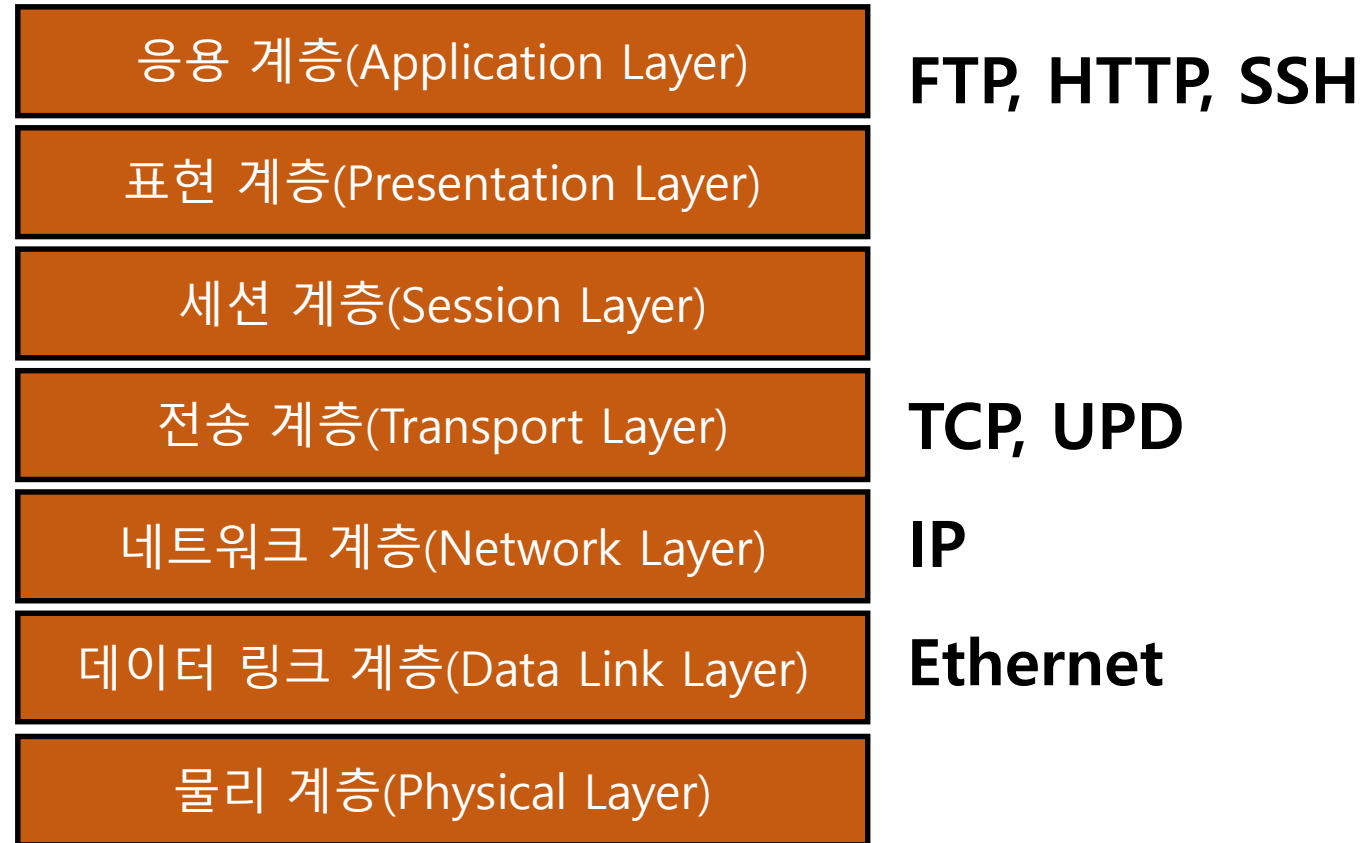
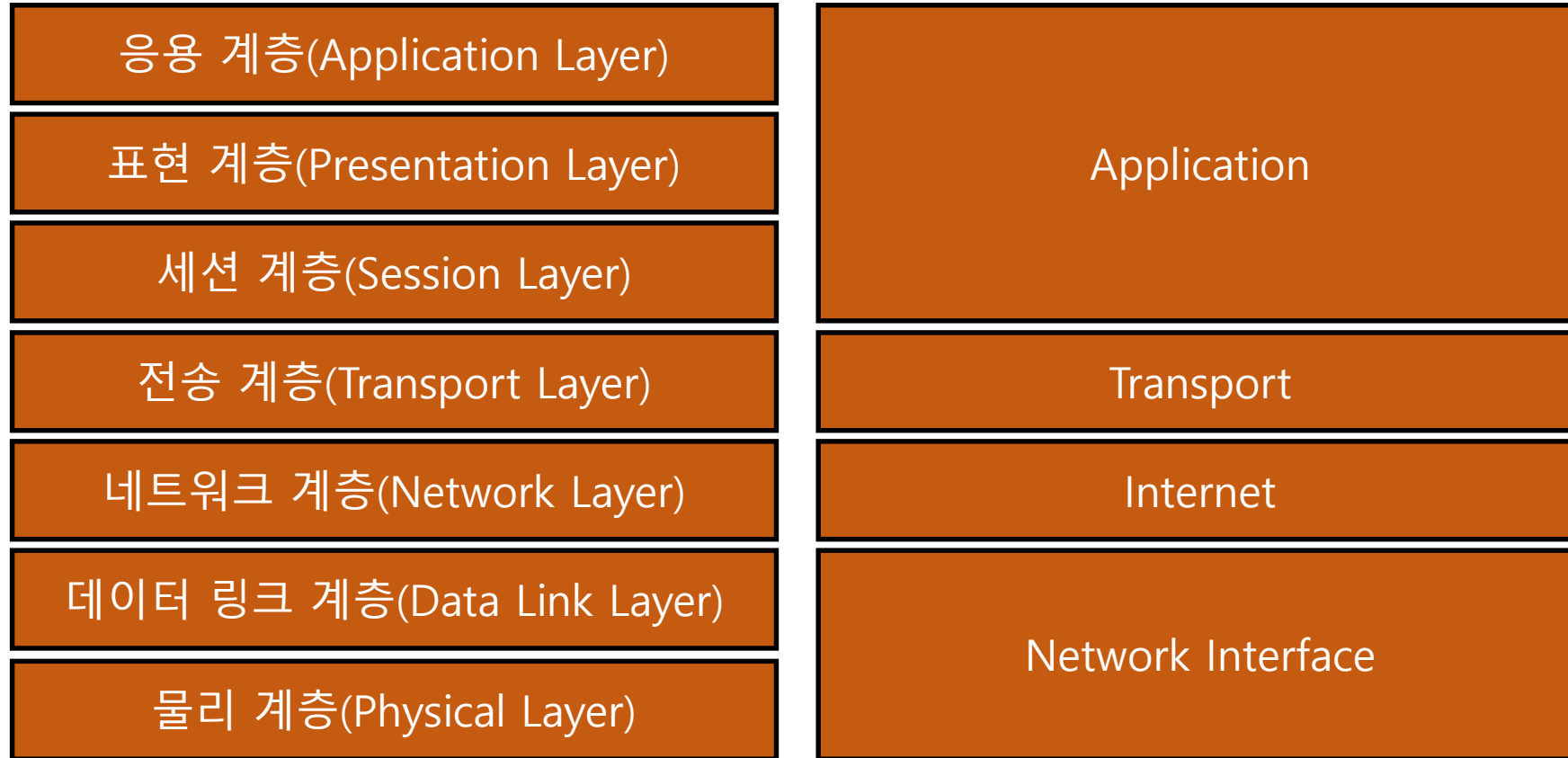


## OSI 7 계층



## TCP/IP



## 1. 물리 계층

**LAN cable : CAT 5 100Mbps, 10/100 BASE-T(IEEE 802.3)  
UTP(Unshielded Twisted Pair)  
RJ-45**

**Repeater : 거리가 멀어지면 노이즈가 생기고 신호가 약해진다  
signal 증폭**

## 2. 데이터 링크 계층

**NIC(network interface card)**

: 일반적으로 랜 카드라고 불린다. 네트워크 어댑터

**MAC(Media Access Control)**

: NIC의 하드웨어 주소

**40-49-0F-80-C3-2F**

제조사

NIC 번호

## 이더넷 프로토콜

1. Preamble : 7 bytes, NIC에 패킷이 들어온다고 알린다.
2. SFD(start frame delimiter) 1 byte, 10101011 → 최초 패킷
- 3. Destination MAC Address : 6 bytes, 패킷 수신 NIC**
- 4. Source Mac Address : 6 bytes, 패킷 송신 NIC**
5. Length or Type : 2 bytes
6. Data : 0 ~ 1500 bytes, 전송 데이터,  
**MTU(maximum transmission unit) : 1500 bytes**
7. Pad : 64 bytes를 맞추기 위해 임의의 데이터를 쓴다
8. FCS(Frame Check Sequence) : 4 bytes, 패킷 오류 검사

### 3. 네트워크 계층

## ARP(Address Resolution Protocol)

: 브로드캐스트로 어떤 IP를 사용하는 호스트의 MAC 주소를 알아낸다.

