

목표: 신입 기술면접을 위한 CS 기본 지식 답변 준비

- 단골 CS 기본 지식 문항에 대해 관련 지식을 압축적으로 이해하고, 답변 준비

금일 목표

- : 다음 단골 CS 네트워크 지식 문항에 대해 구체적으로 이해하고, 답변 준비하기
 1. TCP의 3-way-handshake와 4-way-handshake 방식의 차이점에 대해 간략히 설명해주세요
 2. TCP와 UDP의 차이점과 장단점에 대해서 설명해주세요.
- 과제: 다음 수업까지, 위 문항에 대해 답변을 작성하여, 제출하기 (dream@fun-coding.org)

TCP vs UDP 프로토콜

- TCP: 연결형 프로토콜, 데이터의 전송 순서 보장, 데이터 신뢰성 보장
- UDP: 비연결형 프로토콜, TCP보다 전송속도 빠름, 주로 스트리밍/브로드캐스팅 서비스



TCP/IP 모델

- 인터넷 통신을 위한 모델
- OSI 7 Layer는 표준 모델, 현실에서 인터넷을 위해 사용하는 모델은 TCP/IP 모델
- TCP/IP 모델의 응용 계층은 OSI 모델의 세션+표현+응용 계층 통합

응용 계층 (Application Layer)	HTTP, FTP, DNS, POP3, SMTP 등등	응용 계층 (Application Layer)
표현 계층 (Presentation Layer)		
세션 계층 (Session Layer)		
전송 계층 (Transport Layer)	TCP, UDP	전송 계층 (Transport Layer)
네트워크 계층 (Network Layer)	IPv4, IPv6	인터넷 계층 (Internet Layer)
데이터링크 계층 (Data Link Layer)	Ethernet	네트워크 접속 계층 (Network Access Layer)
물리 계층 (Physical Layer)		

IP

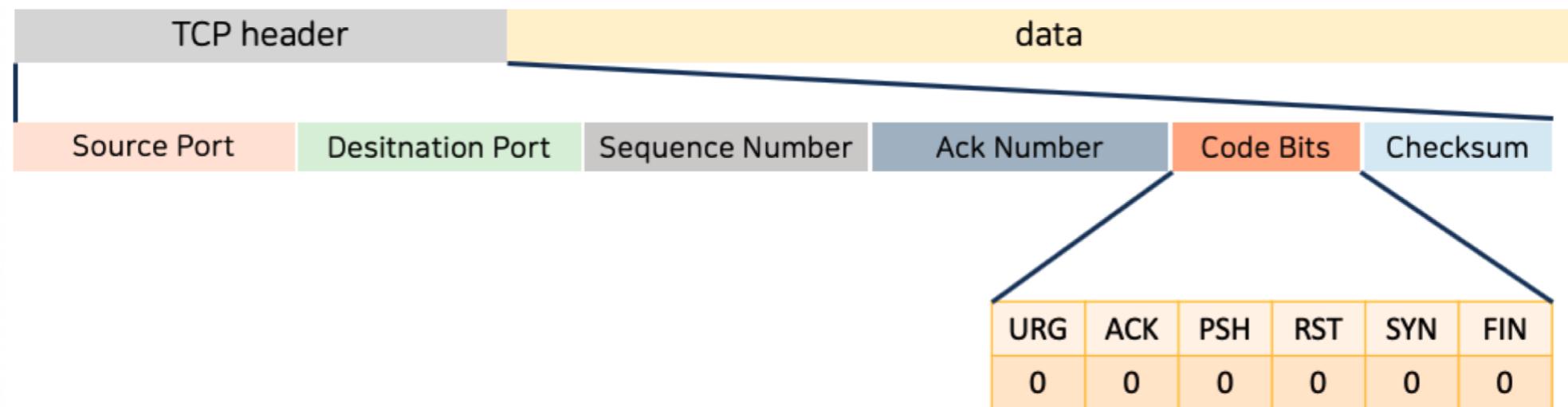
- Internet Protocol version 4
 - 32bit로 구성 (2³²개 IP 주소): 0.0.0.0 ~ 255.255.255.255
 - 비트로 표시하면: 8bit.8bit.8bit.8bit
 - 5개의 클래스로 분리, 이중 상위 3개 클래스가 주로 사용
 - **A(0~127.255.255.255)**
 - **B(128~191.255.255.255)**
 - **C(192~223.255.255.255)**
 - D(224~239.255.255.255)
 - E(240~255.255.255.255)
- 자기 PC의 IP 주소: 127.0.0.1 (DNS로는 localhost)

