 网络协议分析

**运输层协议**

2019/3/21

1. **协议分析**

协议构造：

由于tcp是基于流的协议，需确定传输数据的边界，即封帧。

本实验使用**struct包**进行封帧，传输数据格式为

|  |  |  |
| --- | --- | --- |
| 数据长度 | 数据类型 | 数据 |
| struct.pack('!h', len(data)) | recv\_type.encode('utf8') | data.encode('utf8') |

说明：

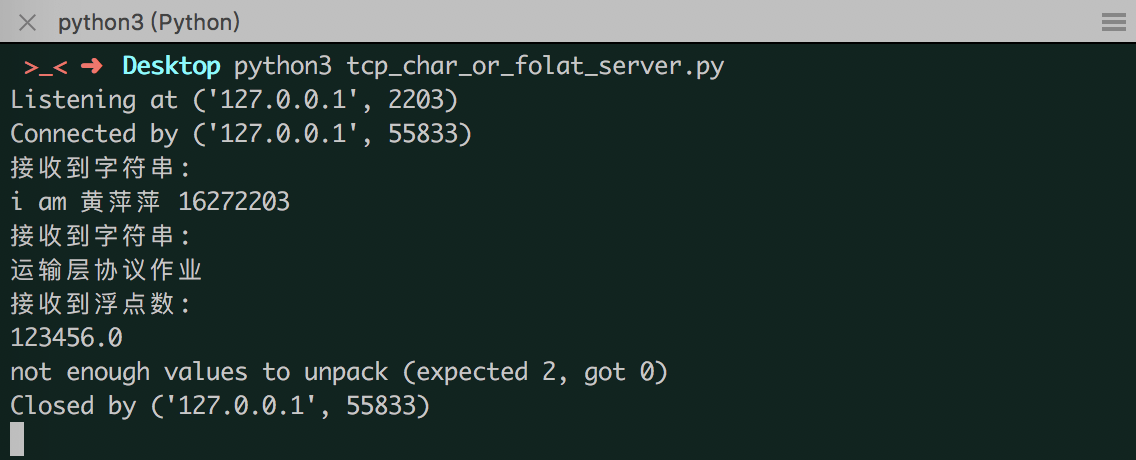
**数据长度**：使用short int封装应用数据的长度，即struct.pack('!h', len(data))；

**数据类型**：用一个字符表示应用数据最后的类型，即'c'--string,'f'—float；

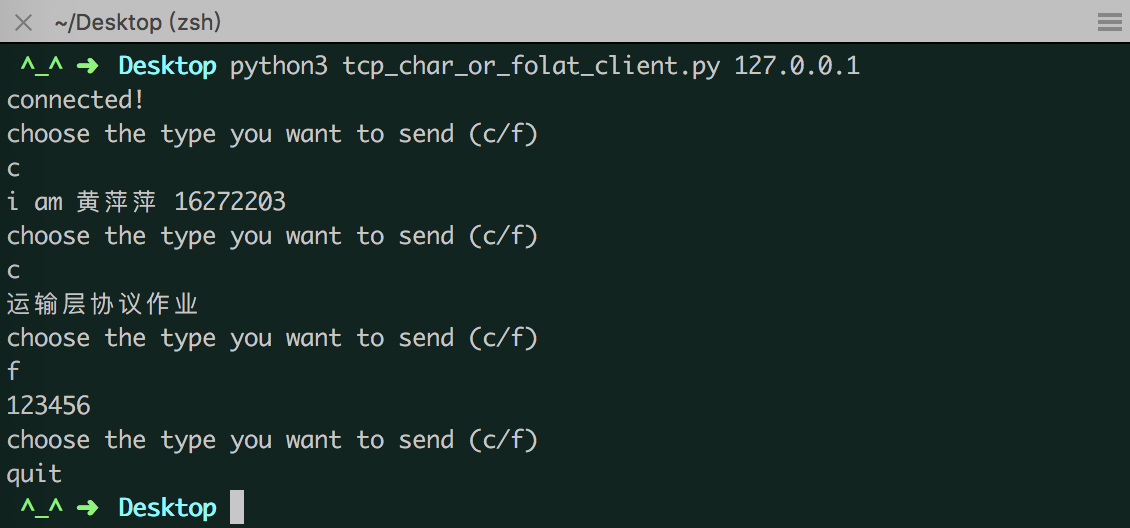
**数据**：data部分,使用utf8编码。

1. **实验结果**

服务端：

****

客户端：

****

1. **源码**

服务端

1. #!/usr/bin/env python3
3. **import** argparse, socket
4. **import** datetime
5. **import** threading
6. **from** sock\_tools **import** \*
7. **import** struct
8. **import** string