Request packet

1. target MAC  
00:00:00:00:00:00
2. target IP  
192.168.1.4

Source Host

**Broadcast**



Local network

Response packet

1. sender MAC  
28:5A:EB:67:44:86
2. sender IP  
192.168.1.4

Destination Host

## ARP

- Source Host

1. ARP cache에서 dest NIC 검색 : 있다면 바로 데이터그램 전송!
2. Cache에 없다면 ARP 요청 프레임 생성
  - 1) Sender Hardware Address(SHA) : Source MAC address  
Sender Protocol Address(SPA) : Source IP address
  - 2) Target Hardware Address(THA) : EMPTY!!  
Target Protocol Address(TPA) : Destination IP address
3. ARP request message Broadcast!!

## ARP

- Destination Host

1. ARP 응답 프레임 생성

1) Sender Hardware Address(SHA) : Destination MAC address

Sender Protocol Address(SPA) : Destination IP address

2) Target Hardware Address(THA) : Source Mac address

Target Protocol Address(TPA) : Source IP address

2. ARP cache 갱신

3. ARP response message UNICAST!!



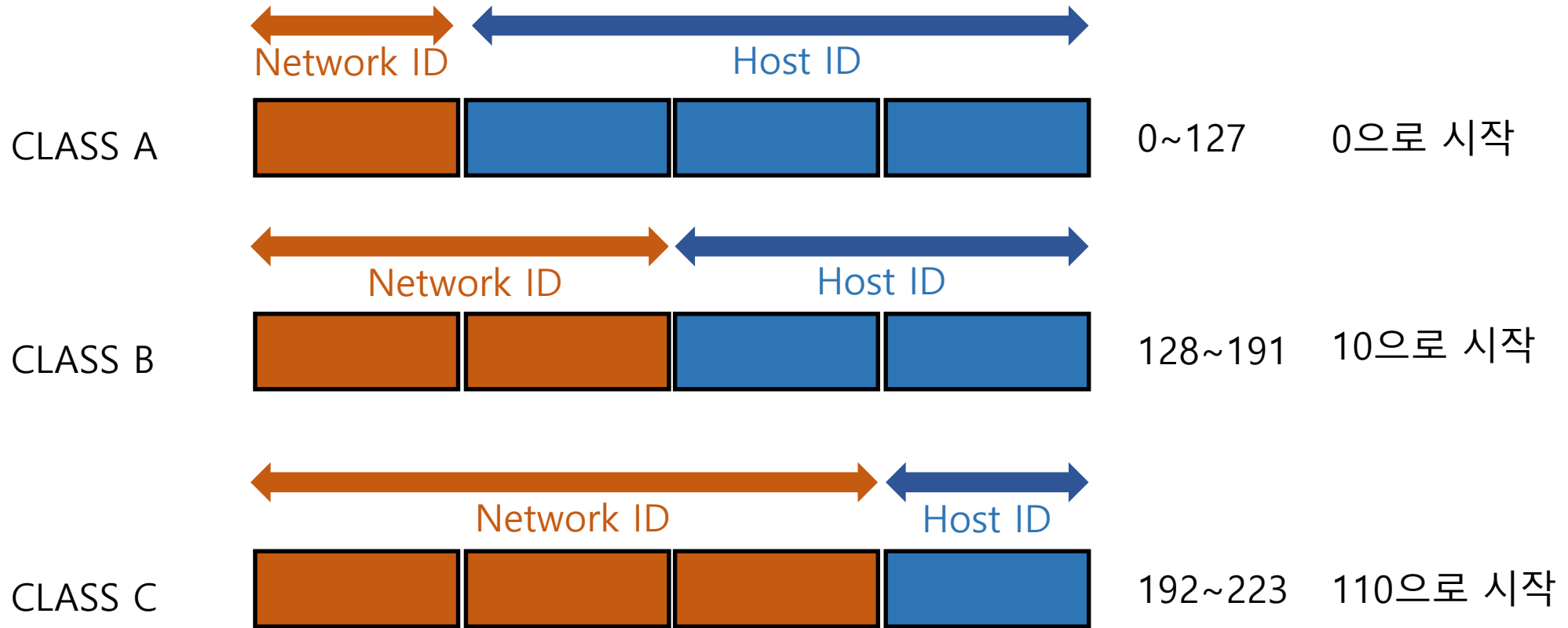
ARP

- Source Host
  1. ARP cache 갱신!!

## IP

1. Version : 4bits, IPv4 → 0x4
2. TTL : 1byte, Time to live, 몇 개 라우터를 지나면 패킷을 버릴 것인가?
3. Protocol: 1byte, 상위 프로토콜, 6:TCP, 17:UDP
- 4. Source Address : 4 bytes, 송신 IP**
- 5. Destination Address : 4 bytes, 수신 IP**
6. Data : 전송 데이터

## IP – 클래스 단위 주소 지정



IP – 서브넷 주소 지정

IP address : 201.175.122.74

Subnet mask : 255.255.255.192

201은 class C

Subnet mask

11111111.11111111.11111111.11000000



## IP – 서브넷 주소 지정

IP address : 201.175.122.74  
Subnet mask : 255.255.255.192



IP address : 201.175.122.74/26

11001001.10101111.01111010.01001010

11111111.11111111.11111111.11000000

---

11001001.10101111.01111010.01000000



Network ID

Subnet ID Host ID

## IP

Public IP(공인 IP 주소)

: globally unique IP

Private IP(사설 IP 주소)

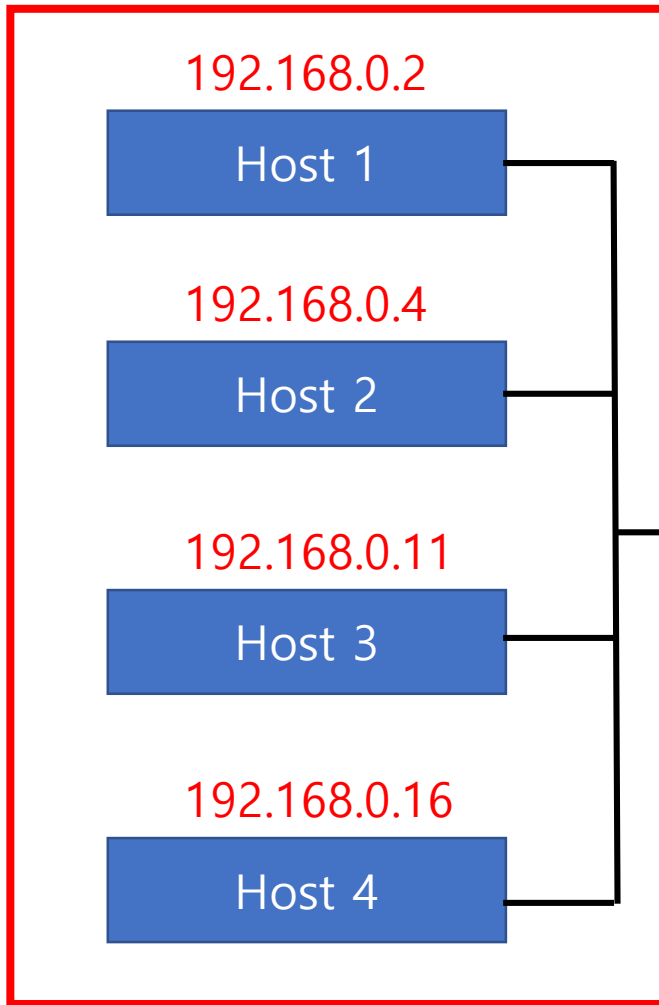
: Private network 상에 존재

NAT를 통해 인터넷에 connect 가능

하지만 인터넷에서 Private IP address로 connect 불가능

구분	Private network
CLASS A	10.0.0.0 ~ 10.255.255.255
CLASS B	172.16.0.0 ~ 172.31.255.255
CLASS C	192.168.0.0 ~ 192.168.255.255

