2. TCP/IP의 데이터를 전기 신호로 만들어 보낸다.

2-1. 소켓을 작성한다.

1. 프로토콜 스택의 내부 구성

- **프로토콜 스택**(Protocol Stack)
    - 여러 프로토콜의 구현체
        - TCP, UDP, ICMP, IP, ARP 등
    - 네트워크 애플리케이션은 Socket 라이브러리를 이용해 프로토콜 스택을 이용할 수 있다.

2. 소켓의 실체는 통신 제어용 제어 정보

- 프로토콜이 동작하는 데 필요한 정보를 **제어 정보**(Control Information)이라 한다.
- 소켓은 제어 정보가 기록되는 파일이다.

3. Socket을 호출했을 때의 동작

- HTTP는 TCP를 사용한다.
- `socket()`의 반환값은 **파일 디스크립터**(File Descriptor)다.
    - 파일 디스크립터는 운영체제에서 파일을 관리할 때 사용하는 식별자다.
- `netstat`으로 네트워크 상태를 확인할 수 있다.

```
~/Project/1-percent-network main !2 ?1 netstat                    ✔ 18:08:59
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  121.168.239.110.56516  117.18.232.200.https   ESTABLISHED
tcp4       0      0  121.168.239.110.56515  20.205.69.80.https     ESTABLISHED
tcp4       0      0  121.168.239.110.56514  lb-140-82-114-25.https ESTABLISHED
tcp4       0      0  121.168.239.110.56512  223.130.192.187.https  CLOSE_WAIT
tcp4       0      0  121.168.239.110.56511  210.89.168.53.https    ESTABLISHED
tcp4       0      0  172.30.1.50.56492      172.30.1.26.56522      SYN_RCVD
tcp4       0      0  121.168.239.110.56510  ec2-54-150-70-46.https ESTABLISHED
tcp4       0      0  121.168.239.110.56509  ec2-54-236-104-1.https ESTABLISHED
tcp4       0      0  121.168.239.110.56508  203.246.172.120.https  ESTABLISHED
tcp6       0      0  choeseonmun-ui-m.56492 fe80::1814:fc06:.56521 ESTABLISHED
```

