

浙大城市学院实验报告

- 课程名称：计算机网络实验
- 实验项目名称：实验十四 传输层网络编程
- 学生姓名：徐彬涵
- 专业班级：软件工程2003
- 学号：32001272
- 实验成绩：
- 指导老师：霍梅梅
- 日期：2022/05/26

一. 实验目的和要求

1. 通过实现使用Java Socket进行通信的UDP客户端和服务端来获得关于使用Java Socket网络编程的经验；
2. 通过实现使用Java Socket进行通信的TCP客户端和服务端来获得关于使用Java Socket网络编程的经验。

二. 实验内容、原理及实验结果与分析

1. UDP编程

阅读讲义，并将源代码（UdpSend.java和UdpRecv.java）在机器上编译运行通过(注意：要根据自己的机器IP地址修改源代码)。

【程序源代码】

- UdpSend.java

```

1  import java.net.*;
2
3  public class UdpSend
4  {
5      public static void main(String[] args) throws Exception
6      {
7          DatagramSocket ds = new DatagramSocket();
8          String str = "Hello World!";
9          DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(),
10         InetAddress.getByAddress("192.168.123.111"), 9000);
11         ds.send(dp);
12         ds.close();
13     }
14 }

```

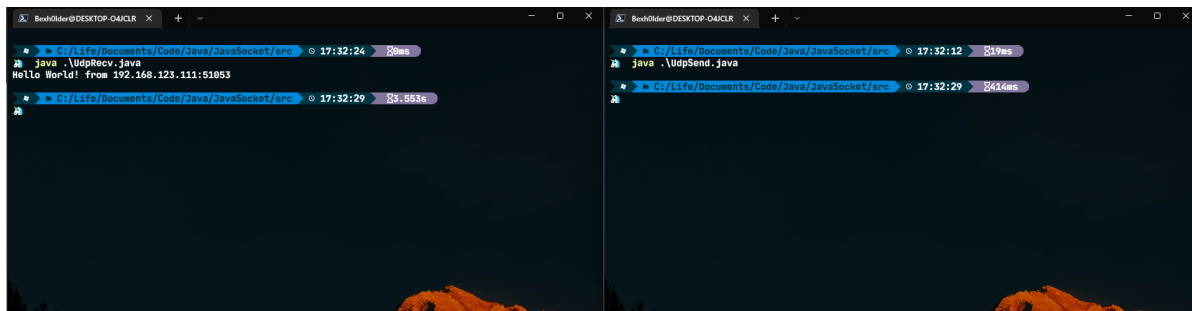
- UdpRecv.java

```

1  import java.net.*;
2
3  public class UdpRecv {
4      public static void main(String[] args) throws Exception
5      {
6          DatagramSocket ds = new DatagramSocket(9000);
7          byte[] buf = new byte[1024];
8          DatagramPacket dp = new DatagramPacket(buf, 1024);
9          ds.receive(dp);
10         String strRecv = new String(dp.getData(), 0, dp.getLength()) + " from " +
11         dp.getAddress().getHostAddress() + ":" + dp.getPort();
12         System.out.println(strRecv);
13         ds.close();
14     }
15 }

```

【实验结果与分析】



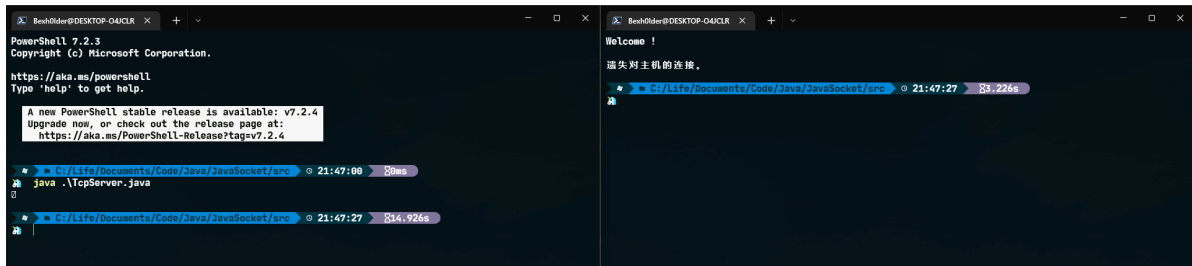
2. TCP编程

阅读讲义，并将源代码（TcpServer.java）在机器上编译运行，然后通过命令行中输入“telnet 自己的IP地址 8001”进行测试。

【程序源代码】

```
1  import java.net.*;
2  import java.io.*;
3  public class TcpServer {
4      public static void main(String [] args){
5          try
6          {
7              ServerSocket ss=new ServerSocket(8001);
8              Socket s=ss.accept();
9              InputStream ips=s.getInputStream();
10             OutputStream ops=s.getOutputStream();
11             ops.write("Welcome !".getBytes());
12             byte [] buf=new byte[1024];
13             int len=ips.read(buf);
14             System.out.println(new String(buf,0,len));
15             ips.close();
16             ops.close();
17             s.close();
18             ss.close();
19         }catch(Exception e)
20         {
21             e.printStackTrace();
22         }
23     }
24 }
25
```