## Private network

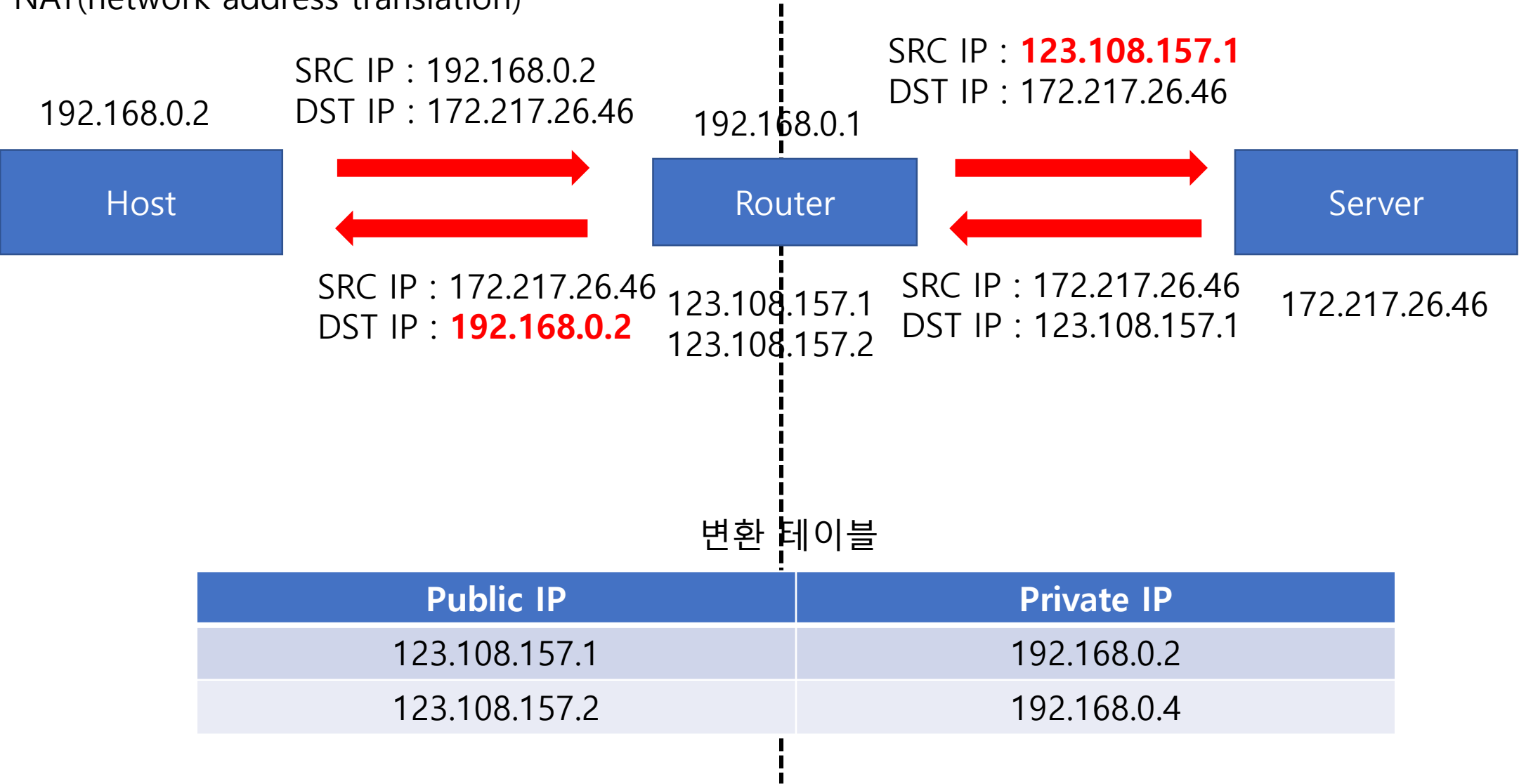


하나의 public IP 주소를 라우터에 할당  
Private network를 구성해 이 공인 IP 주소를 공유

Private IP address  
192.168.0.1

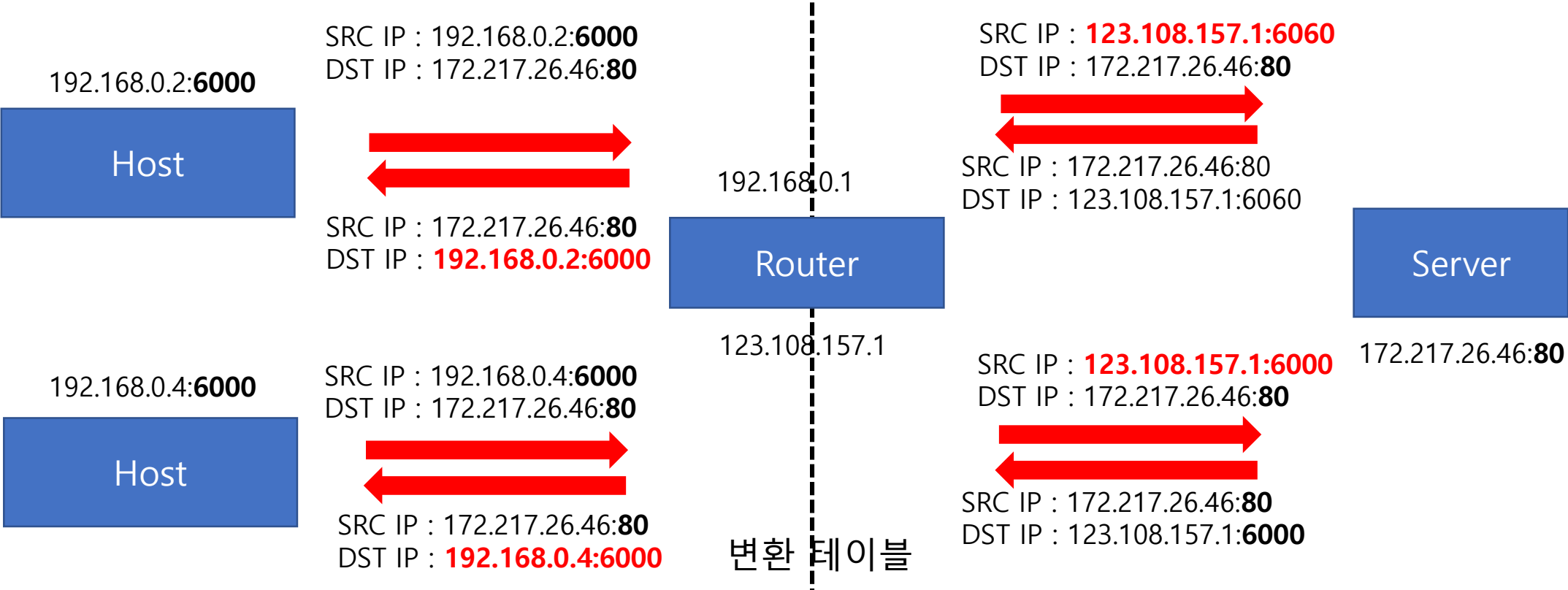
Public IP address  
211.57.44.2

NAT(network address translation)





NAPT(network address port translation)



Public IP:PORT	Private IP:PORT
123.108.157.1:6060	192.168.0.2:6000
123.108.157.1:6000	192.168.0.4:6000

#### 4. 전송 계층

Port : 소켓에 할당된 주소  
특정 프로세스로 데이터를 전달할 수 있다.

well-known port : 0~1023 (server)  
dynamic port : 49152~65535 (client)

Well-known port	Service
21	FTP
22	ssh
23	Telnet
25	SMTP
53	DNS
80	http

TCP

## **TCP(Transmission Control Protocol)**

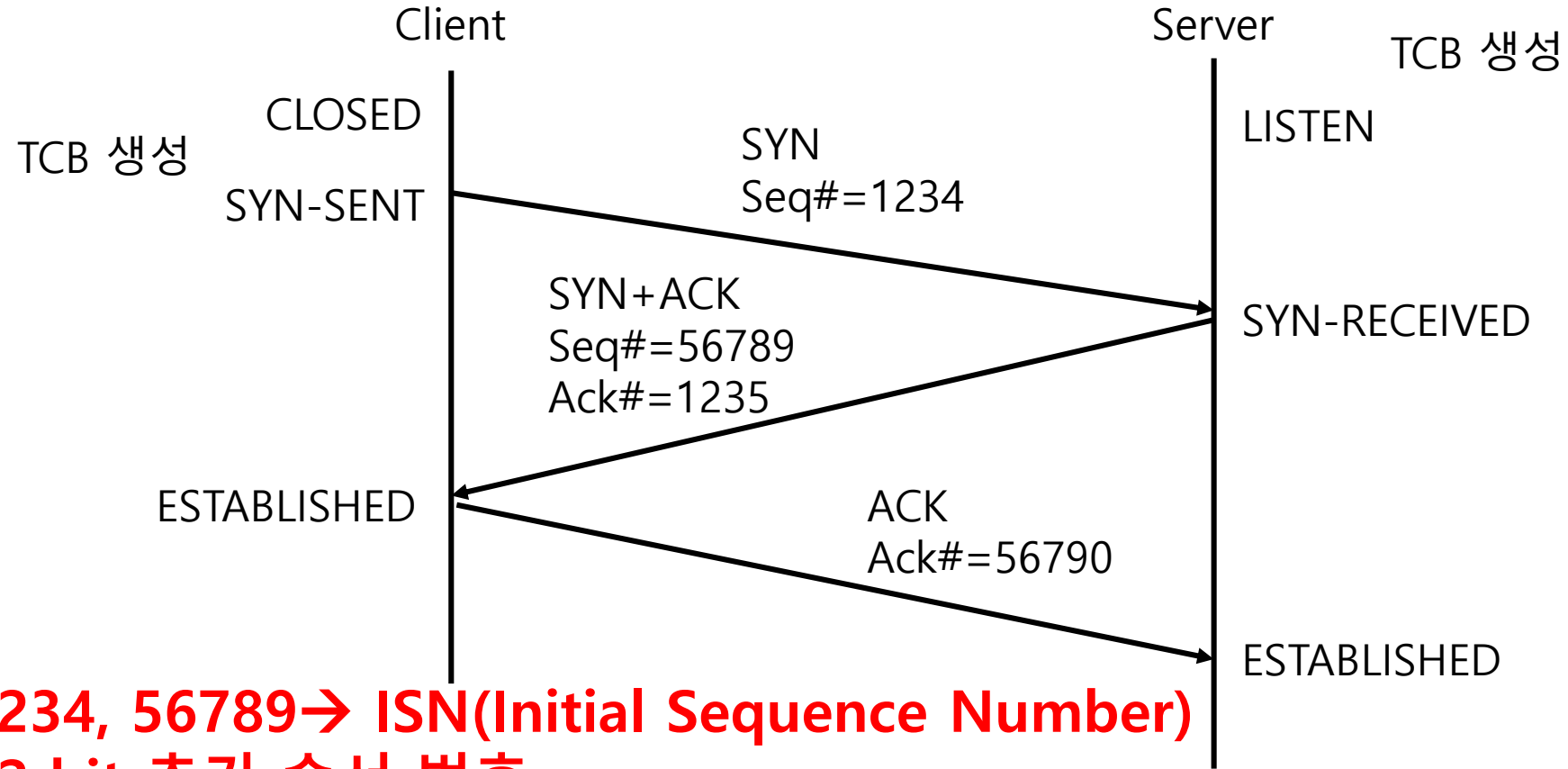
1. 연결 지향형 프로토콜
2. 높은 신뢰성
3. 수신 호스트가 응답하지 않으면 일정 시간 후 데이터를 재전송

