Refactorying Study
03_ TCP & UDP

2020.03.20 권문정

TCP와 UDP의 통신 방식

	TCP	UDP
정의	TCP(Transmission Control Protocol) 전송을 제어하는 프로토콜 = 인터넷상에서 데이터를 메세지의 형태로 보내기 위해 IP와 함께 사용하는 프로토콜	UDP(User Datagram Protocol) 데이터를 데이터그램 단위로 처리하는 프로토콜
연결 방식	연결 상태를 유지하여 통신 송수신지간의 연결 설정 필요	비연결형. 연결을 위해 할당되는 논리적인 경로 없음 정보를 주고 받을 때 신호절차, 연결 설정 X
패킷 교환 방식	가상 회선 방식 - 발신지와 수신지를 연결하여 패킷을 전송하기 위한 논리적 경로를 배정	데이터그램 방식 각각의 패킷은 다른 경로로 전송, 독립적으로 처리
통신 방식	1:1 통신 - Full-Duplex(전이중) 방식 : 하나의 전송선로에서 데이터가 양쪽 방향으로 동시에 전송될 수 있는 것. 수신과 송신이 동시에 가능. - Point to Point(점대점) 방식 : 물리적으로 중개 장치를 통과하지 않고, 한 지점에서 다른 지점으로 가는 방식	1:1 or 1:N or N:M 원칙적으로는 Half-Duplex 어플리케이션 개발 방식에 따라 Full-Duplex도 가능
전송 순서	전송 순서 보장 - 패킷에 번호를 부여하여 분실 확인, 추적, 목적지에서 재조립	전송 순서 바뀔 수 있음
수신 여부	확인	확인 X
에러 복구 여부	에러 복구 - 세그먼트의 오류 검출시 재전송 요구	UDP 헤더의 CheckSum 필드를 통해 최소한의 오류만 검출, 검출된 오류는 폐기 (재전송 X)
흐름제어 혼잡제어	지원	X
신뢰성	높음 (전송 순서 보장, 수신 여부 확인, 에러 복구, 흐름 제어, 혼잡 제어 지원되기 때문)	낮음
속도	느림	빠름

TCP와 UDP의 통신 방식 – 용어 정리

패킷	데이터가 전송될 때 여러 개의 조각으로 나뉜 것
데이터그램	데이터그램 : 독립적인 관계를 지니는 패킷
흐름제어 (Flow Control)	수신측 데이터 처리 속도 < 송신측 데이터 보내는 속도일때, 송신측의 속도를 줄이는 것. WHY? : 수신측의 저장 용량을 초과하면 데이터가 손실될 수 있기 때문
혼잡제어 (Congestion Control)	네트워크가 너무 혼잡할 때, 송신측이 보내는 데이터 전송 속도를 줄이는 것. WHY? : 한 라우터에 데이터가 몰리면 처리할 수 없다. 그러면 호스트들은 데이터를 재전송 하고, 이것이 혼잡을 가중시켜 오버플로우나 데이터 손실이 발생하기 때문
세그먼트	Transport Layer(전송계층)에서 교환되는 데이터 단위를 주로 지칭

TCP의 Flag

Flag	무엇인가를 기억해야 하거나, 또는 다른 프로그램에게 약속된 신호를 남기기 위한 용도로 프로그램에서 사용되는 미리 정의된 비트
SYN(synchronization)	연결 요청 플래그 : 통신 시작 시 세션을 연결하기 위한 플래그
ACK(Acknowledgement)	응답 플래그 : 송신측 으로부터 패킷을 잘 받았다는 걸 알려주기 위한 플래그
FIN(Finish)	연결 종료 플래그 : 더 이상 전송할 데이터가 없으며, 세션 연결을 종료시키겠다는 플래그
RST(Reset)	연결 재설정 플래그 : 비정상적인 세션을 끊기위해 연결을 재설정 하는 플래그
PSH(Push)	넣기 플래그 : 버퍼가 채워지기를 기다리지 않고 받는 즉시 전달함 버퍼링 없이 Application Layer(전송 계층)의 응용프로그램에게 바로 전달하는 플래그.
URG(Urgent)	긴급 데이터 플래그 : 긴급한 데이터의 우선순위를 높여 긴급하게 전달하는 플래그

