Programação com UDP

UDP: não há "conexão" entre cliente e servidor

- Sem negociação (handshaking) antes de enviar dados
- O emissor coloca explicitamente o endereço IP de destino e o nº da porta em cada datagrama
- O recetor extrai o endereço de retorno e a porta do datagrama recebido

UDP: os dados podem ser perdidos ou recebidos fora de ordem

Do ponto de vista da aplicação:

- UDP fornece transferência *não fiável* de grupos de bytes (datagramas ou mensagens) entre cliente e servidor

Interação cliente/servidor usando sockets UDP

servidor (a correr em serverIP)

Criar socket associado à porta X
serverSocket =
socket(AF_INET,SOCK_DGRAM)

Ler datagrama do
serverSocket

Escrever resposta em
serverSocket
especificando
endereço IP do cliente
Número da porta

client

Criar Socket
clientSocket =
socket(AF_INET,SOCK_DGRAM)

Criar datagram e associar serverIP e porta X ;
Enviar datagrama através de clientSocket

Ler datagrama de
clientSocket

fechar
clientSocket

Cliente UDP

Cliente Python UDPClient

Incluir a bibloteca de sockets do Python →
```
from socket import *
serverName = 'hostname'
serverPort = 12000
```
Criar socket UDP →
```
clientSocket = socket(AF_INET,
                      SOCK_DGRAM)
```
Obter entrada do teclado →
```
message = input('Input lowercase sentence:')
```
Definir endereço IP e porta de destino; enviar pelo socket →
```
clientSocket.sendto(message.encode(),
                    (serverName, serverPort))
```
Ler resposta para a string →
```
modifiedMessage, serverAddress =
                    clientSocket.recvfrom(2048)
```
Imprime a mensagem recebida e fecha o socket →
```
print modifiedMessage.decode()
clientSocket.close()
```

message.encode() converte a variável do tipo String do Python para um array de bytes a colocar no datagrama
modifiedMessage.decode() faz o inverso