TCB

## **TCB(Transmission Control Block)**

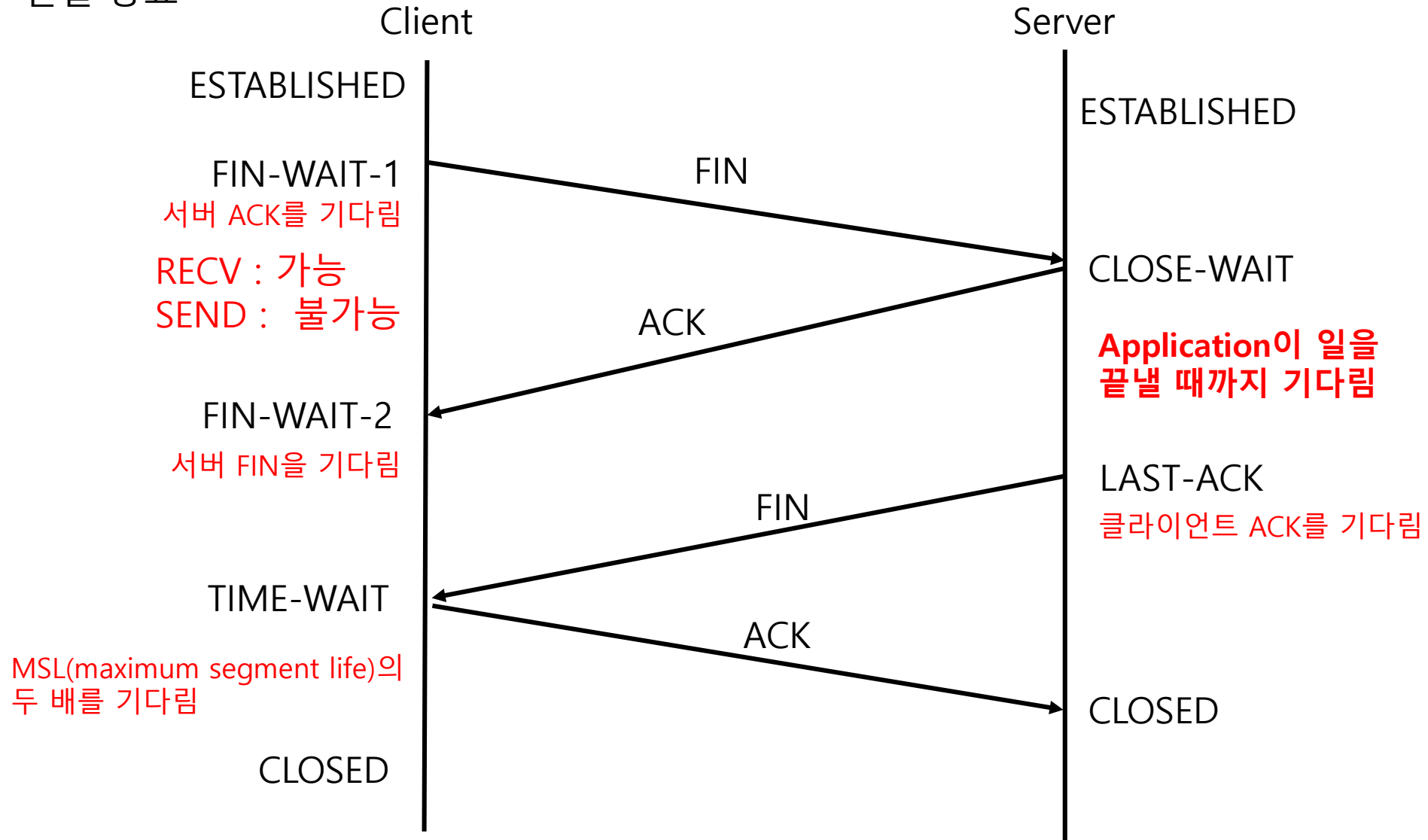
1. 연결을 구분하기 위한 소켓 쌍 번호
2. Send buffer와 receive buffer에 대한 포인터
3. Sent+ACK, Sent+NOT ACK, window 크기

## Three-Way Handshaking



**1234, 56789 → ISN(Initial Sequence Number)**  
**32 bit 초기 순서 번호**

## TCP 연결 종료



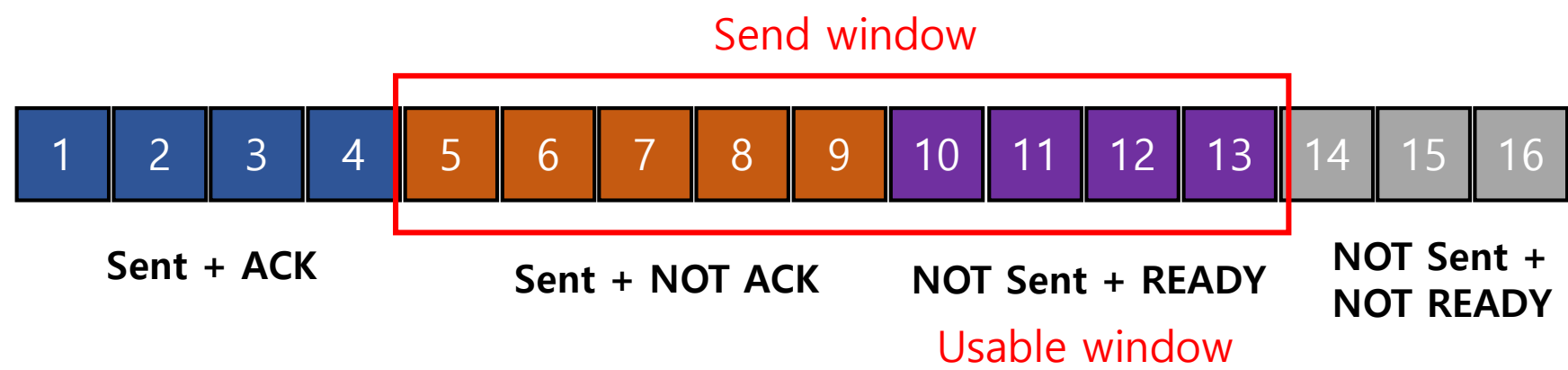
## TCP 헤더

**1. Source Port : 2 bytes, 송신 Port**

**2. Destination Port : 2 bytes, 수신 Port**

- 3. Sequence Number : 4 bytes, 이번에 보내는 데이터의 첫번째 바이트 순서 번호
- 4. Acknowledgement Number : 4 bytes, 그 이전 데이터는 모두 받았다!!
- 5. Window : 2 bytes, 송신자의 수신 윈도우 크기, 수신자의 송신 윈도우 크기와 같다

