计算机网络实验报告

小组成员：秦浩、曲少伟、张广明　　　指导教师：蒋若冰　　　实验日期：**2019/12/16**

一、 实验名称
Programming Assignment 1: Sockets, Mininet, & Performance
二、 实验目的
　　Iperf 是测量网络带宽的常用工具。使用 java sockets 编写自己版本的 Iperfer。然后，使用工具来测量 Mininet 中虚拟网络的性能，并分析链接特性和多路复用如何影响性能。
三、 实验要求
第一步，写 Iperfer。
　　编写名为 Iperfer 的工具，它将使用 sockets 在一对主机之间发送和接收 TCP 数据包。
第二步，安装 mininet。
　　先安装 Ubuntu 虚拟机，在通过：sudo apt-get install mininet，命令安装 mininet。
第三步，在 mininet 中测量
　　使用编写的工具(Iperfer)和标准延迟测量工具 ping（ping 度量 RTT）来测量 Mininet 中的虚拟网络中的带宽和延迟。在报告中包括一些实验的结果和下面问题的答案。对这些问题的回答应该放在一个名为 answers.txt 的文件中。
四、 实验内容
第一步，写 Iperfer 。

伪代码如下：

```
//0 is client mode and 1 is server mode

mode = 0;

// shortest length of args is 3, longest length is 7

minLength = 3, maxLength = 7;

// -c option is client mode, -s is server mode

clientOption = "-c", serverOption = "-s";

hOption = "-h", pOption = "-p", tOption = "-t";

// min port number is 1024, max is 65535

minPort = 1024, maxPort = 65535;


if args.length < minLength || args.length > maxLength        // args length error

    then print Error: missing or additional arguments

        exit

    else if args[0] = clientMode                              // client mode

        then if !(args[1] = hOption) || !(args[3] = pOption) || !(args[5] = tOption)

            then print Error: invalid input

                exit

        then if args[4] < minPort || args[4] > maxPort
```

```
                then print Error: port number must be in the range 1024 to 65535
                        exit
        else if args[0] = serverMode                            // client mode
            then if !(args[1] = pOption)
                then print Error: invalid input
                        exit
            then if args[2]) < minPort || args[2] > maxPort
                then print Error: port number must be in the range 1024 to 65535
                        exit
            mode <-- 1
            else print Error: invalid input
                exit

if mode = 0                                              // client mode
    then print This is client mode
        while true
            do if currentTime - startTime >= totalTime
                    then break
                out.write(packet) // OutputStream out = socket.getOutputStream(), byte[] packet = new
byte[1000];
                ++pachetNumber
        print sent = packetNumber KB rate= packetNumber / (1000 * time) Mbps
    else print This is server mode.
        while (length = in.read(buf)) != -1 // byte[] buf = new byte[1000],InputStream in =
clientSocket.getInputStream()
            do if length == 1000
                    then ++pachetNumber
        print received = packetNumber KB rate = (double)packetNumber / totalTime Mbps
```

