**ЛАБОРАТОРНАЯ РАБОТА №4**

# РАЗРАБОТКА ПРИЛОЖЕНИЙ АРХИТЕКТУРЫ «КЛИЕНТ-СЕРВЕР», ВЗАИМОДЕЙСТВУЮЩИХ ПО ПРОТОКОЛУ UDP

Цель: Научиться создавать приложения архитектуры «клиент-сервер», взаимодействующих по протоколу UDP.

**UDP** (*User Datagram Protocol* — протокол пользовательских датаграмм) — один из ключевых элементов TCP/IP, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в UDP называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.Этот протокол более простой, основанный на сообщениях без установления соединения. Протоколы такого типа не устанавливают выделенного соединения между двумя хостами. Связь достигается путем передачи информации в одном направлении от источника к получателю без проверки готовности или состояния получателя.

Основные свойства UDP:

* *Ненадёжный* — когда сообщение посылается, неизвестно, достигнет ли оно своего назначения — оно может потеряться по пути. Нет таких понятий, как подтверждение, повторная передача, тайм-аут.
* *Неупорядоченность* — если два сообщения отправлены одному получателю, то порядок их достижения цели не может быть предугадан.
* *Легковесность* — никакого упорядочивания сообщений, никакого отслеживания соединений и т. д. Это небольшой транспортный уровень, разработанный на IP.
* *Датаграммы* — пакеты посылаются по отдельности и проверяются на целостность только если они прибыли. Пакеты имеют определенные границы, которые соблюдаются после получения, то есть операция чтения на сокете-получателе выдаст сообщение таким, каким оно было изначально послано.
* *Нет контроля перегрузок* — UDP сам по себе не избегает перегрузок. Для приложений с большой пропускной способностью возможно вызвать коллапс перегрузок, если только они не реализуют меры контроля на прикладном уровне.

**Дейтаграммы**

Дейтаграммы, или пакеты протокола UDP — это пакеты информации, пересылаемые в сети по принципу “fire-and-forget” (выстрелил и забыл). Если вам надо добиться оптимальной производительности, и вы в состоянии минимизировать затраты на проверку целостности информации, пакеты UDP могут оказаться весьма полезными.

UDP не предусматривает проверок и подтверждений при передаче информации. При передаче пакета UDP по какому-либо адресу нет никакой гарантии того, что он будет принят, и даже того, что по этому адресу вообще есть кому принимать такие пакеты. Аналогично, когда вы получаете дейтаграмму, у вас нет никаких гарантий, что она не была повреждена в пути или что ее отправитель все еще ждет от вас подтверждения ее получения.

Java реализует дейтаграммы на базе протокола UDP, используя для этого два класса. Объекты класса **DatagramPacket** представляют собой контейнеры с данными, a **DatagramSocket** — это механизм, используемый при передаче и получении объектов DatagramPacket.

**ОрганизацияUDP**

Необходимые средства поддержки протокола *UDP* находятся в пакете *java.net*. Для создания дейтаграмм в *Java* существует класс *DatagramPacket*. При получении дейтаграммы по протоколу *UDP* класс *DatagramPacket* используется также для чтения данных, адреса отправителя и метаинформации.

Для дейтаграмм не нужно создавать канал, данные посылаются приложению с использованием адреса, состоящего из сокета и номера порта (в дейтаграммах не гарантируются доставка и корректность последовательности передачи пакетов). Для передачи дейтаграмм не нужны ни механизмы подтверждения связи, ни механизмы управления потоком данных.

Чтобы создать дейтаграмму для отправки на удаленную машину, используется следующий конструктор:

*publicDatagramPacket(byte[] ibuf, int length, InetAddressiaddr, intiport);*

Где:

*ibuf* − массив байт, содержащий кодированное сообщение;

*length*− количество байт, которое должно быть помещено в пакет, что определяет размер дейтаграммы;

*iaddr*− это экземпляр класса *InetAddress*, который хранит *IP*-адрес получателя;

*iport* указывает номер порта, на который посылается дейтаграмма.

Чтобы получить дейтаграмму, необходимо использовать другой конструктор для объекта *DatagramPacket*, в котором будут находиться принятые данные. Прототип конструктора имеет вид.

*publicDatagramPacket(byte[] ibuf, intlength);*

Где:

*ibuf* − массив байт, куда должны быть скопированы данные из дейтаграммы;

*length* − количество байт, которое должно быть скопировано.