netstat 동작 화면

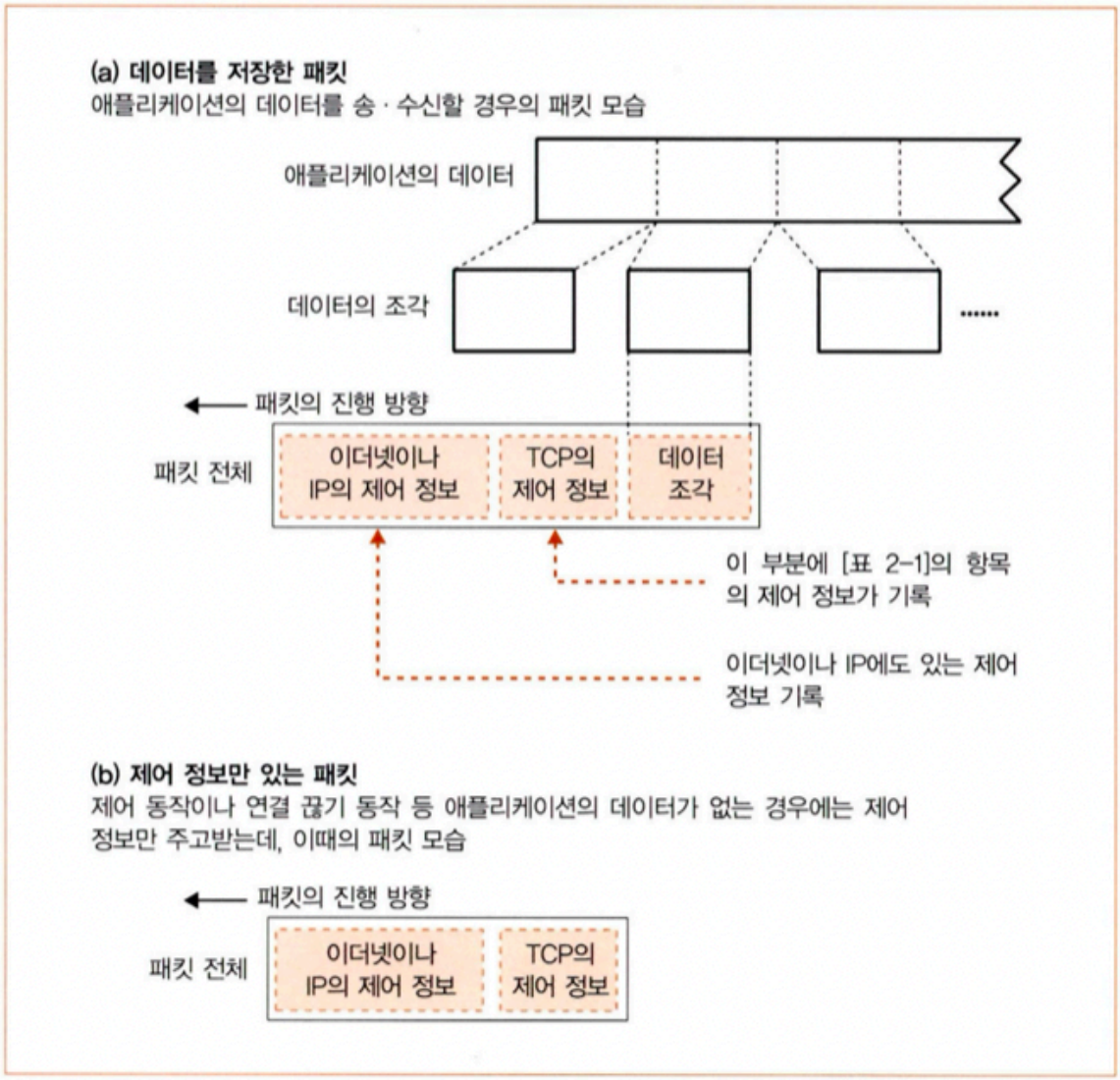
2-2. 서버에 접속한다.

1. 접속의 의미

- 통신 상대와 제어 정보를 주고받아 데이터 송수신이 가능한 상태로 만든다.
- 데이터 송수신에 사용할 버퍼 메모리도 확보한다.

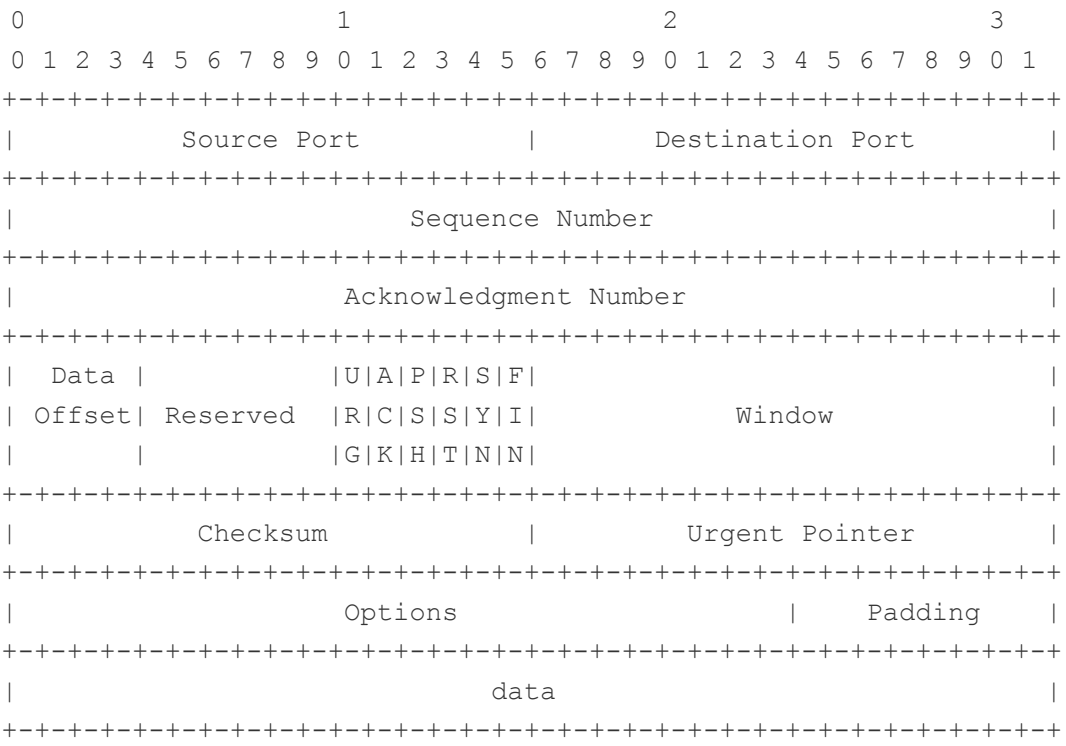
2. 맨 앞부분에 제어 정보를 기록한 헤더를 배치한다.

