МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №3

з дисципліни «Системи з розподіленими даними»

**Виконав:**

Студент 5-го курсу

ТЕФ, гр. ТР-02мп

Круглий Д.В.

**Перевірив:**

Варава І.А.

**Київ-2021**

**Тема:** розробка додатка на основі технології P2P.

**Мета:** навчитись розробляти додатки за допомогою технології P2P.

**Теоретичні відомості**

P2P-технологія - технологія побудови мереж розподілених рівноправних узлів за принципом децентралізації. Одноранговою називають мережу, в якій усі комп'ютери рівноправні, кожен із них має можливість використовувати певні ресурси інших підключених до мережі комп'ютерів або пристроїв і надавати їм доступ до своїх ресурсів.

Переваги P2P:

* Зручність – програма-клієнт самостійно сканує мережу і визначає адреси комп'ютерів
* Швидкість - клієнти з'єднуються напряму, завдяки цьому досягається більша швидкість у порівнянні з ftp
* Надійність - перевіряється контрольна сума файла для гарантування його цілісності
* Доступність - Direct Connect – це простий спосіб поділитися інформацією із іншими користувачами мережі
* Комфорт - можна керувати швидкістю скачування.
* Економічність – абсолютно безплатний доступ до великої кількості різноманітних даних
* Універсальність – пропонуються клієнти з різними можливостями і для різних операційних систем.

**Представлення коду програми**

***Client.py***

from twisted.internet import reactor, protocol

from twisted.internet.protocol import ReconnectingClientFactory as ClFactory

from twisted.internet.endpoints import TCP4ClientEndpoint

from sys import stderr

import json

class Client(protocol.Protocol):

    def \_\_init\_\_(self):

        print("Created")

        reactor.callInThread(self.message\_input)

    @staticmethod

    def \_\_encode\_json(\*\*kwargs):

        return json.dumps(kwargs)

    def send\_message(self, \*\*kwargs):

        self.transport.write(self.\_\_encode\_json(\*\*kwargs).encode("utf-8"))

    def message\_input(self):

        while True:

            self.send\_message(value=input("value:"), type=input("type:"))

    def dataReceived(self, data):

        try:

            data = json.loads(data.decode("utf-8"))

        except UnicodeDecodeError or json.JSONDecodeError:

            print("Something went wrong :(", file=stderr)

            return

        if data['type'] == 'error':

            print(data.get('value', "Unknown error"), file=stderr)

        else:

            print(data.get('value', "No value in the message"))

class ClientFactory(ClFactory):

    def clientConnectionLost(self, connector, unused\_reason):

        self.retry(connector)

    def clientConnectionFailed(self, connector, reason):

        print(reason)

        self.retry(connector)

    def buildProtocol(self, addr):

        return Client()

if \_\_name\_\_ == '\_\_main\_\_':

    endpoint = TCP4ClientEndpoint(reactor, 'localhost', 12345)

    endpoint.connect(ClientFactory())

***server.py***

from twisted.internet import reactor, protocol

from twisted.internet.protocol import ServerFactory as ServFactory, connectionDone

from twisted.internet.endpoints import TCP4ServerEndpoint

import json

# value, type

class Server(protocol.Protocol):

    def \_\_init\_\_(self, clients: dict, my\_id):

        self.my\_id = my\_id

        self.clients = clients

        self.another\_client = None

    def connectionMade(self):

        self.clients[self.my\_id] = self

    @staticmethod

    def \_\_encode\_json(\*\*kwargs):

        return json.dumps(kwargs)

    def send\_message(self, \*\*kwargs):

        if kwargs.get('where'):

            where = kwargs['where']

            del kwargs['where']

            where.transport.write(self.\_\_encode\_json(\*\*kwargs).encode("utf-8"))

        else:

            self.transport.write(self.\_\_encode\_json(\*\*kwargs).encode("utf-8"))

    def dataReceived(self, data):

        try:

            data = json.loads(data.decode("utf-8"))

        except UnicodeDecodeError:

            self.send\_message(value="Cannot decode, use utf-8", type='error')

            return

        except json.JSONDecodeError:

            self.send\_message(value="Cannot decode, use json", type='error')

            return

        if not data.get('type') or not data.get('value'):

            self.send\_message(value=f"Wrong data", type='error')

            return

        if data['type'] == "user\_choose":

            try:

                another\_client = int(data['value'])

                if another\_client in self.clients.keys():

                    self.another\_client = another\_client

                else:

                    raise KeyError

            except ValueError:

                self.send\_message(value="Write another id as int", type='error')

            except KeyError:

                self.send\_message(value="Can't find that client", type='error')

            else:

                self.send\_message(value=f"Talk to {self.another\_client}", type='user\_chosen')

        elif data['type'] == "new\_message":

            if not self.another\_client:

                self.send\_message(value=f"Don't have a client to send your message to", type='error')

            try:

                self.send\_message(value=data['value'], where=self.clients[self.another\_client], type='new\_message')

            except KeyError:

                self.send\_message(value="Something wrong happened, try another client", type='error')

                self.another\_client = None

    def connectionLost(self, reason=connectionDone):

        self.disconnect()

    def disconnect(self):

        del self.clients[self.my\_id]

class ServerFactory(ServFactory):

    def \_\_init\_\_(self):

        self.clients = {}

        self.last\_id = 0

    def buildProtocol(self, addr):

        self.last\_id += 1

        return Server(self.clients, self.last\_id)

if \_\_name\_\_ == '\_\_main\_\_':

    endpoint = TCP4ServerEndpoint(reactor, 12345)

    endpoint.listen(ServerFactory())

    reactor.run()

