Application layer 1

Pinciples of network applications Web and HTTP

어플리케이션 계층

계층	PDU	역할	장비 및 프로토콜
[L7] Appliation Layer	Data	어플리케이션 서비스 수행	FTP, HTTP(S), SMTP
[L6] Presentation (표현) Layer	Data	표현 방식이 다른 Application 간 데이터 형식 동기화 인코딩, 압축, 암호화 등 변환 과정	TLS, SSH
[L5] Session Layer	Data	End-To-End application의 연결 성립 (안정성 유지, 에러 복구, 재전송) TCP/IP 세션 관리(생성, 유지, 해제)	SSH
[L4] Transport (전송) Layer	Segments	데이터가 정상적(패킷의 유실, 순서 유지 등)으로 전송 되도록 제어 (ACK Number, Seq. Number). Application의 Port 구분	[L4]LB(로드 밸런서), 방 화벽 / TCP, UDP, QUIC
[L3] Network Layer	Packets	IP Address 기반 라우팅 (Routing) 최적의 네트워크 경로를 찾아서 Packet을 전송함	라우터 / ARP, IPv4, IPv6, NAT
[L2] Data Link Layer	Frames	0, 1(Bits)의 전기 신호를 Frame 단위로 변환 / 전송 (Flow Control) MAC Address 기반으로 전송할 포트(Switch의) 지정 (MAC Table)	NIC(Network Interface Controller), [L2]스위치
[L1] Physical (물리) Layer	Bits	물리적 연결과 관련 전기 신호(Bits)를 정확히(유실 없이) 전달하는 역할	허브, 리피터, 케이블 (커 넥터) 등

💡 OSI 7 Layer

Application Layer : 7 계층

어플리케이션을

어플리케이션 아키텍처

가장 많이 사용되는 어플리케이션 구조

- 클라이언트-서버 (**client-server architecture)**
- P2P (peer-to-peer)구조

클라이언트-서버 구조(client - server architecture)

💡 가장 널리 알려진 구조

서버

- 항상 켜져있는 호스트를 서버라고 칭함
- 고정 IP 주소
- 데이터 센터에 존재함

클라이언트

- 서버와 통신
- 필요할때 연결
- 동적 IP주소를 가질 수 있다.
- 클라이언트간 직접 통신을 하지 않는다.(서버를 거침)

Peer-to-peer architecture(P2P 구조)

- 서버가 항상 켜져있지는 않는다.
- peer라는 호스트 쌍으로 클라이언트간 직접 통신(peer to peer)
- p2p 구조
  - 파일을 내려받는 피어 : 클라이언트
  - 파일을 올리는 피어 서버

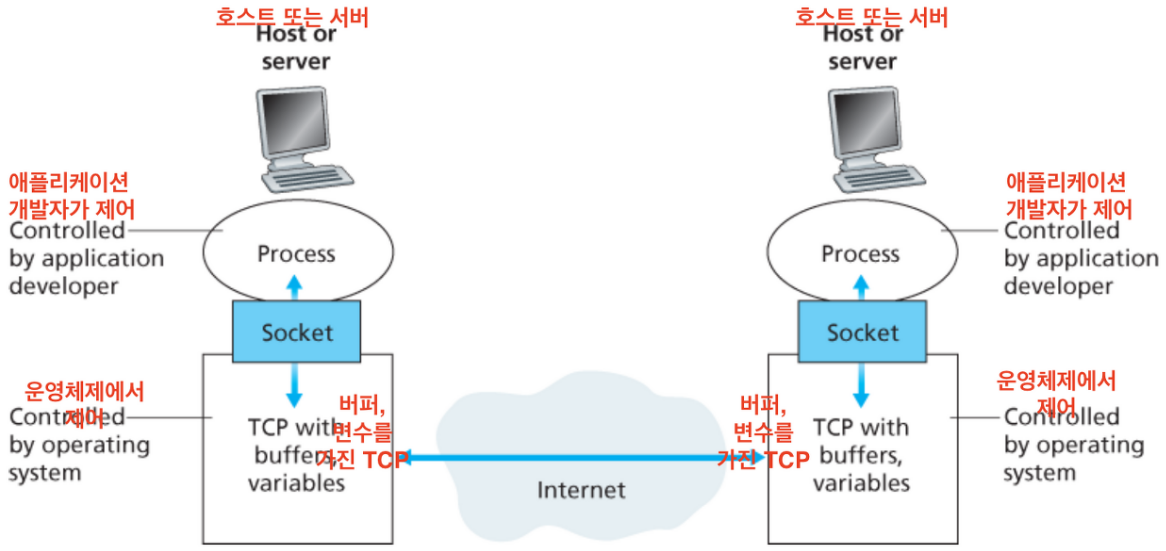
  ⇒ 토렌트 생각해보면됨
- self scalability : 스스로 확장할 수 있음
  - 작업을 다른 피어들에게 분배하여 비용에 효율적
- ip주소가 바뀔 수 있음
- ex) 토렌트, 스카이프

