МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«11» ноября 2017г.

ОТПРАВКА И ПРИЕМ СООБЩЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПРОТОКОЛОВ UDP И TCP/IP

Методические указания по выполнению практических работ по дисциплине «Проектирование защищенных телекоммуникационных систем» для студентов специальности 10.05.02

Курск 2017

УДК 004.056.55

Составители: А.Л. Марухленко

Рецензент
Кандидат технических наук, доцент А.Г. Спеваков

Отправка и прием сообщений с использованием протоколов UDP и TCP/IP: методические указания к выполнению практических работ / Юго-Зап. гос. ун-т; сост. А. Л. Марухленко Курск, 2017. - 14с.

Содержат сведения по вопросам установки и настройки среды разработки Ecplise. Указывается порядок выполнения практической работы, правила оформления, содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Проектирование защищенных телекоммуникационных систем» соответствуют требованиям программы, утвержденной учебно-методическим объединением и предназначены для студентов направления подготовки 10.05.02.

Текст печатается в авторской редакции

Подписано в печать 01.11.2017. Формат 60х84 1/16.
Усл.печ. л. 0,8. Уч.-изд.л. 0,7. Тираж 30 экз. Заказ _____. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Разработать клиент/серверное приложение, в котором сервер может распространять сообщения всем клиентам, зарегистрированным в группе 233.0.0.1, порт 1502. Пользователь сервера должен иметь возможность ввода и отправки текстовых сообщений, а пользователь-клиент просматривает полученные сообщения.

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для решения поставленной задачи необходимо выполнить следующие шаги:

1. Создать новый проект.
2. Реализовать класс сервера для ввода и отправки сообщений.
3. Реализовать класс клиента для получения и просмотра сообщений.
4. Протестировать приложение – запустить сервер и клиент, и отправить сообщение.

2.1 Подготовительный этап

Для реализации проекта необходимо установить и настроить среду разработки Ecplise (см. п. «Установка и настройка программного обеспечения»).

2.2 Создание нового проекта

1) Выберите пункт меню File/New/Project, в окне выбора типа проекта укажите other/Java Project и нажмите Next.
2) Укажите имя проекта Lab2 и нажмите Finish.

2.3 Создание класса Server

Класс `Server` предназначен для отсылки сообщений всем клиентам, зарегистрированным в группе 233.0.0.1. Создание класса Server включает в себя следующие основные задачи:

1. Создание сокета с помощью класса `DatagramSocket`. Сокет сервера выполняет задачу отправки сообщения.
2. Создание объекта `InetAddress`, представляющего адрес сервера. Адреса для групповой (multicast) передачи сообщений выбираются из диапазона 224.0.0.0 - 239.255.255.255. В нашем приложении будет указан адрес 233.0.0.1.
3. Организация ввода строки сообщения с клавиатуры и создание объекта packet класса `DatagramPacket`, который хранит введенные данные и использует метод `send()` объекта класса `DatagramSocket`, для отсылки пакета всем клиентам группы.

1) Для создания класса сервера щелкните правой кнопкой мыши на каталог src в окне Package Explorer и выберите New/Class

2) В появившемся окне в качестве имени пакета (Package) укажите `ru.tpu.javaEELabs.lab2`, а в качестве имени класса (Name) задайте Server. Нажмите Finish.

Код класса `Server` приведен ниже:

```
package ru.tpu.javaEELabs.lab2;

import java.io.*;
import java.net.*;

public class Server {

private BufferedReader in = null;
private String str = null;
private byte[] buffer;
private DatagramPacket packet;
private InetAddress address;
private DatagramSocket socket;

public Server() throws IOException {
      System.out.println("Sending messages");

// Создается объект DatagramSocket, чтобы
// принимать запросы клиента
      socket = new DatagramSocket();

// Вызов метода transmit(), чтобы передавать сообщение всем
// клиентам, зарегистрированным в группе
      transmit();
}

public void transmit() {
      try {
      // создается входной поток, чтобы принимать
      // данные с консоли
            in = new BufferedReader(new InputStreamReader(System.in));
            while (true) {
                  System.out.println(
                        "Введите строку для передачи клиентам: ");
                  str = in.readLine();
                  buffer = str.getBytes();
                  address = InetAddress.getByName("233.0.0.1");
                  // Посылка пакета датаграмм на порт номер 1502
                  packet = new DatagramPacket(
                        buffer,
                        buffer.length,
                        address,
                        1502);

                  //Посылка сообщений всем клиентам в группе
                  socket.send(packet);
            }
      } catch (Exception e) {
            e.printStackTrace();
      } finally {
```

```
            try {
                  // Закрытие потока и сокета
                  in.close();
                  socket.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}

public static void main(String arg[]) throws Exception {
      // Запуск сервера
      new Server();
}}
```