【实验结果与分析】



阅读讲义，并将服务器端（Tcp_Server.java）以及客户端程序的源代码

（Tcp_Client.java）在机器上编译运行，客户端测试命令为“java Tcp_Client 自己的IP地址 8001”。

【程序源代码】

- Tcp_Client.java

```
1  import java.net.*;
2  import java.io.*;
3
4  public class Tcp_Client {
5      public static void main(String[] args) {
6          try {
7              if (args.length < 2) {
8                  System.out.println("Usage:java TcpClient ServerIP ServerPort");
9                  return;
10             }
11             Socket s = new Socket(InetAddress.getByName(args[0]),
Integer.parseInt(args[1]));
12             InputStream ips = s.getInputStream();
13             OutputStream ops = s.getOutputStream();
14             BufferedReader brKey = new BufferedReader(new
InputStreamReader(System.in));
15             DataOutputStream dos = new DataOutputStream(ops);
16             BufferedReader brNet = new BufferedReader(new
InputStreamReader(ips));
17             while (true) {
18                 String strWord = brKey.readLine();
19                 dos.writeBytes(strWord + System.getProperty("line.separator"));
20                 if (strWord.equalsIgnoreCase("quit"))
21                     break;
22                 else
23                     System.out.println(brNet.readLine());
24             }
```

```

25         dos.close();
26         brNet.close();
27         brKey.close();
28         s.close();
29     } catch (Exception e) {
30         e.printStackTrace();
31     }
32 }
33 }

```

- Servicer.java

```

1  import java.net.*;
2  import java.io.*;
3  class Servicer implements Runnable {
4      Socket s;
5      public Servicer(Socket s) {
6          this.s = s;
7      }
8      public void run() {
9          try {
10             InputStream ips = s.getInputStream();
11             OutputStream ops = s.getOutputStream();
12
13             BufferedReader br = new BufferedReader(new InputStreamReader(ips));
14             DataOutputStream dos = new DataOutputStream(ops);
15             while (true) {
16                 String strWord = br.readLine();
17                 System.out.println(strWord + ":" + strWord.length());
18                 if (strWord.equalsIgnoreCase("quit"))
19                     break;
20                 String strEcho = (new StringBuffer(strWord).reverse()).toString();
21                 dos.writeBytes(strWord + "---->" + strEcho +
22                     System.getProperty("line.separator"));
23             }
24             br.close();
25             dos.close();
26             s.close();
27         } catch (Exception e) {
28             e.printStackTrace();
29         }
30     }
31 }

```

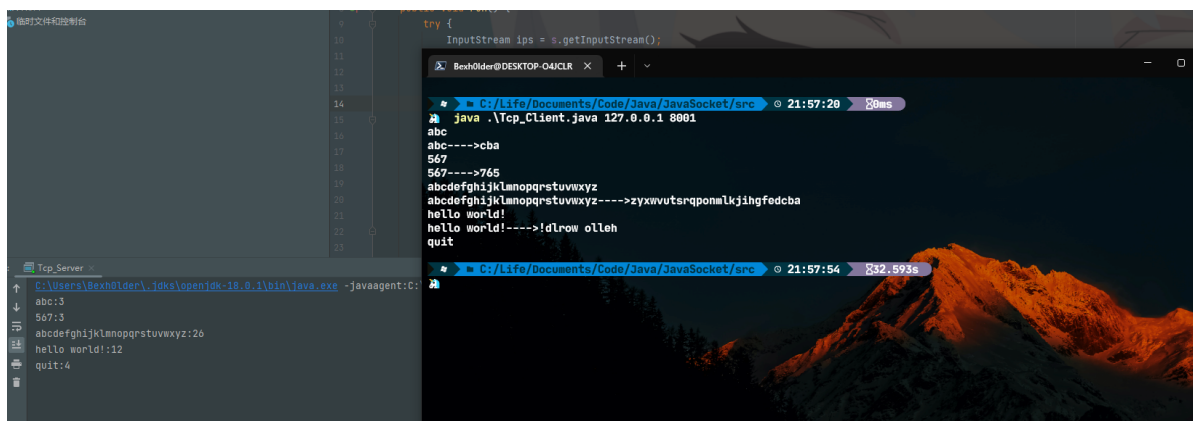
- Tcp_Server.java

```

1  import java.net.ServerSocket;
2  import java.net.Socket;
3
4  class Tcp_Server {
5      public static void main(String[] args) {
6          try {
7              ServerSocket ss = new ServerSocket(8001);
8              while (true) {
9                  Socket s = ss.accept();
10                 new Thread(new Servicer(s)).start();
11             }
12         } catch (Exception e) {
13             e.printStackTrace();
14         }
15     }
16 }
17