프로세스 통신

프로세스 간 통신은 두 개의 프로세스가 메시지를 서로에게 보내는 통신 프로세스 쌍을 이룬다.

- 같은 호스트 안에서 inter-process를 사용해 통신
- 다른 호스트 안에서는 메시지를 주고받음
- 소켓을 통해 네트워크로 메시지를 주고 받는다.

소켓



- 프로세스는 소켓을 통해 메시지 주고받음
- 소켓은 호스트의 어플리케이션 계층과 트랜스포트 계층 간의 인터페이스
- 어플리케이션과 네트워크 사이의 API

Addressing process

- 메시지를 주소받으려면 프로세스는 고유한 주소를 가져야함
- 32-bit ip addresss
- identifier = ip address + port numbers

어플리케이션이 이용할 수 있는 트랜스포트 서비스

프로토콜이 제공하는 서비스

Data Integrity - 신뢰적 데이터 전송

- 데이터 전송시 신뢰성 보장필요
- 프로세스 간 데이터 전송 (데이터 오류 없이 전달)

timing - 시간

- 지연 없이 효과적인 동작필요

throughput - 처리율

- 효율적인 최소 처리량 요구

security - 보안

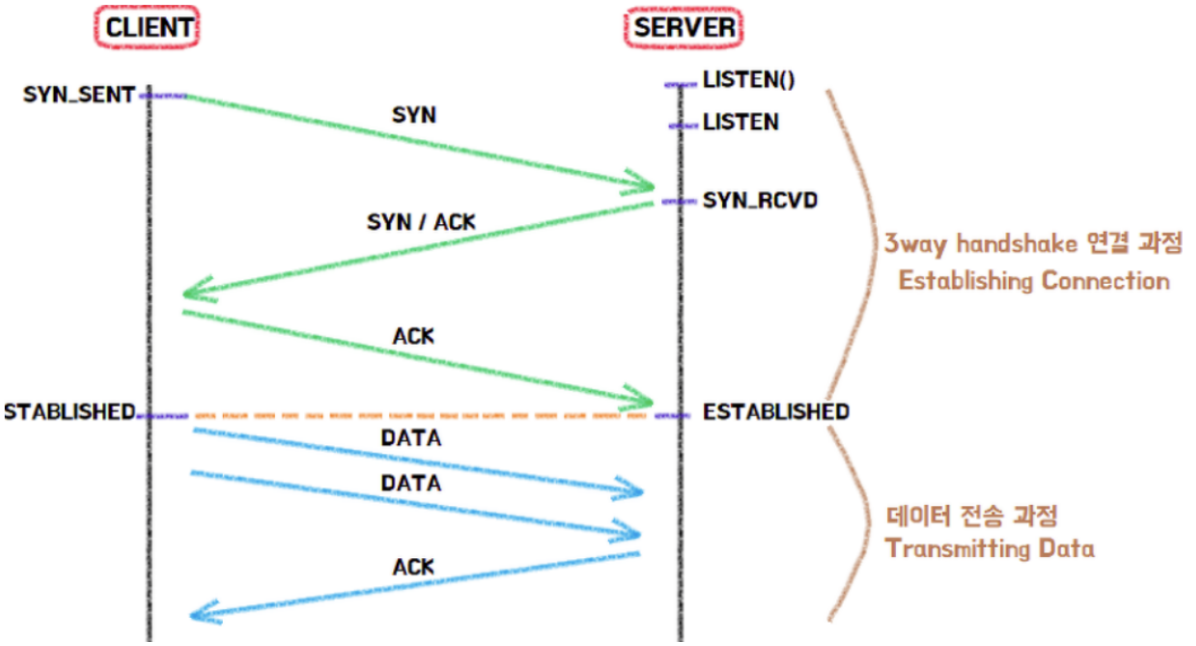