**Приклад роботи програми**

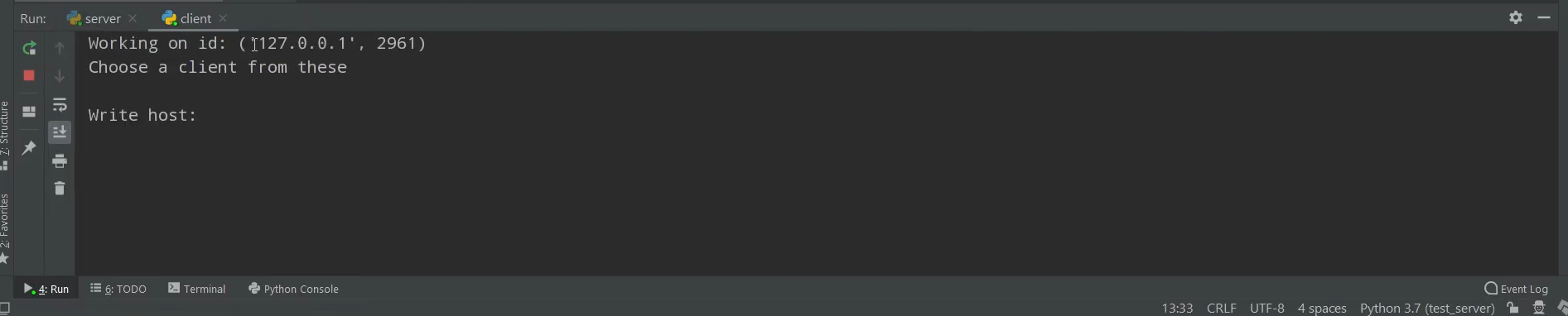


Рис. 1 – Запуск сервера та першого клієнта з портом 2961

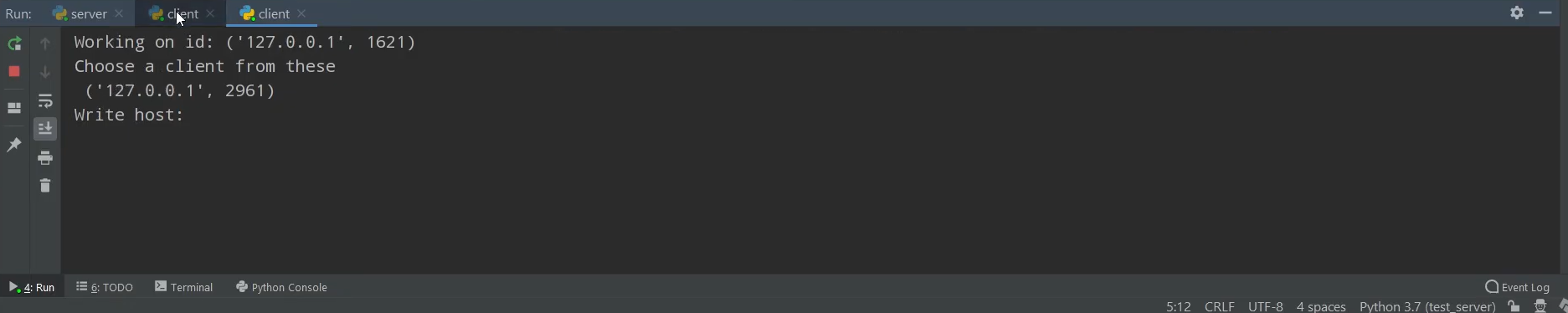
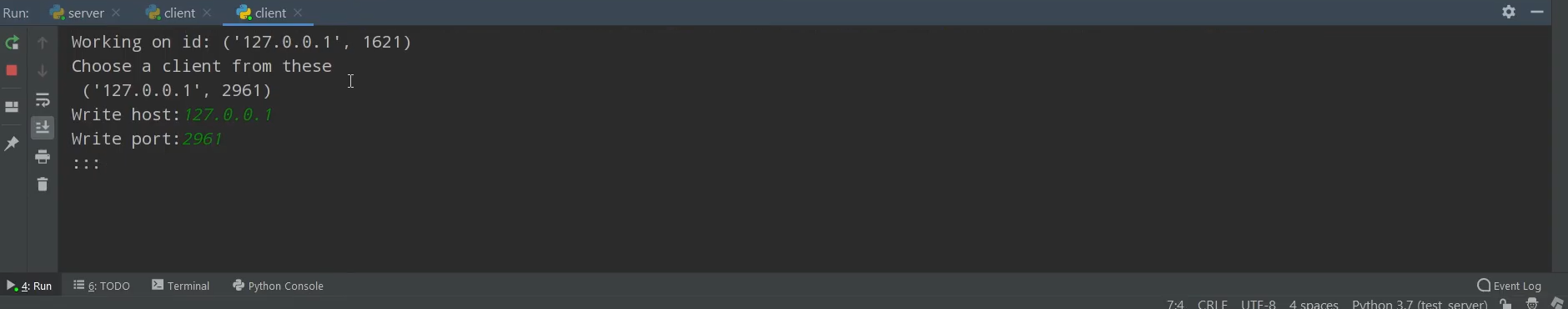


Рис. 2 – Створили ще одного клієнта з портом 1621. Ми бачимо, що нам доступний для підключення клієнт, створений на попередньому кроці.