Java 语言实现

```java
package ouc.cs;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;

/**
 * Iperfer class
 * @author qinhao
 * @date 2019/12/14
```

```java
 * IDEA Community 2019.2.4 + JDK 1.8
 */
public class Iperfer {

    public static void main(String[] args) throws IOException {

        // java Iperfer -c -h <server hostname> -p <server port> -t <time>
        // java Iperfer -s -p <listen port>

        //0 is client mode and 1 is server mode
        int mode = 0;
        // shortest length of args is 3, longest length is 7
        int minLength = 3, maxLength = 7;
        // -c option is client mode, -s is server mode
        String clientOption = "-c", serverOption = "-s";
        String hOption = "-h", pOption = "-p", tOption = "-t";
        // min port number is 1024, max is 65535
        int minPort = 1024, maxPort = 65535;
        // to avoid using magic numbers, I am so difficult.
        int argsZero = 0, argsOne = 1, argsTwo = 2, argsThree = 3, argsFour =
                4,argsFive = 5, argsSix = 6;
        int zero = 0;

        if (args.length < minLength || args.length > maxLength) {
            System.err.println("Error: missing or additional arguments");
            System.exit(1);
        } else if (args[argsZero].equals(clientOption)) {
            // client mode
            if (!(args[argsOne].equals(hOption)) || !(args[argsThree]
                    .equals(pOption)) || !(args[argsFive].equals(tOption))) {
                System.err.println("Error: invalid input");
                System.exit(1);
            }
            if (Integer.parseInt(args[argsFour]) < minPort ||
                    Integer.parseInt(args[argsFour]) > maxPort) {
                System.err.println("Error: port number must be in the range " +
                        "1024 to 65535");
                System.exit(1);
            }
        } else if (args[argsZero].equals(serverOption)) {
            // server mode
            if (!(args[argsOne].equals(pOption))) {
                System.err.println("Error: invalid input");
                System.exit(1);
```

```java
            }
            if (Integer.parseInt(args[argsTwo]) < minPort ||
                    Integer.parseInt(args[argsTwo]) > maxPort) {
                System.err.println("Error: port number must be in the range " +
                        "1024 to 65535");
                System.exit(1);
            }
            mode = 1;
        } else {
            System.err.println("Error: invalid input");
            System.exit(1);
        }


        if (mode == zero) {
            // client mode
            System.out.println("This is client mode.");


            String hostName = args[argsTwo];
            int portNumber = Integer.parseInt(args[argsFour]);
            // second
            int time = Integer.parseInt(args[argsSix]);


            /*
             * Any object that implements java.lang.AutoCloseable, which
             * includes all objects which implement java.io.Closeable, can be
             * used as a resource.
             */
            try (
                    Socket socket = new Socket(hostName, portNumber);
                    OutputStream out = socket.getOutputStream()
            ) {
                byte[] packet = new byte[1000];
                long totalTime = time * 1000;
                long packetNumber = 0;
                long startTime = System.currentTimeMillis();


                while (true) {
                    long currentTime = System.currentTimeMillis();
                    if (currentTime - startTime >= totalTime) {
                        break;
                    }
                    out.write(packet);
                    ++packetNumber;
                }
```

```java
                System.out.println("sent=" + packetNumber + "KB " + "rate=" +
                        (double)packetNumber * 8 / (1000 * time)+ "Mbps\n");

        } catch (UnknownHostException e) {
            e.printStackTrace();
            System.err.println("Don't know about host " + hostName);
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
            System.err.println("Couldn't get I/O for the connection to " +
                    hostName);
            System.exit(1);
        }
    } else {
        // server mode
        System.out.println("This is server mode.");

        int listenPort = Integer.parseInt(args[argsTwo]);

        try (
                ServerSocket serverSocket = new ServerSocket(listenPort);
                Socket clientSocket = serverSocket.accept();
                InputStream in = clientSocket.getInputStream()
        ) {
            byte[] buf = new byte[1000];
            long packetNumber = 0;
            long startTime = System.currentTimeMillis();
            int length;

            while ((length = in.read(buf)) != -1) {
                if (length == 1000) {
                    ++packetNumber;
                }
            }

            long currentTime = System.currentTimeMillis();
            long totalTime = currentTime - startTime;
            System.out.println("received=" + packetNumber + "KB " + "rate" +
                    "=" + (double)packetNumber * 8 / totalTime +
                    "Mbps\n");
        }
    }
}
```

```
}
```

　　在 Windows 系统下测试，客户端 program arguments 设置为-c -h localhost -p 4444 -t 20,服务器 program arguments 设置为-s -p 4444。运行结果如下，基本符合预期：

```
This is server mode.
received=853503KB rate=341.2135325570936Mbps
```

```
This is client mode.
sent=853503KB rate=341.4012Mbps
```