2.4 Создание класса Client

Класс `Client` позволяет клиенту присоединиться к группе 233.0.0.1 для получения сообщений от сервера. Создание класса `Client` включает в себя следующие основные задачи:

1. Создание сокета для просмотра групповых сообщений с помощью класса `MulticastSocket`. Сокет клиента выполняет задачу приема сообщения.
2. Создание объекта `InetAddress`, представляющего адрес сервера и присоединение к группе этого сервера с помощью метода сокета `joinGroup`.
3. Организация чтения пакетов датаграмм (DatagramPacket) из сокета и отображение полученных данных на экране.

1) Для создания класса клиента щелкните правой кнопкой мыши на пакет `ru.tpu.javaEELabs.lab2` в каталоге src окна Package Explorer и выберите New/Class.

2) В появившемся окне в качестве имени класса (Name) задайте Client. Нажмите Finish.

Код класса `Client` приведен ниже:

```
package ru.tpu.javaEELabs.lab2;

import java.net.*;

public class Client {
      private static InetAddress address;
      private static byte[] buffer;
      private static DatagramPacket packet;
      private static String str;
      private static MulticastSocket socket;

      public static void main(String arg[]) throws Exception {
            System.out.println("Ожидание сообщения от сервера");
            try {

                  // Создание объекта MulticastSocket, чтобы получать
                  // данные от группы, используя номер порта 1502
                  socket = new MulticastSocket(1502);
```

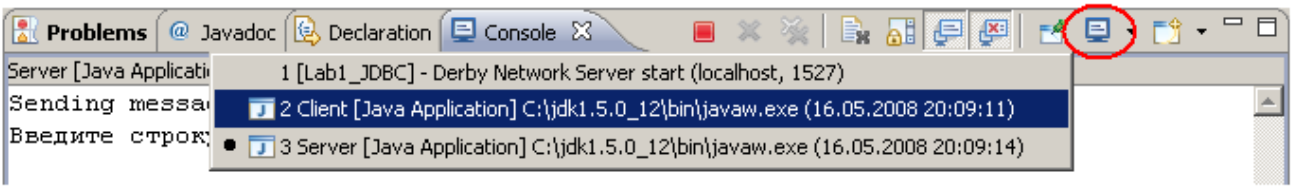
```
            address = InetAddress.getByName("233.0.0.1");

            // Регистрация клиента в группе
            socket.joinGroup(address);
            while (true) {
                  buffer = new byte[256];
                  packet = new DatagramPacket(
                        buffer, buffer.length);
                  // Получение данных от сервера
                  socket.receive(packet);
                  str = new String(packet.getData());
                  System.out.println(
                        "Получено сообщение: " + str.trim());
            }
      } catch (Exception e) {
            e.printStackTrace();
      } finally {
            try {
                  // Удаление клиента из группы
                  socket.leaveGroup(address);

                  // Закрытие сокета
                  socket.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }}}}
```

2.5 Запуск и тестирование

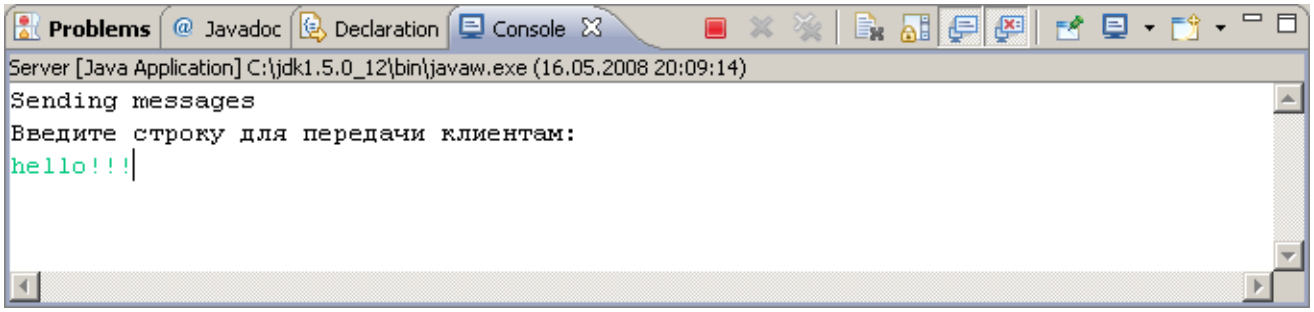
Каждый из построенных классов `Client` и `Server` содержит метод `main()` и является, по сути, отдельным приложением, которое может быть запущено на отдельной машине, подключенной к сети, при этом по умолчанию область видимости передачи групповых сообщений (multicasting scope) ограничивается подсетью сервера. В нашем случае роль клиента и сервера будет выполнять один и тот же компьютер.