- 각 프로토콜의 제어 정보가 기록된 영역을 **헤더**(Header)라고 한다.



헤더는 데이터 앞쪽에 붙는다. 이를 **캡슐화**(Encapsulation)라 한다.

- TCP 헤더는 아래와 같다.



TCP 헤더 구조

필드 명칭	설명
Source Port	송신자의 포트 번호
Destination Port	수신자의 포트 번호
Sequence Number	순서번호(SN)
Acknowledgement Number	확인응답번호(AN)
Data Offset	데이터의 시작 위치
Reserved	예약된 공간으로 사용하지 않는다.
URG(urgent)	긴급 포인터가 활성화 되었음을 나타낸다.
ACK(acknowledge)	데이터가 올바르게 수신되었음을 나타낸다.

필드 명칭	설명
PSH()	flush 동작에 의해 송신된 데이터임을 나타낸다.
RST(reset)	세션을 강제로 종료하고 이상 종료시에 사용한다.
SYN(synchronize)	세션을 성립하기 위해 사용한다.
FIN(finish)	세션 종료를 나타낸다.
Window	윈도우 크기를 나타낸다.
Checksum	체크섬을 나타낸다.
Urgent Pointer	긴급하게 처리해야 할 데이터의 위치를 나타낸다.
Options	헤더 외의 옵션을 사용하기 위한 데이터이나 잘 사용되지 않는다.

TCP 헤더 필드의 의미

3. 접속 동작의 실제

- 연결 성립의 과정을 **3-way Handshake**라고 한다.
    - `connect()`을 호출하는 순간 연결 성립 과정이 일어난다.

```
클라이언트                                                          서버
1.  CLOSED                                                        LISTEN
2.  SYN-SENT    --> < SEQ=100 >< CTL=SYN >                    --> SYN-RECEIVED
3.  ESTABLISHED <-- < SEQ=300 >< ACK=101 >< CTL=SYN,ACK >     <-- SYN-RECEIVED
4.  ESTABLISHED --> < SEQ=101 >< ACK=301 >< CTL=ACK >         --> ESTABLISHED
```

(1) : 클라이언트의 소켓은 닫혀있는 상태고, 서버의 소켓은 대기 상태다.
(2) : 클라이언트가 서버에게 연결을 성립한다는 의미로 SYN 패킷과 함께 순서번호를 전송한다. 순서번호는 임의의 번호다.
(3) : 서버도 클라이언트에게 마찬가지로 연결을 성립한다는 의미와 클라이언트로부터 데이터를 잘 받았다는 의미로 SYN, ACK의 비트를 설정(set)하여 패킷을 보낸다. 이때 확인응답번호 필드에는 다음에 받을 순서번호를 기록해야 하므로 (2)에서 받았던 순서번호에 1을 더하여 기록한다. 즉, 101번의 데이터부터 달라고 클라이언트에게 알려준다.
(4) : 클라이언트가 서버에게 SYN 패킷을 잘 받았다는 의미로 ACK 패킷을 보내면서 연결이 성립된다. 이때부터 데이터를 전송할 수 있기 때문에 데이터를 같이 실어서 보내는 경우도 있다.

