

# 소켓프로그래밍

## 라즈베리파이로 배우는 소켓 통신 프로그래밍



동양미래대학교  
컴퓨터공학부 정석용



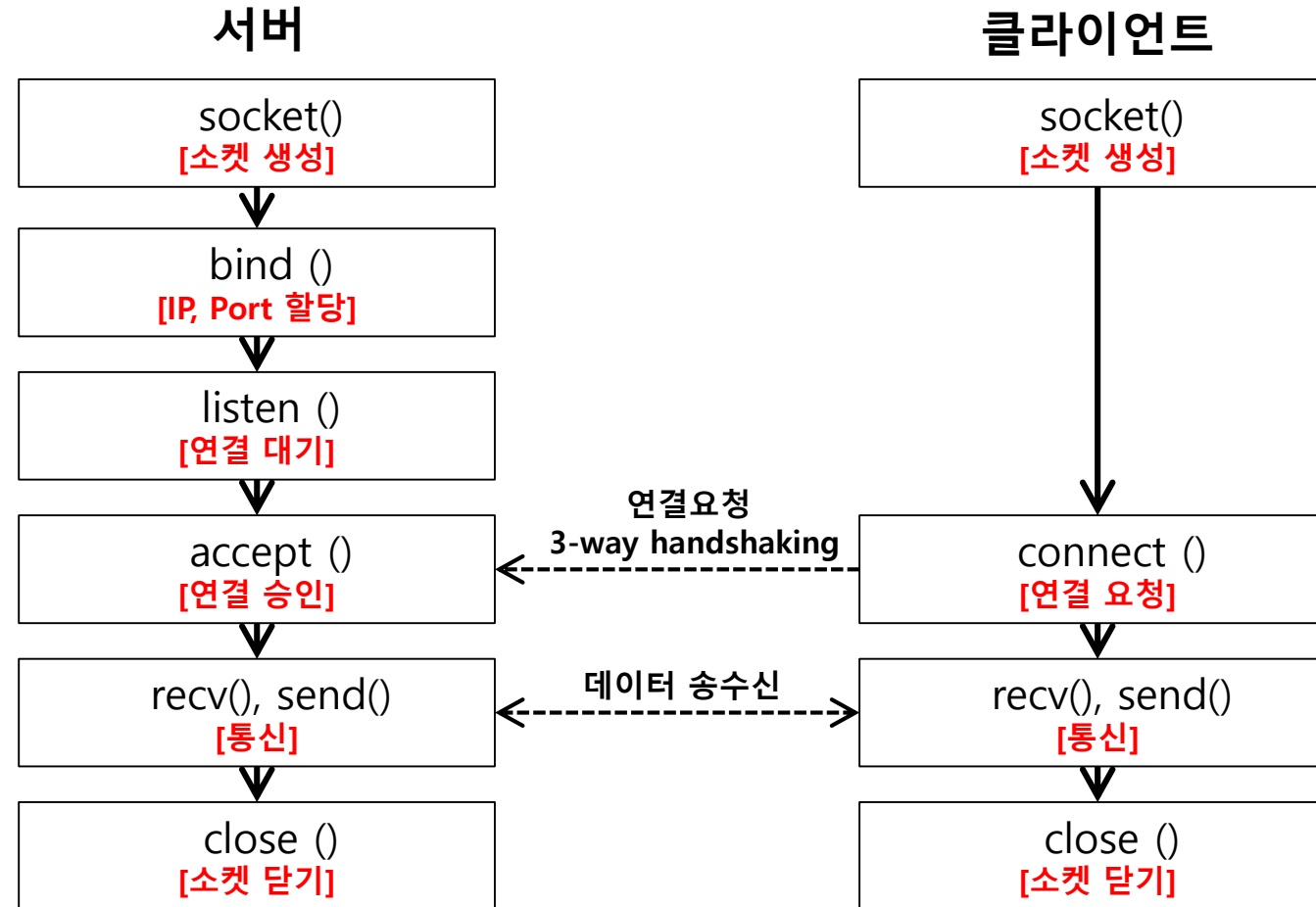
동양미래대학교

- 클라이언트 및 서버 소켓프로그래밍을 배운다.
- 서버 소켓 프로그램 멀티 프로세싱을 배운다.

