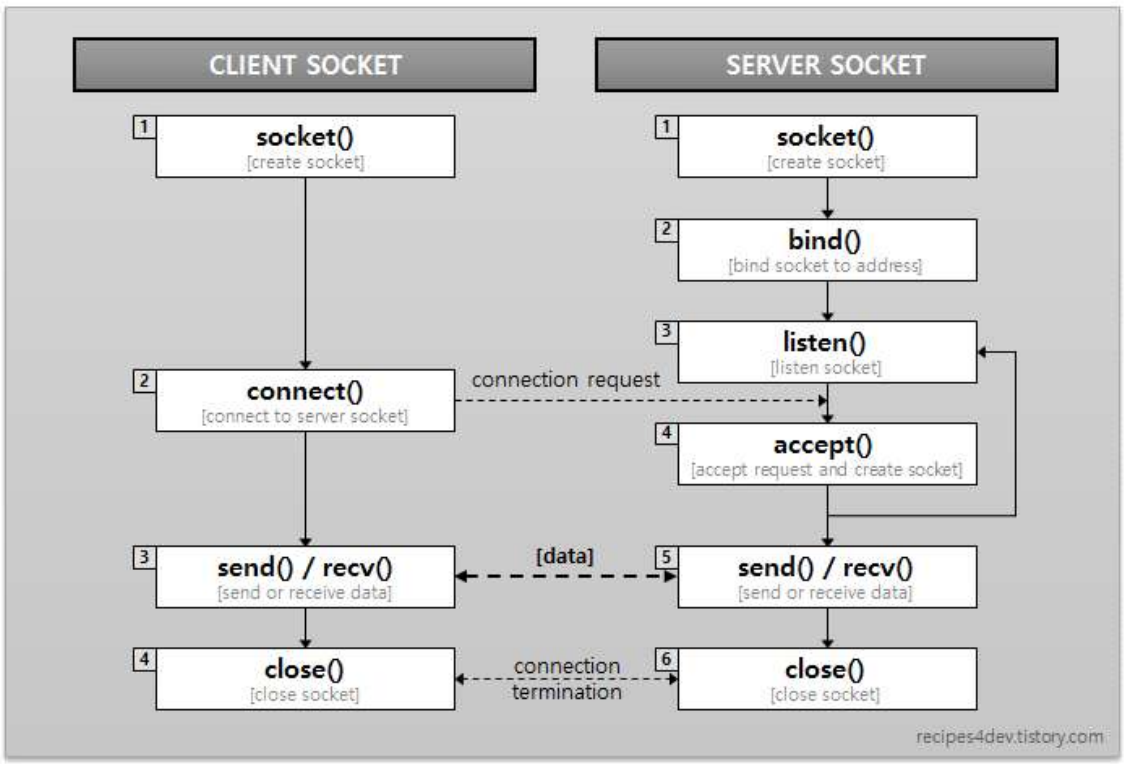
두 개의 시스템 (또는 프로세스) 가 소켓을 통해서 네트워크 연결을 만들기 위해서는 어느 한곳에서 연결 요청을 해야한다. 이 때 IP 주소와 포트 번호로 식별할 대상에게 네트워크 연결을 하고싶다는 요청을 하는 것이다. 하지만 무작정 연결을 요청한다고 해서 네트워크 연결이 체결되는 것이 아니라 수신 측에서 어떤 연결 요청을 받아들일 것인지를 미리 시스템에 등록하여, 요청이 수신되었을 때 요청을 처리할 수 있도록 해야 한다. 기본적인 개념만 머리에 넣어둔다면 어떤 운영체제든, 언어든 소켓 프로그래밍의 기본 틀을 가지고 있기 때문에 프로그래밍할 수 있을 것이다.

클라이언트

1. 소켓 생성 2. 서버 측에 연결 3. 서버 소켓에서 연결을 받으면 데이터를 송수신 4. 모든 처리가 완료되면 소켓을 닫음

서버

1. 소켓 생성 2. 서버가 사용할 IP 주소와 포트 번호를 생성한 소켓에 결합 3. 클라이언트로부터 연결 유청이 수신되는지 주시 4. 요청이 수신되면 accpet 후 소켓 생성 5. 데이터 송수신 6. 소켓 닫음



3. 클라이언트 소켓 프로그래밍

3.1 클라이언트 소켓 생성 : socket()

소켓 통신을 위해 가장 먼저 해야 할 일은 소켓을 생성하는 것이다. 이 때 소켓의 종류를 지정할 수 있는데, TCP 소켓을 위해서는 스트림 타입, UDP 소켓을 위해서는 데이터그램 타입을 지정할 수 있다.

