1. **tcp**와 **udp**의 차이점

tcp는 reliable하고, flow control, congestion cotrol을 지원한다.

udp는 unreliable하고, flow control과 congestion control을 지원하지 않지만, tcp보다 속도가 빠르다.

2. **packet switching** 의 문제점

패킷을 검사하고 목적지를 식별하는 과정에서 processing delay가 유발된다.

패킷이 나가는 속도보다 들어노느 속도가 더 빠르면, router의 버퍼 또는 큐에서 queueing delay가 발생.

3. **socket**의 역할은 무엇인가

프로세스 간 통신을 하기 위한 인터페이스이다.

ip 주소와 포트 번호를 조합하여 네트워크 상의 특정 서비스를 식별한다.

4. 웹 서버들은 왜 공통된 **port**를 쓸까**?**

각 도메인마다 주소가 다른데, port 번호까지 다르면 매우 복잡하게 된다. 공통된 port 사용을 통해 클라이언트와 서버 간 통신을 간편하게 하기 위해서이다.

5. **socket stream**과 **socket datagram**의 차이는 무엇인가**?**

socket stream은 tcp 통신 기반으로, 데이터가 순서대로 전달되며 손실 시 재전송된다.

socket datagram은 udp 통신 기반으로, 개별 패킷으로 데이터가 전송되며 각 패킷은 독립적으로 처리된다. 빠른 전송이 중요할 때 사용된다.

6. **multiplexing**과 **demultiplexing**에 대해서 설명해보기**.**

multiplexing이란, 하위 계층에서 상위 계층으로 데이터를 넘겨줄 때, 데이터를 모아주는 것이다.

demultiplexing은 반대로, 상위 계층에서 하위 계층으로 데이터가 전달될 때, 여러 개로 나뉘어 전달되도록 하는 것이다.

7. **sequence number**의 역할은 무엇인가**?**

패킷에 번호를 부여하여 receiver가 패킷의 순서를 확인할 수 있고 손실된 패킷을 감지할 수 있다.

8. **retransmission**을 하는 경우는 무엇인가**?**

receiver가 nak를 보내거나, sender가 ack를 일정 시간 동안 받지 못하는 경우 (timeout) retransmission을 한다.

9. **pipelined protocol**에 대해서 설명해보라**.**

기존의 데이터 전송 방식은 1개 보내고 ack 받고, 1개 보내고 ack 받는 방식인데, 이것이 비효율적이므로 ack를 받지 않아도 여러 패킷을 보내는 것이다.

go back n 과 selective repeat 방식이 있다.

10. **go back n**에 대해서 설명해보라**.**

sender가 n개의 패킷을 ack를 안 받은 상태에서 전송한다.

이때, 패킷마다 시퀀스 넘버를 부여하며, n개의 패킷을 주고 받을 때, 미수신한 패킷 앞의 패킷의 시퀀스 넘버를 ack로 보내고, 미수신한 패킷을 기준으로 다시 윈도우사이즈만큼 패킷을 재전송하게 된다.

이 과정에서 cummultaive ack를 받게 되고, 이를 처리하는 과정에서 오버헤드가 발생한다.

11. **selective repeat**에 대해서 설명해보라**.**

여러 패킷의 중간에 미수신 패킷이 있을 경우, 그 뒤의 패킷을 모두 버리는 go back n과 달리, 수신한 패킷을 버리지 않는 방식이다.

마찬가지로 ack를 받지 않아도 다음 패킷을 전송할 수 있으며, receiver는 각 패킷에 대해서 ack를 전송한다.

sender는 어느 패킷이 ack를 받았는지 관리하며, 각 패킷에 대한 time을 관리한다. → 각각의 timeout 발생 시 발생한 패킷만 전송한다.

하지만 이 방법은 모든 패킷에 대한 타이머를 관리해야하므로 프로세싱이 증가함 → window size가 커질수록 리소스가 늘어난다.

12. flow control과 **congestion control** 설명하기**.**

패킷에 문제가 발생하는 경우는 크게 두 가지.

network bandwidth의 성능이 부족한 경우.

receiver의 성능이 sender에 비해 부족한 경우.

flow control의 경우 2번 상황에서 receiver의 속도에 맞춰 sender가 데이터 전송 속도를 조절하는 것이다.

congestion control은 1번 상황에서 network bandwidth에 맞춰 데이터 전송 속도를 조절하는 것이다.

13. tcp에서 **3-way handshaking**을 설명하기**.**

3-way handshake는 tcp의 연결 시작 과정이다.

sender가 연결을 시작하기 위해 receiver에게 syn bit를 전송하면, receiver가 이에 대한 ack를 보내고, sender가 다시 승인 번호를 담아 ack를 전송하는 것이다.

14. stop and wait방식에 대해서 설명하기

sender가 데이터 1번 전송 후 receiver의 feedback이 올 때까지 전송을 stop하고 기다린다.

receiver가 ack를 보내고 sender가 이를 수신하면 다음 데이터를 보낸다.

15. tcp congestion control의 **3**단계를 설명하기**.**

크게 slow start, additive increase, multiplicative decrease 로 나뉜다.

1부터 window size를 설정하고 점차 증가시켜나간다. 이때, exponential하게 증가하며, network의 threshold에 도달하면 additive increase로 전환해 linear하게 상승한다.

additive increase를 지속하는 중 packet loss 등이 발생하면 데이터 전송량을 당시 window size의 절반으로 줄인 후, 다시 additive increase를 지속한다.

16. packet delay의 종류 **4**가지 말해보기**,** 간단히 설명

nodal processing

- 패킷을 검사하는 시간.

queueing

- 패킷이 들어오는 속도가 나가는 속도보다 빠를 경우 큐가 막히면서 발생하는 딜레이

transmission delay

- 패킷의 첫 비트부터 마지막 비트까지 나가는 시간

propagation delay

- 마지막 비트가 다음 라우터까지 도달하는 시간.

17. **http is stateless** 라는 건 무엇을 의미하나요**?**

- http는 통신 시 정보를 한 번씩 주고 받은 후 바로 끊는 형태이다. stateless라 함은 서버가 클라이언트의 이전 상태를 보존하지 않는다는 것인데, 매우 많은 사용자를 처리하는 서버에서 각 클라이언트의 상태를 저장, 관리하지 않아도 되므로 효율적이다.

18. **tcp** 연결 방식 중 하나인 **persistent http**에 대해서 설명하기

서버가 응답을 보낸 이후에도 tcp connection이 유지된다. 즉, 클라이언트와 서버 사이 요청/응답은 동일한 connection에서 이루어진다. 이를 통해 응답시간이 줄어드는 효과가 있다.

19. **tcp fairness** 에 대해서 간단히 설명하기

네트워크를 효율적으로 사용하기 위해 tcp 공정성은 네트워크에서 여러 tcp 연결이 공정하게 대역폭을 공유하는 것.

20. **tcp** 연결에서 **packet loss**가 발생하면 **loss**가 일어난 라우터부터 다시 데이터를 전송하나요**?**

첫 라우터부터 데이터를 재전송한다. 엣지에서 loss tracking을 하고, 중간 라우터는 dumb core로 전달하는 역할만 한다.