인터넷 프로토콜 종류

TCP

데이터를 메세지 형태로 보내기 위해 IP와 함께 사용하는 프로토콜

- 신뢰적 데이터 전송 서비스
- 흐름 제어 - 천천히 보냄
- congestion control (혼잡 제어 방식) - 처리할 수 있을 정도로만
- 연결 지향적 프로토콜
    - 어플리케이션 계층 메시지를 전송하기 전에 tcp는 클라이언트와 서버가 서로 전송 제어 정보를 교환하도록 한다. (3-way handshake)
    - 핸드 셰이킹 과정이 클라이언트와 서버에 패킷이 도착할 것이니 준비하라고 알려주는 역할
    - 핸드 셰이킹 단계 이후 tcp연결이 두 프로세스의 소켓 사이에 존재한다고 얘기함
    - 어플리케이션이 메시지 전송을 마치면 연결을 끊음 (4-way handshake)

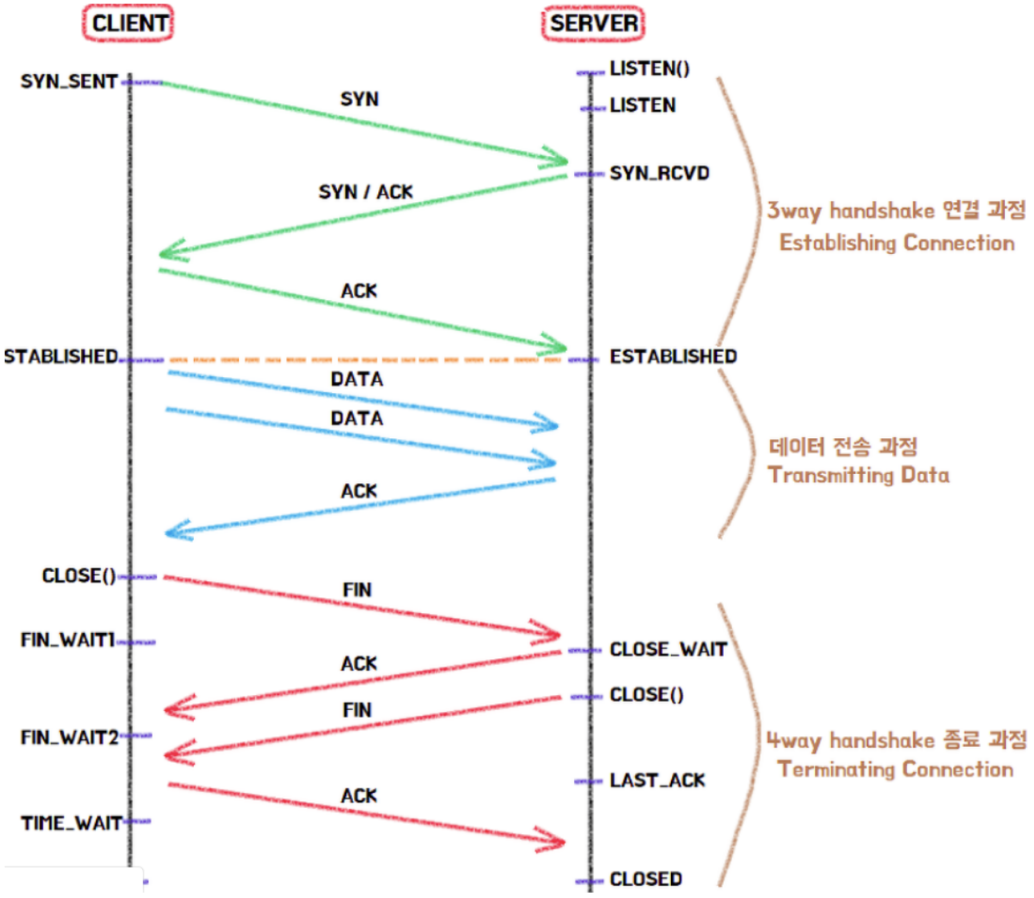
3-way handshake



클라이언트는 서버로 통신을 시작하겠다는 SYN을 보내고,
서버는 그에 대한 응답으로 SYN+ACK를 보낸다.
마지막으로 클라이언트는 서버로부터 받은 패킷에 대한 응답으로 ACK를 보낸다.
이렇게 3-way-handshake를 정상적으로 마친 다음 클라이언트는 서버에 데이터를 요청한다.

