# ЛАБОРАТОРНА РАБОТА № 5 Дослідження роботи з датаграмними сокетами в режимі опиту

## **Мета роботи:** вивчити роботу з датаграмними сокетами в режимі опиту. Створити сервер котрий буде відповідати на запити клієнтів.

* 1. Теоретичні відомості

Датаграмні сокети використовуються в програмах не так часто, оскільки надійність дуже низька. Корисні вони тоді, коли потрібно постійно передавати звук і відео по мережі. Оскільки для датаграмних сокетів не потрібно встановлювати з’єднання, використовувати їх значно простіше.

Реалізація сервера:

1) підготувати бібліотеку до використання;

2) створити об’єкт типу Socket;

3) зв’язати цей об’єкт з локальною адресою/портом;

4) прийняти/передати дані;

5) закрити сокети, звільнити ресурси.

Реалізація клієнта:

1) підготувати бібліотеку до використання;

2) створити об’єкт типу Socket;

3) прийняти/передати дані;

4) закрити сокети, звільнити ресурси.

Створивши сокет за допомогою *socket* та *bind*, ви вже можете використовувати його для відправки і прийому повідомлень за допомогою функцій *sendto* та *recvfrom*.

Для відправлення повідомлення використовують функцію *int* ***sendto*** *(SOCKET s, const char FAR \* buf, int len,int flags, const struct sockaddr FAR \* to, int tolen)*.

Вона майже така як і *send*, але має 2 додаткових параметри:

*- to* – вказуємо структуру *sockaddr*, в котрій є адреса призначення;

*- tolen –* розмір структури.

Для прийому даних використовують функцію *int* ***recvfrom*** *(SOCKET s, char FAR\* buf, int len, int flags, struct sockaddr FAR\* from, int FAR\* fromlen )*.

Відповідно і тут є 2 додаткових параметри, один із них структура *sockaddr,* в котру ми запишемо адресу/порт відправника (на цю ж адресу і будемо відповідати в зворотному направці) і саме розмір цієї структури.

Доречі, можна використовувати і функції *send* та *recv*, зробивши *connect*, але майте на увазі, що ніякого з’єднання не відбувається, ми просто запам’ятовуємо адресу.

**Хід роботи**

Вихідний код сервера та клієнта що працюють на протоколі UDP, зображено в лістингу 1 та лістингу 2 відповідно.

Лістинг 1 – Файл датаграмного серверу

import socket  
udp\_server = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
udp\_server.bind(("127.0.0.1", 5555))  
  
def start():  
 while True:  
 data, adress\_user = udp\_server.recvfrom(4096)  
 number = int(data)  
 for i in range(number):  
 i = str(i).encode()  
 udp\_server.sendto(i, adress\_user)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 start()

Лістинг 2 – Файл датаграмного клієнта

import socket  
import threading  
  
host = "127.0.0.1"  
port = 5555  
addr = (host, port)  
client = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
  
def listen\_server():  
 while True:  
 try:  
 data, ser = client.recvfrom(4096)  
 print(data)  
 except:  
 pass  
  
def send\_data():  
 thread\_for\_listen = threading.Thread(target=listen\_server)  
 thread\_for\_listen.start()  
 times = input()  
 if(int(times) > 10 or int(times) < 0):  
 print("You shoud write in range 0 to 10")  
 times = input()  
  
 client.sendto(str(times).encode("utf-8"), addr)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 send\_data()

Результат виконання програми зображено на рисунку 1. Сервер надсилає ту кількість повідомлень яку вказав клієнт, в данному випадку – 9.

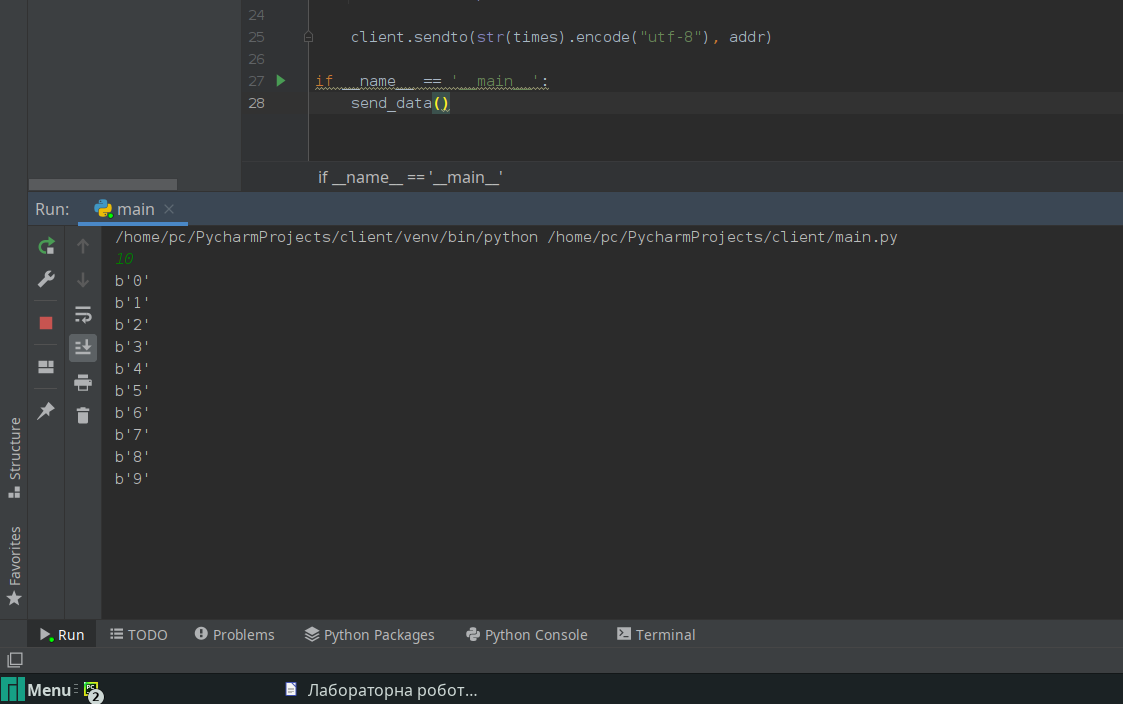


Рисунок 1 – Відповідь клієнту від сервера

Висновки:вивчив роботу з датаграмними сокетами в режимі опиту. Створив сервер котрий буде відповідати на запити клієнтів.