

云南大学数学与统计学院

《计算机网络实验》上机实践报告

课程名称：计算机网络实验	年级：2015 级	上机实践成绩：
指导教师：陆正福	姓名：刘鹏	专业：信息与计算科学
上机实践名称：基于 UDP 协议与 Socket 接口的网络通信编程实验	学号：20151910042	上机实践日期：2018-10-26
上机实践编号：No.04	组号：	

一、 实验目的

1. 熟悉基于 UDP 协议与 Socket 接口的网络通信编程实验
2. 熟悉教材第二章的基本概念

二、 实验内容

1. 掌握基于 UDP 协议与 Socket 接口的网络通信编程的流程
2. 使用 Java 实现基于 UDP 协议与 Socket 接口的 PC 网络通信编程
3. 使用 Java 和 Android 实现基于 UDP 协议与 Socket 接口的移动网络通信编程（选做）
4. 使用 Python 实现基于 UDP 协议与 Socket 接口的网络通信编程（选做）

（说明：3 和 4 至少选做 1 项；调试所用实例可以源自书本和网络，但是应有属于自己的修改，不可以照搬照抄）

三、 实验平台

Windows 10 Pro 1803;

Cygwin GCC 编译器。

JDK 11 for Windows;

四、 程序代码

4.1 基于 UDP 协议与 Socket 接口的网络通信编程的流程

UDP，是 User Datagram Protocol，即用户数据报协议。其关键在于“数据包”，主要就是把数据进行打包然后发送给目标，而不管目标是否接收到数据。

主要流程就是发送者打包数据（Datagram Packet），通过 Datagram Socket 发送，接收者收到数据进行拆解。下面简要地进行分析此过程。

4.1.1 服务端接收、发送流程

```

1. 新建存储空间，等待盛放
byte[] buf = new byte[1024]
DatagramPacket packet = new DatagramPacket(buf, buf.length)

2. 创建UDP Socket
DatagramSocket socket = new DatagramSocket(2222)

3. 进程处于阻塞状态，监听端口直到接收到消息
socket.receive(packet)

4. 返回数据（这一步不是必须的）
DatagramPacket p = new DatagramPacket(buf, buf.length, packet.getAddress(), packet.getPort())
socket.send(p)

5. 关闭Socket
socket.close()

```

4.1.2 客户端发送、接受流程

```

1. 根据服务器地址构建数据包
byte[] bytes = et.getText().toString().getBytes()
InetAddress address = InetAddress.getByName("192.168.1.75")
DatagramPacket packet = new DatagramPacket(bytes, bytes.length, address, 2222)

2. 创建Socket，并发送数据报
DatagramSocket socket = new DatagramSocket()
socket.send(packet)

3. 接收数据报（这一步不是必须的）
byte[] bytes1 = new byte[1024]
DatagramPacket receiverPacket = new DatagramPacket(bytes1, bytes1.length)
socket.receive(receiverPacket)

4. 关闭Socket
socket.close()

```

4.2 使用 Java 实现基于 UDP 协议与 Socket 接口的 PC 网络通信编程

UDP 与 TCP 在应用上的不同，主要体现在使用的 Socket 上面。TCP 是创建 Socket，UDP 是创建 DatagramSocket，这也恰恰反映了两者的区别，UDP 基于数据报，TCP 基于数据流。详见 4.1。

4.3 使用 Java 和 Android 实现基于 UDP 协议与 Socket 接口的移动网络通信编程

本选做题继续使用 Java 语言。UDP 协议的实现核心就是构造数据包。

4.3.1 应用设计

服务器开启一个端口号为 2222 的 UDP 进程，然后一直监听来自外部的访问，直到收到一个 packet。这里的 packet 设置为 1MB 大小。手机 App 是客户端，仅仅向服务器发送一个字符串，然后接受数据并将之打印到屏幕上。

