Writing a UDP Server

In the last lesson, we'd written code to setup an IPV4 socket on TCP. Let's now get into writing a program for a basic server.

We'll cover the following

Setting a Purpose for the Server

In this lesson, we'll finish writing our server program. Our server will reply to every client's message with **a capitalized version of whatever a client program sends to it**. This is just what we're requiring of our server to do, there are other functions that such a server can perform as well.

So, in particular, the server will:

1. Print the original message received from the client.
2. Capitalize the message.
3. Send the capitalized version back to the client.

Let's code!

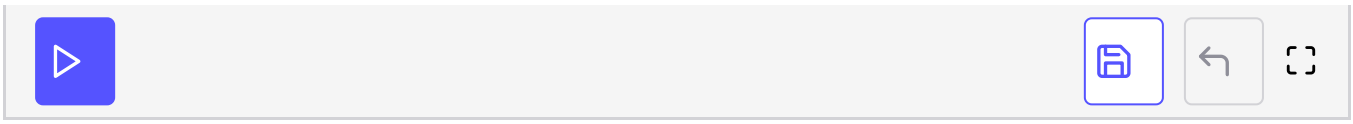
UDP IPV4 Socket

For reference, here is the code for a UDP socket.

```
import socket

# Setting up a socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port)) # Binding the socket to a port and IP address
print('Listening at {}'.format(s.getsockname())) # Printing the IP address and port of socket
```



You can use the `getsockname()` method on an object of the `socket` class to find the current IP address and port that a socket is bound to.

Listening Infinitely

Next, we set up a while loop (**lines 10 and 11**) so that the server listens infinitely. If the rest of this code weren't in this infinite while loop, the server would exit after dealing with one client.

```
import socket

# Setting up a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port)) # Binding the socket to a port and IP address
print('Listening at {}'.format(s.getsockname())) # Printing the IP address and port of socket

while True:
    # The code to handle clients will go here
```

Receiving Messages from Clients

The server can now receive data from clients! The `recvfrom()` method (**line 12**) here accepts data of `MAX_SIZE_BYTES` length (declared on **line 3**) which is the size of one UDP datagram in bytes. This is to make sure that we receive the entirety of each packet. It also returns the IP address of the client that sent the data. We store the data and the client's IP address in the variables `data` and `clientAddress` respectively.

Note that the code stops and waits at `recvfrom()` until some data is received.

```
import socket

MAX_SIZE_BYTES = 65535 # Maximum size of a UDP datagram
```

```
# Setting up a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port)) # Binding the socket to a port and IP address
print('Listening at {}'.format(s.getsockname())) # Printing the IP address and port of socket
while True:
    data, clientAddress = s.recvfrom(MAX_SIZE_BYTES) # Receive at most 65535 bytes at once
```

Capitalizing the Data

Next, we decode the message from the byte stream to ASCII (**lines 13 and 14**). Then we capitalize whatever the client sent.

```
import socket

MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram

# Setting up a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port)) # Binding the socket to a port and IP address
print('Listening at {}'.format(s.getsockname())) # Printing the IP address and port of socket
while True:
    data, clientAddress = s.recvfrom(MAX_SIZE_BYTES)
    message = data.decode('ascii')
    upperCaseMessage = message.upper()
```

Printing the Client's Message & Encoding

We print the client's IP address next since it's always a good idea after a connection has been made in order to keep track of it (**line 15**). We also print the message they sent along with it.

We then encode the capitalized ASCII message to bytes (**line 16**).

```
import socket

MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram

# Setting up a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port))
print('Listening at {}'.format(s.getsockname()))
while True:
    data, clientAddress = s.recvfrom(MAX_SIZE_BYTES)
    message = data.decode('ascii')
    upperCaseMessage = message.upper()
```

```
    print('The client at {} says {!r}'.format(clientAddress, message))
    data = upperCaseMessage.encode('ascii')
```

Sending the Message Back to the Client

Lastly, we send the capitalized message back to the client using the `sendto()` function (**line 17**).

```
import socket

MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram

# Setting up a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port))
print('Listening at {}'.format(s.getsockname()))
while True:
    data, clientAddress = s.recvfrom(MAX_SIZE_BYTES)
    message = data.decode('ascii')
    upperCaseMessage = message.upper()
    print('The client at {} says {!r}'.format(clientAddress, message))
    data = upperCaseMessage.encode('ascii')
    s.sendto(data, clientAddress)
```

Now we have a basic server that accepts messages from clients, has a defined purpose (capitalization), and responds to the client's messages. Let's write code for a client to go with this in the next lesson.