| 도메인주소로 IP 주소 알아내기: ping daum.net

IPv4, IPv6

- Internet Protocol version 6
 - 128bit로 구성 (2^{128} 개 IP 주소)
 - 16bit:16bit:16bit:16bit:16bit:16bit:16bit:16bit

TCP 프로토콜



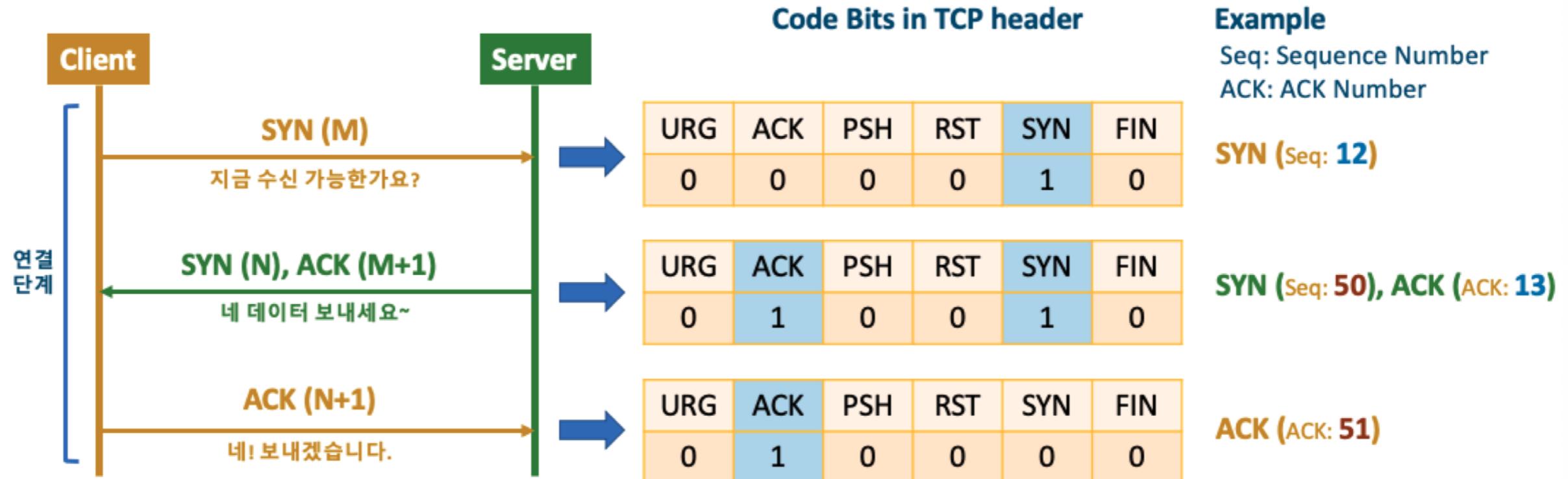
TCP 프로토콜

필드명	길이 (비트)	기능
Source Port	16	송신측 포트
Desination Port	16	수신측 포트
Sequence Number	32	송신된 데이터의 순서번호
Ack Number	32	수신된 데이터 순서번호 + 1 (그 다음 데이터를 보내달라는 의미)
Code Bits	URG	긴급 데이터임을 표시
	ACK	Ack Number 용 데이터임을 표시
	PSH	수신측 버퍼가 다 찰때까지 기다리지 않고, 바로 전달 요청
	RST	접속 리셋 요청시 사용
	SYN	연결 요청시 사용
	FIN	연결 종료시 사용

TCP 연결 방식

- 3-way 핸드쉐이크(handshake): TCP 통신을 위한 연결 설정 과정

TCP 3-way handshake



데모



JUST DO IT.