1) Щелкните правой кнопкой мыши на класс Client в окне Package Explorer и выберите команду Run As/Java Application. Проделайте то же самое с классом Server.

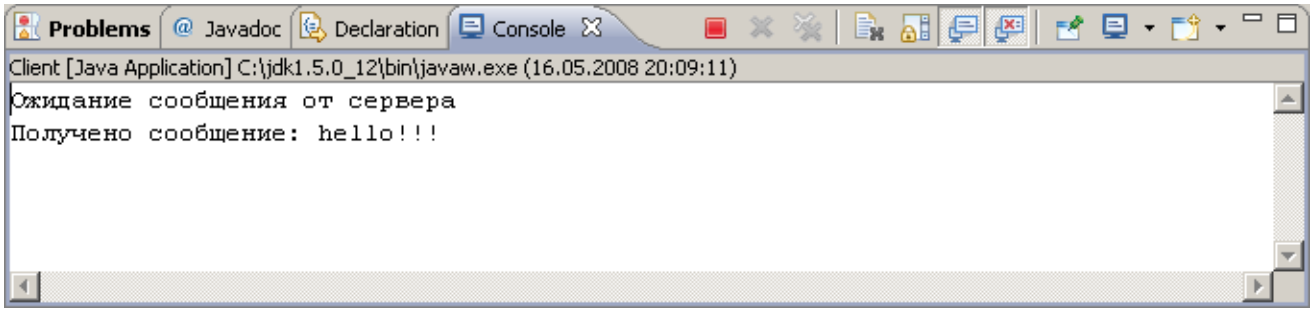
2) В результате будут запущены два приложения, переключаться между которыми можно с помощью кнопки-списка Display Selected Console представления Console:



3) Выберите консоль сервера, введите строку hello и нажмите Enter для подтверждения отправки.



4) Просмотрите консоль клиента и убедитесь, что клиент успешно приял сообщение.



5) Остановка приложения осуществляется с помощью кнопки Terminate представления Console.

3. СОЗДАНИЕ ПРИЛОЖЕНИЯ TCP/IP

3.1 Постановка задачи

Необходимо разработать клиент/серверное приложение, в котором сервер слушает запросы клиентов на порт 1500 и отправляет объект-сообщение содержащий текущую дату/время сервера и строку сообщения. Пользователь-клиент должен иметь возможность просмотра полученного сообщения.

Для решения поставленной задачи необходимо выполнить следующие шаги:

1. Создать класс `DateMessage` с двумя полями: дата и строка – для хранения и передачи сообщения клиенту.
2. Реализовать класс сервера для прослушивания соединений на порту 1500 и отправки сообщений. Задача класса сервера должна выполняться в отдельном потоке.
3. Реализовать класс клиента для получения и просмотра сообщений
4. Протестировать приложение – запустить сервер и клиент, и проверить передачу и получение сообщения.

3.2 Подготовительный этап

Для реализации проекта необходимо установить и настроить среду разработки Ecplise (см. п. «Установка и настройка программного обеспечения»).

3.3 Создание класса DateMessage

1) Создайте новый Java-класс `DateMessage` в пакете `ru.tpu.javaEELabs.lab2`.
2) Скопируйте следующее содержимое класса:

```
package ru.tpu.javaEELabs.lab2;

import java.io.Serializable;
import java.util.Date;

public class DateMessage implements Serializable {


	private Date date;
	private String message;


	public DateMessage(Date date, String message) {
		this.date = date;
		this.message = message;
	}

	public Date getDate() {
		return date;
	}

	public void setDate(Date date) {
		this.date = date;
	}

	public String getMessage() {
		return message;
	}

	public void setMessage(String message) {
		this.message = message;
	}}
```

3.4 Создание класса `ServerTCP`

Создание класса `ServerTCP` включает в себя следующие основные задачи:

1. Создание серверного сокета с помощью класса `ServerSocket`.
2. Ожидание запроса от клиента с помощью метода `accept()` серверного сокета.
3. Формирование объекта-сообщения и отправка его с помощью выходного потока клиентского сокета.