```

【实验结果与分析】



3. Wireshark抓包分析

用Wireshark软件截获上面三个程序运行时客户机和服务器之间发送的数据包，并且根据截获的数据包内容进行分析。

【实验结果与分析】

1. UDP编程

*Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

udp.port == 9000

No.	Time	Source	Destination	Protocol	Length	Info
235	39.900611	192.168.123.111	192.168.123.111	UDP	44	61840 → 9000 Len=12

> Frame 235: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{...}

> Null/Loopback

> Internet Protocol Version 4, Src: 192.168.123.111, Dst: 192.168.123.111

> User Datagram Protocol, Src Port: 61840, Dst Port: 9000

- Source Port: 61840
- Destination Port: 9000
- Length: 20
- Checksum: 0x20ee [unverified]
- [Checksum Status: Unverified]
- [Stream index: 1]
- > [Timestamps]
- [Time since first frame: 0.00000000 seconds]
- [Time since previous frame: 0.00000000 seconds]
- UDP payload (12 bytes)

> Data (12 bytes)

Data: 48656c6c6f20576f726c6421

[Length: 12]

0000 02 00 00 00 45 00 00 28 bf c6 00 00 80 11 00 00E..(.....
0010 c0 a8 7b 6f c0 a8 7b 6f f1 90 23 28 00 14 20 ee ..{o..{o..#(..
0020 48 65 6c 6c 6f 20 57 6f 72 6c 64 21 Hello Wo rld!

源IP地址和目的IP地址均相同，源端口为61840，目的端口为9000，传输数据为"Hello World!"

2. TCP编程

*Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

tcp.port == 8001

No.	Time	Source	Destination	Protocol	Length	Info
2563	115.650218	127.0.0.1	127.0.0.1	TCP	56	65326 → 8001 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
2564	115.650263	127.0.0.1	127.0.0.1	TCP	56	8001 → 65326 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=65495 WS=256 SACK_PERM=1
2565	115.650287	127.0.0.1	127.0.0.1	TCP	44	65326 → 8001 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
2566	115.650850	127.0.0.1	127.0.0.1	TCP	53	8001 → 65326 [PSH, ACK] Seq=1 Ack=1 Win=262400 Len=9 [Malformed Packet]
2567	115.650868	127.0.0.1	127.0.0.1	TCP	44	65326 → 8001 [ACK] Seq=1 Ack=10 Win=2619648 Len=0
2662	127.472813	127.0.0.1	127.0.0.1	TCP	45	65326 → 8001 [PSH, ACK] Seq=1 Ack=10 Win=2619648 Len=1
2663	127.472872	127.0.0.1	127.0.0.1	TCP	44	8001 → 65326 [ACK] Seq=10 Ack=2 Win=2100992 Len=0
2664	127.473680	127.0.0.1	127.0.0.1	TCP	44	8001 → 65326 [FIN, ACK] Seq=10 Ack=2 Win=2100992 Len=0
2665	127.473703	127.0.0.1	127.0.0.1	TCP	44	65326 → 8001 [ACK] Seq=2 Ack=11 Win=2619648 Len=0
2666	127.473828	127.0.0.1	127.0.0.1	TCP	44	65326 → 8001 [FIN, ACK] Seq=2 Ack=11 Win=2619648 Len=0
2667	127.473847	127.0.0.1	127.0.0.1	TCP	44	8001 → 65326 [ACK] Seq=11 Ack=3 Win=2100992 Len=0

[Stream index: 84]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 9]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 1331316000

[Next Sequence Number: 10 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 4179708136

0101 = Header Length: 20 bytes (5)

Flags: 0x018 (PSH, ACK)

Window: 1025

[Calculated window size: 262400]

[Window size scaling factor: 256]

Checksum: 0xff74 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> [Timestamps]

> [SEQ/ACK analysis]

TCP payload (9 bytes)

> [Malformed Packet: LBMSRS]

0000 02 00 00 00 45 00 00 31 88 eb 40 00 06 00 00E..1..@....
0010 7f 00 00 01 7f 00 00 01 1f 41 ff 2e 4f 5a 45 20A..OZE
0020 f9 21 48 e8 50 18 04 01 ff 74 00 00 07 65 6c 63 ..!H.P....t..
0030 6f 64 65 20 21 ood!