11. **def** recvPacket(sock):
12. data = b''
13. len\_data = sock\_readn(sock, 2)
14. **if** (len(len\_data) < 2):
15. **return** data
16. length, = struct.unpack('!h', len\_data)
17. recv\_type = sock\_readn(sock, 1)
18. **if** (len(recv\_type) < 1):
19. **print**('recv type error')
20. recv\_type=recv\_type.decode('utf8')
22. data = sock\_readn(sock, length)
24. **return** recv\_type,data

27. **def** serveClient(sc):
28. **while**(True):
29. **try**:
30. recv\_type,mesg = recvPacket(sc)
31. **if** (mesg == b''):
32. **break**
33. mesg=mesg.decode("utf8")
34. **if**(recv\_type)=='c':
35. mesg = str(mesg)
36. **print**('接收到字符串:')
37. **else**:
38. mesg = float(mesg)
39. **print**('接收到浮点数:')
40. **print**(mesg)
41. **except** Exception as e:
42. **print**(e)
43. **break**
44. **print**('Closed by {}'.format(sc.getpeername()))
45. sc.close()

48. **def** server(host, port):
49. sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)
50. sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)
51. sock.bind((host, port))
52. sock.listen(15)
53. **print**('Listening at', sock.getsockname())
54. **while** True:
55. sc, sockname = sock.accept()
56. **print**('Connected by {}'.format(sockname))
57. serveClientThread = threading.Thread(target = serveClient, args=(sc,))
58. serveClientThread.start()
59. sock.close()

62. **if** \_\_name\_\_ == '\_\_main\_\_':
63. parser = argparse.ArgumentParser(description='TCP Echo Server')
64. parser.add\_argument('-host', help='server host or ip adress',type=str,default='127.0.0.1')
65. parser.add\_argument('-p', '--port', metavar='PORT', type=int, default=2203,
66. help='server port (default 1060)')
67. args = parser.parse\_args()
68. server(args.host, args.port)

客户端

1. #!/usr/bin/env python3
3. **import** argparse, socket
4. **import** threading
5. **import** sys
6. **from** sock\_tools **import** \*
7. **import** struct
8. **import** string
10. BUFSIZE = 65535
12. **def** sendPacket(sock, data,send\_type):
13. **if** send\_type =='c':
14. sock.sendall(struct.pack('!h', len(data)) + 'c'.encode('utf8')+data)
15. **else**:
16. sock.sendall(struct.pack('!h', len(data)) + 'f'.encode('utf8')+data)

19. **def** readKeyboard(sock):
20. **while**(True):
22. **while** True:
23. **print**('choose the type you want to send (c/f)')
24. send\_type = input('')
25. **if** (send\_type == "quit"):
26. sock.close()
27. sys.exit(0)
28. **if** send\_type **in** ('c','f'):
29. **break**
30. **else**:
31. **print**('error type')
33. mesg = input('')
35. sendPacket(sock, mesg.encode("utf8"),send\_type)

38. **def** client(host, port):
39. sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)
40. **try**:
41. sock.connect((host, port))
42. **except** Exception as e:
43. **print**(e)
44. **return**
46. **print**("connected!")
48. keyInputThread = threading.Thread(target = readKeyboard, args=(sock,))
49. keyInputThread.start()

52. **if** \_\_name\_\_ == '\_\_main\_\_':
53. parser = argparse.ArgumentParser(description='TCP Echo Client')
54. parser.add\_argument('host', help='server host or ip adress')
55. parser.add\_argument('-p', '--port', metavar='PORT', type=int, default=2203,
56. help='server port (default 1060)')
57. args = parser.parse\_args()
58. client(args.host, args.port)

班级：16272412

学号：16272203

姓名：黄萍萍