## [실습과제 1]

클라이언트 및 서버 소켓프로그램 (1대 1 통신)

## TCP 소켓 프로그램



TCP 통신으로 1:1 통신 지원

## 파일명 : server.py

```
import socket

s = socket.socket() # 통신용 TCP 소켓 생성
host = '0.0.0.0'    # 서버가 연결될 server IP 주소
port = 9000         # 서버가 연결될 port 번호
s.bind((host, port)) # 통신 소켓을 (host, port)에 연결
s.listen(50)        # 클라이언트 연결 요청 대기

while True:
    c, addr = s.accept()    # 연결 요청 수락
    print ('Got connection from', addr)
    c.send(b'Thank you for connecting') # 데이터 전송(byte Type)
    c.close()
s.close()
```

## 파일명 : client.py

```
import socket

s = socket.socket()
host = '127.0.0.1' # 서버 IP
port = 9000
s.connect((host, port))
print(s.recv(1024).decode())
s.close
```

```
> python server.py
> python client.py
```

## 실행

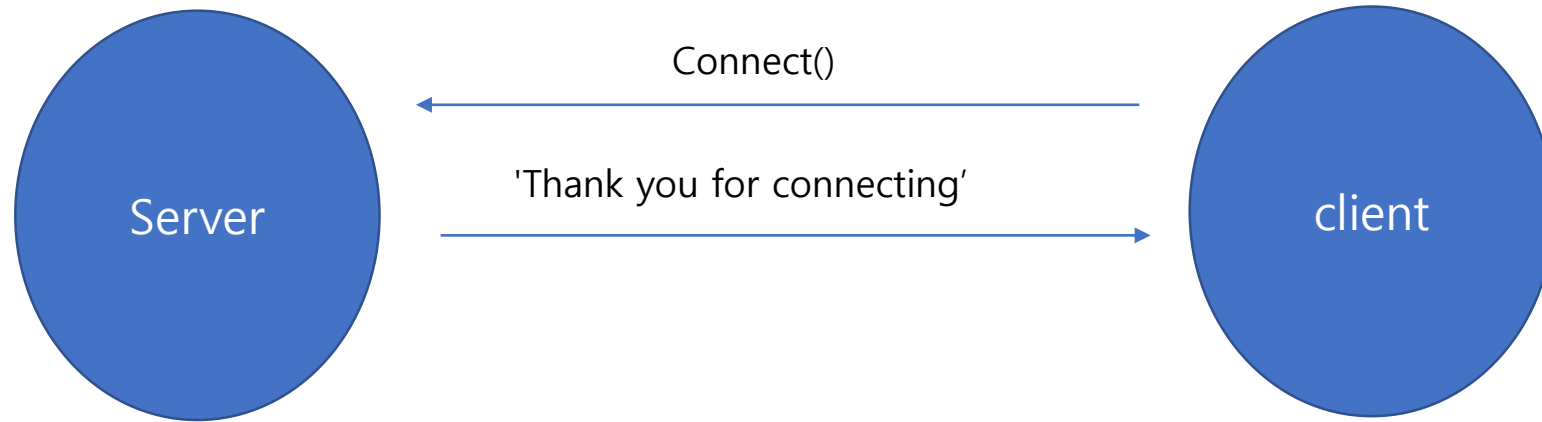
Terminal 1

```
> python server.py
```

Terminal 2

```
> python client.py
```

**질문 : 서버와 클라이언트 중 어느 것을 먼저 실행 시켜야 할까요?**



조금 바꿔 볼까요?

## [실습과제 2]

클라이언트 및 서버 소켓프로그램 (1대 1 통신)



```
import socket
s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
host = '0.0.0.0' # server IP address
Port = 9000
s.bind((host, port))
s.listen(10)
```

### While True:

```
    c, addr = s.accept()
    print ( ' Got connection from ', addr)
    msg = ""
    while True:
        data = c.recv(2048)
        msg = data.decode()
        if msg == 'bye':
            break
        print (msg, 'from client', addr)
        c.send(bytes(msg, 'UTF-8'))
    print('Client at', addr, 'disconnected...')
    c.close()
s.close()
```

```
# 통신용 소켓 생성(TCP 소켓)
# port가 다른 사용 중일 때도 강제 점유
# 서버 프로그램이 연결될 IP 주소(컴퓨터 모든 IP)
# 서버 프로그램이 연결될 포트 번호 9000번
# 소켓을 (host, port)에 연결
# 클라이언트 연결 요청 대기(동시 요청 수락 수)
```

```
# 클라이언트 연결 요청 수락
```

```
# 클라이언트 데이터 수신 (최대 2048Byte 까지)
# Byte 객체를 Unicode(UTF-8) 문자열로 변환
# 수신 문자가 'bye'이면 종료
```

```
# 문자열을 Unicode(UTF-8) Byte 객체로 변환 전송
```

```
import socket
s = socket.socket()
Host = '127.0.0.1'
Port = 9000
s.connect((host, port))

While True:
    out_data = input()
    s.sendall(bytes(out_data, 'UTF-8'))
    if out_data == 'bye':
        break

    in_data = s.recv(1024)
    print('From Server: ', in_data.decode())

s.close
```