第二步，安装 mininet
sudo apt-get install mininet
第三步，在 mininet 中测量
1. 运行 Iperfer 之前，在虚拟机中安装 java，通过以下命令：
sudo apt-get update
sudo apt-get install openjdk-8-jdk
2. sudo python progAssign1_topo.py 创建一个拓扑结构的网络
3. 按照实验要求，在各主机之间运行 Iperfer 和 ping，测量并记录实验结果。

五、实验结果与分析
Q1：

```
mininet> h1 ping -c 20 h2
PING 10.0.0.2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=64 time=87.344 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=95.132 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=81.955 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=82.081 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=81.756 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=80.940 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=82.047 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=81.592 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=85.540 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=81.601 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=81.824 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=82.008 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=81.431 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=81.319 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=81.265 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=83.329 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=81.769 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=82.126 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=81.719 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=81.626 ms
--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 80.940/82.920/95.132/3.174 ms
```

```
mininet> h2 ping -c 20 h3
PING 10.0.0.3 (10.0.0.3): 56 data bytes
64 bytes from 10.0.0.3: icmp_seq=0 ttl=64 time=28.579 ms
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=23.819 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=21.974 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=22.045 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=22.950 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=21.971 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=21.821 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=21.984 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=21.851 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=22.338 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=22.064 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=22.444 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=22.184 ms
64 bytes from 10.0.0.3: icmp_seq=13 ttl=64 time=21.970 ms
64 bytes from 10.0.0.3: icmp_seq=14 ttl=64 time=21.507 ms
64 bytes from 10.0.0.3: icmp_seq=15 ttl=64 time=21.959 ms
64 bytes from 10.0.0.3: icmp_seq=16 ttl=64 time=22.234 ms
64 bytes from 10.0.0.3: icmp_seq=17 ttl=64 time=21.895 ms
64 bytes from 10.0.0.3: icmp_seq=18 ttl=64 time=22.091 ms
64 bytes from 10.0.0.3: icmp_seq=19 ttl=64 time=21.464 ms
--- 10.0.0.3 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.464/22.457/28.579/1.489 ms
mininet> h3 ping -c 20 h4
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: icmp_seq=0 ttl=64 time=67.248 ms
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=61.872 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=61.738 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=61.455 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=62.376 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=61.621 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=61.755 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=61.883 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=61.975 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=61.775 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=62.078 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=61.294 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=62.068 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=62.258 ms
64 bytes from 10.0.0.4: icmp_seq=14 ttl=64 time=61.723 ms
64 bytes from 10.0.0.4: icmp_seq=15 ttl=64 time=61.839 ms
64 bytes from 10.0.0.4: icmp_seq=16 ttl=64 time=62.198 ms
64 bytes from 10.0.0.4: icmp_seq=17 ttl=64 time=61.717 ms
64 bytes from 10.0.0.4: icmp_seq=18 ttl=64 time=62.156 ms
64 bytes from 10.0.0.4: icmp_seq=19 ttl=64 time=62.291 ms
--- 10.0.0.4 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 61.294/62.166/67.248/1.198 ms
```

```
mininet> h2 ping -c 20 h5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=64 time=19.425 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=12.114 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=11.895 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=12.030 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=12.012 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=11.950 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=12.093 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=13.789 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=12.206 ms
64 bytes from 10.0.0.5: icmp_seq=9 ttl=64 time=11.163 ms
64 bytes from 10.0.0.5: icmp_seq=10 ttl=64 time=11.067 ms
64 bytes from 10.0.0.5: icmp_seq=11 ttl=64 time=11.121 ms
64 bytes from 10.0.0.5: icmp_seq=12 ttl=64 time=12.163 ms
64 bytes from 10.0.0.5: icmp_seq=13 ttl=64 time=12.221 ms
64 bytes from 10.0.0.5: icmp_seq=14 ttl=64 time=10.918 ms
64 bytes from 10.0.0.5: icmp_seq=15 ttl=64 time=11.217 ms
64 bytes from 10.0.0.5: icmp_seq=16 ttl=64 time=11.245 ms
64 bytes from 10.0.0.5: icmp_seq=17 ttl=64 time=11.143 ms
64 bytes from 10.0.0.5: icmp_seq=18 ttl=64 time=11.973 ms
64 bytes from 10.0.0.5: icmp_seq=19 ttl=64 time=11.523 ms
--- 10.0.0.5 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 10.918/12.163/19.425/1.783 ms
mininet> h3 ping -c 20 h6
PING 10.0.0.6 (10.0.0.6): 56 data bytes
64 bytes from 10.0.0.6: icmp_seq=0 ttl=64 time=18.757 ms
64 bytes from 10.0.0.6: icmp_seq=1 ttl=64 time=14.443 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=64 time=12.119 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=64 time=11.780 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=64 time=12.027 ms
64 bytes from 10.0.0.6: icmp_seq=5 ttl=64 time=12.086 ms
64 bytes from 10.0.0.6: icmp_seq=6 ttl=64 time=14.415 ms
64 bytes from 10.0.0.6: icmp_seq=7 ttl=64 time=12.105 ms
64 bytes from 10.0.0.6: icmp_seq=8 ttl=64 time=12.252 ms
64 bytes from 10.0.0.6: icmp_seq=9 ttl=64 time=12.296 ms
64 bytes from 10.0.0.6: icmp_seq=10 ttl=64 time=11.783 ms
64 bytes from 10.0.0.6: icmp_seq=11 ttl=64 time=11.211 ms
64 bytes from 10.0.0.6: icmp_seq=12 ttl=64 time=12.248 ms
64 bytes from 10.0.0.6: icmp_seq=13 ttl=64 time=12.500 ms
64 bytes from 10.0.0.6: icmp_seq=14 ttl=64 time=12.265 ms
64 bytes from 10.0.0.6: icmp_seq=15 ttl=64 time=12.521 ms
64 bytes from 10.0.0.6: icmp_seq=16 ttl=64 time=12.099 ms
64 bytes from 10.0.0.6: icmp_seq=17 ttl=64 time=12.143 ms
64 bytes from 10.0.0.6: icmp_seq=18 ttl=64 time=12.347 ms
64 bytes from 10.0.0.6: icmp_seq=19 ttl=64 time=11.062 ms
--- 10.0.0.6 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 11.062/12.623/18.757/1.617 ms
```

