Resolución Práctica 3.2.1: Implementación (parcial) de un cliente TFTP

Fichero *“tftp_cli_rrq.py”*:

```
#!/usr/bin/env python3

import sys
import socket
import time

NULL  = b'\x00'
RRQ   = b'\x00\x01'
WRQ   = b'\x00\x02'
DATA  = b'\x00\x03'
ACK   = b'\x00\x04'
ERROR = b'\x00\x05'

PORT = 50069
BLOCK_SIZE = 512

def get_file(s, serv_addr, filename):
      start = time.time()

      f = open(filename, 'wb')

      req  = RRQ
      req +=  filename.encode() + NULL
      req += 'octet'.encode() + NULL
      s.sendto(req, serv_addr)
      expected_block = 1
      bytes_received = 0
      while True:
            resp, serv_addr = s.recvfrom(4 + BLOCK_SIZE)

            opcode = resp[:2]
            if opcode != DATA:
                  print('Unexpected response.')
                  exit(1)
            else:
                  block = int.from_bytes(resp[2:4], 'big')
                  if block != expected_block:
                        continue
                  data = resp[4:]
                  f.write(data)
                  bytes_received += len(data)
                  req = ACK + expected_block.to_bytes(2, 'big')
                  s.sendto(req, serv_addr)
                  if len(data) < BLOCK_SIZE:
                        break
                  expected_block += 1
      f.close()
      elapsed = time.time() - start
      print('{} bytes received in {:.2e} seconds ({:.2e}
b/s).'.format(bytes_received, elapsed, bytes_received * 8 / elapsed))

if __name__ == '__main__':
      if len(sys.argv) != 3:
            print('Usage: {} server filename'.format(sys.argv[0]))
            exit(1)
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
serv_addr = (sys.argv[1], PORT)

get_file(s, serv_addr, sys.argv[2])
```