# 통신용 소켓 생성  
# 연결할 서버의 IP 주소  
# 연결할 서버의 포트 번호  
# 소켓으로 서버(host, port) 연결 요청

# 키보드 데이터를 읽어옴  
# 문자열을 Unicode(UTF-8) Byte 객체로 변환 전송  
# 키보드 입력이 'bye' 면 종료

# 서버가 전송한 데이터 수신 (최대 1024 바이트)  
# 수신 데이터를 Unicode(UTF-8) 문자열로 변환

```
> python server.py
> python client.py
```

## 실행

Terminal 1

```
> python server.py
```

**질문 : 여러 클라이언트가 동시에 서버에  
접속할 경우 어떤 현상이 발생하는가?**

Terminal 2

```
> python client.py
```

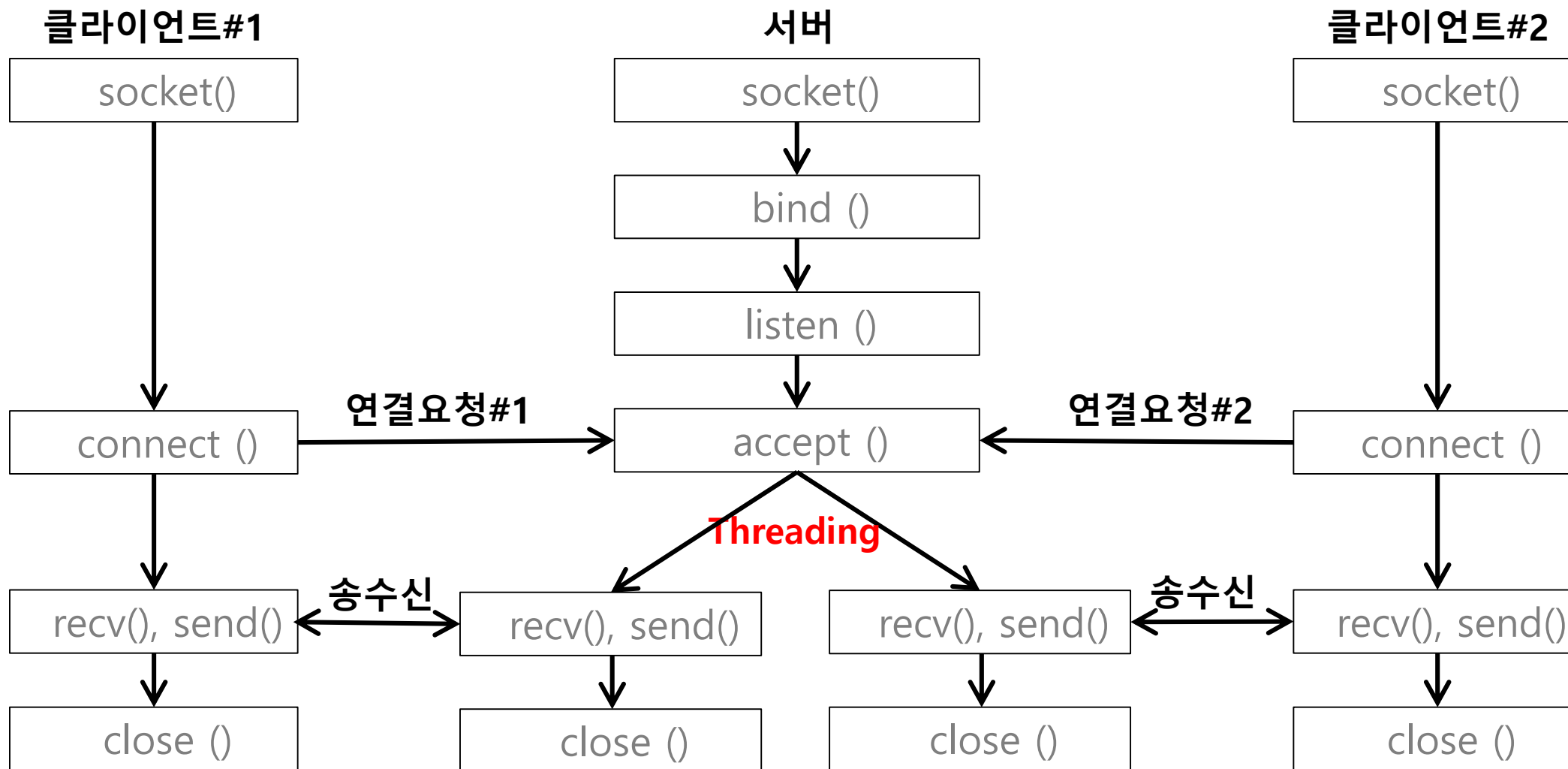
Terminal 3

```
> python client.py
```

## [실습과제 3]

다중 클라이언트 접속 서버 소켓프로그램 (n대 1 통신)

## TCP 소켓서버 확장



TCP 통신으로 1:N 통신 지원

```
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self, addr, c):
        threading.Thread.__init__(self)
        self.csocket = c

    def run(self):
        print ("Connection from : ", addr)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
                break
            print (msg, 'from client', addr)
            self.csocket.send(bytes(msg, 'UTF-8'))
        print ("Client at ", addr, " disconnected...")
```

```
host = "127.0.0.1"
port = 8080
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,
             socket.SO_REUSEADDR, 1)
s.bind((host, port))
print("Server started")
print("Waiting for client request..")
while True:
    s.listen(1)
    c, addr = s.accept()
    n = ClientThread(addr, c)
    n.start()
```

실행

```
> python mserver.py
```

복수 클라이언트에 응답할 수 있도록 thread 화

```
import socket
host = "127.0.0.1"
port = 8080
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

while True:
    out_data = input()
    s.sendall(bytes(out_data, 'UTF-8'))
    if out_data=='bye':
        break

    in_data = s.recv(1024)
    print("From Server :", in_data.decode())

s.close()
```

실행

> python client.py

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

## [실습과제 4]

**TCP** 소켓프로그래밍 프로그래밍 - chatting



```
