### 实验报告

1. **查看课程配套的2个JAVA Applet演示**

1.这个小程序显示了HTTP传输延迟的模型。分为持续连接/非持续连接，并行传输/非并行传输，一共四种情况。非持续连接时，同一个web页面的不同对象（包括HTML页面）在传输时，每个对象都要重新请求建立TCP连接。持续连接时，与该web页面建立一次连接后，所有对象都公用这个连接而不需要重新连接，在RTT远大于传输时间时，这可以节省大量时间。非并行传输时，同一时间只有一个对象在请求或者传输；并行传输时，传完HTML页面后，所有的对象可以同时请求连接或者传输而不需要互相等待。

1. 这个小程序显示了DNS查询过程，分为迭代查询和递归查询；

在递归查询中，查询顺着本地服务器，根服务器，TLD服务器传，直到查询到权威服务器，再把结果原路返回。可以看到这种情况下，如果在本地服务器和根服务器都没有，根服务器就要处理两次往返的查询，会让它负担过重。

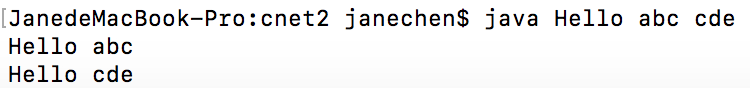
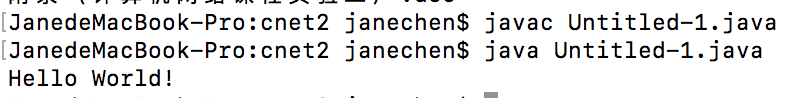
在迭代查询中，若根服务器和TLD服务器查不到，就向本地服务器返回下一级查询的地址，让本地服务器自己查，把负担都放在根服务器上。

在查询中有三种情况：本地服务器缓存中有，直接返回，经过一次往返查询；根服务器有，经过两次往返查询；TLD有，经过三次往返查询；权威服务器有，经过四次。

1. **Socket编程**

**1，2，3.java环境的配置和初步学习**

因为自己的电脑是苹果的，没有Windows的操作系统，所以在VS code上配置了java的环境。



后来觉得不好用又装了eclipse。

**了解Java的基础知识：**

Java程序的基本单位是类，类中主要包含变量与方法；对象是根据类创建的。在Java中，使用关键字new来创建一个新的对象。一个 Java 程序可以认为是一系列对象的集合，而这些对象通过调用彼此的方法来协同工作。

