Resolución Práctica 3.2.3: Implementación (parcial) de un servidor TFTP mejorado

Fichero *“tftp_ser_rrq_mejorado.py”*:

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

import sys
import os
import socket
import select
import signal

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
FILES_PATH ='./data/'
TIMEOUT = 0.5
MAX_RETRANSMISIONS = 3

def send_error(s, addr, code, message):
      resp  = ERROR
      resp += code.to_bytes(2, 'big')
      resp += message.encode()
      resp += NULL
      s.sendto(resp, addr)

def send_file(s, addr, filename):
      try:
            f = open(os.path.join(FILES_PATH, filename), 'rb')
      except:
            send_error(s, addr, 1, 'File not found.')
            exit(1)

      s.connect(addr)
      data = f.read(BLOCK_SIZE)
      resp  = DATA
      resp += b'\x00\x01'
      resp += data
      s.send(resp)

      block_num = 1
      last = False if len(data) == BLOCK_SIZE else True
      while True:
            for trial in range(MAX_RETRANSMISIONS):
                  received, _, _ = select.select([s], [], [], TIMEOUT)
                  if received:
                        break
                  print('Retransmitting ({})...'.format(trial))
                  s.send(resp)
            else:
                  print('Abandoning after {} trials'.format(MAX_RETRANSMISIONS))
                  exit(1)
            resp = s.recv(64)
            opcode = resp[:2]
```

```
            if opcode == ERROR:
                  error_code = int.from_bytes(resp[2:4], 'big')
                  print('Server error {}: {}'.format(error_code, resp[4:-
1].decode()))
                  exit(1)
            elif opcode != ACK:
                  print('Unexpected response.')
                  exit(1)
            else:
                  ack_num = int.from_bytes(resp[2:4], 'big')
                  if ack_num != block_num:
                        continue
                  if last:
                        break
                  block_num += 1
                  block_num %= (1 << 16)
                  data = f.read(BLOCK_SIZE)
                  resp  = DATA
                  resp += block_num.to_bytes(2, 'big')
                  resp += data
                  s.send(resp)
                  if len(data) < BLOCK_SIZE:
                        last = True
      f.close()

if __name__ == '__main__':
      signal.signal(signal.SIGCHLD, signal.SIG_IGN)
      s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
      s.bind(('', PORT))

      while True:
            req, cli_addr = s.recvfrom(64)

            opcode = req[:2]
            if opcode != RRQ:
                  send_error(s, cli_addr, 5, 'Unexpected opcode.')
            else:
                  filename, mode, _ = req[2:].split(b'\x00')
                  if mode.decode().lower() not in ('octet', 'binary'):
                        send_error(s, cli_addr, 0, 'Mode unkown or not
implemented')
                        continue
            filename = os.path.basename(filename.decode()) # For security,
filter possible paths.
            if not os.fork():
                  s.close()
                  dialog = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                  send_file(dialog, cli_addr, filename)
                  dialog.close()
                  exit(0)
```