正在捕获 Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(J) 捕获(C) 分析(A) 统计(S) 电话(W) 工具(T) 帮助(H)

tcp.port == 8001

No.	Time	Source	Destination	Protocol	Length	Info
38	3.114502	127.0.0.1	127.0.0.1	TCP	56	53731 → 8001 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
39	3.114546	127.0.0.1	127.0.0.1	TCP	56	8001 → 53731 [SYN, ACK] Seq=0 Ack=1 Win=0 B192 Len=0 MSS=65495 WS=256 SACK_PERM=1
40	3.114558	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
88	5.039875	127.0.0.1	127.0.0.1	TCP	45	53731 → 8001 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1
89	5.039904	127.0.0.1	127.0.0.1	TCP	44	8001 → 53731 [ACK] Seq=1 Ack=2 Win=262400 Len=0
90	5.039929	127.0.0.1	127.0.0.1	TCP	45	53731 → 8001 [PSH, ACK] Seq=2 Ack=1 Win=2619648 Len=1
91	5.039940	127.0.0.1	127.0.0.1	TCP	44	8001 → 53731 [ACK] Seq=1 Ack=3 Win=262400 Len=0
92	5.039953	127.0.0.1	127.0.0.1	TCP	45	53731 → 8001 [PSH, ACK] Seq=3 Ack=1 Win=2619648 Len=1
93	5.039961	127.0.0.1	127.0.0.1	TCP	44	8001 → 53731 [ACK] Seq=1 Ack=4 Win=262400 Len=0
94	5.039972	127.0.0.1	127.0.0.1	TCP	45	53731 → 8001 [PSH, ACK] Seq=4 Ack=1 Win=2619648 Len=1
95	5.039980	127.0.0.1	127.0.0.1	TCP	44	8001 → 53731 [ACK] Seq=1 Ack=5 Win=262400 Len=0
96	5.039998	127.0.0.1	127.0.0.1	TCP	45	53731 → 8001 [PSH, ACK] Seq=5 Ack=1 Win=2619648 Len=1
97	5.039998	127.0.0.1	127.0.0.1	TCP	44	8001 → 53731 [ACK] Seq=1 Ack=6 Win=262400 Len=0
98	5.050085	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=1 Ack=6 Win=262400 Len=1
99	5.050112	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=2 Win=2619648 Len=0
100	5.050146	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=2 Ack=6 Win=262400 Len=1
101	5.050155	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=3 Win=2619648 Len=0
102	5.050175	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=3 Ack=6 Win=262400 Len=1
103	5.050184	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=4 Win=2619648 Len=0
104	5.050202	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=4 Ack=6 Win=262400 Len=1
105	5.050210	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=5 Win=2619648 Len=0
106	5.050230	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=5 Ack=6 Win=262400 Len=1
107	5.050240	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=6 Win=2619648 Len=0
108	5.050257	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=6 Ack=6 Win=262400 Len=1
109	5.050265	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=7 Win=2619648 Len=0
110	5.050283	127.0.0.1	127.0.0.1	TCP	45	8001 → 53731 [PSH, ACK] Seq=7 Ack=6 Win=262400 Len=1
111	5.050291	127.0.0.1	127.0.0.1	TCP	44	53731 → 8001 [ACK] Seq=6 Ack=8 Win=2619648 Len=0

三次握手

Destination Port: 8001
[Stream index: 3]
[Conversation completeness: Complete, WITH_DATA (63)]
[TCP Segment Len: 0]
Sequence Number: 16 (relative sequence number)
Sequence Number (raw): 175888351

0000 02 00 00 00 45 00 00 28 08 63 40 00 00 06 00 00 ... E { c@ ...
0010 7f 00 00 01 7f 00 00 01 41 e3 1f 41 8a 7b d7 df A { ..

中间传输数据时为一个字节一帧进行传输

三. 讨论、心得

记录实验感受、上机过程中遇到的困难及解决办法、遗留的问题、意见和建议等。