НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Інститут прикладного системного аналізу
Кафедра системного проектування

ЗВІТ

з виконання лабораторної роботи №1
з дисципліни “Комп’ютерні мережі”

На тему “Організація обміну даних у моделі клієнт-сервер за допомогою протоколів UDP і TCP”

Виконав:

Студент групи ДА-82

Муравльов А.Д.

Варіант №18 (51)

Київ — 2021

Зміст

Мета роботи

- Ознайомлення з роботою мережевих транспортних протоколів TCP та UDP;
- Ознайомлення з сокетами і елементами API сокетів;
- Написання програмних модулів клієнт та сервер для організації обміну даними за протоколами TCP і UDP;

Теоретичні відомості

TCP (SOCK_STREAM) – це протокол, заснований на з'єднанні. Зв'язок і обидві сторони ведуть розмову до тих пір, поки з'єднання не буде перервано однією зі сторін або через мережеві помилки.

UDP (SOCK_DGRAM) – це протокол на основі дейтаграм. Дейтаграма — блок інформації, посланий як пакет мережевого рівня через передавальне середовище без попереднього встановлення з'єднання і створення віртуального каналу. Дейтаграма є одиницею інформації у протоколі для обміну інформацією на мережевому і транспортному рівнях еталонної моделі OSI. Ви відправляєте одну дейтаграмму і отримуєте одну відповідь, а потім з'єднання скасовується.
Якщо ви відправите кілька пакетів, TCP обіцяє доставити їх по порядку. UDP цього не робить, тому одержувач повинен перевірити їх.
Якщо пакет TCP втрачений, відправник може повідомити про це. У UDP це не так.
UDP дейтаграми обмежені за розміром, це близько 512 байт.
TCP може посилати набагато більші за розміром дані.
TCP трохи більш надійний і робить більше перевірок.
UDP – більш легкий за вагою (менше навантаження на комп'ютер і мережу).

Завдання

1) Ознайомитись з теоретичними відомостями.

2) Вибрати варіант завдання за номером в списку груп ДА-81 (1-33), ДА-82 (34-65). Завдання полягає у тому, щоб переслати N чисел типу Тип1 і M чисел типу Тип2

№ Варіанта	N	Тип1	M	Тип2
51	6	int	8	char

3) Написати програму для передачі даних між сервером та клієнтом TCP

4) Написати програму для пересилання даних між сервером і клієнтом UDP

5) Перевірити її роботу, запам’ятати скріншоти результатів

Лістинг програм

1. TCP

1.1. Server

```
import time
import socket
import pickle
import sys

HOST = "127.0.0.1"
PORT = 8080

print("Starting server")

SOCKET = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
SOCKET.bind((HOST, PORT))
SOCKET.listen()
CONNECTION, ADDRESS = SOCKET.accept()

print(f"Connected {ADDRESS} to {SOCKET}")
```

```
TIME = time.time()
while True:
    DATA = CONNECTION.recv(1024)
    if not DATA:
        break

    DATA_DECODED = pickle.loads(DATA)
    print(f"Got data\n\tData
size:  {sys.getsizeof(DATA_DECODED)}\n\tData:  {DATA_DECODED}\n\tData
type:  {type(DATA_DECODED)}")
    print(f"{time.time() - TIME}s since server started")

time.sleep(10)
CONNECTION.close()
```

1.2. Client

```
import time
import socket
import pickle
import random
import string

HOST = "127.0.0.1"
PORT = 8080

intArr = []
charArr = []

for i in range(0,7):
    if(i<5):
        intArr.append(random.randrange(0,100))
        charArr.append((random.choice(string.ascii_letters)))
    else:
        charArr.append((random.choice(string.ascii_letters)))

dataToSend = intArr + charArr
```

```
SOCKET = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
SOCKET.connect((HOST,PORT))

for item in dataToSend:
    SOCKET.send(pickle.dumps(item))
    print(f"{item} – sent")
    time.sleep(0.5)

time.sleep(1)
SOCKET.close()
```

2. UDP

2.1. Server

```
import time
import socket
import pickle
import sys

HOST = "127.0.0.1"
PORT = 4200

SOCKET = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
SOCKET.bind((HOST, PORT))
print("UDP server started and listens")

TIME = time.time()
while True:
    DATA, ADDRESS = SOCKET.recvfrom(1024)
    DATA_DECODED = pickle.loads(DATA)
    print(f"Got data from {ADDRESS}.\n\tData
size:  {sys.getsizeof(DATA_DECODED)}\n\tData:  {DATA_DECODED}\n\tData
type:  {type(DATA_DECODED)}")
    print(f"{time.time() - TIME}s since server started")
    if not DATA:
        break
```

