엔지니어링 BASE
2주차 CS 기초

2022.07.27.

18기 엔지니어링 윤정원

Computer Science

- 자료구조
- 알고리즘
- 운영체제 ①
- 네트워크 ②
- 데이터베이스

운영체제

하드웨어를 관리하고, 응용 프로그램과 하드웨어 사이에서 인터페이스 역할을 하며,
시스템의 동작을 제어하는 **시스템 소프트웨어**

즉, 시스템의 자원과 동작을 관리하는 소프트웨어

1. 운영체제

프로그램 파일 단위로 저장 장치에 저장되어 있으며, 아직 실행되지 않은 코드 상태

프로세스 운영체제로부터 메모리 공간을 할당 받아 실행중인 프로그램

스레드 프로세스를 구성하는 독립적인 실행 단위

1. 운영체제

스레드

: 프로세스에서 실행 제어만 분리한 실행 단위

같은 프로세스 내 다른 스레드와 메모리 영역을 공유

각 스레드는 스택 영역을 통해 독립적인 실행 흐름을 가짐

문맥 교환으로 인한 자원 낭비를 최소화

멀티스레드

: 한 프로세스에서 여러 개의 스레드를 동시에 수행하는 것

2. 네트워크 - 네트워크란?

Net + Work

두 개 이상의 노드가 서로 데이터를 공유할 수 있게 하는 통신 환경

Internet : 여러 데이터를 공유할 수 있도록 전세계를 연결한 네트워크 통신망
그 중 웹 서비스 → www

2. 네트워크 - 크기에 따른 분류

LAN
Local Area Network

근거리 네트워크

사무실과 같은 소규모 공간 내의 고속통신회선

WAN보다 빠른 통신 속도

보통 Peer-to-Peer 형태

WAN
Wide Area Network

광대역 네트워크 망

LAN과 LAN을 다시 하나로 묶은 것

etc.

MAN
VLAN
CAN
PAN
.
.
.

2. 네트워크 – 연결 형태에 따른 분류

Star형

장점

- 고속 네트워크에 적합
- 노드 추가가 쉬움
- 개별 링크 장애 시에도 네트워크 영향이 없음

단점

- 중앙 노드 장애 시 전체 네트워크 불통
- 노드 증가에 따라 네트워크 복잡도 증가

Mesh형

장점

- 완벽하게 이중화 되어있으므로 장애에 강함
- 많은 양의 데이터 처리에도 문제 없음

단점

- 구축 비용이 큼

etc.

Tree

Bus

Ring

2. 네트워크 - OSI 7 계층

국제 표준화기구(ISO) 에서 개발한 모델로서,
네트워크 프로토콜 디자인과 통신을 계층으로 나눠 설명한 것.

[1] 물리 계층
- 전기 신호를 이용하여 비트 스트림을 전송하는 계층
- 데이터의 종류나 에러가 있는 지 등은 확인 X

[2] 데이터 링크 계층
- 물리 계층을 이용하여 네트워크 상의 주변 장치들 간 데이터 전송을 하는 계층
- 통신의 오류 찾기, 재전송 등의 기능

