作业：

- 要求实现tcp聊天器
- 客户端

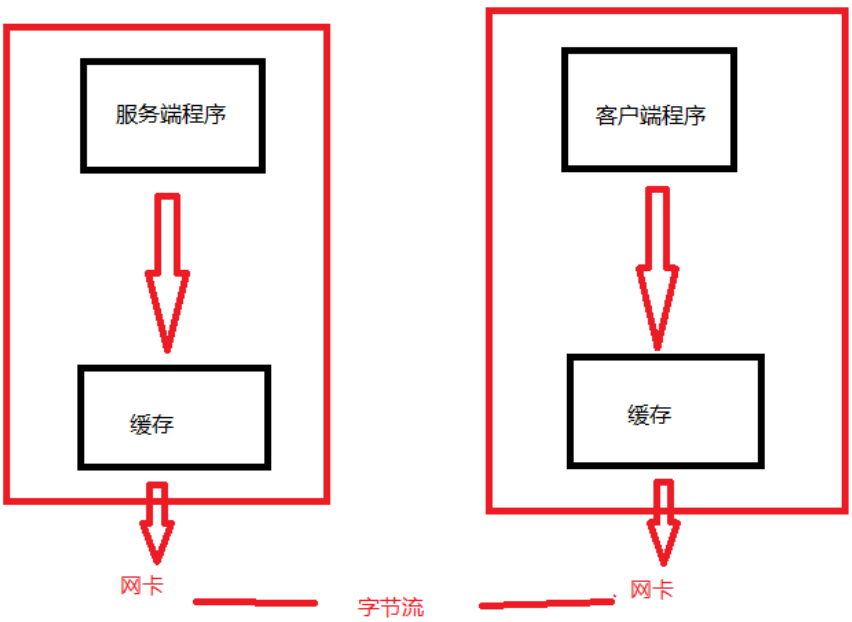
```python
from socket import *

tcp_cli_socket = socket(AF_INET, SOCK_STREAM)

# 连接服务器
serveraddr = ("192.168.61.114", 6666)
tcp_cli_socket.connect(serveraddr)

while True:
    # 提示用户输入数据
    sendata = input("请输入你要发送的内容：").encode("gbk")
    if len(sendata) > 0:
        tcp_cli_socket.send(sendata)
    else:
        break

    # 接收对方发送过来的数据
    recvdata = tcp_cli_socket.recv(1024)
    print(recvdata.decode("gbk"))

# 关闭套接字
tcp_cli_socket.close()
```

- 服务端

```python
from socket import *

tcp_server_socket = socket(AF_INET, SOCK_STREAM)

# 绑定本地信息
address = ("192.168.61.114", 6666)
tcp_server_socket.bind(address)

# 设置监听套接字
tcp_server_socket.listen(5)
```

```python

while True:
    # 如果有新的客户端连接服务器，那么产生一个新的套接字为这个客户端服务
    newsocket, clientaddr = tcp_server_socket.accept()

    while True:
        # 接收对方发送的数据
        recvdata = newsocket.recv(1024)

        if len(recvdata) > 0:
            print(recvdata.decode("gbk"))
        else:
            break

        # 发送一些数据到客户端
        senddata = input("请输入发送给客户端的信息").encode("gbk")
        newsocket.send(senddata)

    newsocket.close()
```

- 什么是粘包（现象）
  - 只有tcp有粘包现象，udp不会有
  - 如下是socket收发消息的原理



- 发送端可以是一K一K地发送数据，而接收端的应用程序可以两K两K地提走数据，当然也有可能一次提走3K或6K数据，或者一次只提走几个字节的数据，也就是说，应用程

序所看到的数据是一个整体，或说是一个流（stream），一条消息有多少字节对应用程序是不可见的，因此TCP协议是面向流的协议，这也是容易出现粘包问题的原因。而UDP是面向消息的协议，每个UDP段都是一条消息，应用程序必须以消息为单位提取数据，不能一次提取任意字节的数据，这一点和TCP是很不同的。怎样定义消息呢？可以认为对方一次性write/send的数据为一个消息，需要明白的是当对方send一条信息的时候，无论底层怎样分段分片，TCP协议层会把构成整条消息的数据段排序完成后才呈现在内核缓冲区。

- 例如基于tcp的套接字客户端往服务端上传文件，发送时文件内容是按照一段一段的字节流发送的，在接收方看来，根本不知道该文件的字节流从何处开始，在何处结束
- 所谓粘包问题主要还是因为接收方不知道消息之间的界限，不知道一次性提取多少字节的数据所造成的。
- 此外，发送方引起的粘包是由TCP协议本身造成的，TCP为提高传输效率，发送方往往要收集到足够多的数据后才发送一个TCP段。若连续几次需要send的数据都很少，通常TCP会根据优化算法把这些数据合成一个TCP段后一次发送出去，这样接收方就收到了粘包数据。

粘包现象示例

```
from socket import *

tcp_server_socket = socket(AF_INET, SOCK_STREAM)

# 绑定本地信息
address = ("192.168.61.114", 7777)
tcp_server_socket.bind(address)

# 设置监听套接字
tcp_server_socket.listen(5)

newsocket, clientaddr = tcp_server_socket.accept()

data1 = newsocket.recv(2)
data2 = newsocket.recv(10)

print('====>', data1.decode("gbk"))
print('====>', data2.decode("gbk"))
```

```
19
20 newsocket.close()
```

```
1  from socket import *
2
3  tcp_cli_socket = socket(AF_INET, SOCK_STREAM)
4
5  # 连接服务器
6  serveraddr = ("192.168.61.114", 7777)
7  tcp_cli_socket.connect(serveraddr)
8
9  tcp_cli_socket.send("hello".encode("gbk"))
10 tcp_cli_socket.send("kniom".encode("gbk"))
```

粘包现象解决方法

- 问题的根源在于，接收端不知道发送端将要传送的字节流的长度，所以解决粘包问题的方法就是围绕，如何让接收端知道发送端将要发送的字节流长度（提前知道）
- 服务端

```
1  from socket import *
2
3  ser = socket(AF_INET, SOCK_STREAM)
4  ser.bind(("192.168.61.114", 8888))
5  ser.listen(5)
6
7  while True:
8      new, addr = ser.accept()
9      while True:
10         msg = new.recv(1024)
11         if not msg:
12             break
13         print(msg.decode("utf-8"))
14         str1 = "8月1日，安理会轮值主"
15         data_len = len(str1.encode("utf-8"))
16         # 服务器第一次响应客户端（发送的是长度）
17         new.send(str(data_len).encode("utf-8"))
18         data = new.recv(1024).decode("utf-8")
19         if data == "recv_ready":
```

```
            new.sendall(str1.encode("utf-8"))

    new.close()
```

- 客户端

```
from socket import *

cli = socket(AF_INET, SOCK_STREAM)
cli.connect(("192.168.61.114", 8888))

while True:
    msg = input("请输入数据:").strip()
    if len(msg) == 0:
        continue
    if msg == "quit":
        break

    cli.send(msg.encode('utf-8'))
    # 拿到服务器将要发送的数据长度
    length = int(cli.recv(1024).decode('utf-8'))
    print(length)
    # 告诉服务端我拿到了长度
    cli.send('recv_ready'.encode('utf-8'))

    send_size = 0
    recv_size = 0

    data = b""  # 1024 2048 3000
    while recv_size < length:  # length = 3000
        data += cli.recv(1024)
        recv_size = len(data)

    print(data.decode('utf-8'))
```