- 패킷 분석 프로그램 (wireshark)

127.0.0.1	127.0.0.1	TCP	68	58974 → 9999	[SYN]	Seq=0
127.0.0.1	127.0.0.1	TCP	68	9999 → 58974	[SYN, ACK]	
127.0.0.1	127.0.0.1	TCP	56	58974 → 9999	[ACK]	Seq=1

1. wireshark 설치 (pcap도 함께 설치)

- <https://www.wireshark.org/download.html>
- Loopback: lo0 선택 후, tcp.port == 9999

2. node.js 설치

- <https://nodejs.org/ko/download/>

3. tcp_server.js, tcp_client.js 실행 (순서대로)

- node tcp_server.js
- node tcp_client.js

이해가요 = **False**

while 이해가요:

설명듣기

책보기

프로그램작성해보기

if 이해가면:

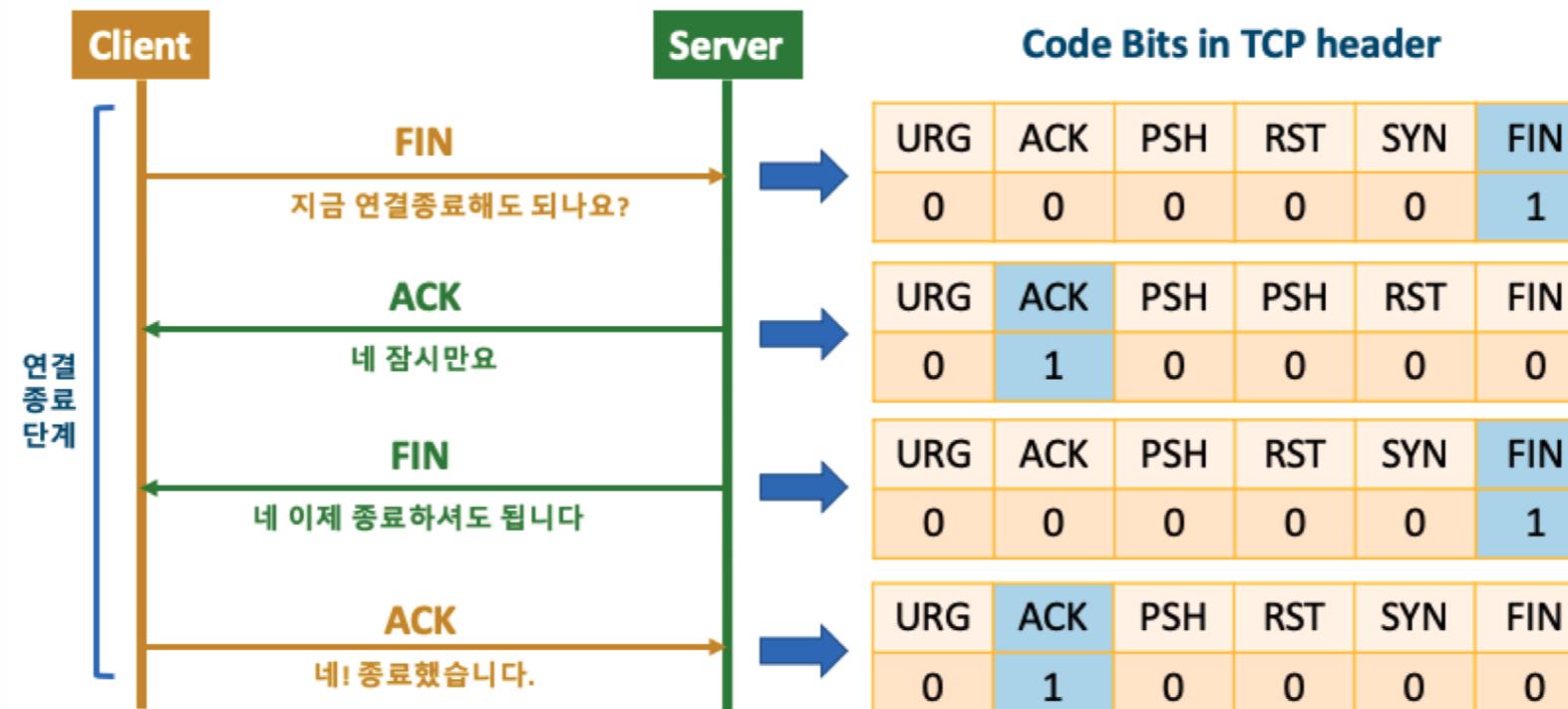
이해가요 = **True**

다음 단계

TCP 연결 해제

- 4-way 핸드쉐이크(handshake): TCP 연결 해제 과정

TCP 4-way handshake

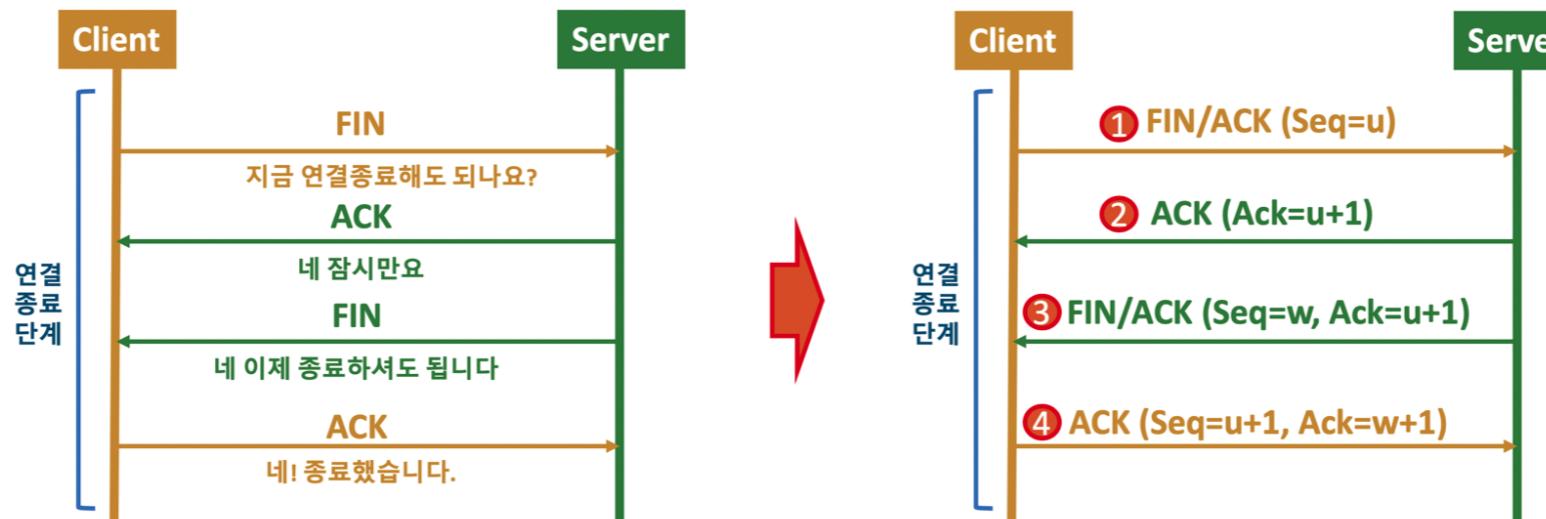


최근에는 FIN/ACK 를 동시에 보냄

TCP 연결 해제

최근에는 FIN/ACK를 동시에 보냄

tcp.port == 9999						
No.	Time	Source	Destination	Protocol	Length	Info
45	25.450006	127.0.0.1	127.0.0.1	TCP	68	64125 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSva
46	25.450143	127.0.0.1	127.0.0.1	TCP	68	9999 → 64125 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344
47	25.450150	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=1316499
48	25.450156	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 9999 → 64125 [ACK] Seq=1 Ack=1 Win=408256
49	25.450174	127.0.0.1	127.0.0.1	TCP	67	64125 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=11 TSval=1
50	25.450185	127.0.0.1	127.0.0.1	TCP	①	64125 → 9999 [FIN, ACK] Seq=12 Ack=1 Win=408256 Len=0 TSval=1