[3] 네트워크 계층
- 라우터를 통해 패킷을 전달해주는 계층

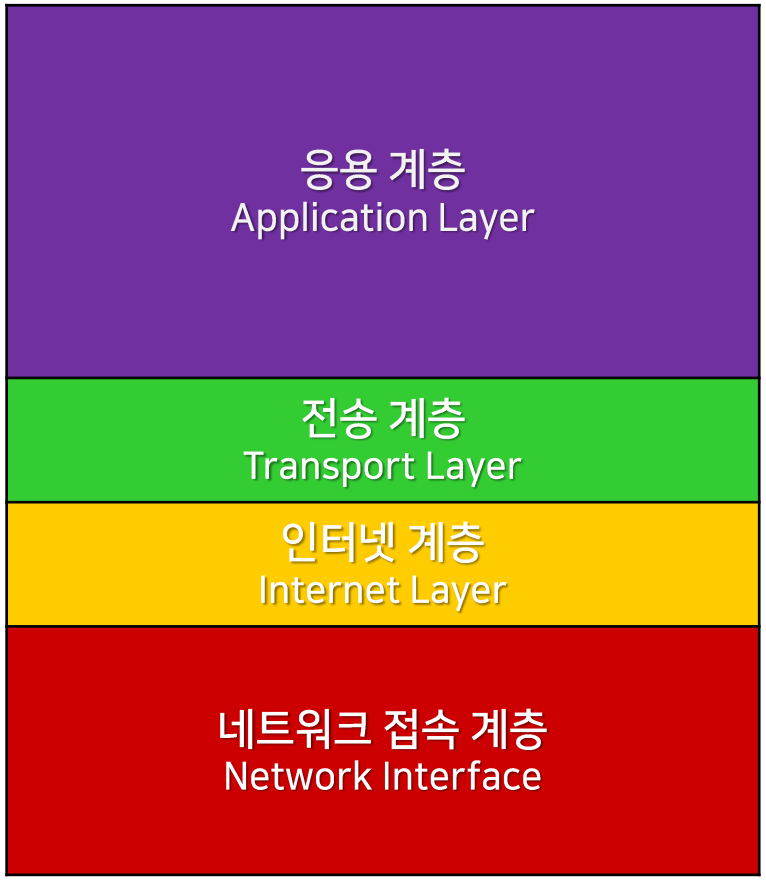
[4] 전송 계층
- 상위 계층의 메시지를 하위 계층으로 전송하는 계층

[5] 세션 계층
- 데이터 교환의 경계(네트워크 양쪽 연결)를 관리
- 동기화와 오류 복구를 수행하는 계층

[6] 표현 계층
- 암호화, 복호화와 같이 코드 간 번역을 담당하는 계층

[7] 응용 계층
- 통신의 최종 목적지로, 일반적인 응용 서비스를 수행하는 계층

2. 네트워크 – TCP/IP



TCP/IP 모델

[응용 계층]
- 응용 프로그램이 사용되는 계층
- 웹 서비스, 이메일 등 실질적 서비스 제공

[전송 계층]
- 송신자와 수신자를 연결하는 통신 서비스 제공
- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)

[인터넷 계층]
- 패킷을 IP 주소(지정된 목적지)로 전송
- 비연결형

[네트워크 접속 계층(링크 계층)]
- TCP/IP 패킷을 네트워크 매체로 전달하고 받아들이는 역할
- Mac 주소 사용

2. 네트워크 – TCP와 UDP

TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
연결형 프로토콜	비연결형 프로토콜
전송 순서를 보장	전송 순서가 바뀔 수 있음
신뢰성이 높다	신뢰성이 낮다
1 : 1(Unicast) 통신	1 : 1, 1 : N(Broadcast), N : N(Multicast) 통신
3-way handshake를 사용하여 신뢰성 있는 전송	TCP 보다 속도가 빠르고 네트워크 부하가 적어, 연속성 있는 전송