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self, addr, c):
        threading.Thread.__init__(self)
        self.csocket = c

    def run(self):
        print ("connection from : ", addr)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
                break
            for client in client_sockets :
                client.send(bytes(msg, 'UTF-8'))
            print ("Client at ", addr , " disconnected...")
            if self.csocket in client_sockets :
                client_sockets.remove(self.csocket)
        self.csocket.close()
```

```
client_sockets = []
host = '0.0.0.0'
port = 9999
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,
             socket.SO_REUSEADDR, 1)
s.bind((host, port))
s.listen(10)
print("Chat Server start")
while True:
    c, addr = s.accept()
    client_sockets.append(c)
    n = ClientThread(addr, c)
    n.start()
    print('# of client:', len(client_socks))
```

Chat - 클라이언트로 부터 수신한 메시지를 모든 클라이언트로 전송

```
import socket, threading
class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
host = "127.0.0.1" # 서버 IP 주소
port = 9999
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
    message = input()
    s.send(message.encode())
    if message == 'bye':
        break

s.close()
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

## 실행

### Terminal 1 (Chat Server)

```
> python server.py
```

1. 자신의 컴퓨터에서 4개 터미널을 생성해서 실행
2. 다른 컴퓨터에서 서버에 접속하여 채팅해 보기

> 자신의 채팅 메시지도 자신에게 보임 ??

### Terminal 2 (Chat Client)

```
> python client.py  
Hi I am Jung  
[chat message] Hi I am Jung
```

### Terminal 3 (Chat Client)

```
> python client.py  
[chat message] Hi I am Jung
```

### Terminal 4 (Chat Client)

```
> python client.py  
[chat message] Hi I am Jung
```

## [도전 과제 4-1]

**TCP** 소켓프로그램 프로그래밍 – chating

실습 과제 6에서 자신의 메시지는 보이지 않도록 처리하시오.