4.3.2 实验过程分析

Java 客户端需要连接网络，所以需要赋予其网络权限。

4.3.3 Server 端 Java 代码

```
1  import java.io.IOException;
2  import java.net.DatagramPacket;
3  import java.net.DatagramSocket;
4  import java.net.InetSocketAddress;
5
6  public class UDPServer {
7      public static void main(String[] args) throws IOException {
8
9          // 生成一个 1MB 的存储空间
10         byte[] buf = new byte[1024];
11
12         // 1. 接收数据
13         // (1) 创建接受数据的数据包
14         DatagramPacket packet = new DatagramPacket(buf, buf.length);
15
16         // (2) 创建 UDP 的 Socket
17         DatagramSocket socket = new DatagramSocket(2222);
18
19         // (3) 接收数据
20         System.out.println("服务端开始监听! ~~~~");
21         socket.receive(packet);
22
23         // (4) 处理数据
24         System.out.println("服务端" + new String(buf, 0, buf.length));
25
26         // 2. 返回数据
27         DatagramPacket p = new DatagramPacket(buf, buf.length, packet.getAddress(),
28         packet.getPort());
29         socket.send(p);
30         socket.close();
31     }
32 }
```

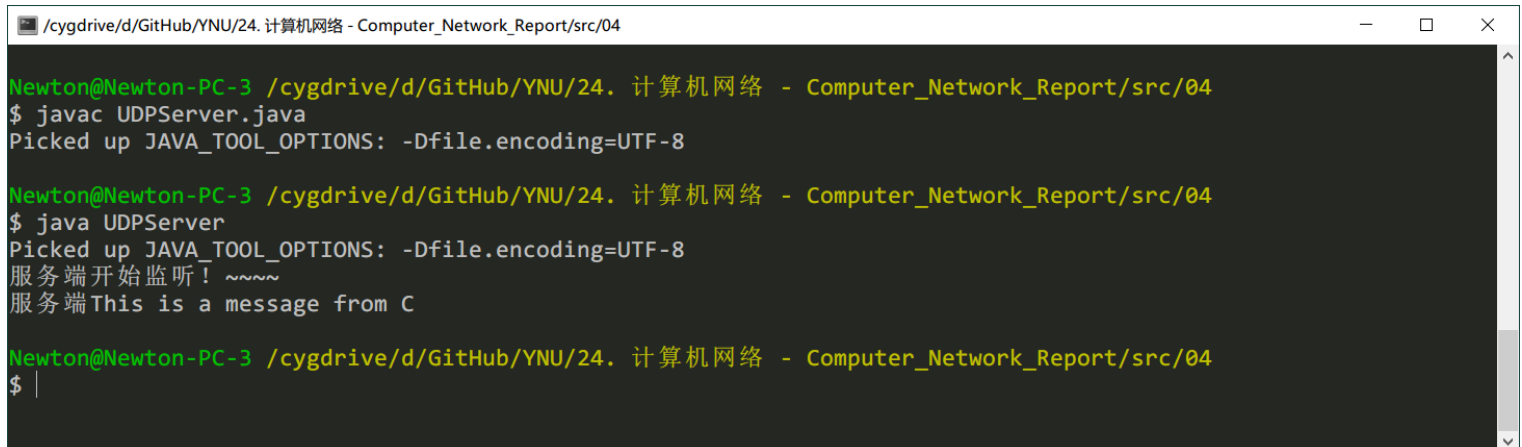
4.3.4 Android App 端 Java 代码

```
1  package com.example.newton.myapplication;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.widget.EditText;
6  import android.widget.TextView;
7  import android.widget.Button;
8
9  import java.io.IOException;
10 import java.net.DatagramPacket;
11 import java.net.DatagramSocket;
```

```
12 import java.net.InetAddress;
13 import java.net.UnknownHostException;
14 import java.net.SocketException;
15 import android.view.View;
16
17 public class MainActivity extends AppCompatActivity {
18
19     EditText et;
20     TextView tv;
21     Button button;
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         et = findViewById(R.id.et);
29         tv = findViewById(R.id.tv);
30         button = findViewById(R.id.button);
31     }
32
33     public void onClick(View view) {
34         new Thread() {
35             @Override
36             public void run() {
37                 super.run();
38
39                 byte[] bytes = et.getText().toString().getBytes();
40                 try {
41                     // 接收数据
42                     InetAddress address = InetAddress.getByName("192.168.1.75");
43
44                     // 1. 构造数据包
45                     DatagramPacket packet = new DatagramPacket(bytes, bytes.length, address,
2222);
46
47                     // 2. 创建数据报套接字并将其绑定到本地主机上的指定端口。
48                     DatagramSocket socket = new DatagramSocket();
49
50                     // 3. 从此套接字发送数据报包。
51                     socket.send(packet);
52
53                     // 接收数据
54                     // 1. 构造 DatagramPacket, 用来接收长度为 length 的数据包。
55                     final byte[] bytes1 = new byte[1024];
56                     DatagramPacket receiverPacket = new DatagramPacket(bytes1, bytes1.length);
57                     socket.receive(receiverPacket);
58                     runOnUiThread(new Runnable() {
59                         @Override
```

```
60         public void run() {
61             tv.setText(new String(bytes1, 0, bytes1.length));
62         }
63     });
64
65     socket.close();
66 } catch (UnknownHostException e) {
67     e.printStackTrace();
68 } catch (SocketException e) {
69     e.printStackTrace();
70 } catch (IOException e) {
71     e.printStackTrace();
72 }
73 }
74 }.start();
75 }
76 }
```

