Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

**Лабораторна робота № 2**

по дисципліні «Програмування компютерних та віртуальних мереж інформаційно-комунікаційних систем»

Тема: «Знайомство з TCP sockets»

|  |  |
| --- | --- |
| Виконав:  студенти групи ІТ-91мн  Давиденко І. В.  Свириденко О. А.  Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_\_\_\_\_ | Перевірено:  ас. Галушко Д. О. |

Київ 2020

Завдання

1. Пригадати API для роботи з TCP-сокетами.

2. Реалізувати наступне клієнт-серверний застосунок:

Сервер:

- Очікує запити на з’єднання від клієнтів (номер порту задається під час запуску застосунку як параметр командного рядка);

- Встановлює та підтримує з’єднання з клієнтами, розсилає інформацію усім клієнтам про те, що до чату приєднався новий клієнт;

- У разі надходження повідомлення від одного з клієнтів надсилає це повідомлення усім клієнтам;

- У разі від’єднання одного з клієнтів розсилає про це інформацію решті клієнтів;

- Має бути реалізований як консольний застосунок;

- Сервер має бути реалізований як багатопоточний застосунок «1 клієнт = 1 потік» використовуючи блокуючий ввід-вивід.

Клієнт:

- Клієнт має з’єднуватись з сервером (адреса сервера, номер порту та ім’я користувача задаються як параметри командного рядка);

- Клієнт має бути реалізований як багатопоточний застосунок. Один потік має очікувати введення тексту користувачем і відправляти його на сервер, а інший потік – виводити на екран повідомлення, які клієнт отримує від сервера;

- Клієнт може бути реалізований як консольний або десктопний застосунок.

Контрольні питання

1. Як співвідносяться модель ISO OSI та стек протоколів TCP/IP?

2. Чим TCP відрізняється від UDP?

3. Що таке порт? Що таке сокет?

4. В чому полягає різниця між Socket та ServerSocket?

5. Що таке блокуючий та неблокуючий ввід-вивід даних?

6. Що таке клас Thread та інтерфейс Runnable? Як ними користуватись для створення багатопоточних застосунків?

Короткі теоретичні відомості

Transmission Control Protocol, TCP (укр. Протоко́л керува́ння переда́чею) — разом із протоколом IP є стрижневим протоколом Інтернету, який дав назву моделі TCP/IP. 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.

Інкапсуляція TCP в моделі OSI

TCP отримує потоки даних від протоколів верхніх рівнів OSI-моделі, початковим джерелом яких є протоколи прикладного рівня, такі як HTTP, FTP та інші. Кожний протокол верхнього рівня має свій визначений TCP-порт.

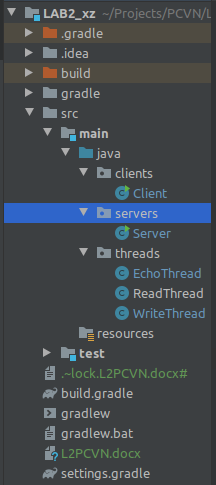
TCP розбиває конкретний потік даних на порції, та додає до кожної з них заголовок з номером послідовності. Отримані таким чином порції даних традиційно називаються TCP-сегментами. Далі кожний сегмент інкапсулюється в IP-пакет і передається через IP-протокол до хоста-отримувача.

Після надходження IP-пакету до хоста-отримувача перевіряється коректність отриманих даних у TCP-сегменті, методом перерахування контрольної суми, та переконується, що попередні сегменти даних також були успішно отримані. Після чого хост-отримувач надсилає запит до хоста-відправника про нову, або повторну передачу порції даних, що одночасно є підтвердженням того, що всі сегменти з номерами послідовності, меншими ніж номер нового запиту, були успішно отримані.

У свою чергу TCP-сегменти деінкапсулюються з IP-пакетів, розміщуються в правильному порядку та з них вилучаються TCP-заголовки. Отриманий таким чином потік даних передається до того протоколу верхнього рівня, з якого первісно надійшли дані на стороні хоста-відправника.

Хід роботи

Створю проект “LAB2”, котрий має наступну структуру:



Файл Client.java:

public class Client {

private String hostname;

private int port;

public Client(String hostname, int port) {

this.hostname = hostname;

this.port = port;

}

public void execute() {

try {

Socket socket = new Socket(hostname, port);

System.out.println("Connected to the chat server");

new ReadThread(socket, this).start();

new WriteThread(socket, this).start();

} catch (UnknownHostException ex) {

System.out.println("Server not found: " + ex.getMessage());

} catch (IOException ex) {

System.out.println("I/O Error: " + ex.getMessage());

}

}

public static void main(String[] args) {

String hostname;

int port;

if (args.length < 2) {

hostname = "localhost";

port = 1917;

} else {

hostname = args[0];

port = Integer.parseInt(args[1]);

}

Client client = new Client(hostname, port);

client.execute();

}

}

Файл Server.java:

public class Server {

protected int PORT;

protected Set<EchoThread> clients = new HashSet<>();

public Server(int PORT) {

this.PORT = PORT;

}

public void run() {

ServerSocket serverSocket = null;

Socket socket = null;

try {

serverSocket = new ServerSocket(PORT);