3-Way Handshake를 하는 이유는 클라이언트, 서버 모두 데이터를 전송할 준비가 되었다는 것을 보장하기 위한 것이다.

4-way handshake



클라이언트는 응답을 주고 연결을 끊기 위해 FIN패킷을 보낸다.
서버는 클라이언트에서 보낸 패킷에 대한 응답으로 ACK 패킷을 보낸다.
서버는 자신이 사용한 소켓을 정리하며 통신을 완전히 끝내도 된다는 의미로 FIN 패킷을 보낸다.
클라이언트는 서버 패킷에 대한 응답으로 ACK패킷을 보낸다.

4-way handshake과정을 거치고 나면 클라이언트와 서버단의 네트워크는 종료가 된다.

UDP

데이터를 데이터그램 단위로 처리하는 프로토콜

- 비신뢰적 데이터 전송 - 데이터 전달 순서, 도착 여부보장하지 않음
- 혼잡제어 방식 없어 속도 저하 없음
- 비연결형 프로토콜
    - tcp의 3-way handshaking 과정없음

💡 서비스를 어떤 프로토콜를 사용할 지 고민

Web and HTTP

web : 웹 어플리케이션 계층 프로토콜의 중심

http : 메시지 구조 및 클라이언트와 서버가 메시지 어떻게 교환하는지 정의

HTTP

hypertext transfer protocol

- 클라이언트

웹 페이지

URL

💡 www.someschool.edu/someDept/pic.gif

host : www.someschool.edu
path : someDept/pic.gif

HTTP와 TCP

http는 tcp를 전송 프로토콜로 사용

- http 클라이언트는 서버에 tcp 연결

- 연결 후 브라우저와 서버 프로세스는 각각 소켓 인터페이스를 통해 TCP 접속
- 클라이언트는 request를 소켓 인터페이스로 보내고, 소켓 인터페이스로부터 response를 받는다.

  마찬가지로, HTTP 서버는 소켓 인터페이스로부터 요청 메시지를 받고 응답 메시지를 소켓 인터페이스로 보낸다.

stateless 프로토콜

비지속/지속 연결

non-persistent 연결

연결수행 과정

- http 클라이언트는 기본 포트 80을 통해 서버로 tcp연결을 시도한다.
- http 클라이언트는 설정된 tcp 연결 소켓을 통해 서버로 http 요청 메시지를 보냄
- http 서버는 tcp 연결 소켓을 통해 요청 메시지를 받는다.
- http서버는 tcp에게 연결 끊으라고 함
- http 클라이언트가 응답 메시지를 받으면 tcp 연결 중단

persistent 연결

HTTP 1.1 지속 연결에서 서버는 응답을 보낸 후에 tcp 연결을 그대로 유지

같은 서버에 있는 여러 웹페이지들을 하나의 지속 tcp연결을 통해 보낼 수 있음.

일반적으로 http서버는 일정 기간 사용되지 않으면 연결을 닫음

http의 디폴트 모드는 파이프라이닝을 이용한 지속 연결을 사용한다.