51	25.450193	127.0.0.1	127.0.0.1	TCP	56	9999 → 64125 [ACK] Seq=1 Ack=12 Win=408256 Len=0 TSval=131649
52	25.450198	127.0.0.1	127.0.0.1	TCP	②	9999 → 64125 [ACK] Seq=1 Ack=13 Win=408256 Len=0 TSval=131649
53	25.450280	127.0.0.1	127.0.0.1	TCP	③	9999 → 64125 [FIN, ACK] Seq=1 Ack=13 Win=408256 Len=0 TSval=1
54	25.450300	127.0.0.1	127.0.0.1	TCP	④	64125 → 9999 [ACK] Seq=13 Ack=2 Win=408256 Len=0 TSval=131649



TCP 전송

- Sequence Number
 - 일정 단위로 데이터를 분할해서 전송
 - 송신측에서 수신측에 데이터가 몇 번째 데이터인지를 알려줌
- ACK Number
 - 다음 번호의 데이터를 알려줌 (어디까지 수신했는지 확인 가능)

```
▼ Transmission Control Protocol, Src Port: 52940, Dst Port: 9999, Seq: 101, Ack: 1, Len: 20 ▼ Transmission Control Protocol, Src Port: 9999, Dst Port: 52940, Seq: 1, Ack: 121, Len: 0
Source Port: 52940                                         Source Port: 9999
Destination Port: 9999                                       Destination Port: 52940
[Stream index: 6]                                         [Stream index: 6]
[TCP Segment Len: 20]                                     [TCP Segment Len: 0]
Sequence number: 101 (relative sequence number)           Sequence number: 1 (relative sequence number)
Sequence number (raw): 3200123400                         Sequence number (raw): 1624755348
[Next sequence number: 121 (relative sequence number)]   [Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)          Acknowledgment number: 121 (relative ack number)
