## 1. 접속의 의미

소켓을 만들면 애플리케이션(브라우저)은 connect를 호출한다.

그러면 프로토콜 스택은 자기 쪽의 소켓을 서버측 소켓에 접속한다.

이때 케이블은 항상 연결되어 있으므로 언제나 신호를 보낼 수 있다.

따라서 데이터를 신호로 변환하여 송신하기만 하면 언제든지 통신이 가능해진다.

- 접속 동작

➔ 소켓을 만든 직후에는 프로토콜 스택에 아무것도 기록되어 있지 않으므로 통신 상대가 누구인지 모른다.

➔ 그러므로 서버의 IP주소나 포트번호를 프로토콜 스택에 알리는 동작이 필요하다.

➔ 접속 동작의 첫 번째 동작은 통신 상대와의 사이에 제어정보를 주고 받아 소켓에 필요한 정보를 기록하고 데이터 송수신이 가능한 상태로 만드는 것이다.

➔ 제어 정보는 데이터 송 수신 동작을 제어하기 위한 정보이며, IP주소, 포트번호도 해당

➔ 송수신하는 데이터를 일시적으로 저장하는 메모리 영역이 필요하고, 이 영역을 '버퍼 메모리' 라고 부른다.

➔ '버퍼 메모리' 확보도 접속 동작을 할 때 실행된다.

## 2. 헤더 배치

통신 동작에 이용하는 제어 정보는 크게 두 종류가 있다.

1. 헤더에 기입되는 정보
2. 소켓(프로토콜 스택의 메모리 영역)에 기록되는 정보

제어 정보를 패킷의 맨 앞에 배치하는 곳부터 헤더라고 부른다.

➔ 헤더는 이더넷 헤더, TCP헤더, IP헤더와 같이 여러가지가 있다.

클라이언트와 서버는 헤더에 필요한 정보를 기록하여 연락을 한다.

소켓에 기록한 제어정보는 상대 측에서 볼 수 없다.

➔ 규칙에 따라 헤더에 제어 정보를 기록하여 대화하기 때문이다.

➔ 이러한 규칙으로 통신이 되기 때문에 내부 구조가 다른 윈도우와 리눅스 OS는 필요한 제어정보가 달라도 문제없이 통신할 수 있다.

소켓에 기록하는 제어 정보는 프로토콜 스택을 만드는 사람에 따라 달라진다.

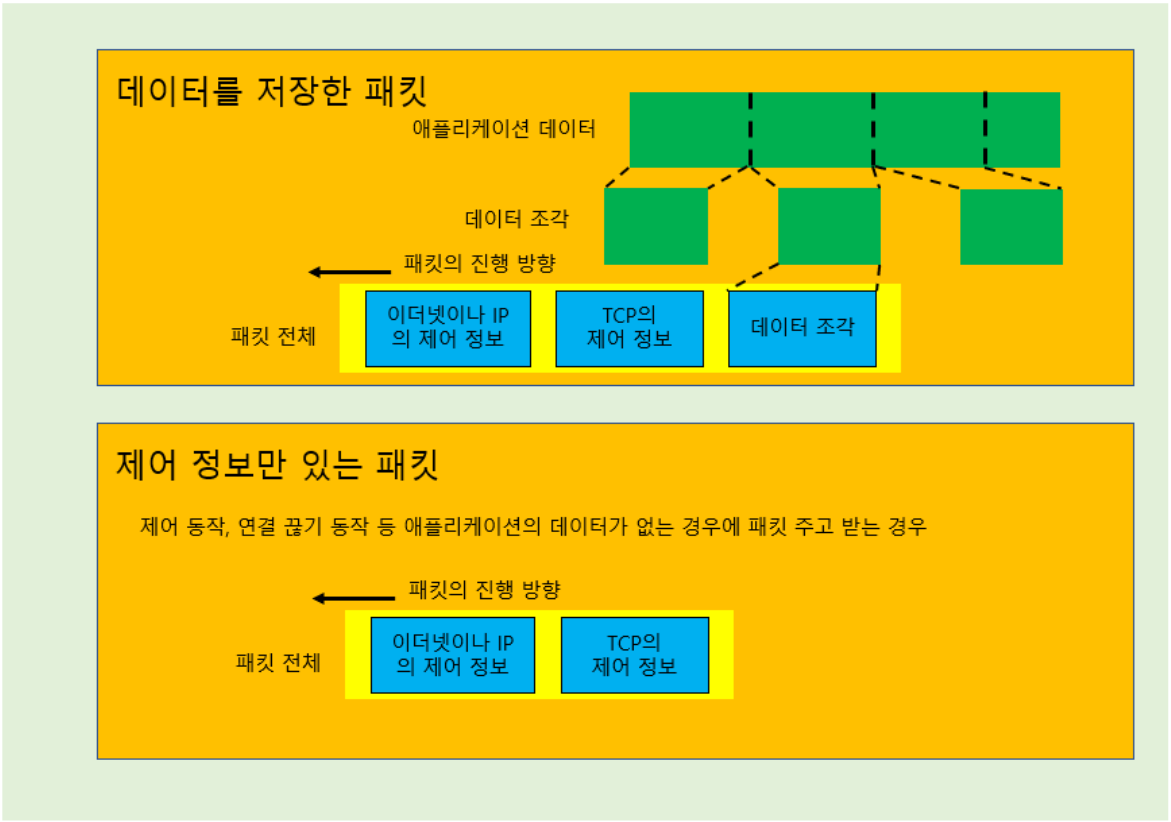


**그림 1. 클라이언트와 서버 사이에 주고받는 제어 정보**

## 3. 접속 동작 실제

애플리케이션이 Socket라이브러리의 connect를 호출하여 서버 측의 IP와 포트번호를 쓰면 명령이 프로토콜 스택의 TCP 담당 부분에 전달 된다.

connect(<디스크립터>,<서버 측의 IP 주소와 포트번호>,... )

이후 TCP 담당 부분은 서버의 TCP담당 부분과의 사이에 제어 정보를 주고 받는다.

접속 동작 과정

➔ TCP 담당 부분에서 접속을 나타내는 제어 정보를 기록한 TCP헤더를 생성한다.

➔ TCP 헤더의 송신처와 수신처의 포트 번호로 접속하는 소켓을 지정한다.

➔ 이후 SYN, ACK 와 같은 비트를 사용하여 서버와 연결을 맺는다.

파이프와 같은 것을 커넥션이라고 하며, 커넥션은 close를 호출하여 연결을 끊을 때까지 계속 존재한다.

커넥션이 이루어지면 프로토콜 스택의 접속 동작이 끝나므로 connect의 실행이 끝나면서 애플리케이션을 제어할 수 있게 된다.