1) Создайте новый Java-класс `ServerTCP` в пакете `ru.tpu.javaEELabs.lab2`.

Код класса `Server` приведен ниже:

```
package ru.tpu.javaEELabs.lab2;

import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Calendar;

/**
 * Класс сервера (выполняется в отдельном процессе)
 */
public class ServerTCP extends Thread {

	// Объявляется ссылка
	// на объект - сокет сервера
	ServerSocket serverSocket = null;

	/**
	 * Конструктор по умолчанию
	 */
	public ServerTCP() {
		try {
			// Создается объект ServerSocket, который получает
			// запросы клиента на порт 1500
			serverSocket = new ServerSocket(1500);
			System.out.println("Starting the server ");
			// Запускаем процесс
			start();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	/**
	 * Запуск процесса
	 */
	public void run() {
		try {
			while (true) {
				// Ожидание запросов соединения от клиентов
				Socket clientSocket = serverSocket.accept();

				System.out.println("Connection accepted from " +
			clientSocket.getInetAddress().getHostAddress());

				// Получение выходного потока,
				// связанного с объектом Socket
				ObjectOutputStream out =
					new ObjectOutputStream(
						clientSocket.getOutputStream());

				// Создание объекта для передачи клиентам
				DateMessage dateMessage = new DateMessage(
						Calendar.getInstance().getTime(),
						"Текущая дата/время на сервере");
				// Запись объекта в выходной поток
				out.writeObject(dateMessage);
				out.close();
			}
		} catch (Exception e) {
```

```
            e.printStackTrace();
      }
}

public static void main(String args[]) {
      // Запуск сервера
      new ServerTCP();
}}
```

3.5 Создание класса `ClientTCP`

Класс `ClientTCP` позволяет клиенту присоединиться к серверу, используя его IP-адрес (в нашем случае localhost) и получить от него сообщение. Создание класса `ClientTCP` включает в себя следующие основные задачи:

1. Создание сокета для доступа к серверу localhost на порт 1500.
2. Получение входного потока сокета.
3. Чтение объекта-сообщения из потока и отображение полученных данных на экране.

1) Создайте новый Java-класс `ClientTCP` в пакете `ru.tpu.javaEELabs.lab2`.

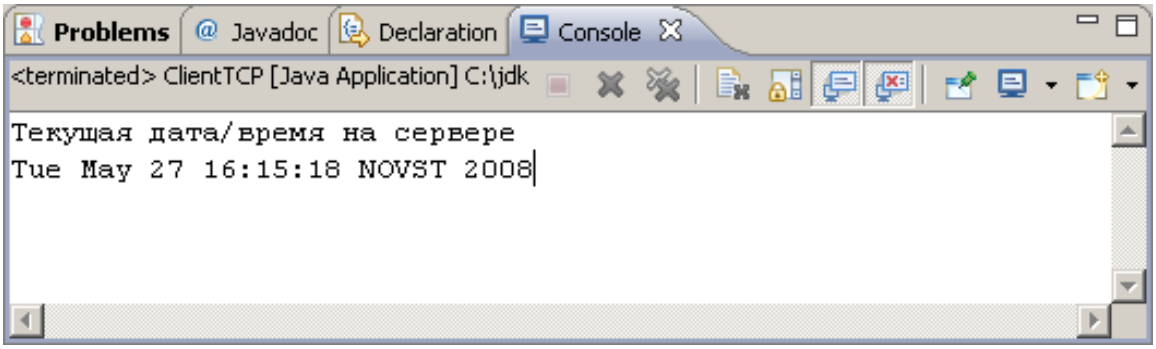
Код класса `Client` приведен ниже:

```
package ru.tpu.javaEELabs.lab2;

import java.io.ObjectInputStream;
import java.net.Socket;

public class ClientTCP {

      public static void main(String args[]) {
            try {
                  // Создается объект Socket
                  // для соединения с сервером
                  Socket clientSocket = new Socket("localhost", 1500);

                  // Получаем ссылку на поток, связанный с сокетом
                  ObjectInputStream in =
                    new ObjectInputStream(clientSocket.getInputStream());

                  // Извлекаем объект из входного потока
                  DateMessage dateMessage =
                        (DateMessage) in.readObject();

                  // Выводим полученные данные в консоль
                  System.out.println(dateMessage.getMessage());
                  System.out.println(dateMessage.getDate());
            } catch (Exception e) {
                  e.printStackTrace();
            }}}
```