"Node: h2"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -c -h 10.0.0
.1 -p 1234 -t 20
This is client mode.
sent=54128KB rate=21.6512Mbps
```

"Node: h1"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -s -p 1234
This is server mode.
received=39905KB rate=13.680737090207842Mbps

root@qinhao-virtual-machine:~/□□面/computer_networks#
```
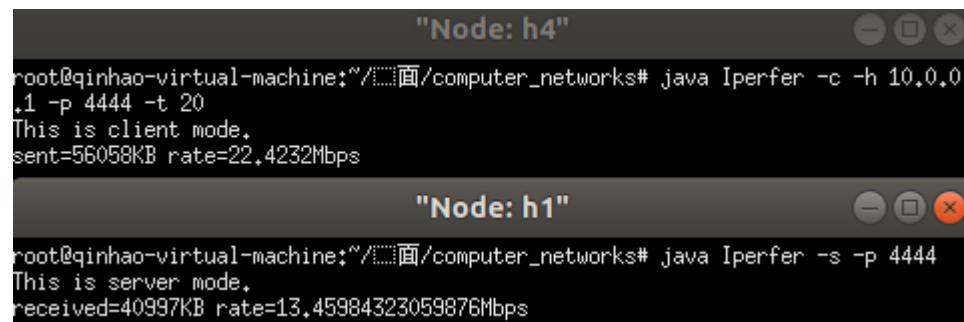
"Node: h4"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -c -h 10.0.0
.3 -p 1234 -t 20
This is client mode.
sent=78246KB rate=31.2984Mbps
```

"Node: h3"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -s -p 1234
This is server mode.
received=57255KB rate=18.43664466269522Mbps
```