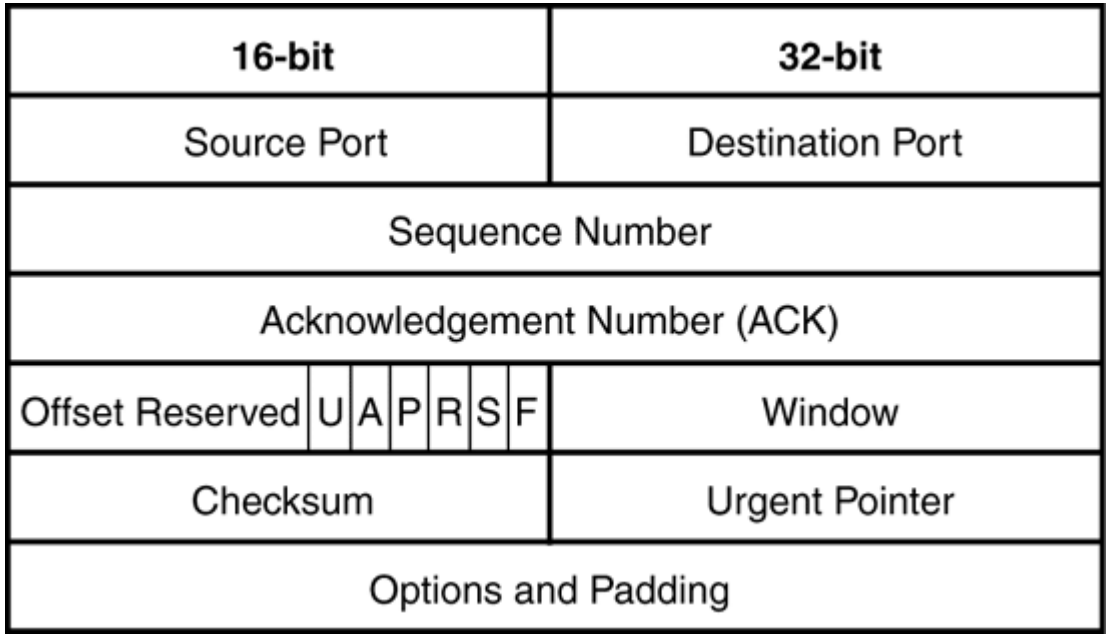
TCP의 헤더 구조

- Source Port (16 bit)
출발지 포트번호를 표시한다. 응용 서비스에 따라 포트번호가 정해져 있는 것도 있지만, 대부분의 경우 처음 세그먼트를 전송하는 측에서 임의의 번호를 사용한다.

- Destination Port (16 bit)
목적지 포트번호를 표시한다. 응용 서비스에 따라 포트번호가 정해져 있다. (EX, Telnet 23)

- Sequence Number (32 bit)
TCP 순서번호를 표시한다. 통신을 시작하는 양단의 장비들의 별개로 임의의 번호부터 시작한다.
(오류로 재전송시, 문제 있는 부분만 전송해야 하기 때문)

- Acknowledgment Number (32 bit)
상대방이 보낸 세그먼트를 잘 받았다는 것을 알려주기 위한 번호이다. (TCP의 특성상 필수)



- Offset (4 bit)
TCP헤더 길이를 4바이트 단위로 표시한다. TCP 헤더는 최소 20, 최대 60 byte 이다.

- Reserved (4 bit)
사용하지 않는 필드이며 모두 0으로 표시한다.
미래를 위해 예약되어 있는 필드이다.

TCP의 헤더 구조

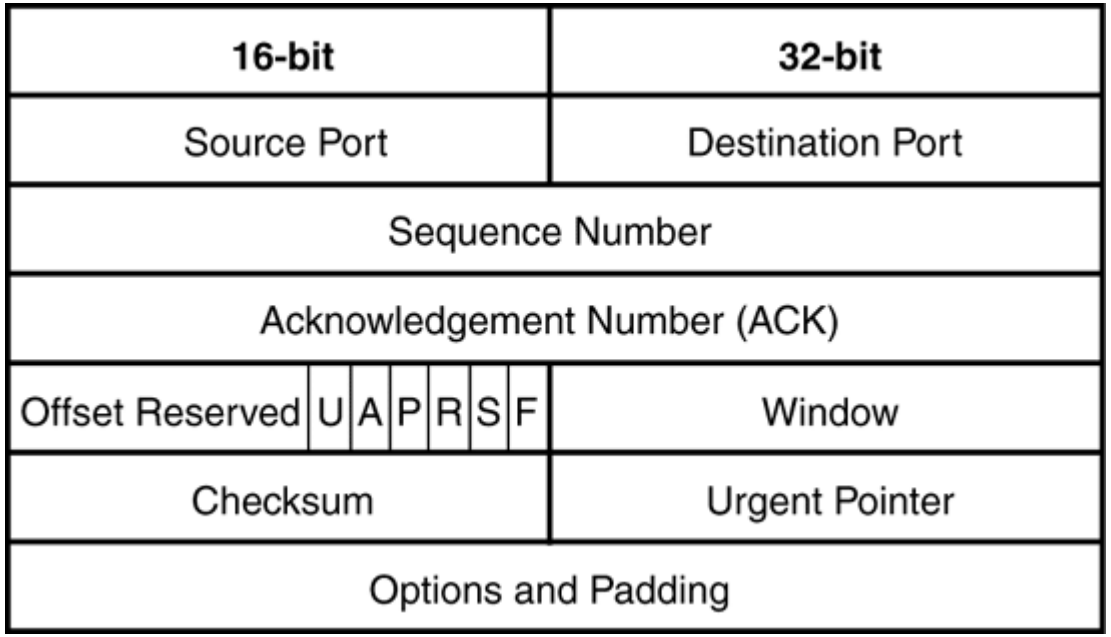
- Flags (8 bit)
제어비트(Control bits) 라고도 하며, 세그먼트의 종류를 표시하는 필드이다.

