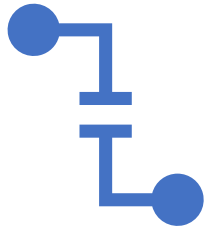
소켓 프로그래밍 With Python

이지훈

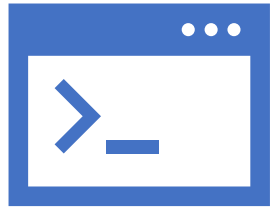
목차



What is Socket?

OSI 7 layer

Socket



Socket Programming

Server.py

Client.py

Next . . .

OSI 7 layer

응용 계층	HTTPS / DNS / FTP	응용 계층
표현 계층	ASCII / JPEG	
세션 계층	SSH / TLS	
전송 계층	TCP / UDP / ARP	
네트워크 계층	IP	
데이터 링크 계층	MAC / Ethernet	
물리 계층	광케이블	

OSI 7 layer

Socket

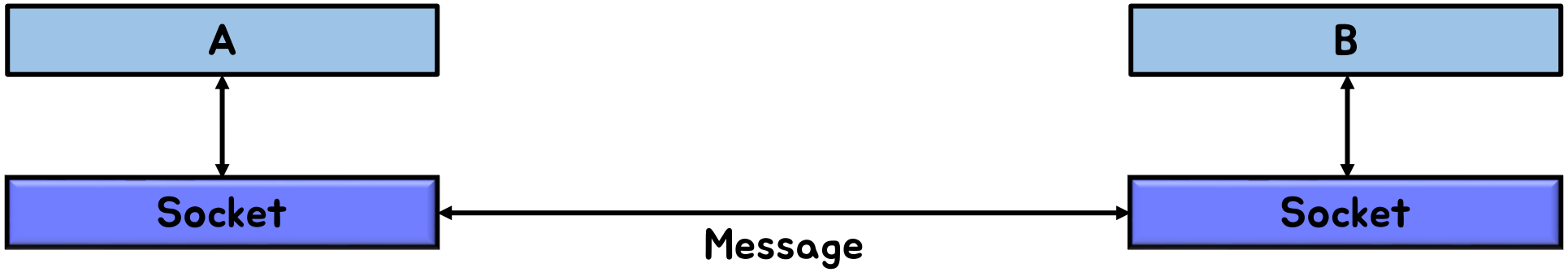
프로그램을 개발할 때 애플리케이션 계층(응용 계층)에 한정되어 있는 경우가 많음
애플리케이션 계층에서 트랜스포트 계층(전송 계층)을 조작할 수 있도록 도움을 주는 게 바로 소켓이다.

소켓은 OS에서 제공하는 인터페이스 -> 어떤 종류의 프로그램이더라도 소켓에 접근하여 통신 가능

소켓은 창구의 기능을 한다

Socket

: 소켓은 창구와 같은 기능을 한다



실제로 통신을 하고 있는 것은 소켓들이고,
A와 B는 자체는 외부 네트워크와 아무런 정보도 주고 받지 않는다.

Server.py

```python
from socket import *

server_sock = socket(AF_INET, SOCK_STREAM)
server_sock.bind(('', 8080))
server_sock.listen(1)

connect_sock, addr = server_sock.accept()

print(str(addr),'에서 접속이 확인되었습니다.')

data = connect_sock.recv(1024)
print('받은 데이터 : ', data.decode('utf-8'))

connect_sock.send('I am a server.'.encode('utf-8'))
print('메시지를 전송완료')
```

Server.py

```python
from socket import *

server_sock = socket(AF_INET, SOCK_STREAM)
server_sock.bind(('', 8080))
server_sock.listen(1)

connect_sock, addr = server_sock.accept()

print(str(addr),'에서 접속이 확인되었습니다.')

data = connect_sock.recv(1024)
print('받은 데이터 : ', data.decode('utf-8'))

connect_sock.send('I am a server.'.encode('utf-8'))
print('메시지를 전송완료')
```

Server.py

```
1  from socket import *
2
3  server_sock = socket(AF_INET, SOCK_STREAM)
4  server_sock.bind(('', 8080))
5  server_sock.listen(1)
6
7  connect_sock, addr = server_sock.accept()
8
9  print(str(addr),'에서 접속이 확인되었습니다.')
10
11 data = connect_sock.recv(1024)
12 print('받은 데이터 : ', data.decode('utf-8'))
13
14 connect_sock.send('I am a server.'.encode('utf-8'))
15 print('메시지를 전송완료')
```

Server.py

```python
from socket import *

server_sock = socket(AF_INET, SOCK_STREAM)
server_sock.bind(('', 8080))
server_sock.listen(1)

connect_sock, addr = server_sock.accept()

print(str(addr),'에서 접속이 확인되었습니다.')

data = connect_sock.recv(1024)
print('받은 데이터 : ', data.decode('utf-8'))

connect_sock.send('I am a server.'.encode('utf-8'))
print('메시지를 전송완료')
```

Server.py

```
1  from socket import *
2
3  server_sock = socket(AF_INET, SOCK_STREAM)
4  server_sock.bind(('', 8080))
5  server_sock.listen(1)
6
7  connect_sock, addr = server_sock.accept()
8
9  print(str(addr),'에서 접속이 확인되었습니다.')
10
11 data = connect_sock.recv(1024)
12 print('받은 데이터 : ', data.decode('utf-8'))
13
14 connect_sock.send('I am a server.'.encode('utf-8'))
15 print('메시지를 전송완료')
```

Client.py

```python
from socket import *

client_sock = socket(AF_INET, SOCK_STREAM)
client_sock.connect(('127.0.0.1', 8080))

print('연결 확인')
msg = input('보낼 메시지 입력:')

client_sock.send(msg.encode('utf-8'))

print('메시지 전송완료.')

data = client_sock.recv(1024)
print('받은 데이터 : ', data.decode('utf-8'))

```

실행화면

```
('127.0.0.1', 55852) 에서 접속이 확인되었습니다.
받은 데이터 :   SCP is Good
메시지를 전송완료
>>>
```

Server

```
연결 확인
보낼 메시지 입력:SCP is Good
메시지 전송완료.
받은 데이터 :  I am a server.
>>>
```

Client

Next . . .

쌍방향에서 채팅을 주고받듯이 통신을 주고받는 방식 . . .

127.0.0.1 에서 하는 방식이 아닌 다른 컴퓨터(VM)에서 각각 통신이 가능하도록 테스트

Q & A