Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання практичної роботи № 3**

**з дисципліни «Програмне забезпечення мережевих технологій»**

**на тему**

**«Створення клієнт-серверних програм за допомогою бібліотек Java»**

**Лектор:**

професор кафедри ПЗ

Мельник Р. А.

**Виконав:**

студент групи ПІ-31

Губ’як Р. І.

**Прийняв:**

старший викладач кафедри ПЗ

Гасько Р. Т.

«\_\_\_» \_\_\_\_\_\_\_\_ 2017 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2017

**ТЕМА РОБОТИ**

Створення клієнт-серверних програм за допомогою бібліотек Java

**МЕТА РОБОТИ**

Навчитись концепціям програмування мовою JAVA, а також засвоїти основні стандартні класи і методи бібліотек java.net, java.io

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Сортування великого масиву на кількох комп’ютерах:

1. Налаштувати прослуховування UDP-порту для читання вхідних повідомлень.
2. Налаштувати прослуховування TCP-порту для встановлення вхідних з’єднань (сервер).
3. Підключитись до серверної частини програми з допомогою TCP-сокета.
4. Переслати дані на опрацювання від сервера до клієнта.
5. Відправити серверу результат роботи клієнта.

**ТЕКСТ ПРОГРАМИ**

**//--------------------------------------------Налаштовування прослуховування UDP-порту**

public class SearchListener extends Thread{

public int Port;

private DatagramSocket socket;

public ArrayList<OnFoundListener> onFoundListeners = new ArrayList<OnFoundListener>();

public SearchListener(int port){

Port = port;

try {

socket = new DatagramSocket(Port);

}

catch (Exception e){

}

this.start();

}

private void OnFound(InetAddress address, int port){

for (OnFoundListener listener : onFoundListeners){

listener.OnFound(address, port);

}

}

public void run(){

byte[] data = new byte[1024];

while (true) {

DatagramPacket receivePacket = new DatagramPacket(data, data.length);

try {

socket.receive(receivePacket);

}

catch(Exception e){

continue;

}

InetAddress address = receivePacket.getAddress();

String message = new String(receivePacket.getData());

message = message.substring(0, message.indexOf(0));

Gson gson = new Gson();

SearchMessage searchMessage = gson.fromJson(message, SearchMessage.class);

OnFound(address, searchMessage.Port);

}

}

}

**//--------------------------------------------------Встановлення з’єднання з сервером, отримання від нього даних, опрацювання і відавка результатів назад до сервера**

public class Worker extends Thread {

public InetAddress Address;

private boolean WaitForResponse;

private int[] MyArray;

public ArrayList<WorkerListener> WorkerListeners = new ArrayList<WorkerListener>();

private ServerSocket WorkerSocket;

public Worker(InetAddress address){

Address = address;

WaitForResponse = true;

}

public void DoWork(){

this.start();

}

public void run(){

WorkerSocket = null;

try {

WorkerSocket = new ServerSocket(2727);

} catch (Exception e) {

}

while (WaitForResponse) {

Socket socket = null;

WorkMessage work = null;

try {

socket = WorkerSocket.accept();

work = new Gson().fromJson(new BufferedReader(new InputStreamReader(socket.getInputStream())).readLine(), WorkMessage.class);

socket.close();

} catch (Exception e) {

continue;

}

if (!work.IsSorted){

MyArray = work.Array;

Sort();

}

else if (work.Array.length == 0){

SendToMergeClient(work.Address);

}

else{

Merge(work.Array);

}

}

}

private void Sort(){

Sorter sorter = new Sorter(MyArray);

sorter.SortProgressListeners.add(new SortProgressListener() {

public void ProgressChanged(int progress) {

new Thread(() -> {

ResultMessage resultMessage = new ResultMessage();

resultMessage.Array = new int[0];

resultMessage.Progress = progress;

try {

Socket socket = new Socket(Address, 2727);

socket.getOutputStream().write(new Gson().toJson(resultMessage).getBytes());

socket.close();

} catch (Exception e) {

}

}).start();

}

});

sorter.Sort();

}

private void SendReadyMerge(){

new Thread(() -> {

ResultMessage resultMessage = new ResultMessage();

resultMessage.Array = new int[0];

resultMessage.Progress = 100;

try {

byte[] data = new Gson().toJson(resultMessage).getBytes();

Socket socket = new Socket(Address, 2727);

socket.getOutputStream().write(data, 0, data.length);

socket.close();

} catch (Exception e) {

}

}).start();

}

private void Merge(int[] array){

MyArray = Merger.Merge(MyArray, array);

SendReadyMerge();

}

private void SendToMergeClient(InetAddress address){

new Thread(() -> {

WorkMessage workMessage = new WorkMessage();

workMessage.Array = MyArray;

workMessage.IsSorted = true;

try {

byte[] data = new Gson().toJson(workMessage).getBytes();

Socket socket = new Socket(address.isLoopbackAddress() ? Address : address, 2727);

socket.getOutputStream().write(data, 0, data.length);

socket.close();

} catch (Exception e) {

return;

}

WaitForResponse = false;

OnFinish();

}).start();

}

}

**ПРОТОКОЛ ВИКОНАННЯ ПРОГРАМИ**

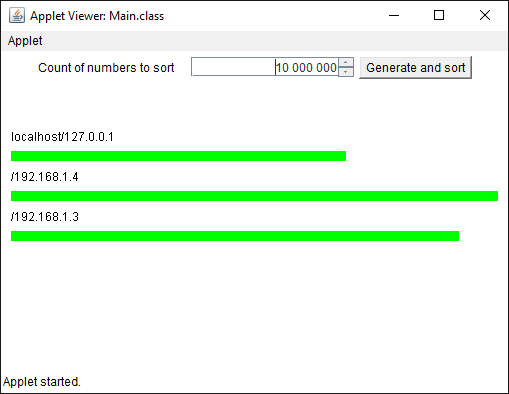


Рис. 1. Прогрес обробки даних на різних комп’ютерах-клієнтах

**ВИСНОВОК**

Під час виконання цієї лабораторної роботи я вивчив концепції сокетів у програмуванні мовою Java, а також засвоїв основні стандартні класи і методи бібліотек java.net, java.io. Створив програму, яка складається з двох частин: клієнта і сервера. Їхня комунікація відбувається з допомогою протоколів TCP та UDP. Результатом роботи системи є відсортований масив чисел.