2.2. Client

```
import time
import socket
import pickle
import random
import string

HOST = "127.0.0.1"
PORT = 4200

intArr = []
charArr = []

for i in range(0,7):
    if(i<5):
        intArr.append(random.randrange(0,100))
        charArr.append((random.choice(string.ascii_letters)))
    else:
        charArr.append((random.choice(string.ascii_letters)))

dataToSend = intArr + charArr

SOCKET = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

for item in dataToSend:
    SOCKET.sendto(pickle.dumps(item), (HOST, PORT))
    print(f"{item} – sent")
    time.sleep(0.5)

time.sleep(1)
SOCKET.close()
```

Весь лістинг коду також викладено у систему контролю версій Github за посиланням: https://github.com/lakub-muravlov/fourth-course-projects/tree/main/Networks/Lab1

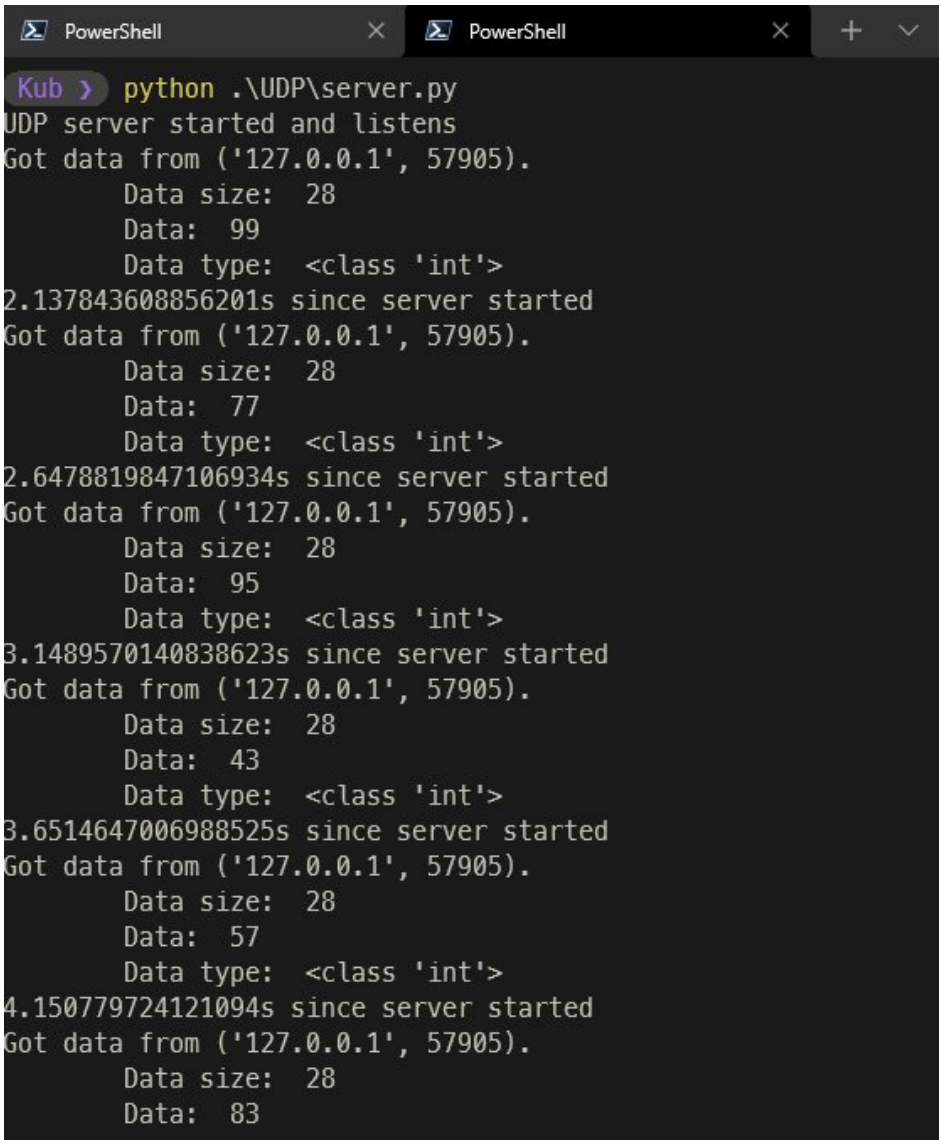
Результат роботи програми

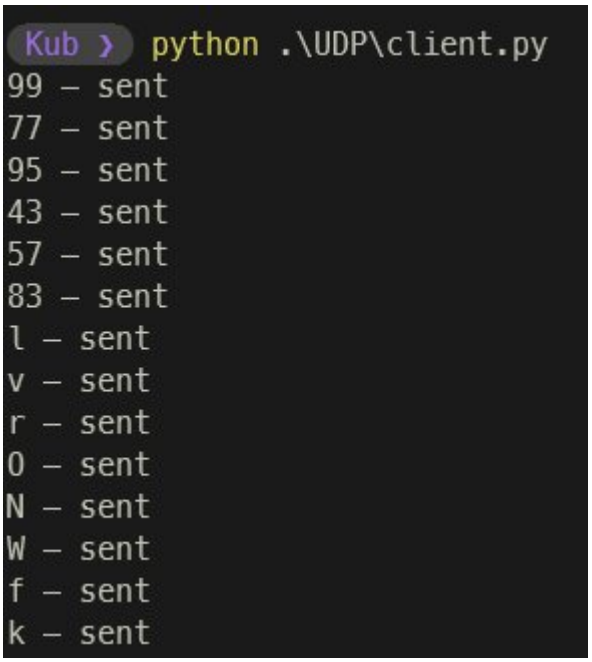
TCP

```
Starting server
Connected ('127.0.0.1', 49743) to <socket.socket fd=144, family=Addre
o=0, laddr=('127.0.0.1', 8080)>
Got data
        Data size:  28
        Data:  47
        Data type:  <class 'int'>
0.0010025501251220703s since server started
Got data
        Data size:  28
        Data:  39
        Data type:  <class 'int'>
0.5013997554779053s since server started
Got data
        Data size:  28
        Data:  85
        Data type:  <class 'int'>
1.0151100158691406s since server started
Got data
        Data size:  28
        Data:  84
        Data type:  <class 'int'>
1.5294280052185059s since server started
Got data
        Data size:  28
        Data:  78
        Data type:  <class 'int'>
2.0319221019744873s since server started
Got data
```

```
Kub ❯ python .\TCP\client.py
47 – sent
39 – sent
85 – sent
84 – sent
78 – sent
28 – sent
e – sent
k – sent
c – sent
Z – sent
w – sent
c – sent
t – sent
A – sent
```

UDP


PowerShell
PowerShell
Kub ❯ python .\UDP\server.py
UDP server started and listens
Got data from ('127.0.0.1', 57905).
Data size: 28
Data: 99