HTTP 메시지 포맷

http 요청 메시지

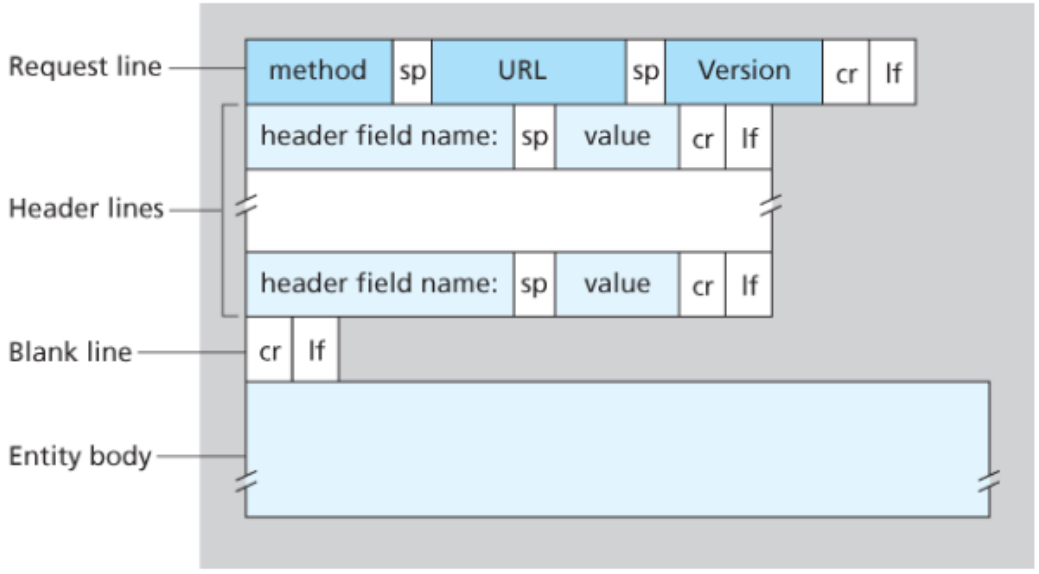


Figure 2.8 General format of an HTTP request message

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

특징

- ASCII 텍스트로 쓰여 있어 사람들이 읽을 수 있음
- 메시지가 5개의 줄로 구성되어있음
    - CR과 LF로 구별

HTTP 요청 메시지의 첫 줄은 요청라인,

이후 줄은 헤더 라인

요청 라인

- 방식필드 - GET, POST, HEAD, PUT, DELETE
- URL 필드
- HTTP 버전 필드

헤더 라인

1. host
    a. 객체 존재하는 호스트 명시
2. connection : 지속 연결 사용인지, 비지속 연결 사용인지
3. user-agent : 서버에게 요청하는 브라우저 타입
4. accept-language : 헤더는 사용자가 객체 어떤 언어 버전을 원하고 있는지를 나타냄

Entity body (개체 몸체)

