**ESCOM**

Practica 2

Axel

Andrés Rodarte López

Estrada Pichardo Jonatan Isúi

3CM#

Introducción

Los sockets de datagrama son un tipo de sockets que proveen una forma sin conexión de enviar y recibir paquetes. Cada paquete enviado por un datagrama socket es individualmente enviado, también es usado para enviar y recibir mensajes broadcast. Usan el protocolo UDP.

UDP no admite numeración de los datagramas, factor que, sumado a que tampoco utiliza señales de confirmación de entrega, hace que la garantía de que un paquete llegue a su destino sea mucho menor que si se usa TCP. Esto también origina que los datagramas pueden llegar duplicados y/o desordenados a su destino. Por estos motivos el control de envío de datagramas, si existe, debe ser implementado por las aplicaciones que usan UDP como medio de transporte de datos, al igual que el reensamble de los mensajes entrantes.

Cuando una aplicación envía datos a través de UDP, éstos llegan al otro extremo como una unidad. Por ejemplo, si una aplicación escribe 5 veces en el puerto UDP, la aplicación al otro extremo hará 5 lecturas del puerto UDP. Además, el tamaño de cada escritura será igual que el tamaño de las lecturas.

Los archivos JSON (JavaScript Object Notation), es un formato de texto sencillo para el intercambio de datos, se puede utilizar en varias formas pero su principal uso es serializar y transmitir datos estructurados entre un servidor y cliente.

Desarrollo

Esta es una aplicación que permite al cliente guardar archivos en el servidor y se envían a través de un socket

**Servidor**

import socket

import sys

import pickle

import json

import os

HOST = '127.0.0.1' # The server's hostname or IP address

PORT = 65435 # The port used by the server

ServerDirectory = './ServerDummy/'

# Create a UDP socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

db = { "alumnos": [], "Grupos": []}

g1\_A = {"Mate 1":"", "Fisica 1":"", "Quimica 1": ""}

g2\_A = {"Mate 2":"", "Fisica 2":"", "Quimica 2": ""}

g3\_A = {"Mate 3":"", "Fisica 3":"", "Quimica 3": ""}

g1\_B = {"Etica 1":"", "Finanzas 1":"", "Gestion 1": ""}

g2\_B = {"Etica 2":"", "Finanzas 2":"", "Gestion 2": ""}

g3\_B = {"Etica 3":"", "Finanzas 3":"", "Gestion 3": ""}

g\_h ={

'Grupo 1-A': {"Mate 2":"7:00-8:00", "Fisica 2":"8:00-9:00", "Quimica 2": "9:00-10:00"},

'Grupo 2-A' : {"Mate 1":"7:00-8:00", "Fisica 1":"8:00-9:00", "Quimica 1": "9:00-10:00"},

'Grupo 3-A' : {"Mate 3":"7:00-8:00", "Fisica 3":"8:00-9:00", "Quimica 3": "9:00-10:00"},

'Grupo 1-B': {"Etica 1":"7:00-8:00", "Finanzas 1":"8:00-9:00", "Gestion 1": "9:00-10:00"},

'Grupo 2-B':{"Etica 2":"7:00-8:00", "Finanzas 2":"8:00-9:00", "Gestion 2": "9:00-10:00"},

'Grupo 3-B': {"Etica 3":"7:00-8:00", "Finanzas 3":"8:00-9:00", "Gestion 3": "9:00-10:00"}

}

if not os.path.exists(ServerDirectory):

os.mkdir(ServerDirectory)

if os.path.isfile('data.json'):

with open('data.json', 'r') as outfile:

db = json.load(outfile)

outfile.close()

else:

with open('data.json', 'w') as outfile:

json.dump(db, outfile)

outfile.close()

# Bind the socket to the port

server\_address = (HOST, PORT)

print('starting up on {} port {}'.format(\*server\_address))

sock.bind(server\_address)

while True:

print('\nwaiting to receive message')

data, address = sock.recvfrom(4096)

data = pickle.loads(data)

print('received {} bytes from {}'.format(len(data), address))