2-3. 데이터를 송·수신한다.

1. 프로토콜 스택에 HTTP 요청 메시지를 넘긴다.

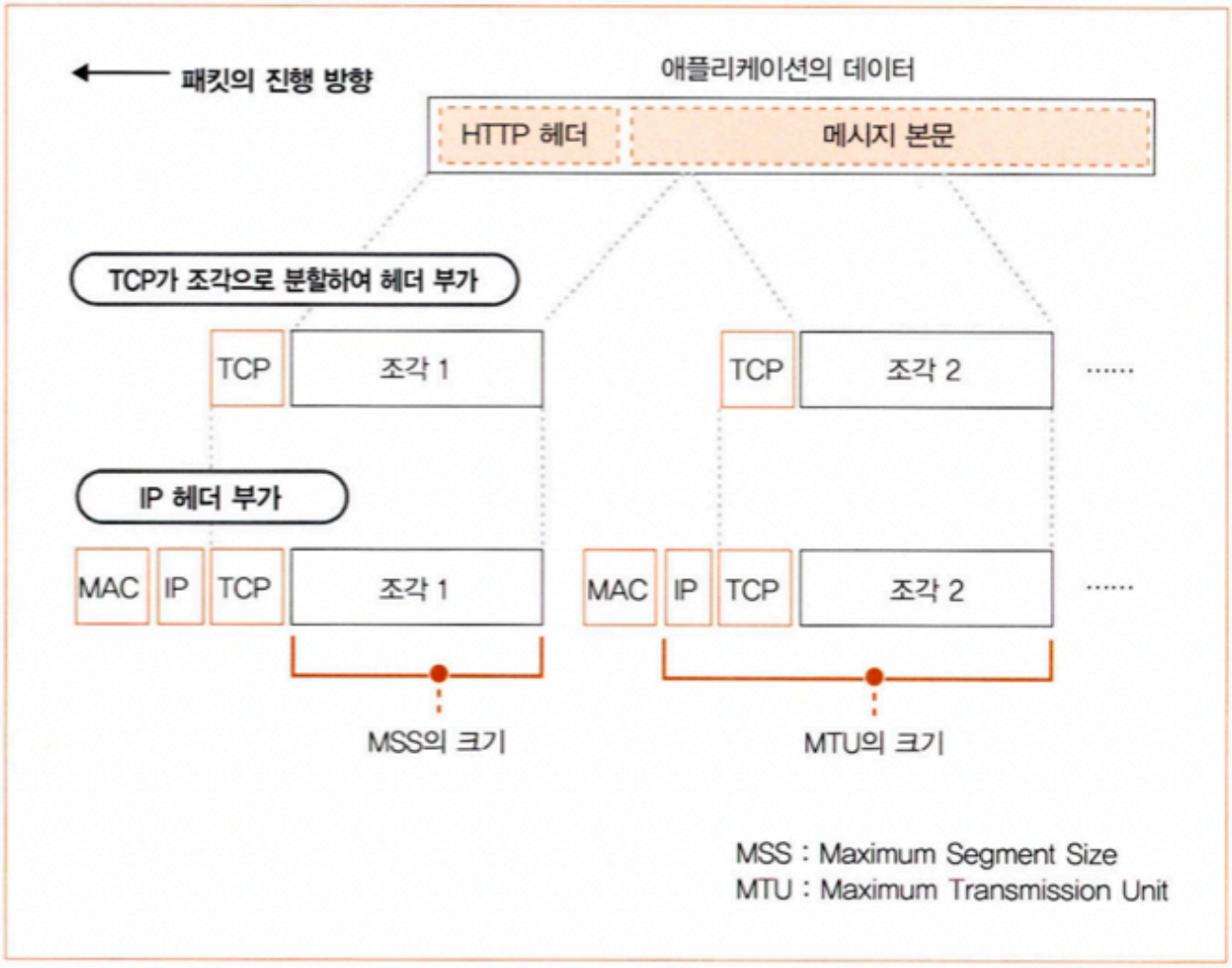
- 망에 부하를 주는 것을 방지하기 위해 데이터 송신 요청이 있을 때 바로바로 보내진 않는다.
- 송신용 버퍼가 있어 여기에 복사본을 저장해둔 후 충분히 데이터가 모일 때까지 기다린 후에 전송한다.
    - 이를 **네이글 알고리즘**(Nagle Algorithm)이라 한다.
- 한 번에 전송할 수 있는 패킷의 크기를 **MTU**(Maximum Transmission Unit)이라 하며, MTU에서 IP 헤더와 TCP 헤더를 뺀 최대 페이로드 크기를 **MSS**(Maximum Segment Size)라 한다.