Data type: <class 'int'>
2.137843608856201s since server started
Got data from ('127.0.0.1', 57905).
Data size: 28
Data: 77
Data type: <class 'int'>
2.6478819847106934s since server started
Got data from ('127.0.0.1', 57905).
Data size: 28
Data: 95
Data type: <class 'int'>
3.1489570140838623s since server started
Got data from ('127.0.0.1', 57905).
Data size: 28
Data: 43
Data type: <class 'int'>
3.6514647006988525s since server started
Got data from ('127.0.0.1', 57905).
Data size: 28
Data: 57
Data type: <class 'int'>
4.150779724121094s since server started
Got data from ('127.0.0.1', 57905).
Data size: 28
Data: 83


Kub ❯ python .\UDP\client.py
99 – sent
77 – sent
95 – sent
43 – sent
57 – sent
83 – sent
l – sent
v – sent
r – sent
O – sent
N – sent
W – sent
f – sent
k – sent

Висновки

У даній лабораторній роботи ми ознайомились з роботою мережевих транспортних протоколів TCP і UDP. Transmission Control Protocol, TCP – протокол, призначений для управління передачею даних у комп'ютерних мережах, працює на транспортному рівні моделі OSI.

На відміну від іншого поширеного протоколу транспортного рівня UDP, TCP забезпечує надійне доправляння даних від хоста-відправника до хоста-отримувача, для цього встановлюється логічний зв'язок між хостами. Таким чином TCP належить до класу протоколів зі встановленим з'єднанням. User Datagram Protocol, UDP — один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання. UDP — це один з найпростіших протоколів транспортного рівня моделі OSI, котрий виконує обмін повідомленнями без підтвердження та гарантії доставки. При використанні протоколу UDP відповідальність за обробку помилок і повторну передачу даних покладена на протокол рівнем вище. Але попри всі недоліки, протокол UDP є ефективним для серверів, що надсилають невеликі відповіді великій кількості клієнтів.

Також ознайомились із сокетами і елементами API сокетів та написали програмні модулі клієнт і сервер для організації обміну даними за протоколами TCP і UDP.