Acknowledgment number (raw): 1624755348                  Acknowledgment number (raw): 3200123420
```

$$101(\text{금번 패킷의 첫 번째 데이터 번호}) + 20(\text{데이터 길이}) = 121$$

만약 ACK가 안오거나, 101로 온다면, 101번 데이터부터 재전송

TCP 전송

tcp.port == 9999						
No.	Time	Source	Destination	Protocol	Length	Info
45	25.450006	127.0.0.1	127.0.0.1	TCP	68	64125 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=1316499
46	25.450143	127.0.0.1	127.0.0.1	TCP	68	9999 → 64125 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344
47	25.450150	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=1316499
48	25.450156	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 9999 → 64125 [ACK] Seq=1 Ack=1 Win=408256
49	25.450174	127.0.0.1	127.0.0.1	TCP	①	64125 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=11 TSval=1
50	25.450185	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [FIN, ACK] Seq=12 Ack=1 Win=408256 Len=0 TSval=1
51	25.450193	127.0.0.1	127.0.0.1	TCP	②	9999 → 64125 [ACK] Seq=1 Ack=12 Win=408256 Len=0 TSval=1316499
52	25.450198	127.0.0.1	127.0.0.1	TCP	56	9999 → 64125 [ACK] Seq=1 Ack=13 Win=408256 Len=0 TSval=1316499
53	25.450280	127.0.0.1	127.0.0.1	TCP	56	9999 → 64125 [FIN, ACK] Seq=1 Ack=13 Win=408256 Len=0 TSval=1
54	25.450300	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [ACK] Seq=13 Ack=2 Win=408256 Len=0 TSval=1316499