Дейтаграммы не ограничены определенной длиной; можно создавать как очень длинные, так и очень короткие дейтаграммы. Заметим, однако, что между клиентом и сервером должно существовать соглашение о длине дейтаграмм, поскольку они оба должны создать массив байт нужного размера перед созданием объекта *DatagramPacket* для посылки или получения дейтаграммы.

Когда дейтаграмма получена, как будет продемонстрировано ниже, можно прочитать ее данные. Другие методы позволяют получить метаинформацию, относящуюся к сообщению:

* *publicintgetLength()*; – возвращает количество байт, из которых состоят данные дейтаграммы;
* *publicbyte[] getData()*; – позволяет получить массив, содержащий эти данные;
* *publicInetAddressgetAddress()*; – возвращает адрес отправителя;
* *publicintgetPort()*; – возвращает номер порта *UDP*, используемый отправителем.

Отправка и получение дейтаграмм осуществляется при помощи класса *DatagramSocket*, который создает сокет *UDP*. У него есть два конструктора, один из которых позволяет системе назначить любой из свободных портов *UDP*. Другой дает возможность задать конкретный порт, что полезно при разработке сервера. Как и для портов *ТСР*, в большинстве операционных систем порты с номерами меньше 1024 доступны только процессам с привилегиями суперпользователя.

*publicDatagramSocket() throws SocketException;*

*publicDatagramSocket(int port) throws SocketException;*

Сокет, созданный первым конструктором, можно использовать для отправки правильно адресованных дейтаграмм при помощи следующего метода класса *DatagramSocket*:

*public void send(DatagramPacket p) throws IOException;*

Если *DatagramSocket*создан вторым конструктором, можно получить дейтаграмму:

*publicsyncronized void receive(DatagramPacket p) throws IOException;*

Заметим, что метод *receive()* блокируется до момента получения дейтаграммы. Поскольку *UDP* является ненадежным протоколом, нельзя быть уверенным, что возврат из *receive()* вообще произойдет.

## **Задания для самостоятельного выполнения**

Разработать приложение на основе *UDP*-соединения, позволяющее осуществлять взаимодействие клиента и сервера по совместному решению задач обработки информации. Приложение должно располагать возможностью передачи и модифицирования получаемых (передаваемых) данных. Возможности клиента: передать серверу исходные параметры (число а, число b и число с вводятся с клавиатуры) для расчета значения функции, а также получить расчетное значение функции. Возможности сервера: по полученным от клиента исходным параметрам считается сумма первого ряда от а до b и сумма второго ряда от b до с. Расчет первой и второй сумм осуществляется в разных потоках.

**Пример:**

**Код реализации**

**Сервер**

packageudpserver;

importjava.net.\*;

importjava.io.\*;

importjava.math.\*;

publicclassUDPServer {

publicvoidrunServer() throws IOException {//методсервераrunServer

DatagramSocket s = null;//созданиеобъектаDatagramSocket

try {

booleanstopFlag = false;//созданиефлагаstopFlagиегоинициализация

//значением false

byte[] buf = newbyte[512];//буфер для приема/передачи дейтаграммы

s = new DatagramSocket(8001);//привязкасокетак

//реальному объекту с портом DEFAULT\_PORT

System.out.println("UDPServer: Started on " + s.getLocalAddress() + ":"

+ s.getLocalPort());//вывод к консоль сообщения

while (!stopFlag) {//цикл до тех пор, пока флаг не примет значение true

DatagramPacketrecvPacket = newDatagramPacket(buf, buf.length);//создание объекта дейтаграммы для получения данных

s.receive(recvPacket);//помещение полученного содержимого в

//объект дейтаграммы

Stringcmd = newString(recvPacket.getData());//извлечение

String[] str; //создание дополнительного массива строк

str = cmd.split("\\D"); //разделение полученной строки на подстроки

for (int i = 0; i <str.length; i++) {

System.out.println("Серверпринялчисло [" + i + "]" + str[i]);

}

Integer a = Integer.valueOf(str[0]); //присвоение каждому числу своей подстроки

Integer b = Integer.valueOf(str[1]);

Integer c = Integer.valueOf(str[2]);

//произведение рассчета

Double sum1 = 0.0, sum2 = 0.0, res;

for (int n = a; n <= b; n++) {

sum1 += Math.pow(n, n - 1);