```
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self, addr, c):
        threading.Thread.__init__(self)
        self.csocket = c

    def run(self):
        print ("connection from : ", addr)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
                break
            for client in client_sockets :
                if client != self.csocket :
                    client.send(bytes(msg, 'UTF-8'))
            print ("Client at ", addr , " disconnected...")
            if self.csocket in client_sockets :
                client_sockets.remove(self.csocket)
            self.csocket.close()
```

```
client_sockets = []
host = '0.0.0.0'
port = 9999
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,
             socket.SO_REUSEADDR, 1)
s.bind((host, port))
s.listen(10)
print("Chat Server start")
while True:
    c, addr = s.accept()
    client_sockets.append(c)
    n = ClientThread(addr, c)
    n.start()
    print('# of client:', len(client_socks))
```

Chat - 클라이언트로 부터 수신한 메시지를 모든 클라이언트로 전송

```
import socket, threading
class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
host = "127.0.0.1" # 서버 IP 주소
port = 9999
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
    message = input()
    if message == 'bye':
        s.send(message.encode())
        break
    s.send(message.encode())
s.close()
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

## [추가 과제 4-2]

**TCP** 소켓프로그램 프로그래밍 – chating

클라이언트 실행시 서버 IP 주소와 port 번호를 입력 받자...

```
import socket, threading
Import sys

class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
Print(len(sys.argv), sys.argv)
host = sys.argv[1]
port = int(sys.argv[2])
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')
```

```
n = ChatThread(s)
n.daemon = True
n.start()
```

```
while True:
    message = input()
    if message == 'bye':
        s.send(message.encode())
        break
    s.send(message.encode())
s.close()
```

실행

```
> python chat_client.py 127.0.0.1 9999
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료



```
import socket, threading
Import sys

class ChatThread(threading.Thread):
    def __init__(self,s):
        threading.Thread.__init__(self)
        self.csocket = s

    def run(self):
        while True:
            data = self.csocket.recv(2048)
            print('receive : ', data.decode())
```

```
if len(argv) != 3 :
    print(f'usage : python {sys.argv[0]} ip# port#')
    quit()
host = sys.argv[1]
port = int(sys.argv[2])

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
Print('chat client start')

n = ChatThread(s)
n.daemon = True
n.start()

while True:
    message = input()
    if message == 'bye':
        s.send(message.encode())
        break
    s.send(message.encode())
s.close()
```

실행 : 오류 처리 기능 추가

> python chat\_client.py 127.0.0.1

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

### [도전 과제 4-3]

채팅 클라이언트에서 자신의 이름(별명)을 입력 받고  
채팅 메시지에 붙여서 보낸다.

- name : syjung
- [syjung] hello

## [더 알아보기 과제 4-4]

딕셔너리(dictionary) 데이터 전송

```
import socket
import json
```

```
s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
host = '0.0.0.0' # server IP address
Port = 9000
s.bind((host, port))
s.listen(10)
```

```
while True:
```

```
    c, addr = s.accept()
    print('Got connection from', addr)
```

```
    for i in range(10)
```

```
        data = c.recv(2048)
        r_data = json.loads(data.decode())
        print('r_data :', r_data)
```

```
        s_data = json.dumps(r_data)
        c.sendall(bytes(s_data, 'utf-8'))
```

```
        print('client at', addr, 'disconnected...')
```

```
s.close()
```

```
# 통신용 소켓 생성(TCP 소켓)
# port가 다른 사용 중일 때도 강제 점유
# 서버 프로그램이 연결될 IP 주소(컴퓨터 모든 IP)
# 서버 프로그램이 연결될 포트 번호 9000번
# 소켓을 (host, port)에 연결
# 클라이언트 연결 요청 대기(동시 요청 수락 수)
```

```
# 클라이언트 연결 요청 수락
```

```
# json 파일 수신(byte type)
# byte 타입의 json 데이터를 str으로 변환 후, dictionary 로 변환
```

```
# dictionary를 Json으로 변환
```

```
import socket
import json
import time

s = socket.socket()
Host = '127.0.0.1'
Port = 9000

s.connect((host, port))
data = {}
data['name'] = 'jung'
for count in range(10):
    data['count'] = count
    body = json.dumps(data)
    print('send message: ', bytes(body, 'utf-8'))
    s.sendall(bytes(out_data, 'UTF-8'))

    in_data = s.recv(1024)
    r_data = json.loads(in_data.decode())
    print('From Server: ', r_data)
    time.sleep(3)

s.close
```

# 통신용 소켓 생성  
# 연결할 서버의 IP 주소  
# 연결할 서버의 포트 번호

# 소켓으로 서버(host, port) 연결 요청

# 키보드 데이터를 읽어옴  
# dic to json

# 문자열을 Unicode(UTF-8) Byte 객체로 변환 전송

# 서버가 전송한 데이터 수신 (최대 1024 바이트)