MTU와 MSS

MTU와 MSS

2. 데이터가 클 때는 분할하여 보낸다.

- 보내려고 하는 데이터가 MSS를 초과하는 경우 MSS의 크기에 맞게 분할하여 보낸다.



데이터가 크면 분할한다.

3. ACK 번호를 사용하여 패킷이 도착했는지 확인한다.

- TCP는 **수신 확인응답**을 통해 송신측에게 어떤 데이터까지 수신했는지를 알려준다.
    - 수신 확인응답에 사용되는 정보는 순서번호와 확인응답번호다.
        - 순서번호에는 보내려고 하는 페이로드의 첫 번째 바이트 번호가 저장된다.
        - 확인응답번호에는 다음에 받고자 하는 순서번호가 저장된다.
- 순서번호는 악의적인 공격을 방지하기 위해 난수를 바탕으로 산출한 초기값으로 시작한다.
    - 그래서 세션을 성립할 때 서로 순서번호를 주고받는다.
- TCP는 데이터가 누락될 경우를 대비하여 송신한 데이터를 **재전송 큐**(Retransmission Queue)에 보관하고, 만일 제대로 전송되지 않았을 경우 다시 재전송한다.
    - 이를 **재전송 기반 오류 제어**(Retransmission Error Control) 혹은 검출 후 재전

송 방식이라고 한다.
    - 반면 송신한 데이터에 대해 ACK 패킷을 받았다면 복사본을 즉시 재전송 큐에서 삭제한다.
    - 수신자의 확인응답번호로 데이터의 재전송이 일어나는 것을 **자동 재전송 요청**(ARQ; Automatic Retransmission Request)이라 한다.
    - 몇 번의 재전송이 일어났음에도 데이터가 제대로 수신되지 못한다면 회복의 전망이 없는 것으로 보고, 데이터 송신 동작을 강제로 종료하고 애플리케이션에 오류를 통지한다.

4. 패킷 평균 왕복 시간으로 ACK 번호의 대기 시간을 조정한다.

- TCP는 데이터를 송신한 후 이에 대한 ACK 패킷을 일정 시간동안 대기하며, 이 시간 동안 ACK 패킷을 받지 못한다면 데이터가 제대로 도착하지 못했다고 판단한다.
    - 대기 시간 동안 ACK 패킷이 도착하지 못한 경우를 **재전송 타임아웃**(RTO; Retransmission Timeout)이라 한다.
    - 여기에 사용되는 타이머를 **재전송 타이머**(Retransmission Timer)라 한다.
    - 타임아웃 값은 **RTT**(Round Trip Time)을 기반으로 계산한다.
        - 네트워크의 상황은 시시각각 다르기 때문에 동적으로 계속 변할 수 밖에 없다. 그래서 이를 **적응적 재전송**(Adaptive Retransmission)이라 한다.

5. 윈도우 제어 방식으로 효율적으로 ACK 번호를 관리한다.

- TCP는 수신 버퍼를 가지고 있어 수신된 데이터를 일시 보관해둔다.
- TCP는 ACK 패킷을 기다리는 동안에도 계속 데이터를 송신한다.
    - 수신 할 수 있는 메모리 영역을 **윈도우**(Window)라고 하며, 윈도우 크기를 서로 통지하여 상대방이 수신할 수 있는 만큼만 데이터를 전송한다.
        - 이를 **슬라이딩 윈도우**(Sliding Window) 혹은 **연속적 ARQ**(Continuous ARQ / Go-back-N ARQ)라고도 한다.