# Sliding Window

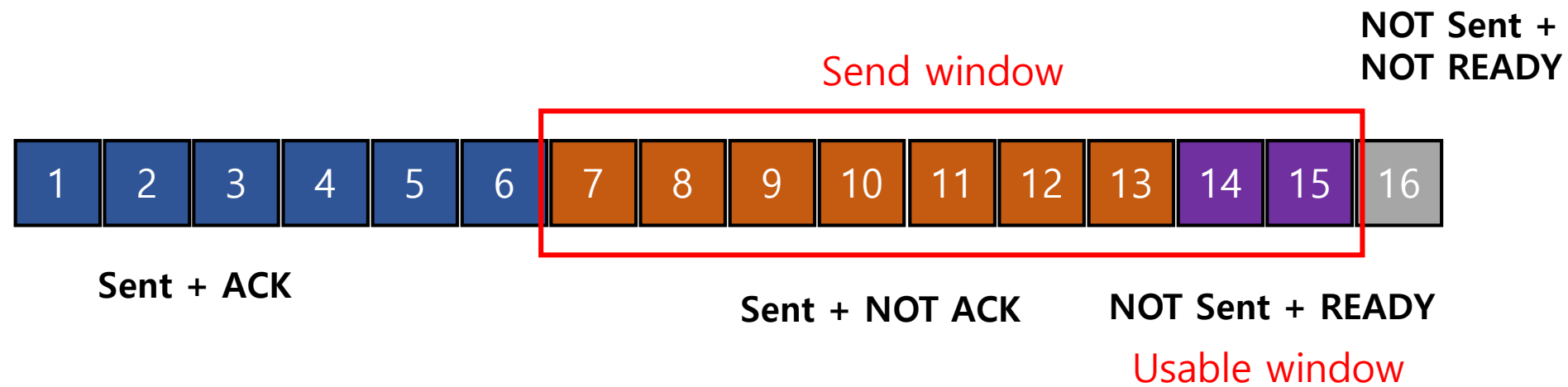




# Sliding Window

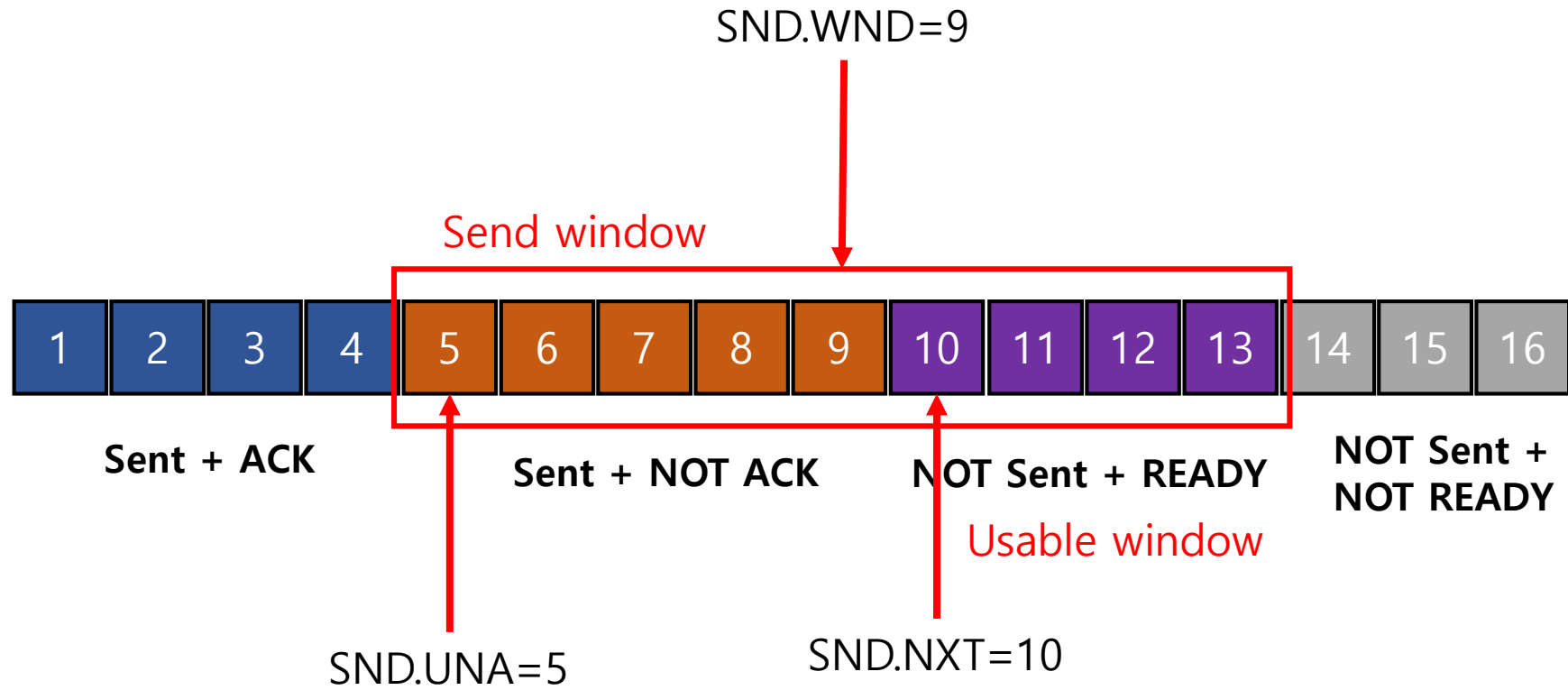


Sliding Window

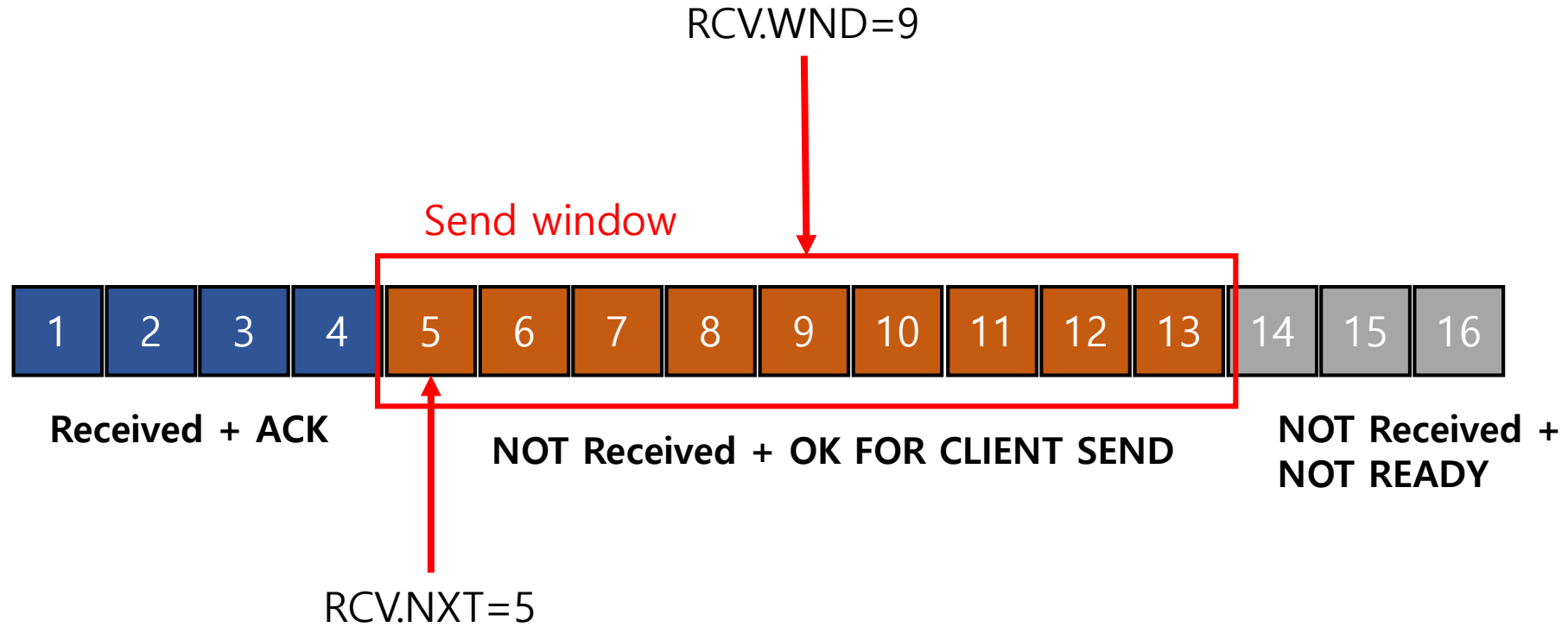


**ACK: #7**

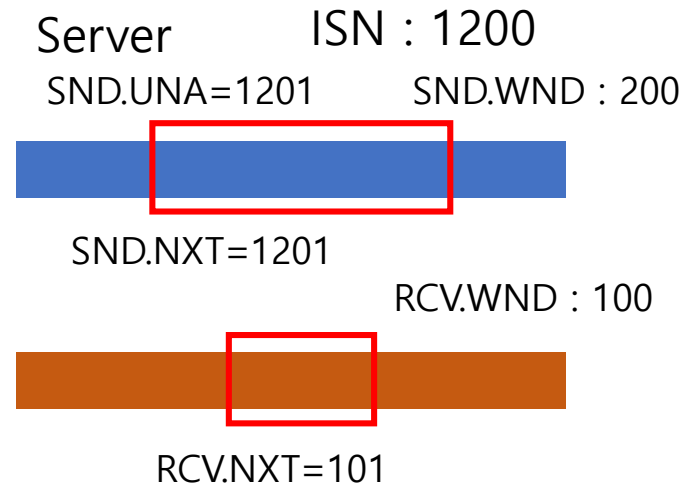
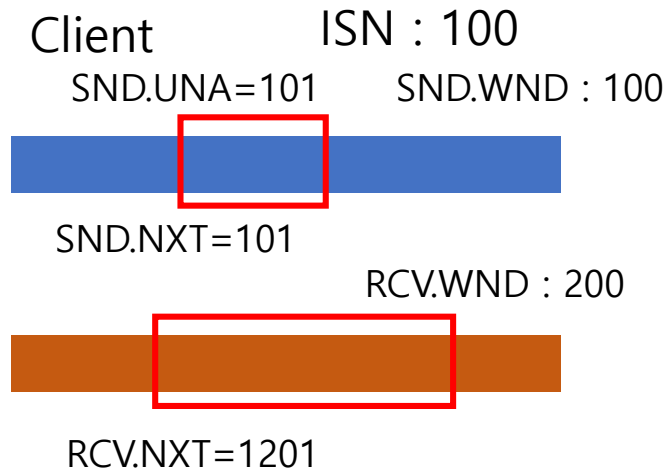
SND 포인터(client)