```
> python server.py
> python client.py
```

## [실습과제 5]

**UDP** 소켓프로그래밍 프로그래밍

## TCP v.s. UDP

	TCP	UDP
신뢰성 reliability	<ul style="list-style-type: none"> <li>● 오류제어 (사본 보유 및 재전송)</li> <li>● 흐름제어 (flow control)</li> </ul>	<ul style="list-style-type: none"> <li>● 오류제어 기능 없음</li> <li>● 흐름제어 기능 없음</li> </ul>
	신뢰성이 높다 (reliable)	신뢰성이 낮다 (unreliable)
연결 connection	<ul style="list-style-type: none"> <li>● 경로 설정 (path setup)</li> <li>● 자료의 순서 유지</li> </ul>	<ul style="list-style-type: none"> <li>● 경로 설정 절차가 없음</li> <li>● 자료의 순서가 뒤바뀔 수 있음</li> </ul>
	연결형(connection-oriented)	비연결형(connection-less)
비용 cost	<ul style="list-style-type: none"> <li>● more space, more time</li> </ul>	<ul style="list-style-type: none"> <li>● less space, less time</li> </ul>
	고비용	저비용
응용 분야 application		<ul style="list-style-type: none"> <li>● 한 패킷으로 서비스(DNS, time)</li> <li>● 안전한 LAN 환경 (NFS)</li> </ul>
	TCP	UDP

## TCP v.s. UDP

### TCP 소켓 생성

```
t1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
```

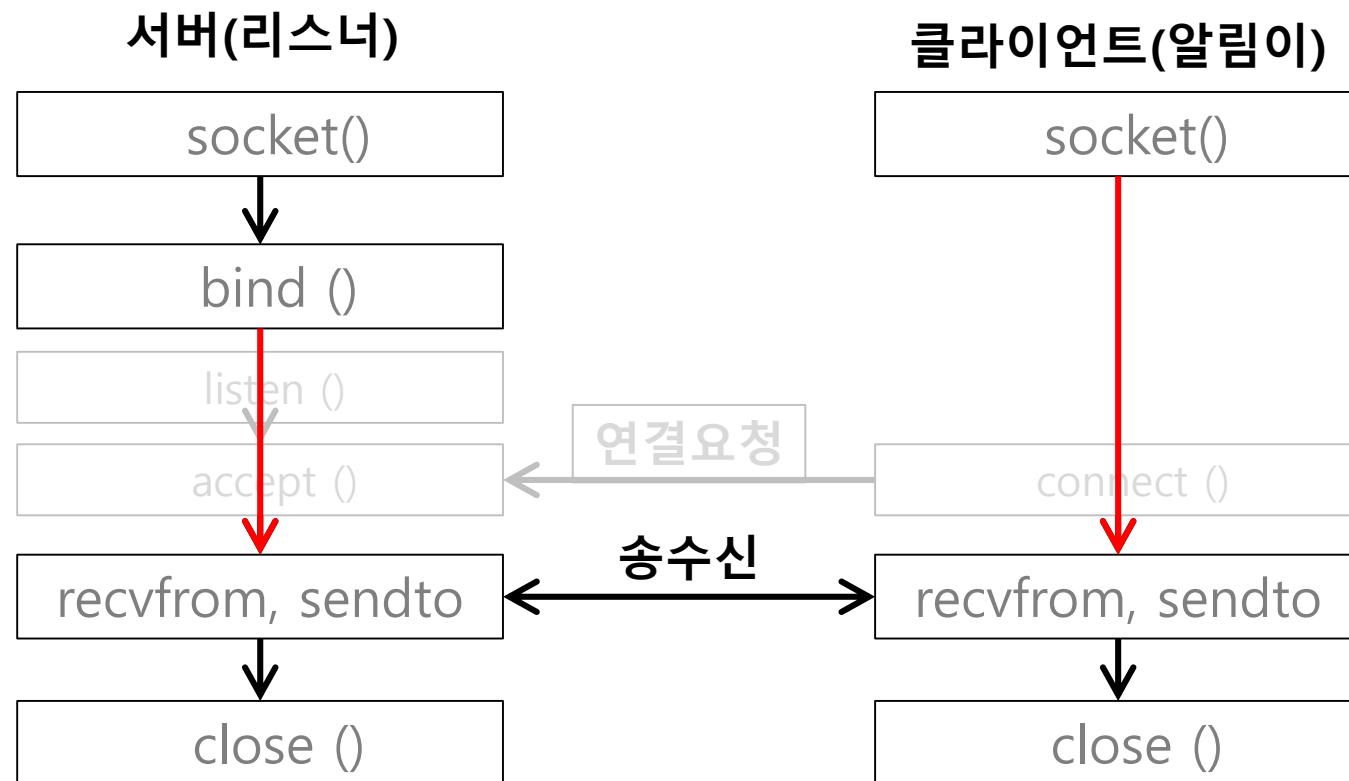
```
t2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