Servidor UDP

Servidor Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
                        clientAddress)
```

Criar socket UDP →

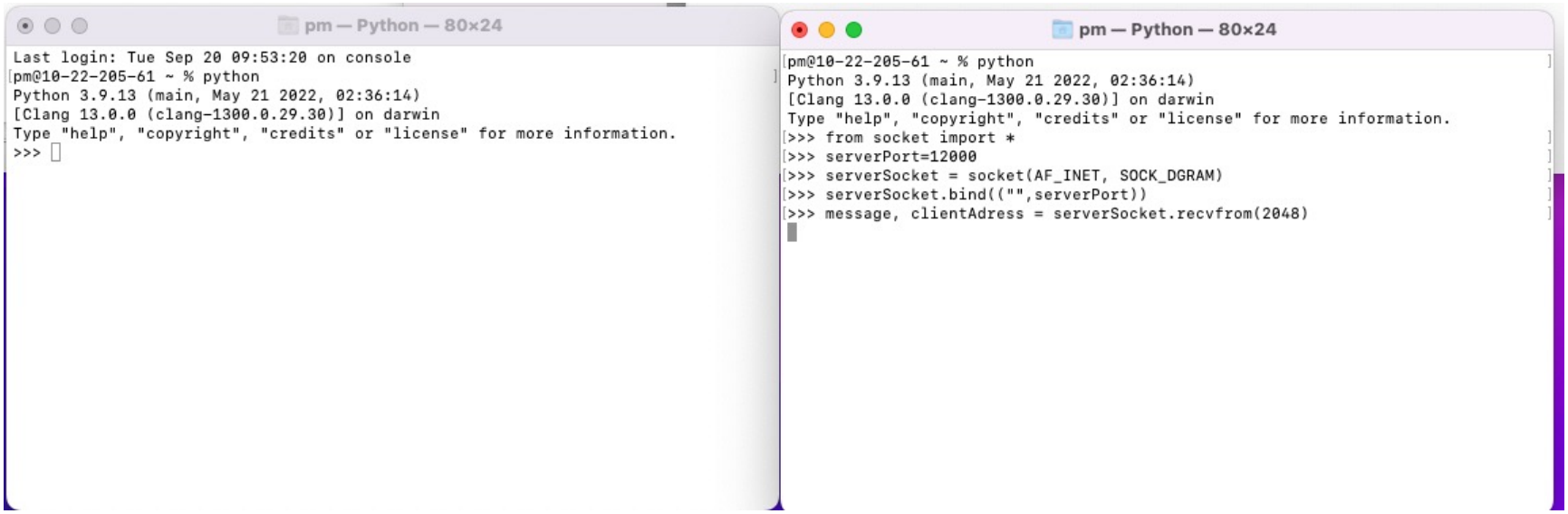
Associar à porta local 12000 →

Em ciclo →

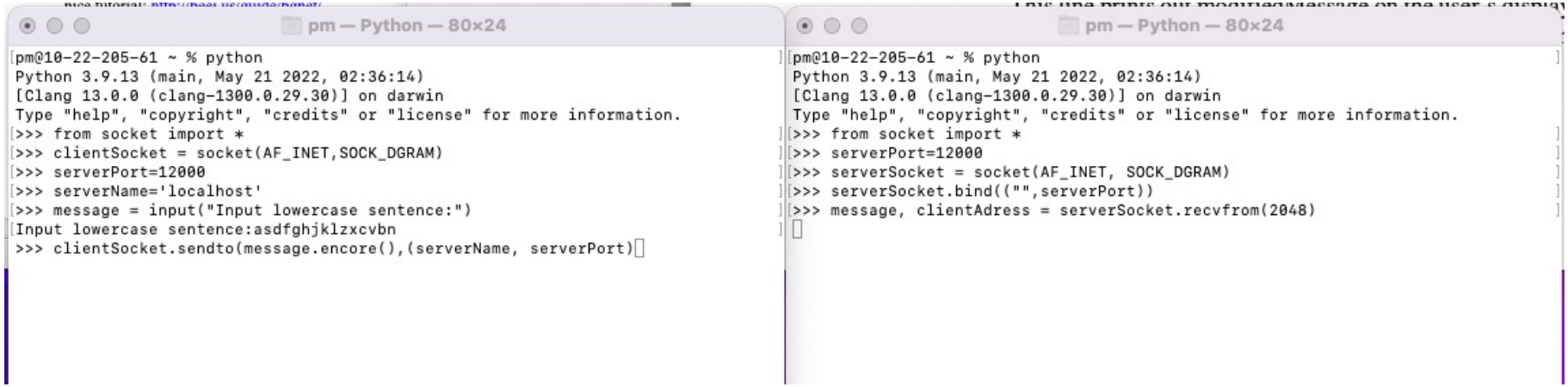
Ler do socket UDP para message, obtém endereço do cliente(client IP and port) →

Envia mensagem de volta para o cliente →

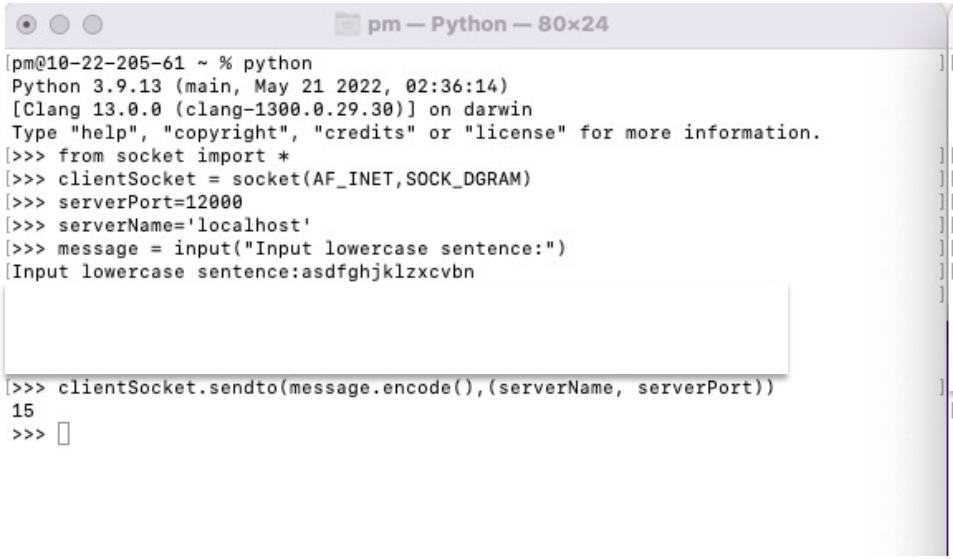
Servidor arranca e está disponível

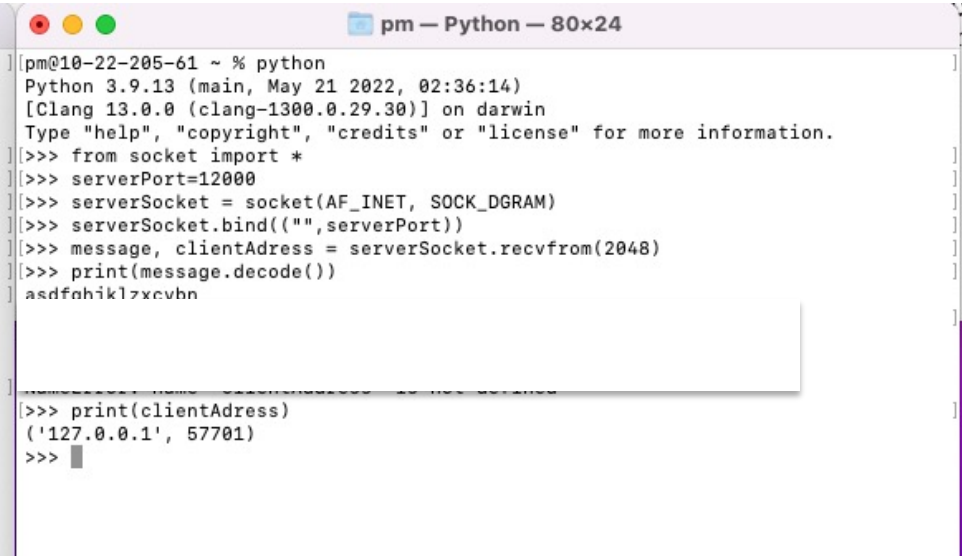


Cliente antes de enviar

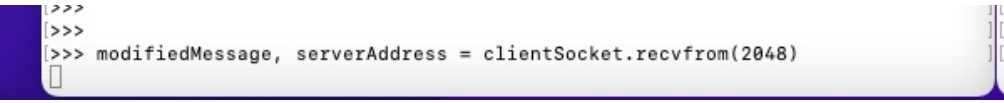


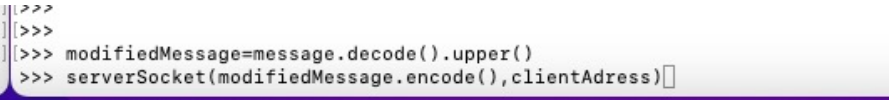
Após envio pelo cliente e receção pelo servidor

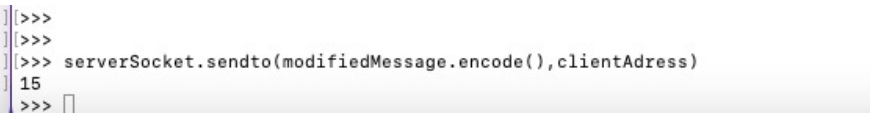




Envio da resposta pelo servidor







Cliente e servidor