"Node: h5"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -c -h 10.0.0
.2 -p 4444 -t 20
This is client mode.
sent=67181KB rate=26.8724Mbps
```

"Node: h2"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -s -p 4444
This is server mode.
received=48903KB rate=17.097456516038807Mbps
```

"Node: h6"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -c -h 10.0.0
.3 -p 4444 -t 20
This is client mode.
sent=69649KB rate=27.8596Mbps
```

"Node: h3"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -s -p 4444
This is server mode.
received=50610KB rate=17.05188679245283Mbps
```

"Node: h3"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -c -h 10.0.0
.2 -p 1234 -t 20
This is client mode.
sent=94095KB rate=37.638Mbps
```

"Node: h2"

```
root@qinhao-virtual-machine:~/□□面/computer_networks# java Iperfer -s -p 1234
This is server mode.
received=86910KB rate=31.00606493043168Mbps
```

Q2：
预测 RTT：167.543ms（延迟 83.7715ms）
预测吞吐量：
sent=54128KB rate=21.6512Mbps
received=39905KB rate=13.680737090207842Mbps
实测：


```
mininet> h1 ping -c 20 h4
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: icmp_seq=0 ttl=64 time=180.756 ms
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=169.708 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=165.933 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=164.885 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=166.208 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=164.776 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=165.709 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=166.285 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=163.742 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=164.703 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=165.132 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=164.556 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=164.699 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=165.926 ms
64 bytes from 10.0.0.4: icmp_seq=14 ttl=64 time=165.824 ms
64 bytes from 10.0.0.4: icmp_seq=15 ttl=64 time=164.391 ms
64 bytes from 10.0.0.4: icmp_seq=16 ttl=64 time=163.641 ms
64 bytes from 10.0.0.4: icmp_seq=17 ttl=64 time=163.705 ms
64 bytes from 10.0.0.4: icmp_seq=18 ttl=64 time=163.533 ms
64 bytes from 10.0.0.4: icmp_seq=19 ttl=64 time=165.728 ms
--- 10.0.0.4 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 163.533/165.992/180.756/3.643 ms
```

解释：
RTT 为三条独立链路上 RTT 之和
吞吐量是三条独立链路最小的吞吐量
Q3:
两个主机：
预期值：
H1 - H4

延迟 ： 80 ms
吞吐量 ： 10 Mbps
H7 - H9
延迟 ： 80 ms
吞吐量 ： 10 Mbps
实测：


```
"Node: h4"
root@qinhao-virtual-machine:~/桌面/computer_networks# java Iperfer -c -h 10.0.0
.1 -p 4444 -t 20
This is client mode.
sent=56552KB rate=22.6208Mbps
```

```
"Node: h7"
root@qinhao-virtual-machine:~/桌面/computer_networks# java Iperfer -c -h 10.0.0
.9 -p 4444 -t 20
This is client mode.
sent=19719KB rate=7.8876Mbps
```

```
"Node: h1"
root@qinhao-virtual-machine:~/桌面/computer_networks# ping -c 20 10.0.0.4
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: icmp_seq=0 ttl=64 time=178.937 ms
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=177.542 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=163.591 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=165.877 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=165.697 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=164.460 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=164.744 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=164.695 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=165.618 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=164.210 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=163.767 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=165.610 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=164.247 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=165.162 ms
64 bytes from 10.0.0.4: icmp_seq=14 ttl=64 time=165.010 ms
64 bytes from 10.0.0.4: icmp_seq=15 ttl=64 time=164.670 ms
64 bytes from 10.0.0.4: icmp_seq=16 ttl=64 time=163.459 ms
64 bytes from 10.0.0.4: icmp_seq=17 ttl=64 time=164.772 ms
64 bytes from 10.0.0.4: icmp_seq=18 ttl=64 time=165.711 ms
64 bytes from 10.0.0.4: icmp_seq=19 ttl=64 time=164.316 ms
--- 10.0.0.4 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 163.459/166.105/178.937/4.110 ms
root@qinhao-virtual-machine:~/桌面/computer_networks# []
```