### UDP 소켓 생성

```
u1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
```

```
u2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```





TCP 통신으로 1:1 통신 지원

## 파일명 : listener.py

```
import socket
host = "0.0.0.0"
port = 20001

u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
u.bind((host, port))
print("UDP server up and listening")
while(True):
    c, addr = u.recvfrom(1024)
    print(f'Message [{c}] from {addr}')
    u.sendto(b'Hello UDP Client', addr)
```

실행

```
> python listener.py
```

## 파일명 : notify.py

```
import socket
name = socket.gethostname()
host = "127.0.0.1"
port = 20001

u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
u.settimeout(5.0)
u.sendto("Hello UDP Listener".encode('UTF-8'), (host, port))
c, addr = u.recvfrom(1024)
print(f'Message from Server {addr}')
Print(f'[Recv Data] {c.decode()}')
```

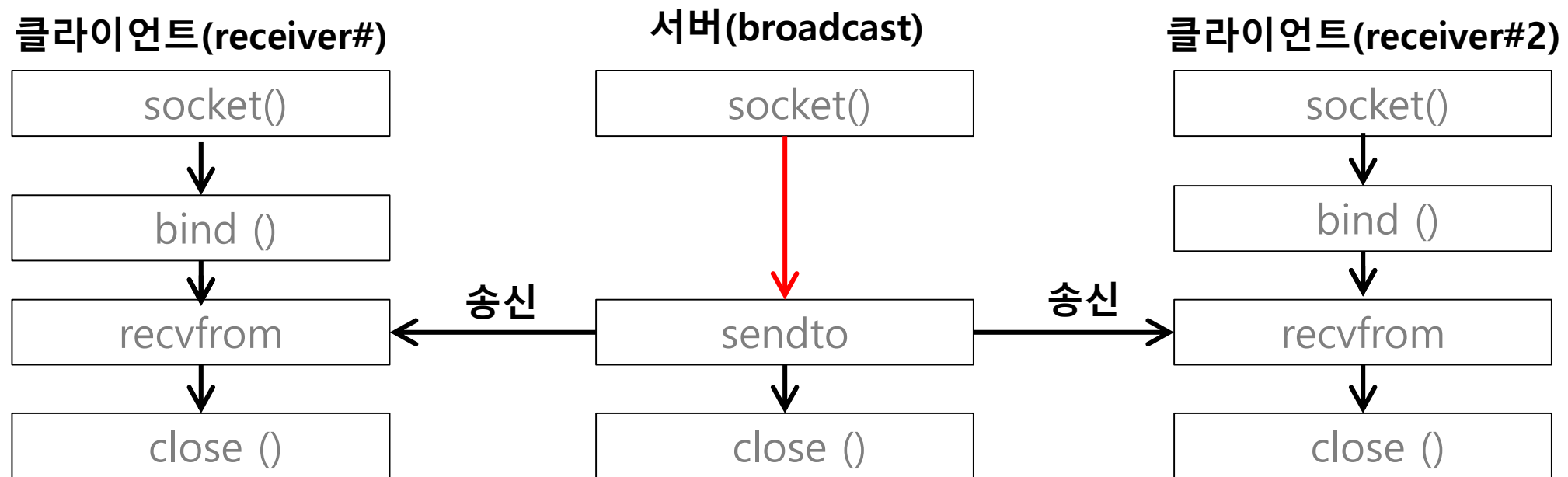
### 실행

```
> python notify.py
```

## [실습과제 6]

**UDP 브로드캐스팅 소켓프로그래밍 프로그래밍**

## UDP 브로드캐스트 - 메시지 방송



UDP 통신으로 다수 컴퓨터에 메시지 전송 (응답 기다리지 않음)

## UDP 브로드캐스트 소켓 프로그램 - 서버

## 파일명 : broadcaster.py