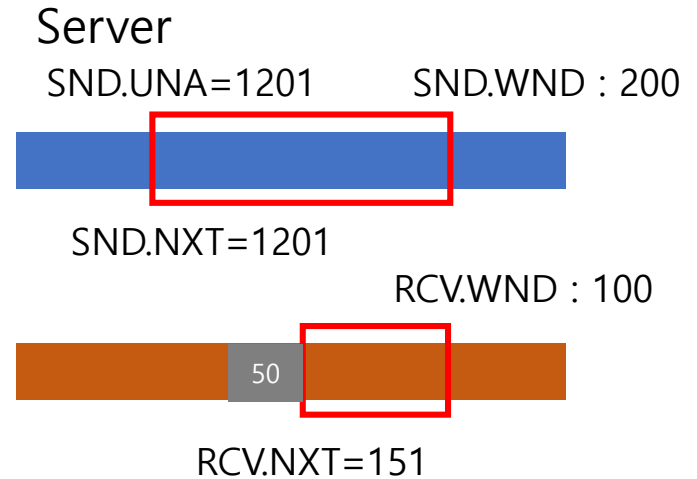
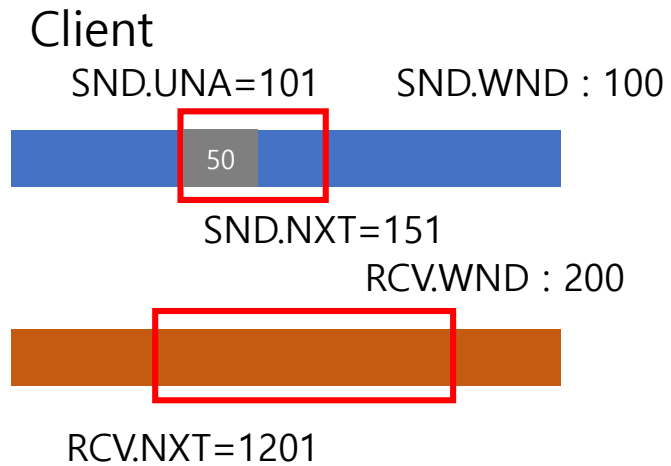
RCV 포인터(server)



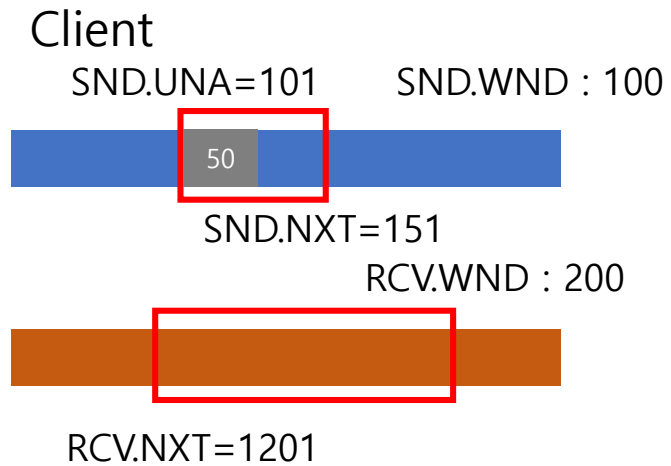
## Sliding window



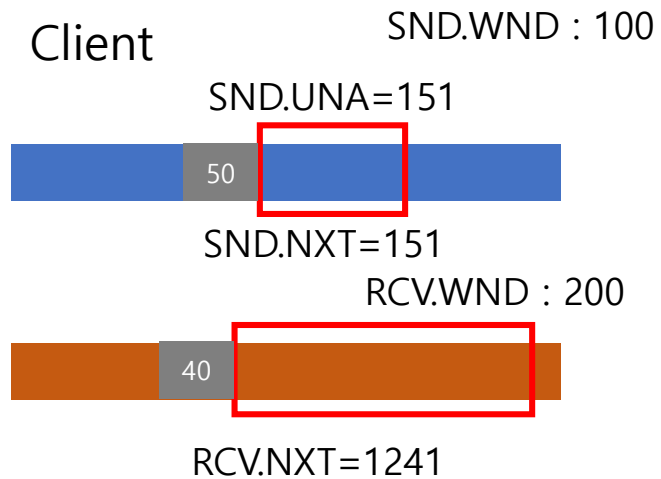
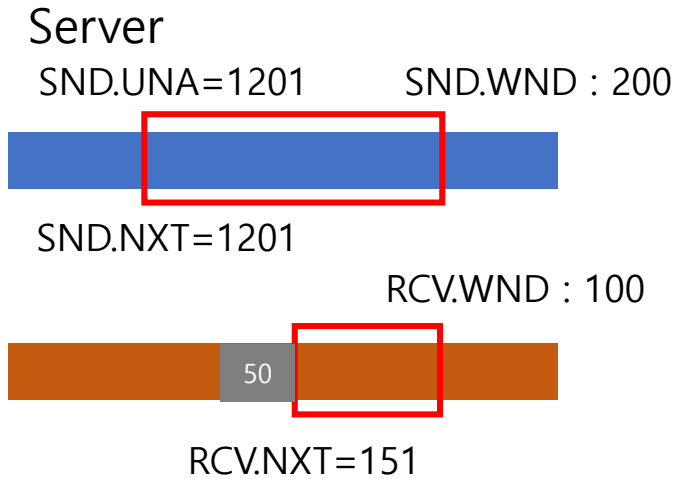
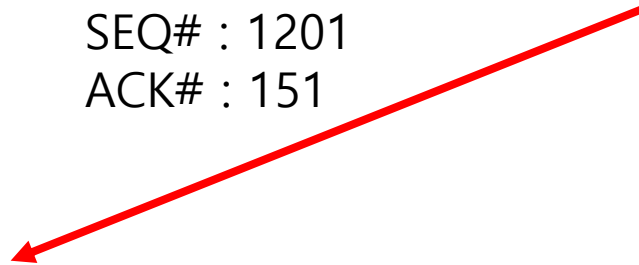
LEN : 50  
SEQ# : 101



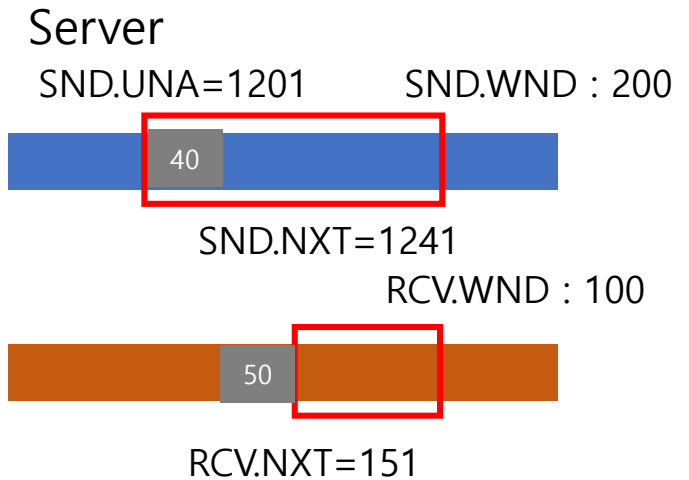
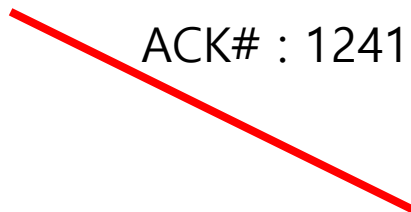
## Sliding window



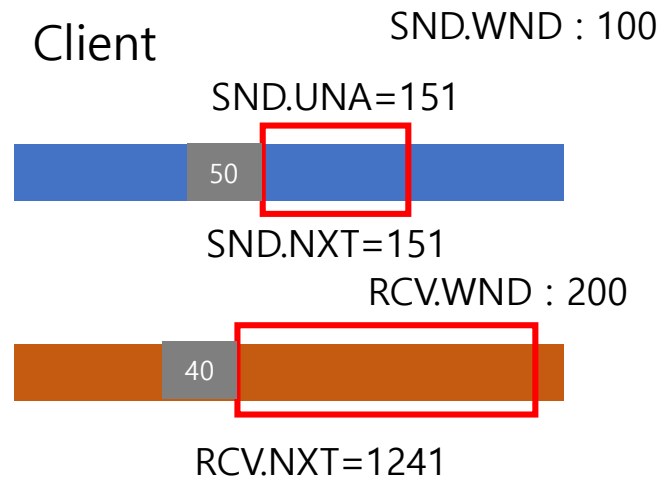
LEN : 40  
SEQ# : 1201  
ACK# : 151



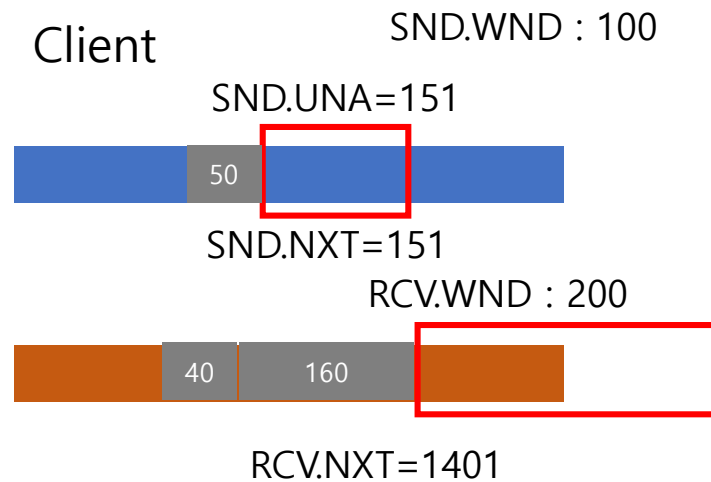
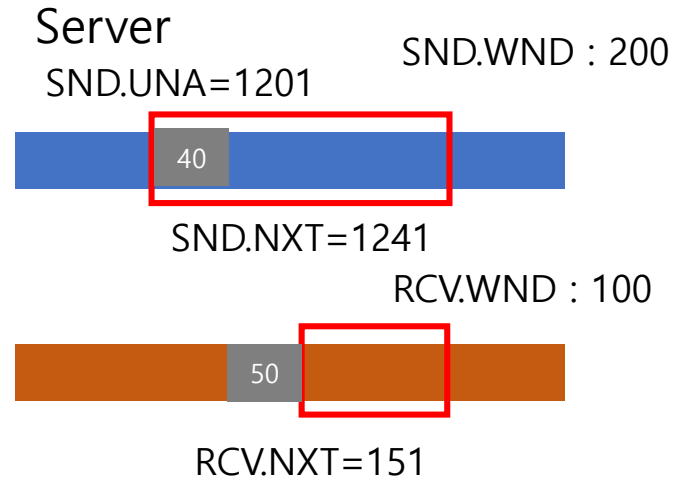
ACK# : 1241