} catch (IOException e) {

e.printStackTrace();

}

while (true) {

try {

socket = serverSocket.accept();

System.out.println("New user connected");

EchoThread thread = new EchoThread(socket, this);

this.clients.add(thread);

thread.start();

} catch (IOException e) {

System.out.println("I/O error: " + e);

}

}

}

public void broadcast(String message, EchoThread excludeUser) {

for (EchoThread client : this.clients) {

if (client != excludeUser) {

client.write(message);

}

}

}

public void disconnected(EchoThread user) {

clients.remove(user);

System.out.println("The user " + user.getUuid().toString() + " quitted");

}

public static void main(String[] args) {

int port = 1917;

if (args.length >= 1) {

port = Integer.parseInt(args[0]);

}

Server server = new Server(port);

server.run();

}

}

Файл EchoThread.java

public class EchoThread extends Thread {

/\*\*

\* UUID socket

\*/

protected UUID uuid;

/\*\*

\* List subscribed topics

\*/

protected Server server;

/\*\*

\* Client socket

\*/

protected Socket socket;

/\*\*

\* List clients

\*/

protected List<EchoThread> clients;

/\*\*

\* XZ

\*/

protected PrintWriter writer;

public UUID getUuid() {

return uuid;

}

public EchoThread(Socket clientSocket, Server server) {

this.socket = clientSocket;

this.uuid = UUID.randomUUID();

this.server = server;

}

public void run() {

InputStream inp = null;

BufferedReader brinp = null;

try {

inp = socket.getInputStream();

brinp = new BufferedReader(new InputStreamReader(inp));

OutputStream output = socket.getOutputStream();

writer = new PrintWriter(output, true);

} catch (IOException e) {

e.printStackTrace();

}

String serverMessage = "New user connected: " + this.uuid;

server.broadcast(serverMessage, this);

String line;

try {

do {

line = brinp.readLine();

serverMessage = "[" + this.uuid + "]: " + line;

server.broadcast(serverMessage, this);

} while (!line.equalsIgnoreCase("bye"));

socket.close();

server.disconnected(this);

serverMessage = this.uuid + " has quitted.";

server.broadcast(serverMessage, this);

} catch (IOException e) {

e.printStackTrace();

}

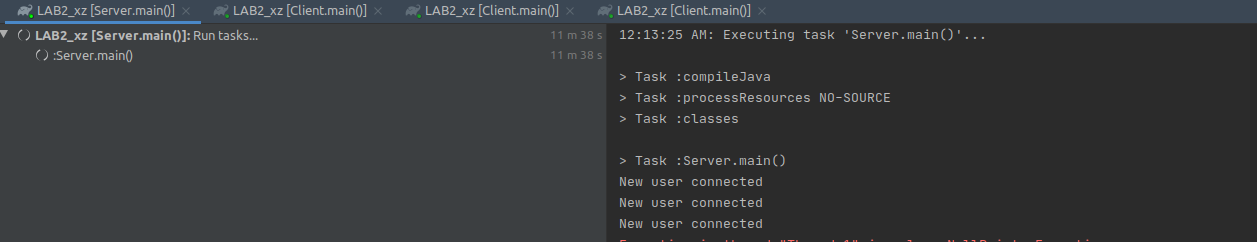
}

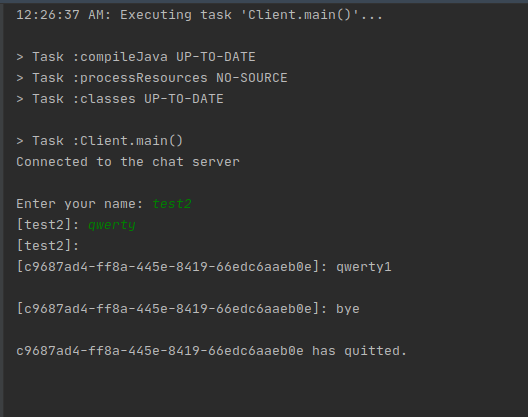
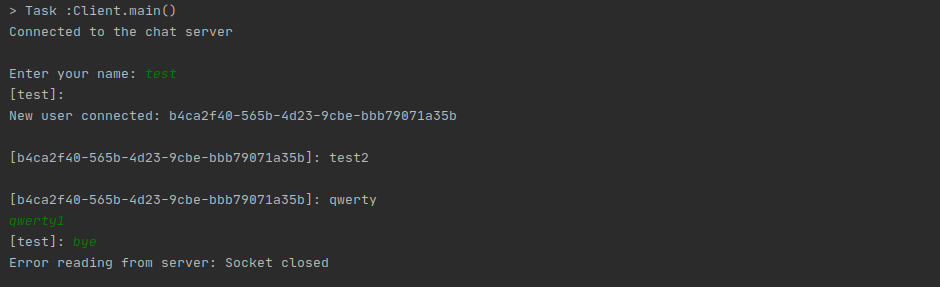
public void write(String message) {

writer.println(message);

}

В результаті виконання програми отримаю наступне:





Висновок: Під час лабораторної роботи я навчився створювати проект котрий за допомогою протоколу TCP/IP здійснює відправку повідомлень в реальному часі. Також створив програму котра отримує повідомлення та виводить їх на екран.