- Window size (16 bit)
상대방의 확인 없이 전송할 수 있는 최대 바이트 수를 표시한다.

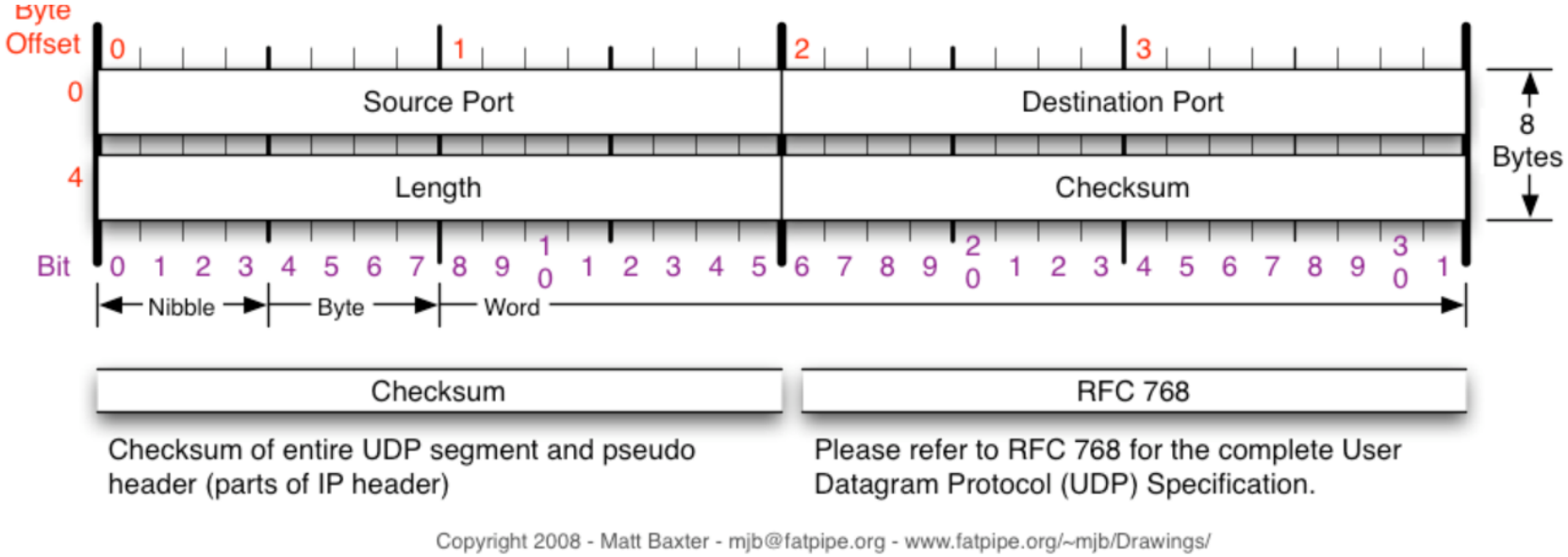
- Checksum (16 bit)
헤더와 데이터의 에러를 확인하기 위한 필드이다.
(TCP 특성상 필수)

- Urgent Pointer (16 bit)
현재의 순서 번호부터 긴급포인트에 표시된 바이트까지가 긴급한 데이터임을 표시한다. (URG 플래그 있을 때만 사용)

- Option (0~40 byte)
최대 세그먼트 사이즈 지정 등 추가적인 옵션이 있을 경우 표시한다.



UDP의 헤더 구조



- Source Port (16 bit)
출발지 포트번호를 표시한다. 응용 서비스에 따라 포트번호가 정해져 있는 것도 있지만, 대부분의 경우 처음 세그먼트를 전송하는 측에서 임의의 번호를 사용한다.

- Destination Port (16 bit)
목적지 포트번호를 표시한다. 응용 서비스에 따라 포트번호가 정해져 있다. (EX, DNS 53)

- Length (16 bit)
헤더와 데이터를 포함한 전체 길이를 바이트 단위로 표시한다.

- Checksum (16 bit)
헤더와 데이터의 에러를 확인하기 위한 필드이다.
UDP 헤더에는 에러복구를 위한 필드가 존재 X.
TCP 헤더에 비해 간단하다. 필수가 아니다. 최소한의 기능만.