**源文件声明规则：**

1 一个源文件中只能有一个public类（所以在eclipse中只有新建class，没有新建java文件）

2 一个源文件可以有多个非public类

3 源文件的名称应该和public类的类名保持一致。例如：源文件中public类的类名是Employee，那么源文件应该命名为Employee.java。

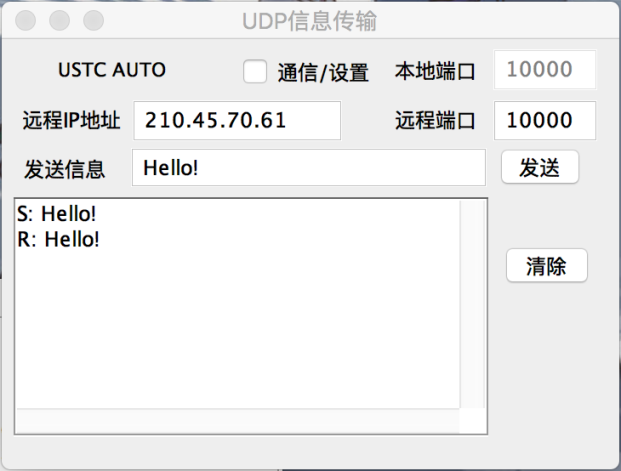
4 如果一个类定义在某个包中，那么package语句应该在源文件的首行。

5 如果源文件包含import语句，那么应该放在package语句和类定义之间。如果没有package语句，那么import语句应该在源文件中最前面。

I6 mport语句和package语句对源文件中定义的所有类都有效。在同一源文件中，不能给不同的类不同的包声明。

**4. Socket套接字应用演示**

这个udp应用实现的是利用udp连接发送一个字符串给指定主机和端口并显示，接收一个字符串并显示，有图形化界面。

在远程IP地址栏输入127.0.0.1（默认本地主机）或者本地主机IP，可以得到回复并显示。

输入其他地址没有回复，因为其他主机并没有接收该信息的应用。

**代码阅读：**

UdpApp.java的作用是实例化类Frame1并进行初始化。

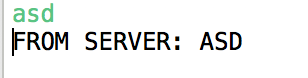
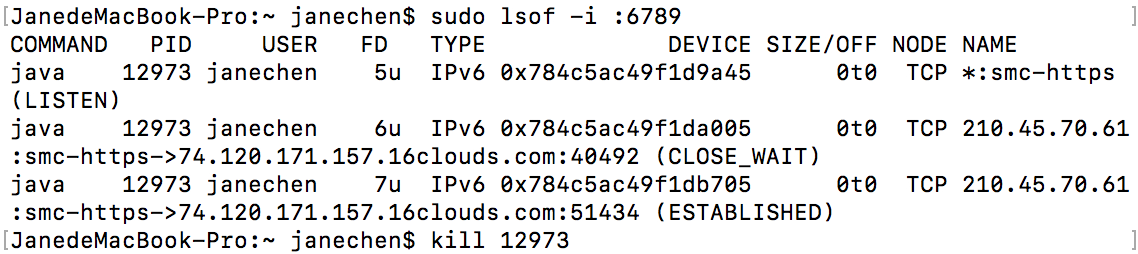
Frame1.java的作用是定义类Frame1和操作；其中jbInit()初始化图形界面（画框，栏，按钮 ，填写标签），Botton1监听点击事件，如果出现则调用类button1\_actionPerformed，将信息发送给指定IP和port，并且在下面显示；checkbox通信/设置监听点击事件，根据点击切换状态，并修改变量Run来控制run的执行，run中创建一个并发的线程进行等待接收或者忙等Run变为true，接收后显示。

**java图形界面的编写**：利用库swing和awt中定义好的类，在此基础上画窗口。

Frame1 继承（extends）了swing中定义的类JFrame（窗体），在里面可以增加swing中创建的别的组件，比如Jpanel，bottom，label，textfield，Jpanel，textarea，checkbox。在创建之后，需要对组件初始化，比如颜色，大小，初始值，标签；对按钮还可以定义按下触发的事件（类），textfield可以用getText得到框内输入的值。定义完之后把他们全部添加到定义好的Jpanel即主题窗中。

1. **编写简单tcp应用**

按照书上的例子编写了用户操作和服务器操作，得到结果如下：

  
发现一个问题，由于服务器等待用户连接请求是一个死循环，没有释放welcome socket的机会，所以再次运行会报错，说6789号端口已经被占用，此时可以查出进程号kill掉。

为了编写tcp应用，希望得到图形界面，所以使用了udp应用的框架，把实现udp发送接收的部分改为tcp连接与接收，得到的结果：



核心代码：

用户方：按下发送按钮触发的事件：

**void** button1\_actionPerformed(ActionEvent e) {

String str = textField2.getText();

**if** (str.compareTo("") != 0) {

**try** {

    textArea1.append("S: " +

                     textField1.getText() + "\n");

**byte** data[] = textField1.getText().getBytes();

    String datas = **new** String(data); //数据转为string格式发送

String sentence;

String modifiedSentence;

Socket clientSocket = **new** Socket("JanedeMacBook-Pro.local", 6789);

//建立用户socket，目标主机为本主机，端口为6789

DataOutputStream outToServer = **new** DataOutputStream(

clientSocket.getOutputStream()); //实例化向服务器发送这一操作的类

BufferedReader inFromServer =

**new** BufferedReader(**new** InputStreamReader(

clientSocket.getInputStream())); //实例化从服务器接受这一操作的类

outToServer.writeBytes(datas + '\n'); //发送

modifiedSentence = inFromServer.readLine(); //接收

System.***out***.println("FROM SERVER: " +

modifiedSentence);

textArea1.append("R: " +

    modifiedSentence + "\n"); //显示接收到的数据

  }

**catch** (IOException exc) {

    textArea1.append(exc.toString() + "\n");

    exc.printStackTrace();

  }

}

服务器：并行的等待连接线程：

**public** **void** run() {

Thread thisThread = Thread.*currentThread*(); //建立并行线程

**while** (**true**) {

**if** (!Run) {

**try** {

      thisThread.*sleep*(100);

    }

**catch** (InterruptedException e) {

    }

**continue**;

  }

**try**{ //Run == true时等待连接

  String clientSentence;

  String capitalizedSentence;

  ServerSocket welcomeSocket = **new** ServerSocket(6789); //建立欢迎连接的socket，端口号与用户socket的目标对应

Socket connectionSocket = welcomeSocket.accept(); //等待到连接请求后接受，建立连接

BufferedReader inFromClient =

**new** BufferedReader(**new** InputStreamReader(

connectionSocket.getInputStream())); //实例化接收用户发送的数据的类

DataOutputStream outToClient =

**new** DataOutputStream(

connectionSocket.getOutputStream()); //实例化发送给用户的类

clientSentence = inFromClient.readLine(); //接收数据

capitalizedSentence =

clientSentence.toUpperCase() + '\n'; //转化为大写

outToClient.writeBytes(capitalizedSentence); //发送给用户

welcomeSocket.close(); //关闭socket，不然就只能接受一次，下一次报端口已经被占用的错误

}

**catch** (IOException se) {

      textArea1.append(se.toString() + "\n");

      se.printStackTrace();

    }

}

}

与udp的主要区别在于需要socket建立连接，之后再进行发送接收。