4.3.5 Server 端运行结果

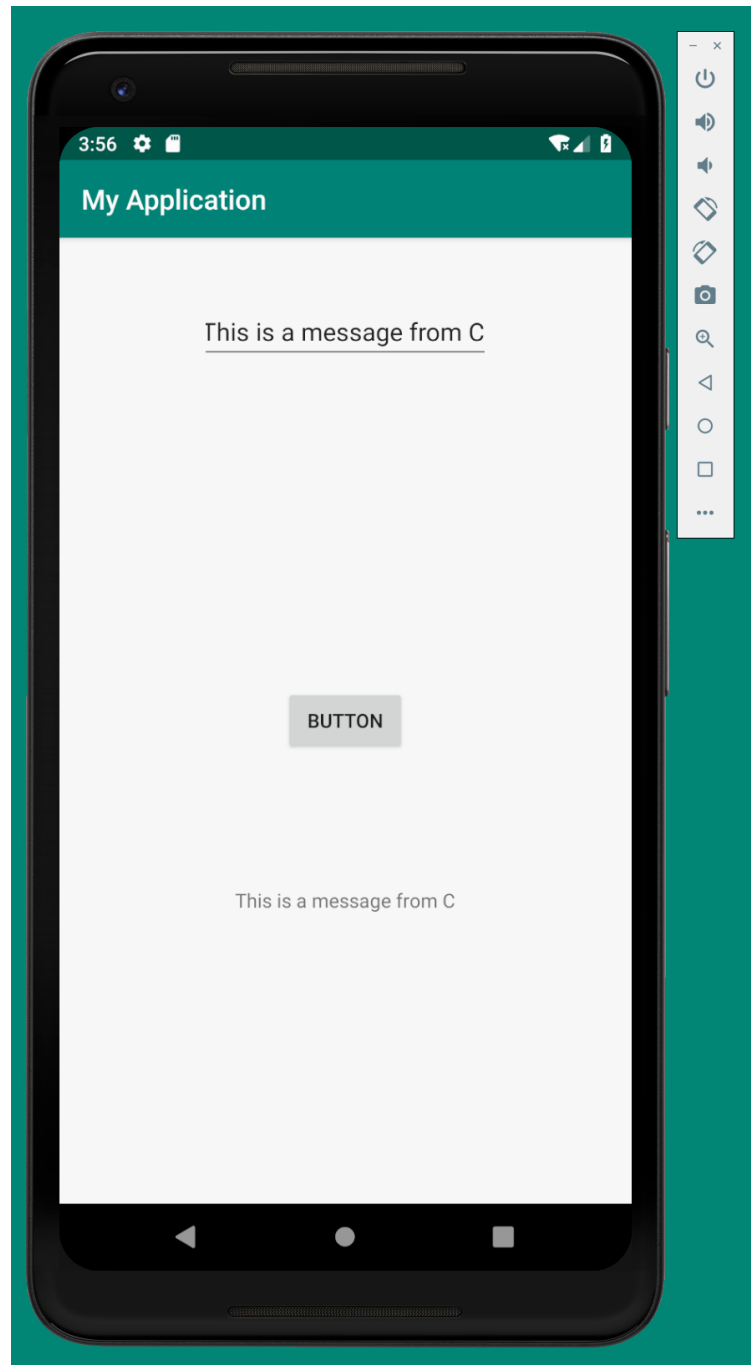


```
/cygdrive/d/GitHub/YNU/24. 计算机网络 - Computer_Network_Report/src/04
Newton@Newton-PC-3 /cygdrive/d/GitHub/YNU/24. 计算机网络 - Computer_Network_Report/src/04
$ javac UDPServer.java
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8

Newton@Newton-PC-3 /cygdrive/d/GitHub/YNU/24. 计算机网络 - Computer_Network_Report/src/04
$ java UDPServer
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
服务端开始监听! ~~~~
服务端This is a message from C

Newton@Newton-PC-3 /cygdrive/d/GitHub/YNU/24. 计算机网络 - Computer_Network_Report/src/04
$ |
```

