TCP vs UDP

날짜	@2023년 4월 22일
주차	10
카테고리	Network

TCP

> Transmission Control Protocol, 패킷을 세그먼트 단위로 처리하는 프로토콜

Ö 네트워크 계층의 **전송 계층**에서 사용하는 프로토콜

Ö **Reliable Data Transmission : 신뢰성 있는 데이터 전송으로 안정성 제공**

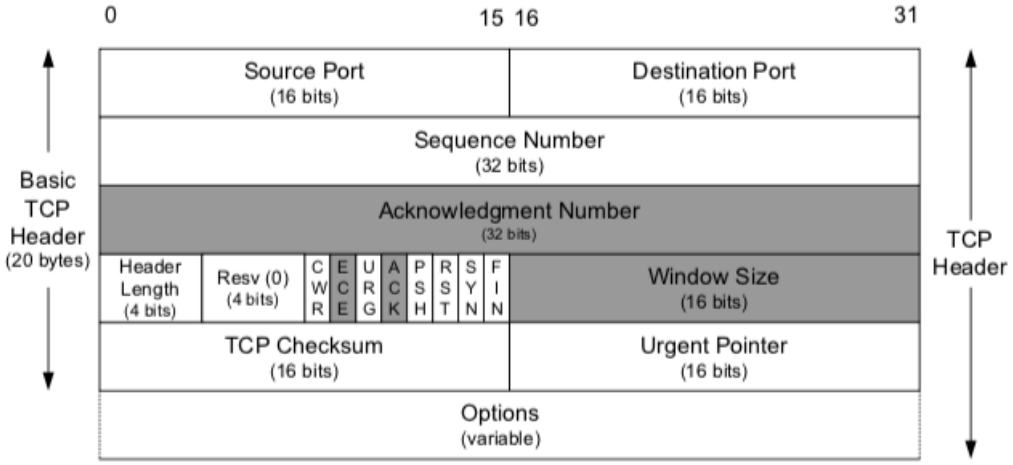
Ö 장치들 간 논리적인 접속을 위해 연결을 설정한 신뢰성을 보장하는 연결형 서비스 (connection-oriented protocol)

- 3-way handshaking으로 연결 및 4-way handshaking으로 연결 해제
- 웹 브라우저(HTTP), 메일, 파일 전송 등 신뢰성이 요구되는 애플리케이션에 사용

Ö 데이터가 확실하게 송•수신될 수 있도록 속도를 조절하고, 순서를 보장 (ordered)

- UDP에 비해 상대적으로 속도가 느림

TCP Header



Field	Contents	size
송수신자의 포트 번호	TCP로 연결되는 가상 회선 양단의 송수신 프로세스에 할당되는 포트 주소	16

Field	Contents	size
시퀀스 번호 (sequence number)	송신자가 지정하는 순서 번호로, 전송되는 바이트 수를 기준으로 증가	32
응답 번호 (ACK number)	수신 프로세스가 제대로 수신한 바이트 수를 응답하기 위해 사용	32
데이터 오프셋 (Data Offset)	TCP 세그먼트 시작 위치를 기준으로 데이터 시작 위치를 표현	4
예약 필드 (Reserved)	사용하지 않지만 나중으로 위한 예약 필드로, 0으로 채워져 있어야 함	4
제어 비트 (Flag Bit)	SYN, ACK, FIN, PSH, RST 등의 제어 번호	6
윈도우 크기 (Window Size)	수신 윈도우의 버퍼 크기 지정 시 사용, 0이면 전송 중지	16
체크섬 (Checksum)	TCP segment에 포함되는 프로토콜 헤더와 데이터에 대한 오류 검출	16
긴급 위치 (Urgent Pointer)	긴급 데이터 처리, URG 플래그 비트가 지정된 경우에만 유효	16

TCP의 대표적인 3가지 제어

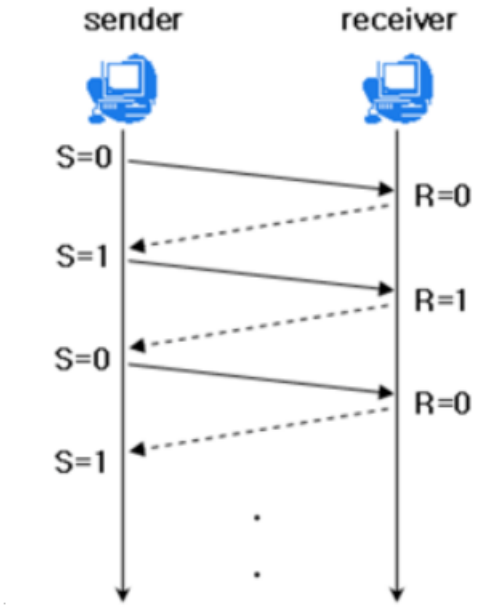
- 전송 데이터의 양을 조절하는 흐름제어
- 통신 중 데이터가 유실되거나 잘못 수신되었을 경우에 대처하기 위한 오류제어
- 네트워크 혼잡에 대처하기 위한 혼잡제어

1. 흐름제어 (Flow Control)

> 송신 측과 수신 측의 데이터 처리 속도가 상이할 시 발생 가능한 오류에 대비하기 위해 **전송되는 데이터 양을 조절하는 흐름 제어 기능**이 있다.