#print(data)

if data["op"] == 'sign':

data.pop('op', None)

db["alumnos"].append(data)

if data["group"] == 'Grupo 1-A':

db[data["boleta"]] = {"Materias":g1\_A, "Grupo":data["group"]}

elif data["group"] == 'Grupo 2-A':

db[data["boleta"]] = {"Materias":g2\_A, "Grupo":data["group"]}

elif data["group"] == 'Grupo 3-A':

db[data["boleta"]] = {"Materias":g3\_A, "Grupo":data["group"]}

elif data["group"] == 'Grupo 1-B':

db[data["boleta"]] = {"Materias":g1\_B, "Grupo":data["group"]}

elif data["group"] == 'Grupo 2-B':

db[data["boleta"]] = {"Materias":g2\_B, "Grupo":data["group"]}

elif data["group"] == 'Grupo 3-B':

db[data["boleta"]] = {"Materias":g3\_B, "Grupo":data["group"]}

with open('data.json', 'w') as outfile:

json.dump(db, outfile, sort\_keys=True, indent=4)

if data:

sent = sock.sendto(b'end', address)

print('sent {} bytes back to {}'.format(sent, address))

f = open(ServerDirectory + data["foto"], 'wb')

while True:

data, address = sock.recvfrom(4096)

f.write(data)

sent = sock.sendto(b'next', address)

if len(data) < 4096:

break

f.close()

print("Job done")

elif data["op"] == 'grades':

with open('data.json', 'r') as outfile:

db = json.load(outfile)

outfile.close()

if data["calOf"] in db:

sent = sock.sendto(pickle.dumps(db[data["calOf"]]['Materias']), address)

else:

sent = sock.sendto(b'No\_user', address)

elif data["op"] == 'horario':

with open('data.json', 'r') as outfile:

db = json.load(outfile)

outfile.close()

if data["calOf"] in db:

sent = sock.sendto(pickle.dumps(g\_h[db[data["calOf"]]['Grupo']]), address)

else:

sent = sock.sendto(b'No\_user', address)

#sent = sock.sendto(pickle.dumps(alumno), address)

**Cliente**

import json

import sys

import os

import socket

import pickle

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import QIcon

from PyQt5.QtCore import Qt

HOST = '127.0.0.1' # The server's hostname or IP address

PORT = 65435 # The port used by the server

# Subclass QMainWindow to customise your application's main window

class MainWindow(QMainWindow):

def \_\_init\_\_(self, parent=None):

super(MainWindow, self).\_\_init\_\_(parent)

self.setWindowTitle("MogSaes")

windw = Window(self)

windw.setMinimumSize(1,1)

# Set the central widget of the Window. Widget will expand

# to take up all the space in the window by default.

self.setCentralWidget(windw)

def on\_button\_clicked\_alumno(self):

alumno = Alumno(self)

self.setCentralWidget(alumno)

#alert = QMessageBox()

#alert.setText('You clicked the button!')

#alert.exec\_()

def on\_button\_clicked\_maestro(self):

pass

#Maestro = Maestro(self)

#self.setCentralWidget(Maestro)

def on\_button\_clicked\_salir(self):

windw = Window(self)

self.setCentralWidget(windw)

class Alumno(QWidget):

def \_\_init\_\_(self, parent):

super(Alumno, self).\_\_init\_\_(parent)

self.layout = QVBoxLayout(self)

self.mainAlumno()

def mainAlumno(self):

self.clean()

#self.parent().resize(500,250)

self.mainWidget = QWidget()

self.mainWidget.layout = QVBoxLayout(self)

self.button1 = QPushButton("Inscribir")

self.mainWidget.layout.addWidget(self.button1)

self.button1.clicked.connect(self.inscribir)

self.button2 = QPushButton("Ver calificaciones")

self.button2.clicked.connect(self.calificaciones)

self.mainWidget.layout.addWidget(self.button2)

self.button3 = QPushButton("Ver horario")

self.button3.clicked.connect(self.horario)

self.mainWidget.layout.addWidget(self.button3)

self.button4 = QPushButton("Salir")

self.mainWidget.layout.addWidget(self.button4)

self.button4.clicked.connect(self.parent().on\_button\_clicked\_salir)

self.mainWidget.setLayout(self.mainWidget.layout)

self.layout.addWidget(self.mainWidget)

self.setLayout(self.layout)

def inscribir(self):

self.clean()

self.tabs = QTabWidget()

self.info = QWidget()

self.grupo = QWidget()

self.foto\_path = ''

self.group = ''

self.tabs.addTab(self.info, "Información personal")

self.tabs.addTab(self.grupo,"Grupos")

self.info.layout = QFormLayout(self)

self.boleta = QLineEdit(self)

LB = QLabel('Boleta', self)

self.nombre = QLineEdit(self)

LN = QLabel('Nombre', self)

self.ApeMate = QLineEdit(self)

LAm = QLabel('Apellido Materno', self)

self.ApePate = QLineEdit(self)