- Get 일 때는 비어있고, Post 일 때 사용

HTTP응답 메시지

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)
```

상태 라인과 상태 코드

- status code
    - 200 : OK / 요청 성공, 정보가 응답으로 잘 보내짐

- 301 : Moved Permanently / 요청 객체 영원히 이동됨
- 400 : Bad Request / 서버가 요청을 이해할 수 없음
- 404 : Not Found / 서버에 요청한 것이 존재하지 않음
- 505 : HTTP Version Not Supported : 요청 HTTP 프로토콜 버전을 서버가 지원하지 않음

쿠키

statelsee http

> 💡 서버가 사용자 접속을 제한하거나 사용자에 따라 콘텐츠를 제공하기 원하므로 사용자를 확인하는 것이 바람직할 때가 있는데 이때 쿠키 사용

쿠키 동작

- 웹 서버에 http 요청 메시지 전달
- 웹 서버는 유일한 식별번호 만들고
- http 응답 메시지에 “set-cookie : 식별번호”의 헤더를 포함해서 전달
- 브라우저는 헤더를 보고, 관리하는 특정한 쿠키 파일에 그 라인을 덧붙임

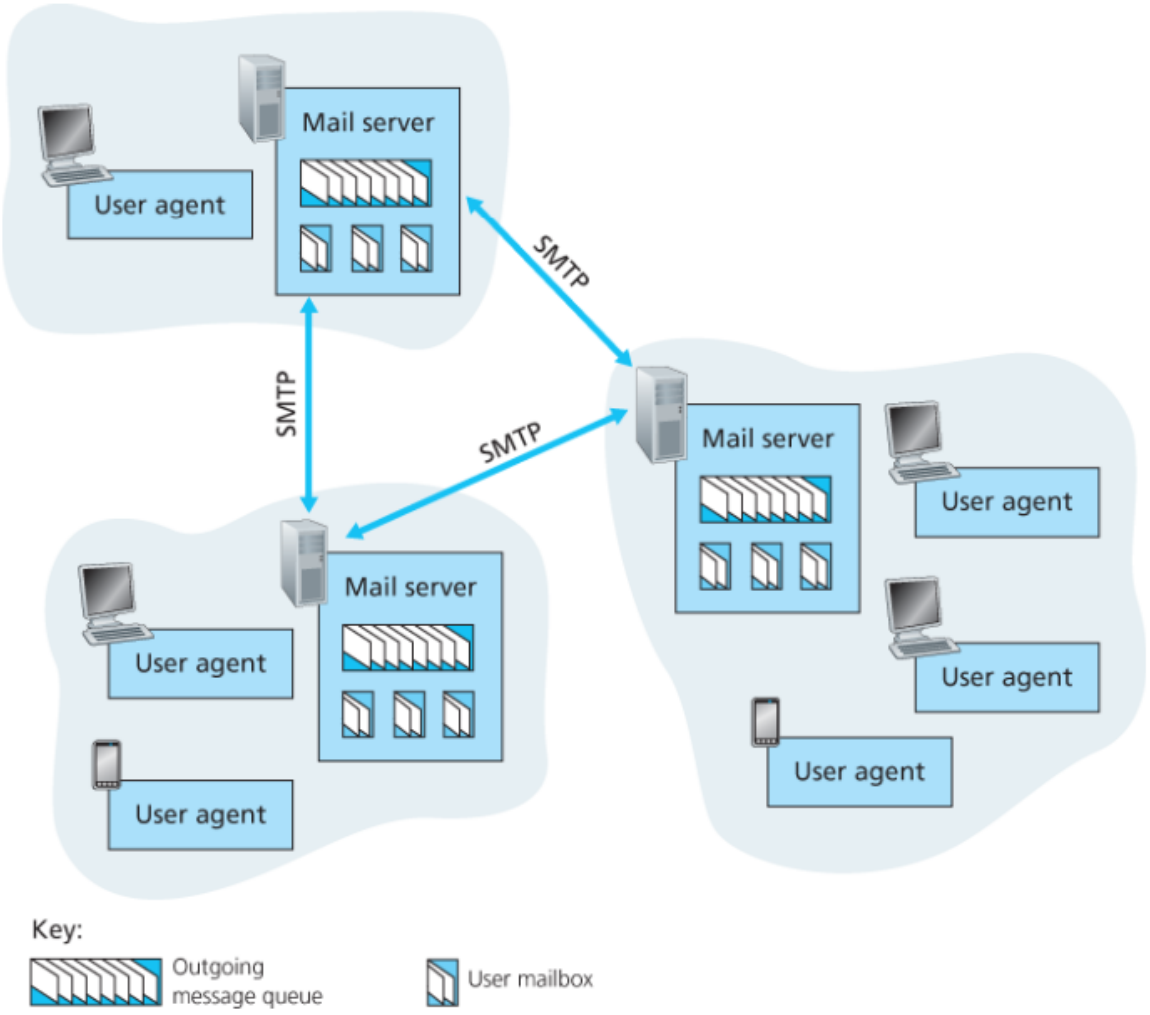
웹 캐싱

> 💡 웹 캐시는 자체의 저장 디스크를 갖고 있어, 최근 호출된 객체의 사본을 저장 및 보존

프록시 동작 과정

- 브라우저는 웹 캐시와 tcp연결을 설정하고 웹 캐시에 있는 객체에 대한 http요청을 보낸다.
- 웹 캐시는 객체의 사본이 저장되어있는지 확인 후 저장되어있다면 클라이언트 브라우저로 http응답메시지와 함께 객체를 전송
- 갖고 있지 않으면 tcp연결 설정

전자 메일



메일 서버

SMTP

- SMTP는 메일을 송신자의 메일 서버로부터 수신자의 메일 서버로 전송하는 데에 tcp의 신뢰적인 데이터 전송 서비스를 이용한다.
- SMTP는 대부분의 애플리케이션 계층 프로토콜처럼, 클라이언트와 서버를 갖고 있다.
- SMTP의 클라이언트와 서버 모두가 모든 메일 서버에서 수행되고, 상대 메일로 송신할 때는 클라이언트가 되고 수신할 때는 서버가 된다.

SMTP

전송 용량 제한이 되어 커다란 첨부 파일이나 비디오 파일을 보낼 때 문제를 일으킴

- SMTP는 메일을 송신자의 메일 서버로부터 수신자의 메일 서버로 전송→ `TCP의 신뢰적인 데이터 전송 서비스` 이용
- Client와 Server를 갖고 있다
  - Client와 Server 모두가 모든 메일 서버에서 수행
  - 상대 메일로 송신할 때는 client가 되고 수신할 때는 서버가 됨
- client → server로 이메일 보낼 때 : TCP 이용, port 25
- direct transfer : 보내는 서버 → 받는 서버
- SMTP에서는 모든 메일 메시지의 몸체는 단순한 `7-bit ASCII` → 전송 용량이 제한되어 커다란 첨부 파일이나 비디오 파일을 보낼 때 문제
- 전송의 3과정
  1. handshaking (greeting)
  2. transfer of messages (data exchange)
  3. closure