3way-Handshaking

3-way Handshaking 정확한 전송을 보장하기 위해 상대방 컴퓨터와 사전에 세션을 수립하는 과정

- 양쪽 모두 데이터를 전송할 준비가 되었다는 것을 보장하고, 실제로 데이터 전달이 시작되기 전에 한쪽이 다른 쪽이 준비되었다는 것을 알 수 있도록 한다.
- 양쪽 모두 상대편에 대한 초기 순차 일련번호를 얻을 수 있도록 한다.

Step1) Client(SYN_SENT) → Server : TCP SYN

A클라이언트는 B서버에 접속을 요청하는 SYN 패킷을 보낸다. 이때 A클라이언트는 SYN 을 보내고 SYN/ACK 응답을 기다리는SYN_SENT 상태가 된다.

Step2) Server(SYN_RECEIVED) → Client : TCP SYN ACK

B서버는 SYN요청을 받고 A클라이언트에게 요청을 수락한다는 ACK 와 SYN flag 가 설정된 패킷을 발송하고 A가 다시 ACK으로 응답하기를 기다린다. 이때 B서버는 SYN_RECEIVED 상태가 된다.

Step 3) Client → Server(ESTABLISHED) : TCP ACK

A클라이언트는 B서버에게 ACK을 보내고 이후로부터는 연결이 이루어지고 데이터가 오가게 된다.
이때의 B서버는ESTABLISHED 상태이다.

TCP 의 재전송 기능 (타임아웃)

• TCP는 그 특성상 자신이 보낸 데이터에 대해서 상대방이 받았다는 의미의 응답 패킷을 다시 받아야 통신이 정상적으로 이뤄졌다고 생각한다.

• 따라서 3-way Handshaking에서 데이터를 보낸 쪽이 받는 쪽의 ACK를 받아야 한다.

• 만약 자신이 보낸 데이터에 대한 응답 패킷을 받지 못하면 패킷이 유실되었다고 판단하고 보냈던 패킷을 다시한번 보낸다. 이 과정을 TCP 재전송이라고 한다.

• 여기서 ACK를 얼마나 기다려야하는지에 대한 값을 RTO(Retransmission Timeout)라고 부른다. RTO안에 ACK를 받지 못하면 보내는 쪽에서 재전송을 진행한다.

TCP 의 재전송 기능 (타임아웃)

• RTO에는 일반적인 RTO와 Init RTO가 있다.

- 일반적인 RTO : RTT(Round TripTime, 두 종단 간 패킷 전송에 필요한 시간)를 기준으로 설정된다. 예를 들어 두 종단간 패킷 전송에 필요한 시간이 1초라면, 최소한 1초는 기다려야 내가 보낸 패킷이 손실되었는지 아닌지를 판단할 수 있다.

- InitRTO : TCP Handshake 중 첫번째 SYN 패킷에 대한 RTO이다. 맨처음 연결을 맺을 때는 두 종단 간 패킷 전송의 소요 시간을 전혀 알 수 없기 때문에 임의의 설정값으로 RTO를 계산한다.

• TCP 재전송은 보냈던 패킷을 다시한번 보내기 때문에 네크워크 성능에 저하를 가져올 수밖에 없지만, TCP 통신의 특성상 반드시 필요한 과정이다.

References

- tcp, udp 비교, 통신 방법 : https://mangkyu.tistory.com/15

- 3웨이 핸드셰이킹

https://blog.naver.com/minki0127/220696648179

https://mindnet.tistory.com/entry/%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EC%89%BD%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-22%ED%8E%B8-TCP-3-WayHandshake-4-WayHandshake

- 헤더

https://m.blog.naver.com/PostView.nhn?blogId=minki0127&logNo=220804490550&proxyReferer=https%3A%2F%2Fwww.google.com%2F](https://m.blog.naver.com/PostView.nhn?blogId=minki0127&logNo=220804490550&proxyReferer=https%3A%2F%2Fwww.google.com%2F)

https://mindnet.tistory.com/entry/%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EC%89%BD%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-19%ED%8E%B8-TCP-Header-4%EA%B3%84%EC%B8%B5-TCP-%ED%97%A4%EB%8D%94-%EA%B5%AC%EC%A1%B0?category=702276](https://mindnet.tistory.com/entry/%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EC%89%BD%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-19%ED%8E%B8-TCP-Header-4%EA%B3%84%EC%B8%B5-TCP-%ED%97%A4%EB%8D%94-%EA%B5%AC%EC%A1%B0?category=702276)

- TCP Flag
[https://hongpossible.tistory.com/entry/TCP-Flag란](https://hongpossible.tistory.com/entry/TCP-Flag%EB%9E%80)

- TCP 의 타임아웃과 재전송
https://jihooyim1.gitbooks.io/linuxbasic/contents/09.html