Seq=1(금번 패킷의 첫 번째 데이터 번호) + 11(데이터 길이) = 12

TCP 제어

네트워크 트래픽에 따른 효율적/신뢰적 송신을 위한 제어 알고리즘 적용

- 흐름 제어(Flow Control): Sliding Window
- 혼잡 제어(Congestion Control)

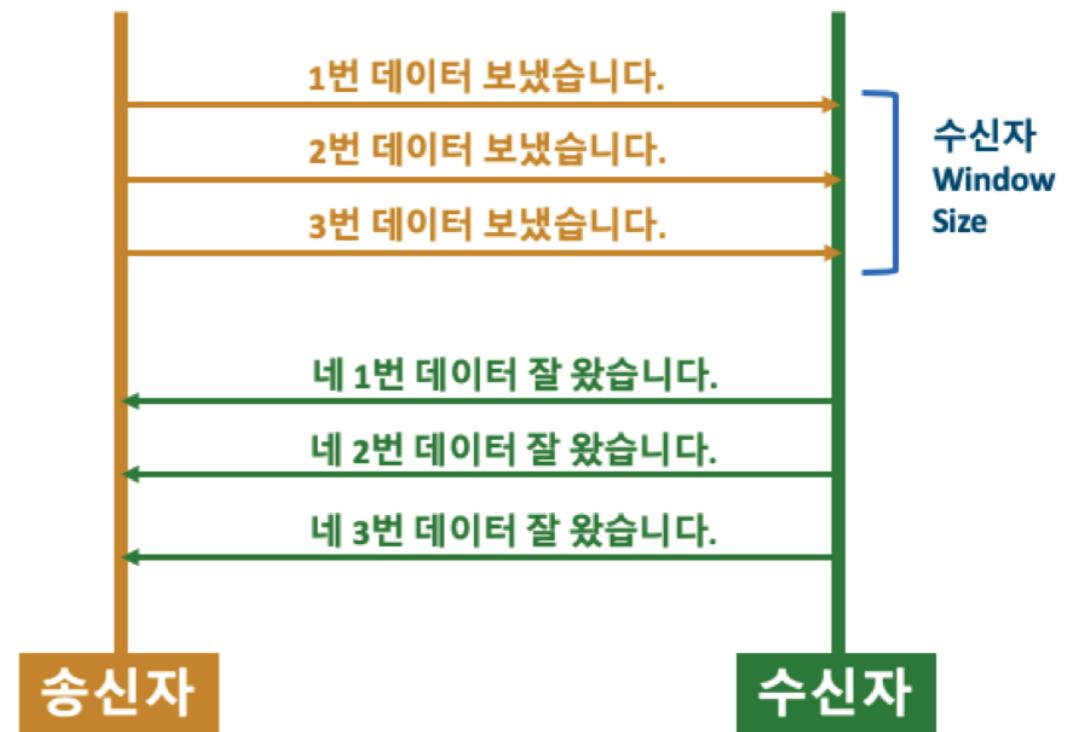
흐름 제어: 슬라이딩 윈도우 (Sliding Window)

- 매번 ACK를 기다리지 않고, 여러 패킷을 연속해서 송신하기 위해,
- 각 컴퓨터의 윈도우 사이즈를 확인하고, 윈도우 사이즈만큼 ACK 없이 연속해서 송신

Stop and Wait



Sliding Window



흐름 제어: 슬라이딩 윈도우 (Sliding Window)

전송 (초기 수신측 윈도우 사이즈: 3)