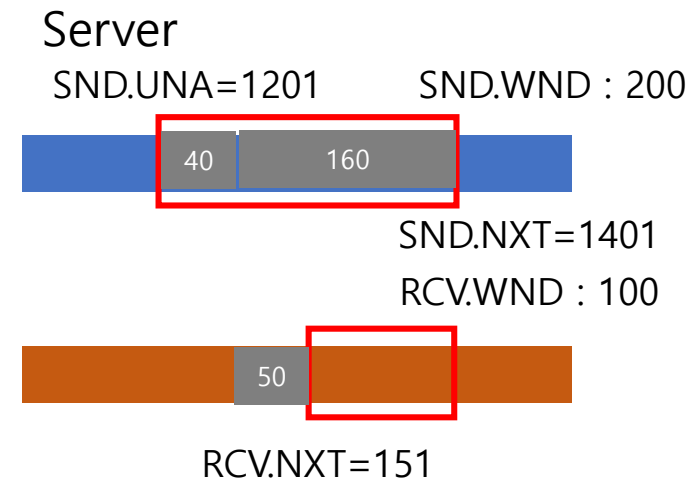
## Sliding window



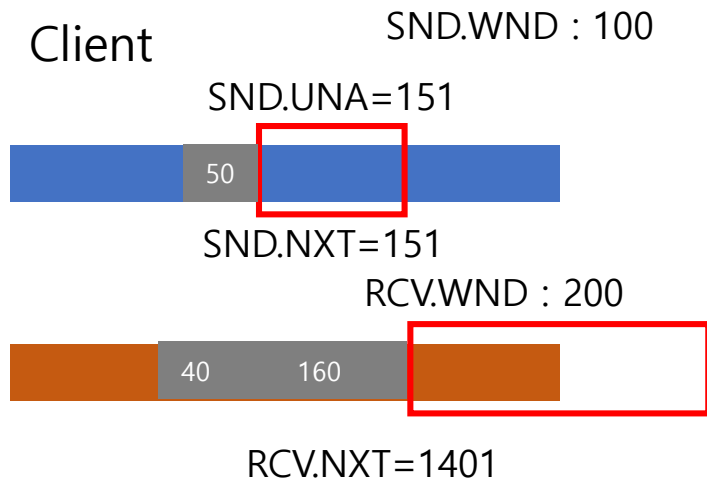
LEN : 160  
SEQ# : 1241



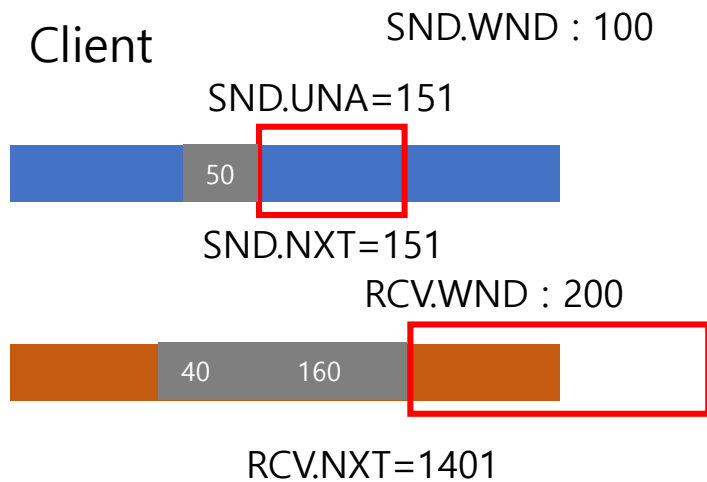
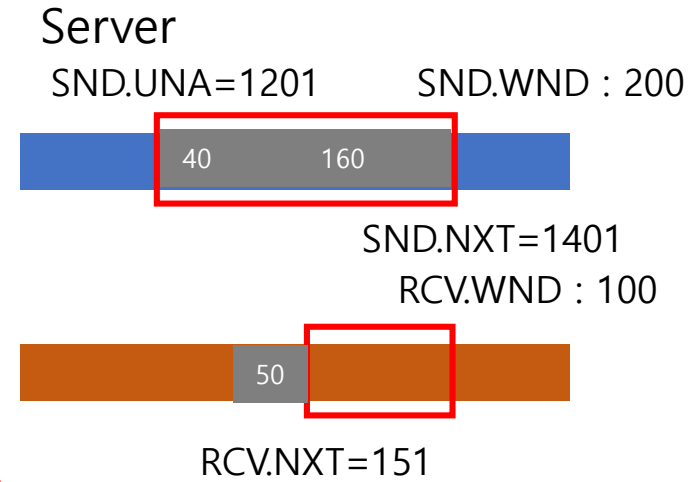
ACK# : 1241