}

System.out.println("Перваясумма: " + sum1);

for (int n = b; n <= c; n++) {

double d=(4 \* n) + 1;

sum2 += 3/d;

}

System.out.println("Втораясумма: " + sum2);

res = sum1 - sum2;

System.out.println("Результат: " + res);

String str1 = Double.toString(res);

byte[] buf1 = str1.getBytes(); // преобразуем строку в массив байтов для отправки

DatagramPacketsendPacket = newDatagramPacket(buf1, buf1.length, recvPacket.getAddress(), recvPacket.getPort());//формирование объекта дейтаграммы для отсылки данных

s.send(sendPacket);//послать сами данные

stopFlag = true;//остановкасервера

} // while(server is not stopped)

System.out.println("UDPServer: Stopped");

} finally {

if (s != null) {

s.close();//закрытиесокетасервера

}

}

}

public static void main(String[] args) {//метод main

try {

UDPServerudpSvr = new UDPServer();//созданиеобъектаudpSvr

udpSvr.runServer();//вызовметодаобъектаrunServer

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

**Клиент**

packageudpclient;

import java.net.\*;

import java.io.\*;

importjava.util.Scanner;

publicclassUDPClient {//описание класса клиента

public void runClient() throws IOException {//методклиентаrunClient

DatagramSocket s = null;//созданиедейтаграммы

try {

Scanner sc = new Scanner(System.in);

int a = 0, b = 0, c = 0;

System.out.print("Введитецелоечисло a: ");

if (sc.hasNextInt()) { // возвращает истинну если с потока ввода можно считать целое число

a = sc.nextInt(); // считывает целое число с потока ввода и сохраняем в переменную

} else {

System.out.println("Вы ввели не целое число");

}

System.out.print("Введите целое число b: ");

if (sc.hasNextInt()) { // возвращает истинну если с потока ввода можно считать целое число

b = sc.nextInt(); // считывает целое число с потока ввода и сохраняем в переменную

} else {

System.out.println("Вы ввели не целое число");

}

System.out.print("Введите целое число c: ");

if (sc.hasNextInt()) { // возвращает истинну если с потока ввода можно считать целое число

c = sc.nextInt(); // считывает целое число с потока ввода и сохраняем в переменную

} else {

System.out.println("Вы ввели не целое число");

}

Stringstr=(a + "|" + b + "|" + c); //склеиваем все числа в одну строку

byte[] buf = newbyte[512]; //буфер для приема/передачи дейтаграммы

s = newDatagramSocket();//привязка сокета к реальному объету

System.out.println("UDPClient: Started");

byte[] verCmd = str.getBytes(); // преобразуем строку в массив байтов для отправки

DatagramPacketsendPacket = new DatagramPacket(verCmd, verCmd.length, InetAddress.getByName("127.0.0.1"), 8001);//создание

//дейтаграммы для отсылки данных

s.send(sendPacket);//посылка дейтаграммы

DatagramPacketrecvPacket = newDatagramPacket(buf, buf.length);//создание дейтаграммы для получения данных

s.receive(recvPacket);//получениедейтаграммы

String result = new String(recvPacket.getData()).trim();//извлечение

//данных (версии сервера)

System.out.println("Полученный результат: " + result);

System.out.println("UDPClient: Ended");

} finally {

if (s != null) {

s.close();//закрытиесокетаклиента

}

}

}

public static void main(String[] args) {//метод main

try {

UDPClient client = new UDPClient();//созданиеобъекта client

client.runClient();//вызовметодаобъекта client

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

## 

Варианты индивидуальных заданий:

Соответственно для каждого варианта рассчитанный результат (при a = 2, b=4, c=6):

**Контрольные вопросы:**

1. Что такое UDP?
2. Для чего используется протокол соединения UDP?
3. Какие отличия UDP от TCP/IP?
4. Что такое дейтаграмм?
5. Гарантирует ли протокол соединения UDP доставку данных?
6. Какие действия необходимо предпринять для обмена данными по UDP протоколу?
7. Методы для работы с дейтаграммами?

**Литература:**

Web-ресурсы:

1) http://www.softzenware.com/java/14.htm

2) http://www.codeguru.com.ua/article/a-339.html

3) http://www.cyberguru.ru/networks/protocols/udp.html

4) http://citforum.ru/nets/comer/gl11.shtml

Печатные издания:

1) Книга «Изучаем Java» Берт Бейтс, «ЭКСМО»,2012г.