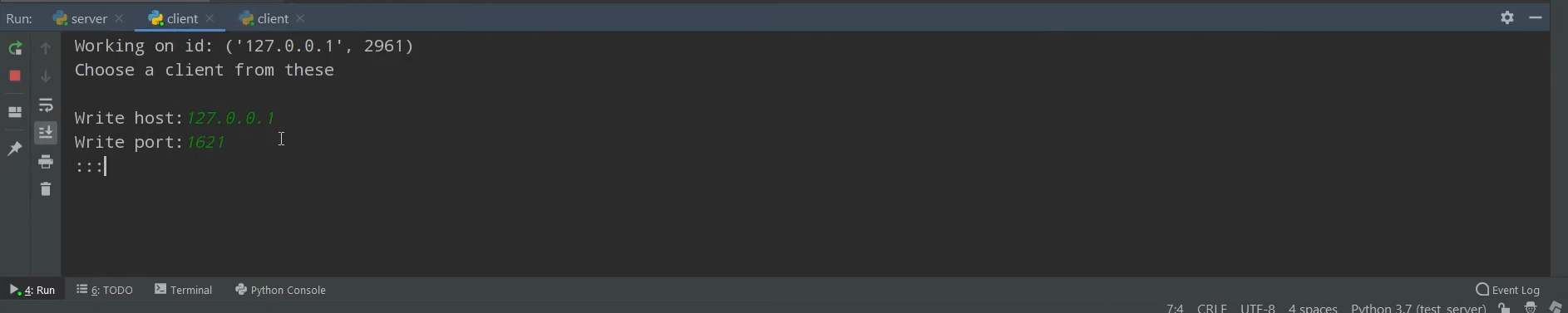
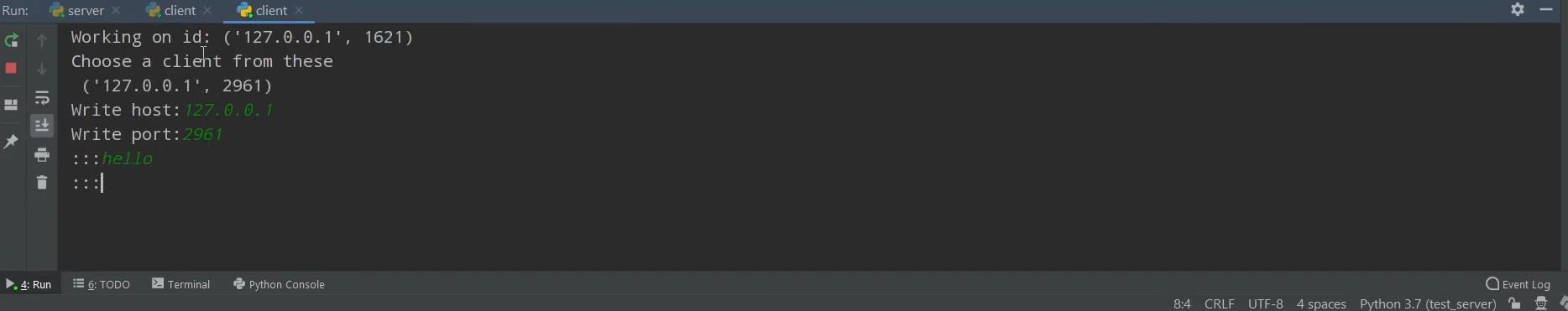