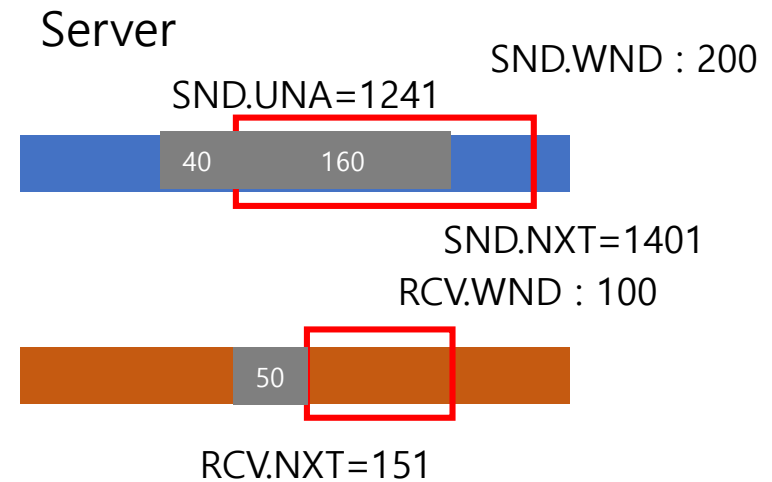
## Sliding window



ACK# : 1241

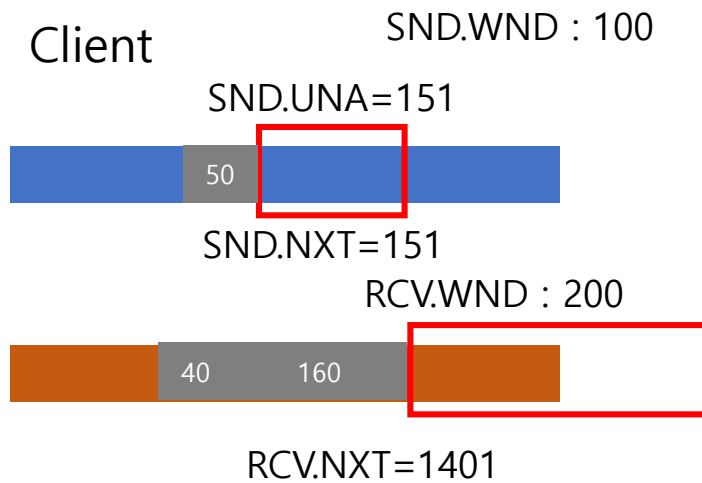


ACK# : 1401

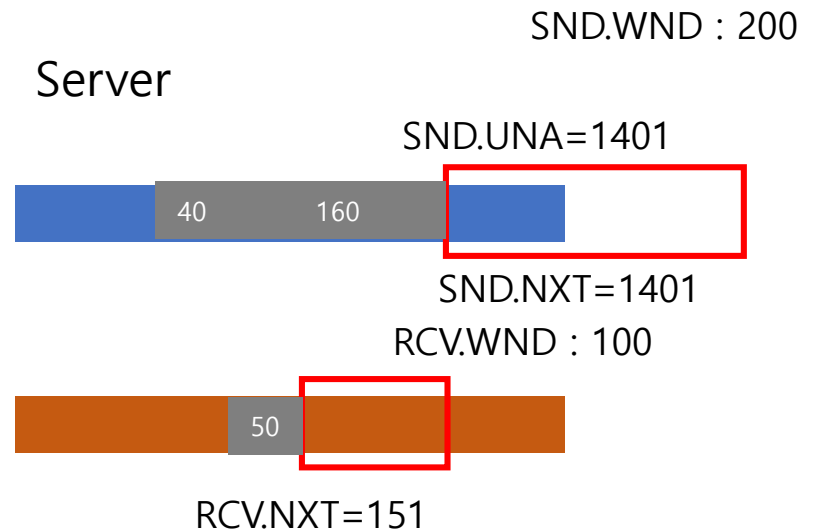
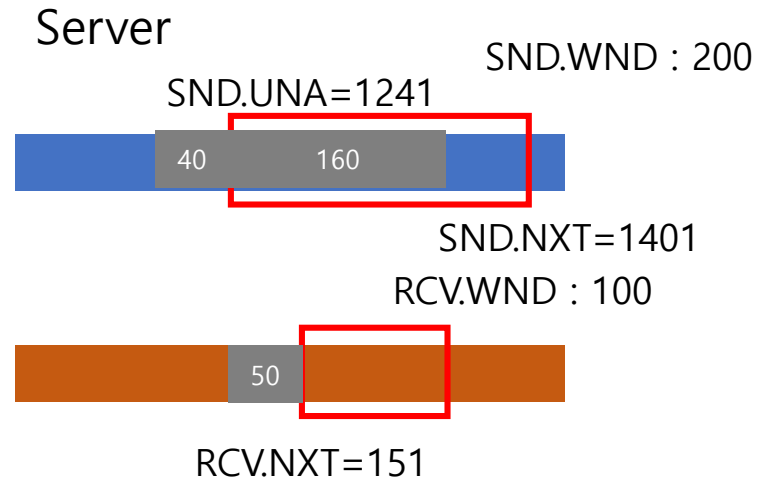
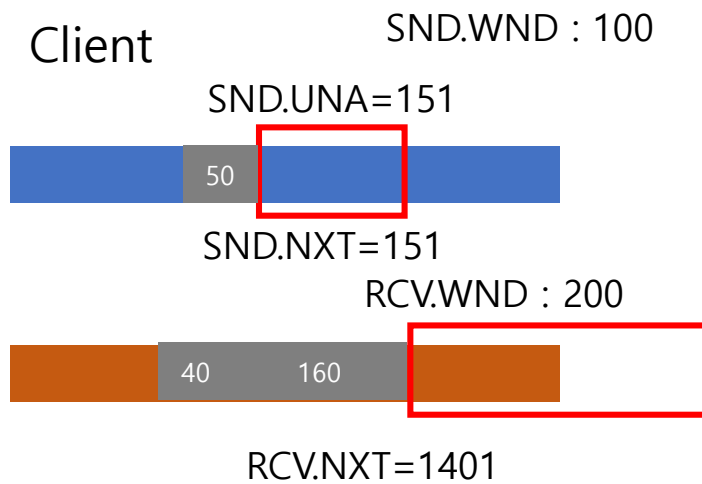




## Sliding window



ACK# : 1401



# TCP socket

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more i
nformation.
>>> import socket
>>> server=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> server.bind(('127.0.0.1', 3030))
>>> server.listen()
>>> data_sock, clnt=server.accept()
>>> data=data_sock.recv(1024)
>>> data
b'I am your fahter'
>>> data_sock.send('My name is John'.encode())
15
...
```

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more i
nformation.
>>> import socket
>>> clnt=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> clnt.connect(('127.0.0.1', 3030))
>>> clnt.send('I am your fahter'.encode())
16
>>> data=clnt.recv(1024)
>>> data
b'My name is John'
>>> |
```

UDP

## **UDP(User Datagram Protocol)**

1. 비 연결 지향형 프로토콜
2. 신뢰할 수 없다.
3. 데이터를 재전송하지 않는다.
4. 전송된 데이터 일부가 손실될 수 있다.

# UDP socket

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more i
nformation.
>>> import socket
>>> sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
>>> sock.bind(('127.0.0.1', 3030))
>>> data, clnt=sock.recvfrom(1024)
>>> data
b'I am your father!'
>>> clnt
('127.0.0.1', 62711)
>>> sock.sendto('My name is John'.encode(), clnt)
15
>>> sock.close()
>>> |
```

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more i
nformation.
>>> import socket
>>> clnt=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
>>> clnt.sendto('I am your father!'.encode(), ('127.0.0.1', 3
030))
17
>>> data, server=clnt.recvfrom(1024)
>>> data
b'My name is John'
>>> server
('127.0.0.1', 3030)
>>> clnt.close()
>>>
```

## UDP 헤더

- 1. Source Port : 2 bytes, 송신 Port**
- 2. Destination Port : 2 bytes, 수신 Port**
3. Length : 2 bytes, UDP 헤더와 데이터 필드를 모두 포함한 전체 패킷의 길이

## 5. 응용 계층

1. FTP : File Transfer Protocol(20, 21), 파일 전송을 위한 프로토콜
2. Telnet : 23번 포트, 사용자가 원격에 있는 서버에 로그인
3. SSH : 22번 포트, 텔넷과 유사하나 암호화를 통해 보안을 강화했다.
4. SMTP : Simple Mail Transfer Protocol(25) : 메일 서비스
5. DNS : Domain Name System(53) : 도메인 이름을 IP 주소로 변환하는 프로토콜
6. HTTP : HyperText Transfer Protocol(80) : 웹에서 데이터를 주고 받을 수 있는 프로토콜
7. HTTPS : HTTP over Secure Socket Layer(443), 텍스트를 SSL/TLS로 암호화해 보안을 강화했다.

# HTTP

## URL

`http://<user>:<password>@<host>:<port>/<urlpath>?<query>#<bookmark>`

잘 쓰이지 않음

기본으로 80을 쓰므로  
일반적으로 생략

HTTP

## **HTTP/1.0**

- 일시적 연결

## **HTTP/1.1**

- 지속적 연결(Persistent Connection)
- 효율적인 캐싱과 프록싱
- 콘텐츠 협상



# HTTP

1. 일시적 연결 (HTTP/1.0)  
: TCP 연결 후 하나의 요청/응답 후 연결을 끊는다
2. 지속적 연결 (HTTP/1.1)  
: TCP 연결을 그대로 유지한다.  
Connection: Close 헤더를 포함해 일시적 연결을 할 수 있다.

# HTTP

일반 메시지 형식(generic message format)

<시작줄(start-line)>

<메시지 헤더> : Host 헤더는 HTTP/1.1 필수

<빈 줄>

[<메시지 본문>]

[<메시지 트레일러>] : chunking에서 메시지 본문 뒤에 위치

## HTTP 요청

```
GET /index.html HTTP/1.1  
Date: Sun, 25 March 2019 00:15:45 GMT  
Host: www.csbootcamp.com  
Accept: text/html, text/plain  
<CRLF>
```

요청 줄

헤더

빈 줄

메시지 본문

## HTTP 응답

```
HTTP/1.1 200 OK
Date: Sun, 25 March 2019 00:15:45 GMT
Connection: close
Content-type: text/html
<CRLF>
<html>
<head> </head>
<body> </body>
</html>
```

상태 줄

헤더

빈 줄

메시지 본문

## 메서드

1. GET
  - URL이 지정하는 자원을 찾아 클라이언트에 전송
2. HEAD
  - GET과 같으나 메시지 본문이 없다. 테스트, 확인 용도
3. POST
  - 클라이언트가 임의의 데이터를 서버로 보낸다.
  - 주로 form 형식을 사용하며 서버의 프로그램에 전달

## 메서드

1. PUT
  - 지정한 URL에 요청의 본문 내용을 저장
  - 서버로 파일을 복사할 수 있게 한다 → 잘 쓰이지 않는 이유
2. DELETE
  - 지정한 자원을 지우도록 요청
3. TRACE
  - 클라이언트가 서버에 보낸 요청의 복사본을 돌려받는다

## 상태 코드

1. 1XX – 정보 제공 메시지  
: 일반적인 정보를 제공
2. 2XX – 성공  
: 서버가 메소드를 받아 수행했다
3. 3XX – 리다이렉션  
: 자원이 여러 종류가 있거나 새로운 URL로 이동하는 등  
추가 행동이 필요
4. 4XX – 클라이언트 에러  
: 요청이 잘못되었거나 자원을 찾을 수 없다
5. 5XX – 서버 에러  
: 요청은 유효하나 수행 방법을 모르거나 서버 문제로 처리 불가

## 컨텐츠 협상

### 1. 서버 주도(Server-driven) 협상

: 최선 추측(best guess) → 항상 클라이언트가 원하는 형식 데이터를 받는 것은 아님.

### 2. 에이전트 주도(Agent-driven) 협상

: 클라이언트가 자원을 고를 수 있다.

자료에 접근하는데 두번의 요청과 응답이 필요 → 효율성이 떨어짐

Accept(매체 유형)

Accept-Charset(문자 집합)

Accept-Encoding(컨텐츠 인코딩)

Accept-Language(언어)

Accept-Language: en, sp



## 컨텐츠 협상

**Accept : text/html, text/\*;q=0.5, \*/\*;q=0.2**  
default : 1

**Accept-Language : kr;q=0.7, fr;q=0, en;q=0.4**

청킹(chunking)

이미 본문 내용을 알고 있을 때  
Content-Length 헤더를 이용

**HTTP/1.1 200 OK**

**Content-Type: text/html**

**Content-Length: 114**

**Expires: Mon, 26 Mar 2019 00:30:40 GMT**

**<html><head></head><body>This is generated by Django app. So this length of this  
Content is dynamic.</body></html>**

청킹(chunking)

본문 내용이 동적으로 생성될 때  
Transfer-Encoding: chunked

**HTTP/1.1 200 OK**  
**Content-Type: text/html**  
**Transfer-Encoding: chunked**  
**Trailer: Expires**

**20**  
**<html><head></head><body>This is**  
**26**  
**generated by Django app. So this length**  
**2A**  
**of this Content is dynamic.</body></html>**  
**0**  
**Expires: Mon, 26 Mar 2019 00:30:40 GMT**