- 송신 측의 속도가 빠르면, 수신 측 버퍼가 넘치는 오버플로우 문제 발생 가능
  ⇒ 윈도우 크기로 송신 측 데이터 전송량을 조절

해결방법1) Stop and Wait : 상대에게 데이터를 보낸 후 응답 신호를 받고 나서 다음 패킷을 보내는 방법



해결방법2) Sliding Window : 송신 측이 수신 측에서 받은 윈도우 크기를 참고하여 데이터 흐름을 제어하는 방법

Ö 윈도우 크기 : 처리할 수 있는 데이터 양

Ö 수신 측(sender)이 한 번에 처리 가능한 윈도우 크기 만큼의 데이터를 3-way handshaking할 때 송신 측(receiver)에 전달한다.

- 상대방에게 응답을 받지 않아도 범위 내에서 데이터를 전송 가능
- 패킷의 왕복시간이 크면 네트워크가 혼잡하다고 간주 ⇒ 윈도우 크기를 실제 버퍼보다 작게 설정
- 윈도우 크기는 통신 과정 중에 유동적으로 설정

2. 오류제어

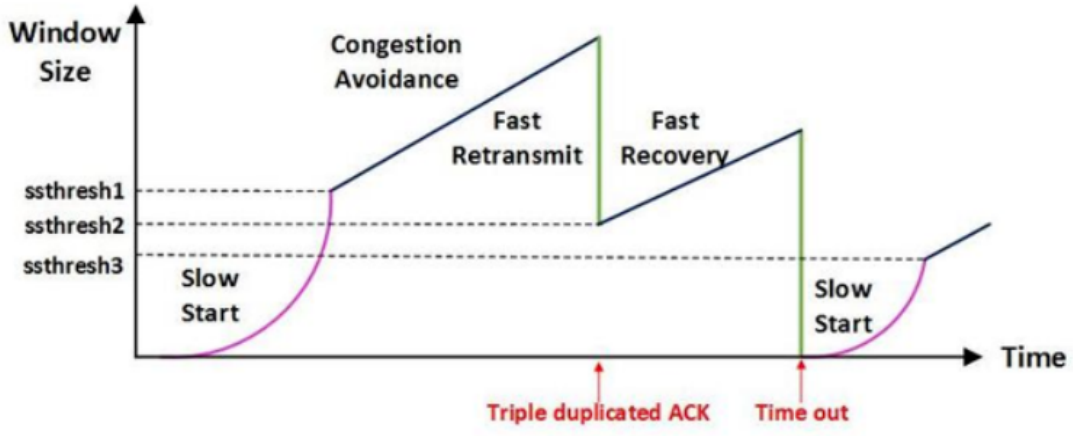
> TCP는 통신 중에 오류가 발생하면 해당 데이터를 재전송하고, 이때 재전송 기반 오류 제어인 ARQ(Automatic Repeat Request)를 사용한다.

Ö 오류인지 알 수 있는 방법?

- 송신 측이 보낸 데이터가 유실된 경우
- 수신 측이 보낸 응답(ACK) 데이터가 유실되어 송신 측이 ACK를 받지 못한 경우
- 중복된 ACK를 받은 경우
- 수신 측이 부정 응답(NACK)을 보낸 경우

3. 혼잡 제어 (Congestion Control)

> **네트워크 내에 패킷 수가 과도하게 증가하는 혼잡 현상을 방지**하고 제거하기 위한 제어 방법이다.



4가지의 혼잡제어 해결방법

해결방법 1) AIMD (Addictive Increase / Multiplicative Decrease)

Ö 패킷을 하나씩 보내다가 문제가 발생하지 않으면 윈도우 크기를 +1 증가

- 그러나 패킷 전송에 실패하거나 timeout 발생 시, 전송 속도를 절반으로 줄임

Ö 단점 : 처음에 전송 속도를 올리는 데 시간이 오래걸리고, 네트워크 혼잡 상황을 미리 감지하지 못하여 사전대처가 어렵다

해결방법 2) Slow Start (느린 시작)

Ö ACK를 받을 때마다 각 패킷마다 윈도우 크기를 +1 증가시키고, 전송 속도는 지수 함수 꼴로 증가시킨다.

