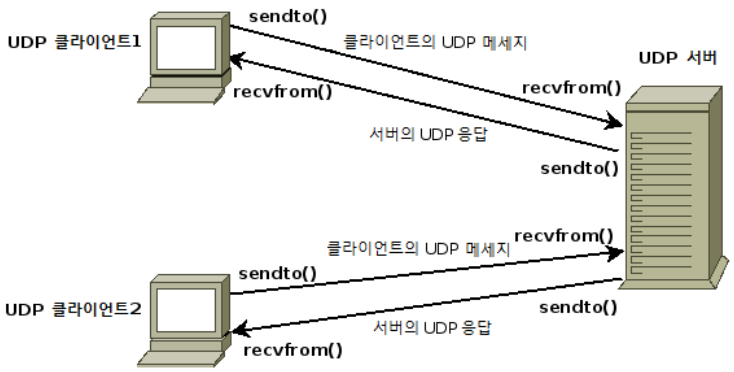
UDP

- User Datagram Protocol
- TCP/IP 슈트의 인터넷 프로토콜 사용
- UDP 사용한 통신에서 클라이언트 프로그램은 메시지 패킷을 대상 서버로 전송하며 대상 서버도 UDP에서 실행



속성

- 메시지 패킷 전달을 보장하지 않는다. 네트워크에서 문제 발생 시 패킷 영구 손실 가능성 있음.
- 신뢰할 수 없는 프로토콜로 간주
- UDP를 구현하는 기본 매커니즘에는 연결 기반 통신이 필요하지 않음.
    - UDP 서버 또는 UPD 클라이언트 간에 데이터 스트리밍 없음
- UDP 클라이언트는 n개의 별개 패킷을 서버에 전송 가능 & 서버의 응답으로 n개의 별개 패킷을 수신 가능
- 비 연결 프로토콜이므로 UDP와 관련된 오버 헤드는 TCP와 같은 연결 기반 프로토콜에 비해 적음.

출처: https://sdrl982.tistory.com/268

예시: 파이썬을 사용한 UDP 서버

```
import socket

localIP     = "127.0.0.1"
localPort   = 20001
bufferSize  = 1024

msgFromServer       = "Hello UDP Client"
bytesToSend         = str.encode(msgFromServer)

# 데이터그램 소켓을 생성
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# IP&PORT Bind
UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")

# 들어오는 데이터그램 Listen
while(True):
    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
    message = bytesAddressPair[0]
    address = bytesAddressPair[1]

    clientMsg = "Message from Client:{}".format(message)
    clientIP  = "Client IP Address:{}".format(address)

    print(clientMsg)
    print(clientIP)

    # Sending a reply to client
    UDPServerSocket.sendto(bytesToSend, address)
```

출력

```
UDP server up and listening
Message from client:b "Hello UDP Server"
Client IP Address: ("127.0.0.1", 57696)
```

예시: 파이썬을 사용한 UDP 클라이언트

```
import socket

msgFromClient       = "Hello UDP Server"
bytesToSend         = str.encode(msgFromClient)
serverAddressPort   = ("127.0.0.1", 20001)
bufferSize          = 1024

# 클라이언트 쪽에서 UDP 소켓 생성
UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# 생성된 UDP 소켓을 사용하여 서버로 전송
UDPClientSocket.sendto(bytesToSend, serverAddressPort)

# Server에서 메시지 수신
msgFromServer = UDPClientSocket.recvfrom(bufferSize)
msg = "Message from Server {}".format(msgFromServer[0])
print(msg)
```

Message from Server b" Hello UDP Client"

UDP 통신 python 소켓 프로그래밍

[서버]

```
import socket

sock = socket.socket( socket.AF_INET , socket.SOCK_DGRAM)
sock.bind( ('192.168.2.179',8080) )
data , addr = sock.recvfrom( 200 )

print ( " Server is received data : " , data.decode() )
print ( " Send Client IP : " , addr[0] )
print ( " Send Client Port : " , addr[1] )
```

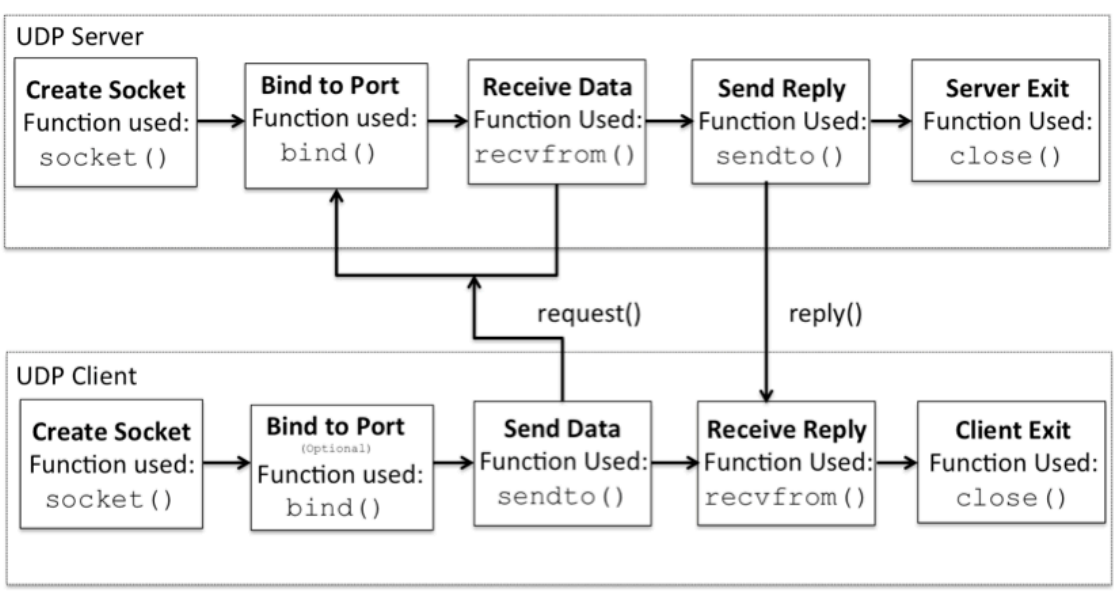
소켓 생성
서버 IP·PORT 고정
데이터 수신 대기, 최대 수신가능 데이터 200 byte
⇒ 반환값: (데이터, (IP, PORT))
msg & IP·PORT
msg
받은 데이터 출력
보내온 client IP 출력
보내온 client PORT 출력 (랜덤번호)

[클라이언트]

```
import socket

sock = socket.socket( socket.AF_INET , socket.SOCK_DGRAM )
sock.sendto( "Hello".encode() , ('100.100.100.110',8080) )
```

소켓 생성
Hello 문자열 encoding → 서버전송



- udp : 신뢰성이 없는 데이터그램으로서 데이터가 중복되거나 누락될 수 있지만 속도가 빠릅니다.
- tcp : 신뢰성이 있는 바이트 스트림 지향으로서 메시지 수신을 확인하며 통신하게 됩니다.
- 소켓 : 프로세스 통신의 종착점입니다.

udp

1. 클라이언트와 서버 모두 소켓을 열어주고, 클라이언트에서 서버 호스트와 포트로 데이터그램을 생성합니다.
2. 서버에서는 생성된 데이터 그램을 읽어들여서 소켓을 이용하여 서버 호스트와 포트로 반환합니다.
3. 다시 클라이언트에서는 소켓으로 받은 데이터 그램을 읽어 값을 얻게 됩니다.