LAp = QLabel('Apellido Paterno', self)

foto = QPushButton("Subir foto")

foto.clicked.connect(self.getFoto)

self.LF = QLabel('', self)

self.info.layout.addRow(LB, self.boleta)

self.info.layout.addRow(LN, self.nombre)

self.info.layout.addRow(LAp, self.ApePate)

self.info.layout.addRow(LAm, self.ApeMate)

self.info.layout.addRow(self.LF, foto)

self.info.setLayout(self.info.layout)

self.grupo.layout = QGridLayout()

self.b1 = QRadioButton('Grupo 1-A')

self.b1.toggled.connect(lambda:self.btnstate(self.b1))

self.b2 = QRadioButton('Grupo 1-B')

self.b2.toggled.connect(lambda:self.btnstate(self.b2))

self.b3 = QRadioButton('Grupo 2-A')

self.b3.toggled.connect(lambda:self.btnstate(self.b3))

self.b4 = QRadioButton('Grupo 2-B')

self.b4.toggled.connect(lambda:self.btnstate(self.b4))

self.b5 = QRadioButton('Grupo 3-A')

self.b5.toggled.connect(lambda:self.btnstate(self.b5))

self.b6 = QRadioButton('Grupo 3-B')

self.b6.toggled.connect(lambda:self.btnstate(self.b6))

self.grupo.layout.addWidget(self.b1,0,0)

self.grupo.layout.addWidget(self.b2,0,1)

self.grupo.layout.addWidget(self.b3,1,0)

self.grupo.layout.addWidget(self.b4,1,1)

self.grupo.layout.addWidget(self.b5,2,0)

self.grupo.layout.addWidget(self.b6,2,1)

self.grupo.setLayout(self.grupo.layout)

sign = QPushButton("Inscribisrse")

rtrn = QPushButton("Regresar")

sign.clicked.connect(self.sendInfo)

rtrn.clicked.connect(self.mainAlumno)

auxBox = QWidget()

auxBox.layout = QHBoxLayout()

auxBox.layout.addWidget(sign)

auxBox.layout.addWidget(rtrn)

auxBox.setLayout(auxBox.layout)

self.layout.addWidget(self.tabs)

self.layout.addWidget(auxBox)

self.setLayout(self.layout)

def calificaciones(self):

self.clean()

self.parent().resize(250,100)

getUser = QWidget()

getUser.layout = QFormLayout(self)

self.boletaS = QLineEdit(self)

LB = QLabel('Boleta', self)

getUser.setLayout(getUser.layout)

snd = QPushButton("Aceptar")

snd.clicked.connect(self.lookGrades)

rtrn = QPushButton("Regresar")

rtrn.clicked.connect(self.mainAlumno)

getUser.layout.addRow(LB, self.boletaS)

getUser.layout.addRow(snd, rtrn)

self.layout.addWidget(getUser)

self.setLayout(self.layout)

#self.parent().setFixedSize(self.layout.sizeHint())

def horario(self):

self.clean()

self.parent().resize(250,100)

getUser = QWidget()

getUser.layout = QFormLayout(self)

self.boletaS = QLineEdit(self)

LB = QLabel('Boleta', self)

getUser.setLayout(getUser.layout)

snd = QPushButton("Aceptar")

snd.clicked.connect(self.lookHorario)

rtrn = QPushButton("Regresar")

rtrn.clicked.connect(self.mainAlumno)

getUser.layout.addRow(LB, self.boletaS)

getUser.layout.addRow(snd, rtrn)

self.layout.addWidget(getUser)

self.setLayout(self.layout)

def lookHorario(self):

if self.boletaS.text() != '':

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = (HOST, PORT)

alumno = {

"op": 'horario',

"calOf": self.boletaS.text()

}

try:

# Send data

sock.sendto(pickle.dumps(alumno), server\_address)

data, server = sock.recvfrom(4096)

try:

self.grades = pickle.loads(data)

self.showGrades()

except Exception as e:

print(e)

if data.decode() == 'No\_user':

QMessageBox.question(self, 'Error', "Boleta no encontrada", QMessageBox.Ok, QMessageBox.Ok)

except:

print('closing socket')

sock.close()

#QMessageBox.question(self, 'Exito', "El alumno ha sido inscrito", QMessageBox.Ok, QMessageBox.Ok)

self.mainAlumno()

else:

QMessageBox.question(self, 'Error', "Faltan campos", QMessageBox.Ok, QMessageBox.Ok)

def lookGrades(self):

if self.boletaS.text() != '':

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = (HOST, PORT)

alumno = {

"op": 'grades',

"calOf": self.boletaS.text()

}

try:

# Send data

sock.sendto(pickle.dumps(alumno), server\_address)

data, server = sock.recvfrom(4096)

try:

self.grades = pickle.loads(data)

self.showGrades()

except Exception as e:

print(e)

if data.decode() == 'No\_user':