```python
from socket import *

clientSocket = socket(AF_INET,SOCK_DGRAM)

serverName="localhost"
serverPort= 12000

message=input("Input lowercase sequence: ")

clientSocket.sendto( message.encode(),(serverName, serverPort))

modifiedMessage, serverAddres = clientSocket.recvfrom(2048)

print(modifiedMessage.decode())

clientSocket.close()
```

```python
from socket import *

serverSocket = socket(AF_INET,SOCK_DGRAM)

serverPort= 12000

serverSocket.bind(("",serverPort))

while  True:
    message, clientAddress = serverSocket.recvfrom( 2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto( modifiedMessage.encode(), clientAddress)

serverSocket.close()
```

```
Desktop — Python UDPserver.py — 80×24
pm@10-22-205-61 Desktop %
pm@10-22-205-61 Desktop %
pm@10-22-205-61 Desktop %
pm@10-22-205-61 Desktop % python UDPserver.py
```

```
Desktop — -zsh — 80×24
pm@10-22-205-61 Desktop % python UDPClient.py
Input lowercase sequence: qwertyuiop
QWERTYUIOP
pm@10-22-205-61 Desktop % python UDPClient.py
Input lowercase sequence: asdfghjkl
ASDFGHJKL
pm@10-22-205-61 Desktop % python UDPClient.py
Input lowercase sequence: zxcvbnm
ZXCVBNM
pm@10-22-205-61 Desktop %
```