```
"Node: h9"
root@qinhao-virtual-machine:~/桌面/computer_networks# ping -c 20 10.0.0.7
PING 10.0.0.7 (10.0.0.7): 56 data bytes
64 bytes from 10.0.0.7: icmp_seq=0 ttl=64 time=182.917 ms
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=173.808 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=165.898 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=165.647 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=165.903 ms
64 bytes from 10.0.0.7: icmp_seq=5 ttl=64 time=163.728 ms
64 bytes from 10.0.0.7: icmp_seq=6 ttl=64 time=164.943 ms
64 bytes from 10.0.0.7: icmp_seq=7 ttl=64 time=166.485 ms
64 bytes from 10.0.0.7: icmp_seq=8 ttl=64 time=165.248 ms
64 bytes from 10.0.0.7: icmp_seq=9 ttl=64 time=165.390 ms
64 bytes from 10.0.0.7: icmp_seq=10 ttl=64 time=164.550 ms
64 bytes from 10.0.0.7: icmp_seq=11 ttl=64 time=164.217 ms
64 bytes from 10.0.0.7: icmp_seq=12 ttl=64 time=165.646 ms
64 bytes from 10.0.0.7: icmp_seq=13 ttl=64 time=165.397 ms
64 bytes from 10.0.0.7: icmp_seq=14 ttl=64 time=164.531 ms
64 bytes from 10.0.0.7: icmp_seq=15 ttl=64 time=165.579 ms
64 bytes from 10.0.0.7: icmp_seq=16 ttl=64 time=165.358 ms
64 bytes from 10.0.0.7: icmp_seq=17 ttl=64 time=165.185 ms
64 bytes from 10.0.0.7: icmp_seq=18 ttl=64 time=165.228 ms
64 bytes from 10.0.0.7: icmp_seq=19 ttl=64 time=165.533 ms
--- 10.0.0.7 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 163.728/166.560/182.917/4.234 ms
root@qinhao-virtual-machine:~/桌面/computer_networks#
```

```
"Node: h4"
root@qinhao-virtual-machine:~/桌面/computer_networks# java Iperfer -c -h 10.0.0
.1 -p 4444 -t 20
This is client mode.
sent=56552KB rate=22.6208Mbps
```

```
"Node: h7"
root@qinhao-virtual-machine:~/桌面/computer_networks# java Iperfer -c -h 10.0.0
.9 -p 4444 -t 20
This is client mode.
sent=19719KB rate=7.8876Mbps
```

```
"Node: h1"
root@qinhao-virtual-machine:~/桌面/computer_networks# java Iperfer -s -p 4444
This is server mode.
received=41797KB rate=11.819582891481089Mbps
```

```
"Node: h9"
root@qinhao-virtual-machine:~/桌面/computer_networks# java Iperfer -s -p 4444
This is server mode.
received=14806KB rate=5.030493502081033Mbps
```

三个主机：
预期值：
H1 - H4
延迟 ： 80 ms
吞吐量 ： 6.67 Mbps
H7 - H9
延迟 ： 80 ms
吞吐量 ： 6.67 Mbps
H8 - H10
延迟 ： 80 ms
吞吐量 ： 6.67 Mbps
实测：

解释：

可以看到多对主机通信时对时延几乎没有影响，而对吞吐量的影响则比较大，具体到不同的连接影响程度不同

预期延迟是路径上链接的延迟之和。

延迟不会随如此小的一对主机一次通信而变化，因为传输和传播时间比排队延迟少得多。

随着主机数量的增加，延迟可能会增加，但很难在此有限设置中模拟此类行为。

预期吞吐量是沿路径（瓶颈）吞吐量最少的链路除以同时连接数。

测量的输出似乎与预期值不一致。

造成这种情况的主要原因是（根据我们的说法）实验错误。

在各种主机上启动服务器和客户端以收集测量值时存在延迟，但实验要求我们同时进行。

由于无法实际执行模拟测试，导致第一次连接的时间稍高一些，而接下来的连接时间较低。

Q4：

预期值：

H1 - H4

延迟： 80 ms

吞吐量：17.5 Mbps

H5 - H6：

延迟：20 ms

吞吐量：22.5 Mbps

实测：