6. ACK 번호와 윈도우를 합승한다.

- 윈도우 통지는 수신 버퍼의 빈 영역이 늘어났을 때 일어난다.
- 윈도우 통지는 ACK 패킷을 보낼 때 함께 알리게 된다.
    - 만일 더이상 수신할 수 없을 경우, 즉 윈도우 크기가 0이라면 송신자는 더이상 데이터를 송신하지 않고 대기하게 되고, 수신자는 송신자로부터 데이터가 오지 않기

때문에 더이상 수신 윈도우 변화를 통지할 수 없게 된다. 이러한 교착상태를 방지하기 위해 **윈도우 프로브**(Window Probe)가 있다.

- 송신자는 수신측의 상황을 알아보기 위해 주기적으로 **윈도우 프로브 패킷**(Window Probe Packet)을 전송한 후 이에 대한 ACK 세그먼트로 수신 윈도우 변화를 통지 받는다.
    - 이때 사용하는 타이머를 **영속 타이머**(Persistence Timer)라고 한다.

7. HTTP 응답 메시지를 수신한다.

- 응답 메시지 패킷이 도착하면 수신 버퍼에서 이를 복사해 애플리케이션에 건네준다.
- 수신 버퍼의 크기가 변화했으므로 송신 측에 윈도우를 통지한다.

2-4. 서버에서 연결을 끊어 소켓을 말소한다.

1. 데이터 보내기를 완료했을 때 연결을 끊는다.

- 연결 해제의 과정을 **4way Handshake**라고 한다.
    - 연결을 끊고자 할 때, `shutdown()`을 사용한다.

```
클라이언트                                                          서버
1.  ESTABLISHED                                                   ESTABLISHED
2.  (Close)
    FIN-WAIT-1  --> < SEQ=100 >< ACK=300 >< CTL=FIN,ACK >  --> CLOSE-WAIT
3.  FIN-WAIT-2  <-- < SEQ=300 >< ACK=101 >< CTL=ACK >      <-- CLOSE-WAIT
4.                                                             (Close)
    TIME-WAIT   <-- < SEQ=300 >< ACK=101 >< CTL=FIN,ACK >  <-- LAST-ACK
5.  TIME-WAIT   --> < SEQ=101 >< ACK=301 >< CTL=ACK >      --> CLOSED
6.  (2 MSL)
    CLOSED
```

(1) 서로 연결된 상태다.
(2) 연결 해제 요청은 보통 클라이언트가 먼저 한다. FIN 패킷을 서버에게 전송한다.
(3) 서버는 FIN 패킷을 받았다는 의미로 ACK 패킷을 전송한다. 이때를 **TCP Half-close**라 한다.
(4) 서버에서도 FIN 패킷을 전송한다.
(5) 클라이언트가 서버의 FIN 패킷을 받았다는 의미로 ACK 패킷을 전송한다.
(6) 혹시 모를 상황에 대비하여 바로 소켓을 닫지 않고 잠시 기다리다 소켓을 닫는다.

2. 소켓을 말소한다.

- 연결을 해제할 때 마지막 ACK 패킷을 보낸 후 일정 시간 동안 연결을 유지하는 이유는..
    - 마지막 ACK 패킷이 유실될 경우 재전송하기 위함이다.
    - 뒤늦게 도착한 패킷을 폐기하기 위함이다.
        - 바로 소켓을 닫아버리면 해당 포트 번호를 다른 소켓이 점유할 수도 있으며 이는 오동작을 일으킬 수 있다.
- 이때 사용되는 타이머를 **대기 시간 타이머**(Time-waited Timer)라고 한다.

참고자료

- https://www.ietf.org/rfc/rfc793.txt
- https://datatracker.ietf.org/doc/html/rfc9293