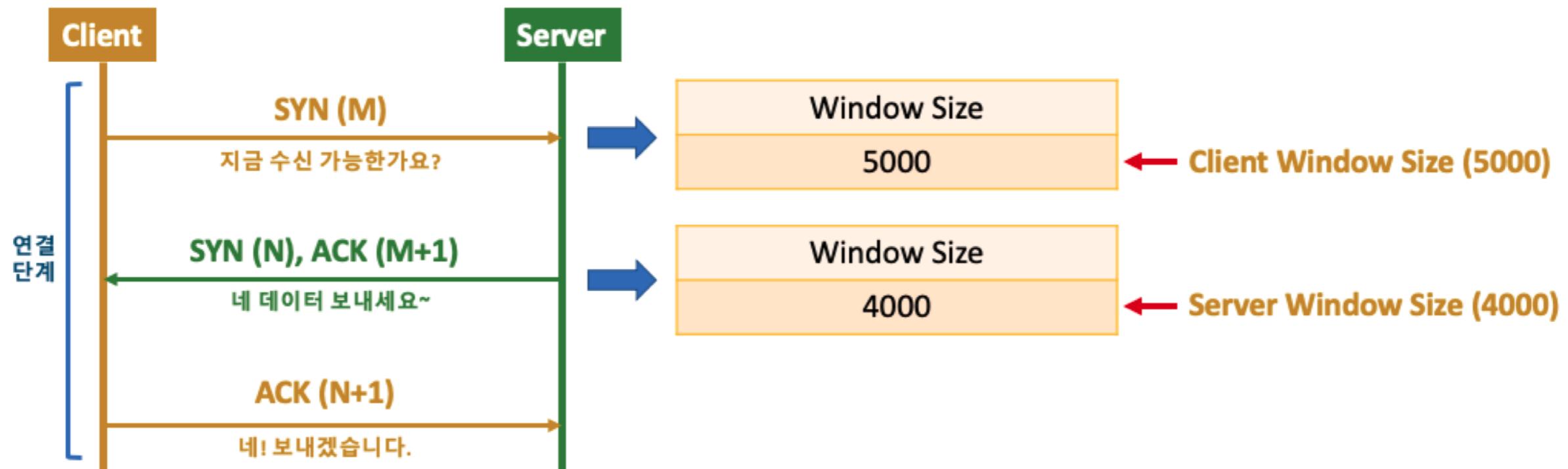
ACK 받은 데이터 이후로
윈도우 이동

전송



참고: TCP 윈도우 사이즈 설정

TCP 3-way handshake



참고: TCP 윈도우 사이즈 설정

tcp.port == 9999						
No.	Time	Source	Destination	Protocol	Length	Info
45	25.450006	127.0.0.1	127.0.0.1	TCP	68	64125 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=1316499999 Ack=1 Win=408256 Len=0 TSval=1316499999
46	25.450143	127.0.0.1	127.0.0.1	TCP	68	9999 → 64125 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 TSval=1316499999 Ack=2 Win=408256 Len=0 TSval=1316499999
47	25.450150	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=3 Win=408256 Len=0 TSval=1316499999
48	25.450156	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 9999 → 64125 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=4 Win=408256 Len=0 TSval=1316499999
49	25.450174	127.0.0.1	127.0.0.1	TCP	67	64125 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=11 TSval=1316499999 TSval=1316499999 Ack=5 Win=408256 Len=11 TSval=1316499999
50	25.450185	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [FIN, ACK] Seq=12 Ack=1 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=6 Win=408256 Len=0 TSval=1316499999
51	25.450193	127.0.0.1	127.0.0.1	TCP	56	9999 → 64125 [ACK] Seq=1 Ack=12 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=7 Win=408256 Len=0 TSval=1316499999
52	25.450198	127.0.0.1	127.0.0.1	TCP	56	9999 → 64125 [ACK] Seq=1 Ack=13 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=8 Win=408256 Len=0 TSval=1316499999
53	25.450280	127.0.0.1	127.0.0.1	TCP	56	9999 → 64125 [FIN, ACK] Seq=1 Ack=13 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=9 Win=408256 Len=0 TSval=1316499999
54	25.450300	127.0.0.1	127.0.0.1	TCP	56	64125 → 9999 [ACK] Seq=13 Ack=2 Win=408256 Len=0 TSval=1316499999 TSval=1316499999 Ack=10 Win=408256 Len=0 TSval=1316499999