```
import socket
import time
host = '<broadcast>'
port = 37020
u = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.settimeout(0.2)
print("UDP Broadcast server up and announce message")
while(True):
    u.sendto(b"broadcast message", (host, port))
    print("Broadcast message")
    time.sleep(1)
```

## 실행

```
> python broadcaster.py
```

## UDP 브로드캐스트 - 클라이언트

## 파일명 : receiver.py

```
import socket

host = ""
port = 37020
u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM, proto=socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.bind((host, port))
print("Broadcast receiver start")
while True:
    c, addr = u.recvfrom(1024)
    print(f'Message [{c}] from Server {addr}')
```

실행

&gt; python receiver.py

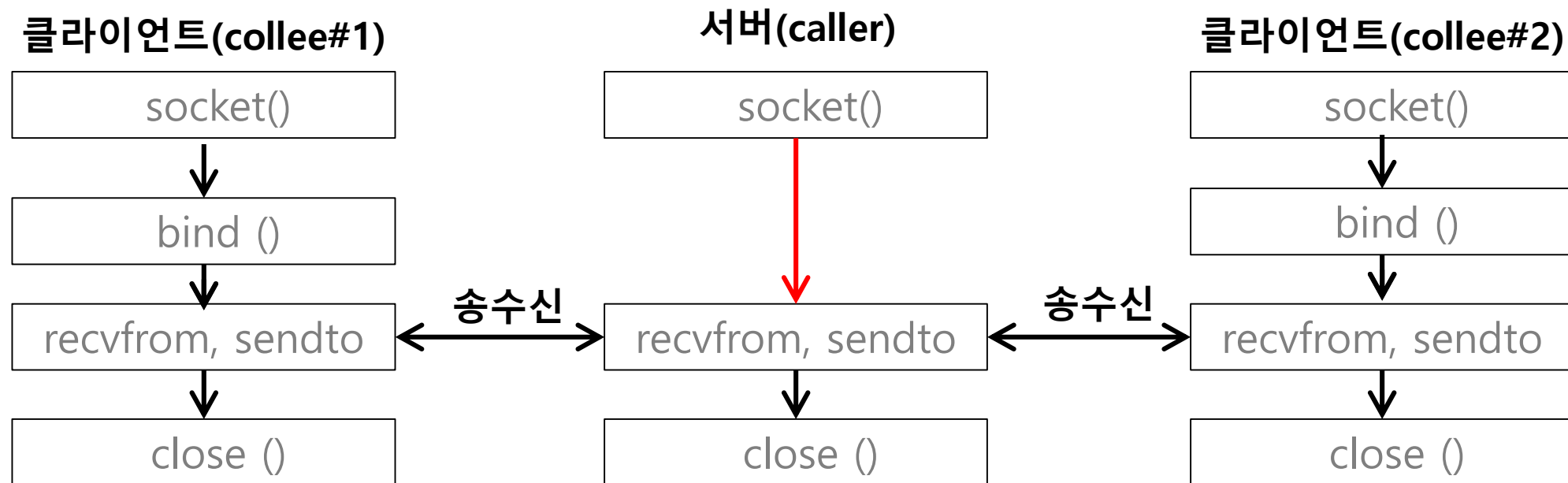
## [실습과제 7]

**UDP 브로드캐스팅** 소켓프로그램 프로그래밍 확장



## UDP broadcast 확장

## UDP 1:N caller, collee



UDP 1:N 통신으로 네트워크 주변 장비 확인

## UDP 브로드캐스트 소켓 프로그램 - 서버

## 파일명 : caller.py

```
import socket
import time
host = '<broadcast>'
port = 37021
u = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.settimeout(3)
print("UDP Broadcast to check devices")
while(True):
    u.sendto(b"broadcast message", (host, port))
    print("Braodcast message")
    try:
        while True:
            c, addr = u.recvfrom(1024)
            print(f'Message from client {addr}')
    except:
        pass
    time.sleep(1)
```

실행

&gt; python caller.py

## UDP 브로드캐스트 - 클라이언트

## 파일명 : callee.py

```
import socket
host = ""
port = 37021
u = socket.socket(family=socket.AF_INET,
                  type=socket.SOCK_DGRAM,
                  proto=socket.IPPROTO_UDP)

u.setsockopt(socket.SOL_SOCKET,
             socket.SO_BROADCAST, 1)
u.bind((host, port))
print("Answer to caller")
while True:
    c, addr = u.recvfrom(1024)
    print(f'Call from Server {addr}')
    u.sendto(b"receive message", addr)
```

## 실행

```
> python callee.py
```