- 혼잡이 감지되면 윈도우 크기 -1 감소
- 한 번 혼잡 현상 발생 시 네트워크 수용량 예측이 가능

  ⇒ 혼잡 현상이 발생하는 윈도우 사이즈의 절반까지 지수 함수 꼴로 증가시키다가 그 후에는 +1 증가

해결방법 3) Fast Retransmit (빠른 전송)

Ö 수신자 측에서 먼저 도착해야 할 패킷이 아닌 다음 패킷을 받아도 ACK 패킷을 전송

- 지금까지 받은 데이터 중 연속되는 패킷의 마지막 순번 이후를 ACK 패킷에 실어 보낸다.
- 중복된 순번의 패킷을 3개 받으면 데이터 재전송하고, 이 상황을 혼잡하다고 판단하여 윈도우 크기를 줄인다.

UDP

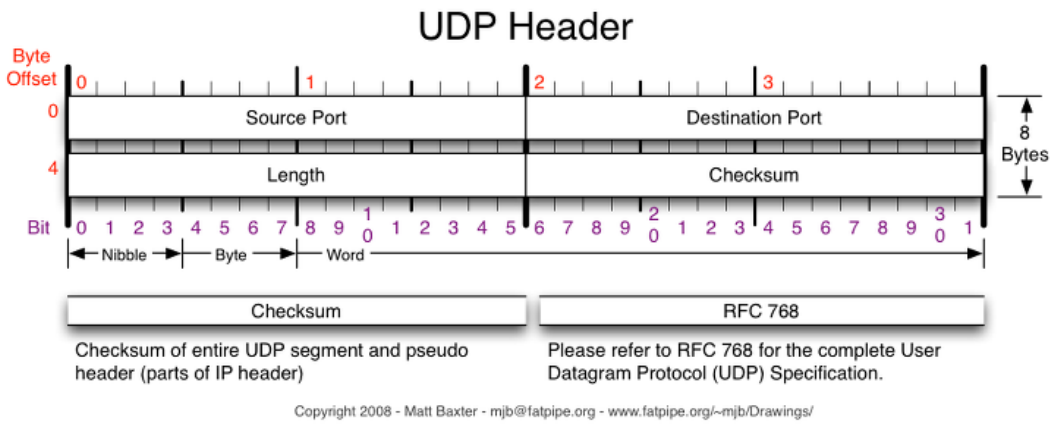
> User Datagram Protocol, 데이터를 데이터그램 단위로 처리하는 프로토콜

Ö 일방적인 데이터 전송, 비연결형 서비스 → 확인 응답이 불가능하여 TCP보다 신뢰도가 떨어짐

Ö TCP에 비해 간단하여 용량이 가볍고, 순서가 보장되지 않으며, 송신 속도가 빠름 → **데이터의 신속성**

Ö 데이터를 빠른 속도로 전송하고자 하는 애플리케이션에 사용

UDP Header



Field	Contents	size
송신자의 포트 번호 (Source Port)	데이터를 보내는 애플리케이션의 포트 번호	16
수신자의 포트 번호 (Destination Port)	데이터를 받을 애플리케이션의 포트 번호	16
데이터 길이 (length)	UDP 헤더와 데이터의 총 길이	16
체크섬 (Checksum)	TCP와 마찬가지로, 데이터 오류 검사에 사용	16

Comparison Table

공통점

Index	Contents
1	포트 번호를 이용하여 주소 지정
2	데이터 오류 검사를 위한 체크섬 사용

차이점

Index	TCP	UDP
1	연결형 프로토콜	비연결형 프로토콜

Index	TCP	UDP
2	byte 스트림을 통한 연결	message 스트림을 통한 연결
3	흐름제어, 오류제어, 혼잡제어 기능 제공	제어 기능 지원 X
4	신뢰성 있는 데이터 전송 (재전송 가능)	일방적인 데이터 전송으로 신뢰성이 낮음 (재전송 X)
5	안정적이고 속도가 상대적으로 느림	속도가 상대적으로 빠름
6	일대일 (Unicast) 통신	일대일, 일대다(Broadcast), 다대다(Multicast) 통신
7	데이터의 분실, 중복, 순서가 뒤바뀜 등을 자동으로 보정해줘 송수신 데이터	

TCP와 UDP의 오류 해결 방법

TCP	UDP
데이터의 분실, 중복, 순서가 뒤바뀜 등을 자동으로 보정해주어 송수신 데이터를 정확하게 전달할 수 있도록 해준다.	TCP와 달리 에러가 발생할 수 있고, 재전송과 데이터의 뒤바뀜을 자동으로 처리해주지 않아 애플리케이션에서 처리해야 하는 번거로움 존재