- 송신측과 수신측 모두 자신의 윈도우 사이즈(버퍼 크기)를 65535로 설정
- 하지만, RTT 값을 기반으로 바로 윈도우 사이즈 재설정
 - SYN과 SYN/ACK 사이의 시간 RTT(Round Trip Time)을 측정하여, 이를 기반으로 윈도우 사이즈 재설정

실제 TCP는 기본 알고리즘 이외에 보다 복잡한 기능이 들어가 있음

혼잡 제어: Congestion Window

- 송신 제어를 위한 윈도우는 2개
 - Receiver Window (RWND): 흐름 제어(Sliding Window)에서 활용
 - Congestion Window (CWND): 네트워크 혼잡 제어를 위해 활용
- 송신측 최종 윈도우 크기 = $\min(\text{RWND}, \text{CWND})$

참고

- Congestion Window 초기값: 1 MSS

$$\text{MSS} = \text{MTU} - (\text{IP 헤더길이}) - (\text{TCP 헤더길이})$$

- MTU(Maximum Transmission Unit): 한번 전송때 보낼 수 있는 최대 단위
- 보통 $1500 - 20 - 20 = 1460$ bytes

ifconfig 또는 ipconfig 으로 확인해보기

혼잡 제어: AIMD

수십년동안 꾸준한 개선, 그 중 가장 기본이 되는 알고리즘만 이해하면 됨

- AIMD (Additive Increase/Multiplicative Decrease)
 - 처음에는, $CWND = 1$
 - ACK가 도착하면 (패킷 전송 성공시) -> $CWND = CWND + 1$
 - ACK가 일정 시간동안 도착하지 않음 (패킷 전송 실패) -> $CWND = CWND / 2$

혼잡 제어: Slow Start 및 혼잡 회피 (Congestion Avoidance)

AIMD의 느린 초기 송신을 빠르게 하기 위한 기술

- 느린 시작 (Slow Start)
 - 일반적으로 처음에는, CWND = 1
 - ACK가 도착하면 (패킷 전송 성공시) -> CWND = CWND의 2배
 - ACK가 일정 시간동안 도착하지 않음 (패킷 전송 실패) - CWND = 1 (초기값)
- 혼잡 회피 (Congestion Avoidance)
 - CWND가 일정 크기에 도달하면,
 - ACK 도착시 (패킷 전송 성공시) -> CWND = CWND + 1

UDP

- 사용자 데이터그램 프로토콜(User Datagram Protocol)
- UDP와 TCP 비교

	TCP	UDP
연결 설정	3-way handshake	없음
연결 해제	4-way handshake	없음
혼잡 제어	Sliding Window, Slow Start	없음
데이터 신뢰성	데이터 송수신 확인 (ACK)	없음

UDP

- UDP는 데이터 효율성 중시
 - 일반적으로 TCP보다 빠를 수 있음
- UDP 헤더도 단순함



UDP

udp_server.js, udp_client.js

wireshark 확인해보기

Loopback: lo0 선택 후, udp.port == 3000

UDP 장점

- 동영상 스트리밍등의 서비스에서 많이 사용됨
 - 데이터 유실이 일부 있어도 문제 없고, 빠른 전송이 필요한 서비스
- TCP와 달리 브로드캐스팅 지원
 - 브로드캐스팅: 동일 네트워크에 연결된 모든 컴퓨터에 데이터 송신 가능

팁: 특정 포트 사용 프로세스 종료하기

```
sudo lsof -i :"포트번호"  
sudo kill -9 프로세스번호
```