Рис. 3 – З’єднання клієнтів між собою без участі сервера



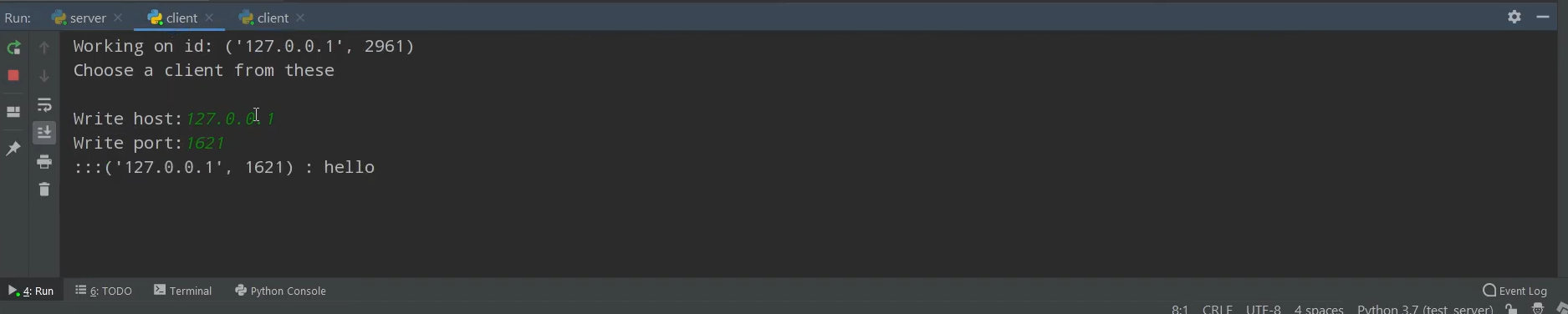
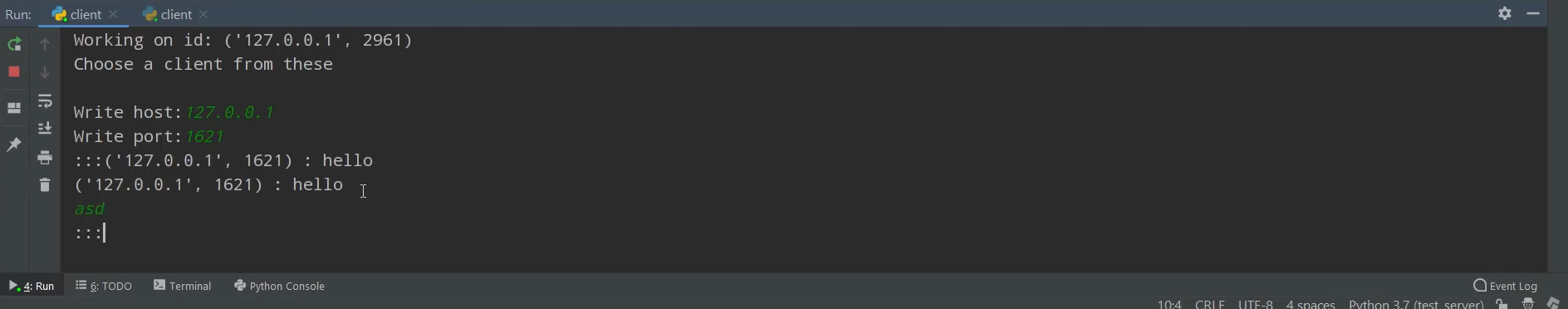


Рис. 4 – Надсилання повідомлення від другого клієнта до першого



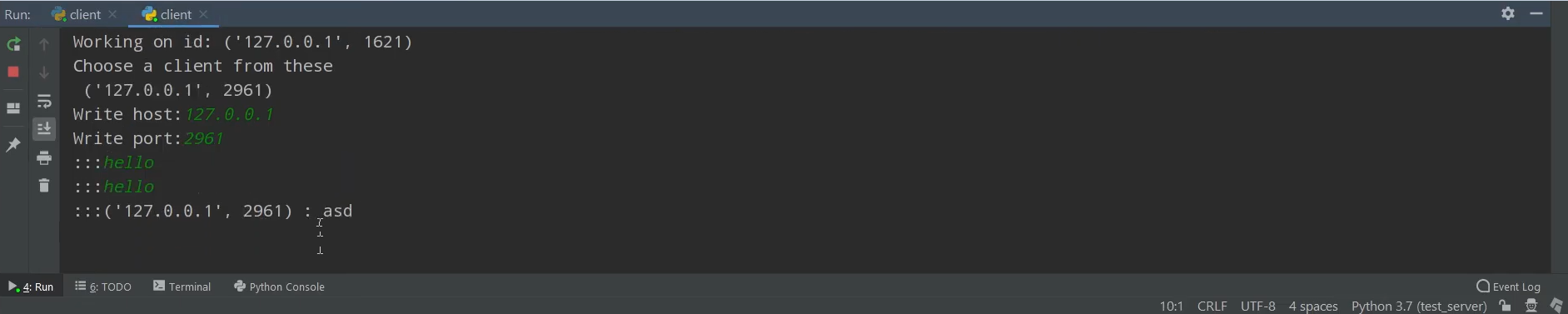


Рис. 5 – Вимкнення сервера та надсилання повідомлення від першого клієнта до другого для підтвердження того, що клієнти продовжують взаємодіяти без участі сервера

**Висновок:** у ході виконання лабораторної роботи, я більш детально ознайомився з технологією P2P, а також навчився писати додатки з використанням даної технології.