QMessageBox.question(self, 'Error', "Boleta no encontrada", QMessageBox.Ok, QMessageBox.Ok)

except:

print('closing socket')

sock.close()

#QMessageBox.question(self, 'Exito', "El alumno ha sido inscrito", QMessageBox.Ok, QMessageBox.Ok)

self.mainAlumno()

else:

QMessageBox.question(self, 'Error', "Faltan campos", QMessageBox.Ok, QMessageBox.Ok)

def showGrades(self):

grades = self.grades

print('grades in:', grades)

self.clean()

self.parent().resize(262,len(grades)\*62)

gradetab = QWidget()

gradetab.layout = QVBoxLayout(self)

rtrn = QPushButton("Regresar")

rtrn.clicked.connect(self.mainAlumno)

tableWidget = QTableWidget()

tableWidget.setRowCount(len(grades))

tableWidget.setColumnCount(2)

auxCount = 0

for key in grades:

tableWidget.setItem(auxCount,0, QTableWidgetItem(key))

tableWidget.setItem(auxCount,1, QTableWidgetItem(grades[key]))

auxCount += 1

#field, value = grades.items()[0]

#print(field, value)

gradetab.layout.addWidget(tableWidget)

gradetab.layout.addWidget(rtrn)

gradetab.setLayout(gradetab.layout)

self.layout.addWidget(gradetab)

self.setLayout(self.layout)

def clean(self):

try:

for i in reversed(range(self.layout.count())):

self.layout.itemAt(i).widget().setParent(None)

except:

pass

def sendInfo(self):

if self.boleta.text() != '' and self.nombre.text() != '' and self.ApeMate.text() != '' and self.ApePate.text() != '' and self.group != '' and self.foto\_path != '':

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = (HOST, PORT)

alumno = {

"op":'sign',

"boleta": self.boleta.text(),

"name": self.nombre.text(),

"ap": self.ApePate.text(),

"am": self.ApeMate.text(),

"group":self.group,

"foto": os.path.basename(self.foto\_path)

}

try:

# Send data

print('sending', alumno)

sock.sendto(pickle.dumps(alumno), server\_address)

# Receive response

print('waiting to receive')

data, server = sock.recvfrom(4096)

print('received {!r}'.format(data))

if data.decode() == 'end':

f = open(self.foto\_path, 'rb')

chonk = f.read(4096)

while chonk:

sock.sendto(chonk, server\_address)

data, server = sock.recvfrom(4096)

#print('received {!r}'.format(data))

chonk = f.read(4096)

f.close()

finally:

print('closing socket')

sock.close()

QMessageBox.question(self, 'Exito', "El alumno ha sido inscrito", QMessageBox.Ok, QMessageBox.Ok)

self.mainAlumno()

else:

QMessageBox.question(self, 'Error', "Faltan campos", QMessageBox.Ok, QMessageBox.Ok)

def getFoto(self):

fnme = QFileDialog.getOpenFileName(self, 'Open file', 'c:\\',"Image files (\*.jpg \*.gif)")

self.LF.setText(os.path.basename(fnme[0]))

self.foto\_path = fnme[0]

def btnstate(self,b):

self.group = b.text()

class Window(QWidget):

def \_\_init\_\_(self, parent):

super(Window, self).\_\_init\_\_(parent)

self.parent().resize(250,100)

self.layout = QVBoxLayout(self)

self.button\_alumno = QPushButton('Alumno')

self.button\_maestro = QPushButton('Maestro')

self.layout.addWidget(self.button\_alumno)

self.layout.addWidget(self.button\_maestro)

self.button\_alumno.clicked.connect(self.parent().on\_button\_clicked\_alumno)

self.button\_maestro.clicked.connect(self.parent().on\_button\_clicked\_maestro)

self.setLayout(self.layout)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication([])

window = MainWindow()

window.show()

app.exec\_()

**data.JSON**

**{**

**"0001": {**

**"Grupo": "Grupo 1-A",**

**"Materias": {**

**"Fisica 1": "10",**

**"Mate 1": "10",**

**"Quimica 1": "10"**

**}**

**},**

**"alumnos": [**

**{**

**"am": "Bishop",**

**"ap": "Varo",**

**"boleta": "0001",**

**"foto": "FB\_IMG\_1544209290263.jpg",**

**"group": "Grupo 1-A",**

**"name": "Andrew"**

**}**

**]**

**}**

Conclusiones

Para esta practica se utilizo un archivo JSON como método para almacenar datos, los almacenamos como objetos que tienen atributos. Esto nos permite evitar el uso de una base de datos y facilita el intercambio de información del datagram socket.