```
"Node: h1"                              ⊖ ⊡ ⊗
root@qinhao-virtual-machine:~/▭▭面/computer_networks# java Iperfer -c -h 10.0.0
.4 -p 4444 -t 20
This is client mode.
sent=40391KB rate=16.1564Mbps
```

```
"Node: h4"                              ⊖ ⊡ ⊗

root@qinhao-virtual-machine:~/▭▭面/computer_networks# ping -c 20 10.0.0.1
PING 10.0.0.1 (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=64 time=187.412 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=170.681 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=165.649 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=166.219 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=163.735 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=165.901 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=165.656 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=164.742 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=165.509 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=165.498 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=165.472 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=165.506 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=165.255 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=165.535 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=165.541 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=164.961 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=165.403 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=165.681 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=164.508 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=167.610 ms
--- 10.0.0.1 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max/stddev = 163.735/166.824/187.412/4.910 ms
root@qinhao-virtual-machine:~/▭▭面/computer networks# □
```

```
"Node: h1"                              ⊖ ⊡ ⊗
root@qinhao-virtual-machine:~/▭▭面/computer_networks# java Iperfer -c -h 10.0.0
.4 -p 4444 -t 20
This is client mode.
sent=40391KB rate=16.1564Mbps
```

```
"Node: h4"                              ⊖ ⊡ ⊗
root@qinhao-virtual-machine:~/▭▭面/computer_networks# java Iperfer -s -p 4444
This is server mode.
received=29945KB rate=10.457482102322333Mbps
```

解释：

预期延迟是路径上链接的延迟之和。

不共享线路的 H1 - H4 的预期吞吐量为：20 Mbps;

和 H5 - H6 不共享线路是 25Mbps。但是，L2 在路径之间共享，

最大带宽为 40 Mbps，小于需要支持的总流量。

它比所需的流量少 5 Mbps，并且假设两个路由都受相同数量的降低，

2.5 Mbps，我们从路由的不间断流量带宽中减去。

六、遇到的问题与解决方法

1. 创建拓扑网络失败：

解决：sudo apt-get install openvswitch-testcontroller

sudo cp /usr/bin/ovs-testcontroller /usr/bin/ovs-controller

2.编译 java 文件提示找不到主类

解决：删除代码中的 package **.**，重新编译。

七、心得体会

通过这个实验，我们初步了解如何使用 JavaSocket 编写测量网络带宽的工具，初步学习了 ping 和 Iperfer 的使用方法，如何度量延迟和吞吐量。了解了链路平均延迟和吞吐量与此链路中小链路的平均延迟和吞吐量的关系：多对主机通信时对时延几乎没有影响，而对吞吐量的影响则比较大，具体到不同的连接影响程度不同。

八、小组成员及分工

成员任务：

任务	执行人
Iperfer 代码编写	秦浩
第三步（1-2 数据测量）	秦浩、曲少伟
第三步（3-4）	张广明
解释如何运行代码的自述文件	秦浩
实验报告撰写	曲少伟、张广明
PPT 编写	秦浩、曲少伟
汇报	秦浩

成员任务量：

秦浩 40%

曲少伟 30%

张广明 30%