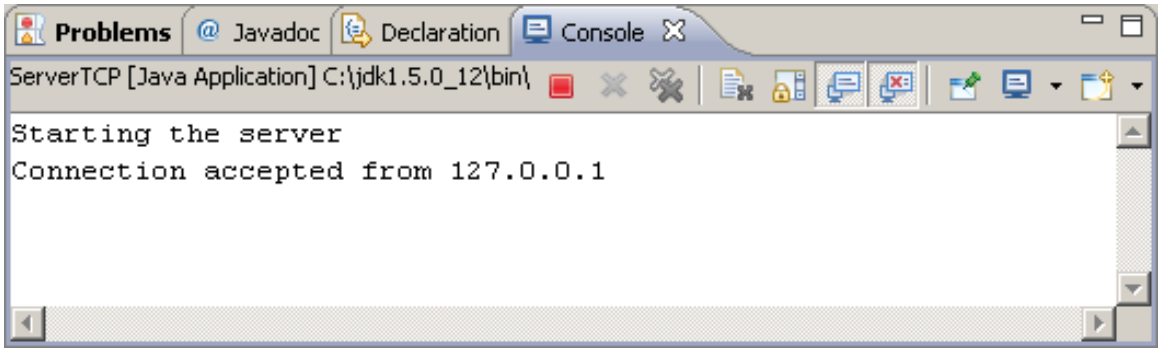
3.6 Запуск и тестирование

Щелкните правой кнопкой мыши на класс ServerTCP в окне Package Explorer и выберите команду Run As/Java Application. В консоли отображается сообщение `Starting the server`.

Проделайте то же самое с классом ClientTCP. При запуске клиент пытается соединиться с сервером и обрабатывает полученное сообщение. В результате в консоли клиента выводится следующее:



Выберите консоль сервера и просмотрите сообщение о приеме соединения от клиента:



4. ВАРИАНТЫ ЗАДАНИЙ

1. Необходимо разработать клиент/серверное приложение, в котором сервер каждые 10 секунд распространяет некоторое текстовое сообщение, например, о погоде, всем *промежуточным* клиентам, зарегистрированным в группе 233.0.0.1, порт 1502 с помощью UDP. Текст сообщения хранится в текстовом файле на сервере. Промежуточный клиент фильтрует полученные сообщения и в случае изменения содержимого отображает его в консоли. Для *конечного* клиента промежуточный клиент выступает сервером. Конечный клиент присоединяется к промежуточному и получает тексты последних пяти отфильтрованных сообщений (с помощью протокола TCP/IP). Необходимо снабдить приложение конечного клиента графическим интерфейсом.

2. Разработать приложение для широковещательного общения пользователей (чат). Клиент отсылает сообщение серверу (с помощью

протокола TCP/IP). Сервер накапливает порции сообщений и каждые 10 секунд распространяет очередную порцию сообщений всем клиентам, зарегистрированным в группе 233.0.0.1, порт 1502 с помощью UDP. Если за указанный период не поступило ни одного нового сообщения, то рассылка не производится. Клиент принимает сообщения и отображает их на экране. Клиентское приложение должно иметь удобный графический интерфейс для ввода новых и просмотра полученных сообщений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Шувалов В.П., Величко В.В., Субботин Е.А., Ярославцев А.Ф. Телекоммуникационные системы и сети. Том 3. Мультисервисные сети (2005)

2) Петраков А.В. Основы практической защиты информации. 2-е изд. Учебн. пособие. – М.: Радио и связь. 2000. – 368 с.

3) Цифровые и аналоговые системы передачи: Учебник для вузов/ В.И.Иванов, В.Н.Гордиенко, Г.Н.Попов и др.; Под ред. В.И.Иванова. – 2-е изд. – М.: Горячая линия – Телеком, 2003. – 232 с.

4) Гольдштейн Б.С., Соколов Н.А., Яновский Г.Г. Сети связи: Учебник для ВУЗов. - СПб.: БХВ-Петербург, 2010. - 400 с.

5) Башарин Г.П. Лекции по математической теории телетрафика: Учеб. пособие. Изд. 3-е, испр. и доп. - М.: РУДН, 2009. - 342 с.

6) Буч Г. Объектно-ориентированный анализ и проектирование. – М.: Вильямс, 2008.

7) Леоненков А.В. Самоучитель языка UML. – СПб.: БХВ-Петербург, 2004.

8) Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов. – М.: ДМК Пресс, 2002.

9) А.В. Росляков. Виртуальные частные сети. Основы построения и применения. - М.: Эко-Трендз, 2006. - 242 с.