최초 소켓이 만들어지는 시점에는 어떠한 연결 대상에 대한 정보도 들어 있지 않다. 그러기에 연결 대상 즉, IP 와 Port 을 지정하고 연결 요청을 전달하기 위해서는 생성한 소켓을 사용하여 connect() API를 호출해야 한다.

3.2 연결 요청 : connect()

connect() API는 IP주소와 포트 번호로 식별되는 대상으로 연결 요청을 보낸다.

connect() API는 블록 방식으로 동작하기에, 연결 요청에 대한 결과가 결정되기 전에는 connect()의 실행이 끝나지않고 대기한다.

호출이 성공하면 send() / recv() API 를 통해 데이터를 송수신 할 수 있다.

3.3 데이터 송수신 : send() / recv()

연결된 소켓을 통해 데이터를 보낼 때는 send() 수신에는 recv() API를 사용한다.두 API 모두 connect와 동일하게 블럭 방식으로 동작한다.

그러므로 두 호출이 모두 결과가 결정되기 전까지 API가 리턴되지 않는다.

특히 recv() 같은 경우는 한번 실행되면 언제 어떤 데이터가 전송되어 올 것인지 알 수 없기때문에, 데이터 수신을 위해서 별도의 스레드를 실행하여 데이터 수신을 기다린다.

3.4 소켓 닫기 : close()

데이터 송수신이 필요없게 되면, 소켓을 닫기 위해 close() API를 호출한다. 해당 소켓은 닫힌 이후에는 재사용이 불가능하다.

3. 서버 소켓 프로그래밍

4.1 서버 소켓 생성 : socket()

클라이언트와 동일하게 소켓을 생성한다.

4.2 서버 소켓 바인딩 : bind()

bind의 사용되는 파라미터는 포트 번호 혹은 IP 주소 + 포트 번호이다. 시스템 상에서 많은 수의 프로세스가 동작하기에 서버에 접근할 수 있는 가상화된 포트를 지정해야 한다. 운영체제는 소켓들이 중복된 포트 번호를 사용하지 않도록, 내부적으로 포트 번호와 소켓 연결 정보를 관리하는데, bind() 호출과정에서 중복되는 포트 사용이 있으면 운영체제가 포트 할당을 거부하고, API는 에러를 리턴한다.

4.3 클라이언트 연결 요청 대기 : listen()

서버 소켓에 포트 번호를 결합하고 나면, 서버 소켓을 통해 클라이언트의 요청을 받을 준비가 되었다. 클라이언트의 연결 요청이 수신될때까지 한없이 세월아 네월아 기다려야 한다. 대기 상태를 빠져나오는 경우도 있는데 이때는 요청이 수신되는 경우와, 에러가 발생하는 경우다.

클라이언트 연결 요청에 대한 정보는 시스템 내부적으로 관리되는 큐에 쌓이게 됨으로, 대기 중인 연결 요청을 큐로부터 꺼내와서, 연결을 수립하기 위해서는 accept() 를 호출하면 된다.

4.4 클라이언트 연결 수립 : accept()

최종적으로 accept() 을 호출해서 소켓 간 연결이 수립이 된다. 여기서 한가지 독특한 부분은 accept() 가 호출되어 소켓 간 연결을 수립할 때 새로운 소켓을 만들어 연결하게 된다. 그럼 여기서 서버 소켓의 역할은? 단순히 클라이언트 연결을 대기하고 연결 수립 요청을 하고 소켓을 닫는 것이다.

4.5 데이터 송수신 : send() / recv()

클라이언트와 동일하다.

4.6 소켓 연결 종료 : close()

클라이언트와 동일하다. 하지만 서버 소켓의 경우 클라이언트와 연결을 수립했던 소켓뿐만 아니라 클라이언트의 연결을 대기하는 소켓 또한 닫아줘야 함으로 유의해야 한다.