2. 네트워크 - HTTP

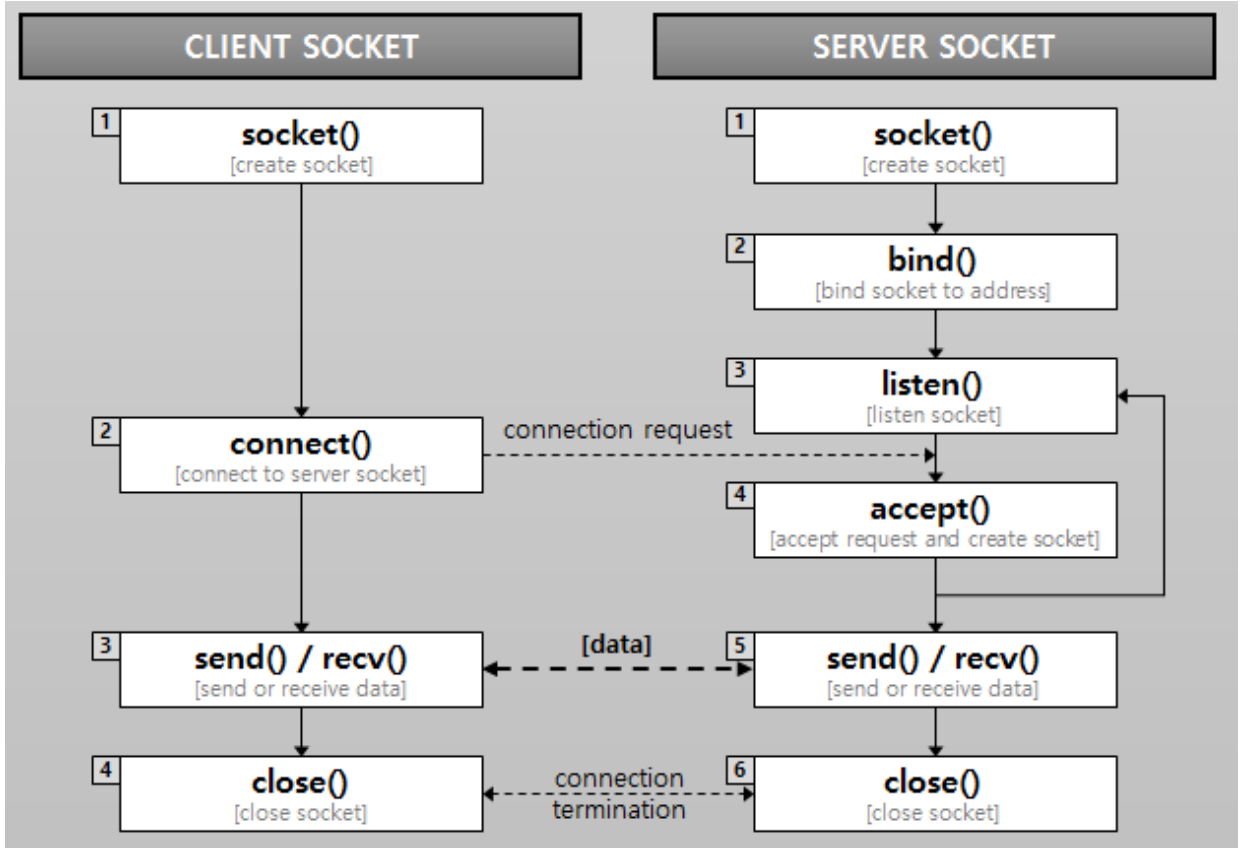
HTTP

Hyper Text Transfer Protocol

HTTP 통신
: 클라이언트의 요청이 있을 때만 서버가 응답하여
해당 정보를 전송하고, 곧바로 연결을 종료하는 방식

단방향 통신 / 비실시간 / 요청을 보낸 후 서버의 응답을 기다리는 어플리케이션 개발에 주로 사용

2. 네트워크 – Socket



→ 양방향 통신 / 실시간 데이터 처리

[Server]

- 클라이언트의 요청 발생 시 응답 전송
- 클라이언트와 통신을 위해 소켓 사용

[client]

- 소켓을 사용하여 서버와 연결
- 데이터 송수신

[Socket]

- TCP와 UDP 모두 지원
- 식별 주소 : IP + 포트번호
- 두 컴퓨터 연결 전, 반드시 양쪽 소켓 생성

[Port]

- 포트 번호는 0 ~ 2^16-1 사이 숫자
- 0 ~ 1023번 포트는 주요 프로토콜에 할당
- 특정 네트워크 서비스 또는 리스너를 식별하기 위해 사용

2. 네트워크 – Socket 통신 메소드

SOCK_STREAM : TCP 사용

SOCK_DGRAM : UDP 사용

socket() // 소켓 생성

.close() // 서버와 연결 끊기

- 서버 연결 메소드

.bind(ip, port) // 각 종단점을 소켓과 결합

.listen(n) // n개의 클라이언트 연결 대기

.accept() // 클라이언트 연결 수용

- 클라이언트 연결 메소드

.connect() // IP 주소와 Port 번호 이용해 서버에 연결 요청

.connect_ex() // 문제 발생 시 오류코드 반환

- TCP 데이터 송수신 메소드

.recv() // TCP 소켓을 통해 메시지 수신, 수신 데이터 반환

.recv_into() // TCP 소켓을 통해 메시지 수신하여 버퍼에 저장

.send() // TCP 소켓으로 메시지 전송, 송신 바이트 수 반환

.sendall() // TCP 소켓으로 메시지를 버퍼에 남기지 않고 모두 전송

- UDP 데이터 송수신 메소드

.recvrom() // UDP 소켓을 통해 메시지를 수신

.recvfrom_into() // UDP 소켓을 통해 메시지 수신하여 버퍼에 저장

.sendto() // UDP 소켓으로 메시지 송신

실습

간단한 TCP Socket Programming 실습 - Python

실습 결과

Client

```
connected
Client Message: Hello!
send a Message
The client received:  Hi Boaz
```

Server

```
The server is running
connected
The server received:  Hello!
Server Message: Hi Boaz
```

실습

간단한 TCP Socket Programming 실습 - Python

client 코드

```python
from socket import *

serverName = '127.0.0.1'  # IP address: localhost
serverPort = 13000  # port number: 13000

clientSocket = socket(AF_INET, SOCK_STREAM)

clientSocket.connect((serverName, serverPort))
print('connected')

sentence = input('Client Message: ')
clientSocket.send(sentence.encode())
print('send a Message')

modifiedSentence = clientSocket.recv(1024)
modifiedSentence = modifiedSentence.decode()
print('The client received: ', modifiedSentence)

clientSocket.close()

```

실습

간단한 TCP Socket Programming 실습 - Python

server 코드

```python
from socket import *

ip = '127.0.0.1'
serverPort = 13000  # port number: 12000

serverSocket = socket(AF_INET, SOCK_STREAM)

serverSocket.bind((ip, serverPort))
serverSocket.listen(1)
print('The server is running')

connectionSocket, addr = serverSocket.accept()
print('connected')

data = connectionSocket.recv(1024)
sentence = data.decode()
print('The server received: ', sentence)

sendMessage = input('Server Message: ')
connectionSocket.send(sendMessage.encode())

connectionSocket.close()
```

감사합니다!

참고자료

- https://realpython.com/python-sockets

→ 파이썬 소켓 통신 가이드

- https://hi0seon.tistory.com/entry/컴퓨터-네트워크-및-실습-소켓-프로그래밍
- https://youngq.tistory.com/71?category=776361
- https://github.com/sangwoo24/CS
- https://helloworld-88.tistory.com/215
- https://1d1cblog.tistory.com/69