4.3.6 Android 客户端运行结果



五、 实验体会

因为之前做过有关 TCP 的 Java 编程，所以许多框架上的问题可以直接拿过来套用。重点是关心 TCP 与 UDP 的区别。从 Java 的实现来看，核心区别是采用的类不同，TCP 采用 Socket 类，而 UDP 采用 DatagramPacket Socket。这两种类的具体实现是有本质差异的。

下面引用之前曾经出现过的图进行对比：

TCP Server

1. 新建serverSocket对象，并指定端口
`serverSocket = new ServerSocket(port)`
2. 进行监听
`socket = serverSocket.accept()`
3. 拿到输入流（客户端发来的消息）
`InputStream = socket.getInputStream()`
4. 解析数据
`reader = new InputStreamReader(inputStream)`
`bufReader = new BufferedReader(reader)`
一系列操作
5. 关闭输入流
`socket.shutdownInput()`
6. 拿到输出流
`OutputStream = socket.getOutputStream()`
`OutputStream.someOperations()`
`OutputStream.flush()`
7. 关闭输出流
`socket.shutdownOutput()`
`OutputStream.close()`
8. 关闭IO资源
`bufReader.close()`
`reader.close()`
`InputStream.close()`
9. 关闭socket
`socket.close()`
`serverSocket.close()`

UDP Server

1. 新建存储空间，等待盛放
`byte[] buf = new byte[1024]`
`DatagramPacket packet = new DatagramPacket(buf, buf.length)`
2. 创建UDP Socket
`DatagramSocket socket = new DatagramSocket(2222)`
3. 进程处于阻塞状态，监听端口直到接收到消息
`socket.receive(packet)`
4. 返回数据（这一步不是必须的）
`DatagramPacket p = new DatagramPacket(buf, buf.length, packet.getAddress(), packet.getPort())`
`socket.send(p)`
5. 关闭Socket
`socket.close()`

TCP Client

1. 直接创建socket对象
`socket = new Socket(ip_address, port)`
2. 拿到客户端要发送的数据流
`OutputStream = socket.getOutputStream()`
3. 写入要发给服务器的数据
`OutputStream.someOperations()`
4. 关闭输出流
`OutputStream.flush()`
`socket.shutdownOutput()`
5. 经过些许等待，拿到服务器返回的数据，即输入流
`InputStream = socket.getInputStream()`
6. 解析服务器返回的数据
7. 关闭IO资源
8. 关闭socket
`InputStream.close()`
`OutputStream.close()`
`socket.close()`

UDP Client

1. 根据服务器地址构建数据包
`byte[] bytes = et.getText().toString().getBytes()`
`InetAddress address = InetAddress.getByName("192.168.1.75")`
`DatagramPacket packet = new DatagramPacket(bytes, bytes.length, address, 2222)`
2. 创建Socket，并发送数据报
`DatagramSocket socket = new DatagramSocket()`
`socket.send(packet)`
3. 接收数据报（这一步不是必须的）
`byte[] bytes1 = new byte[1024]`
`DatagramPacket receiverPacket = new DatagramPacket(bytes1, bytes1.length)`
`socket.receive(receiverPacket)`
4. 关闭Socket
`socket.close()`

具体的区别，是由 TCP 和 UDP 的传输协议的差异决定的。

六、参考文献

- [1] 林锐. 高质量 C++/C 编程指南 [M]. 1.0 ed., 2001.
- [2] java IO: <https://zhuanlan.zhihu.com/p/21444494>
- [3] java NIO: <https://www.jianshu.com/p/093b7c408dba>
- [4] java NIO: <https://docs.oracle.com/javase/7/docs/api/java/nio/package-summary.html>
- [5] java NET: <https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html>