**TCP Socket Programming**

1. wireshark包捕获截图（client端口号为52332，server端口号为20000）

① TCP建立连接，client向server发送SYN报文请求建立连接，server发送SYN+ACK报文，client发送ACK报文完成三次握手过程。结果如图1所示。

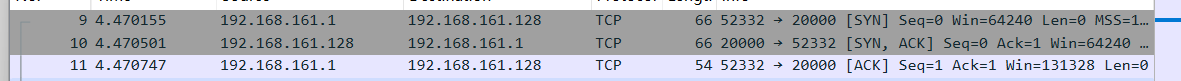


图1 三次握手

② client在启动时命令行输入SERVER\_IP、SERVER\_PORT、每一块数据的最小长度LMIN、最大长度LMAX以及需要读取的原文件FILE\_NAME。程序会读取文件并随机进行分块，输出总块数，向server发送包含总块数的Initialization报文，收到server回复的agree报文后，client发送包含待反转字符串的request报文开始发送数据，server每收到一条request报文后回复包含反转字符串的answer报文，数据交互结束后client需要输出一个新的文件，是原文件的全部反转。client具体过程如图2所示。四种类型报文分别如图3、4、5、6所示。

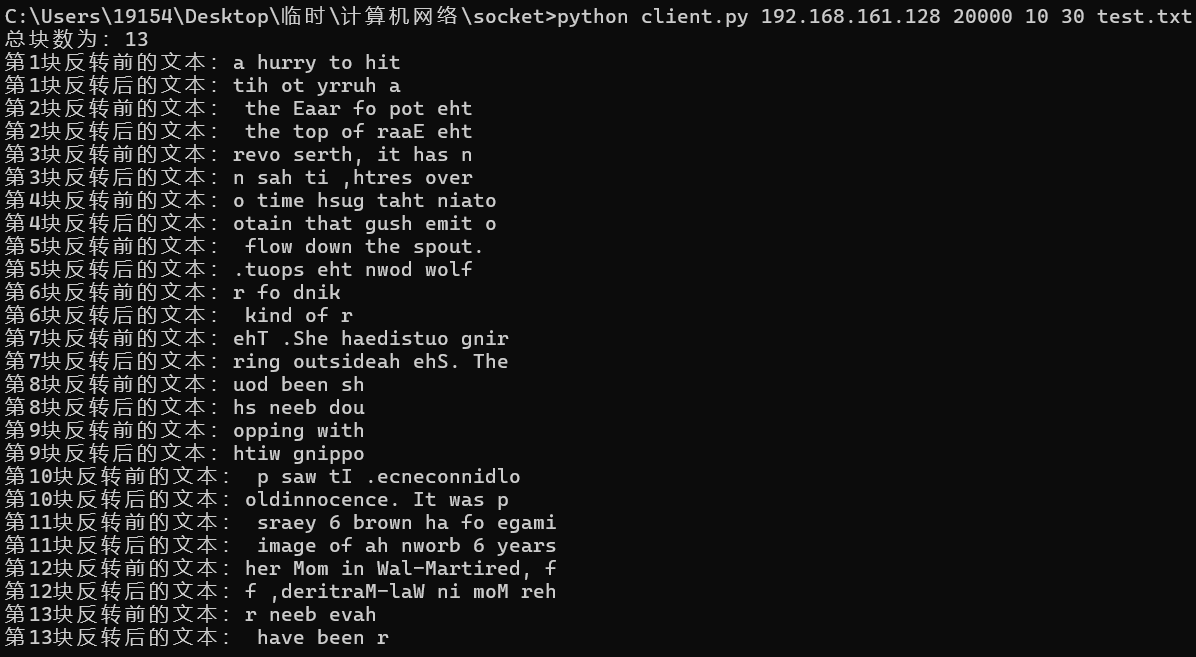


图2 client具体过程

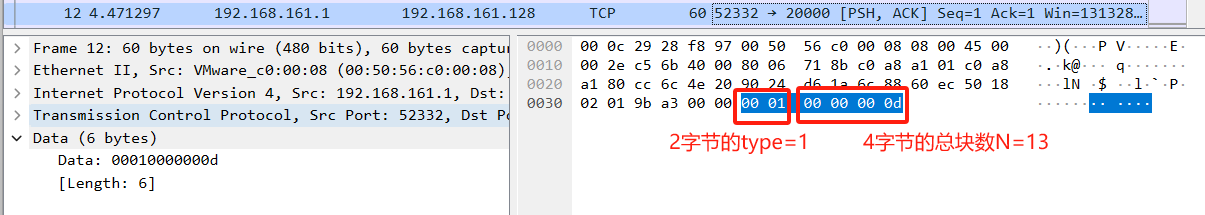


图3 client向server发送包含总块数的Initialization报文

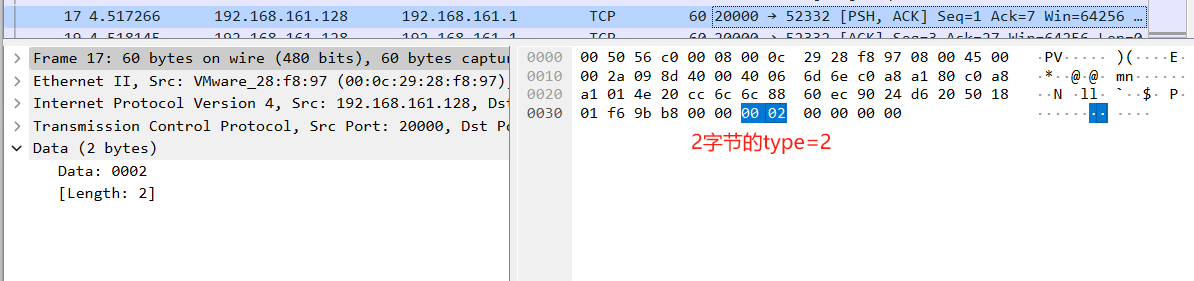


图4 server向client回复agree报文

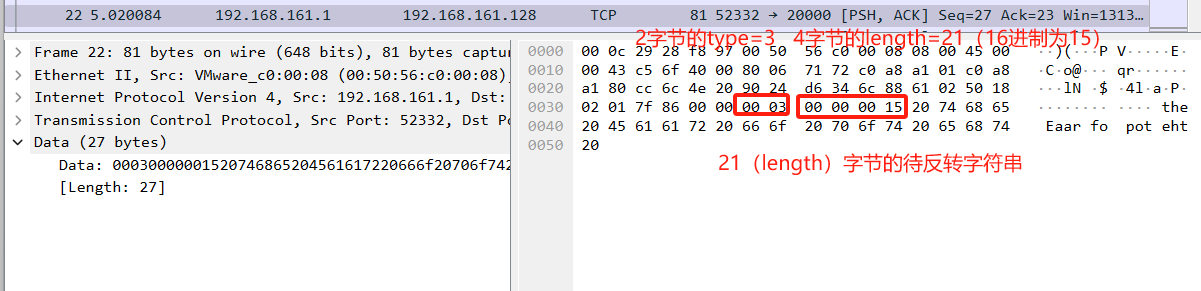


图5 client向server发送包含待反转字符串的request报文

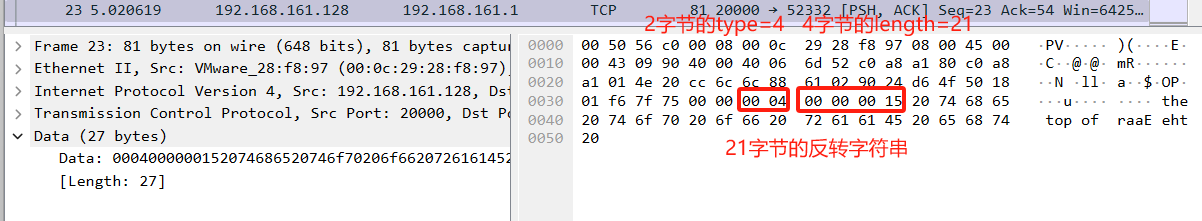


图6 server向client发送包含反转字符串的answer报文

③ TCP连接释放过程，client向server发送FIN+ACK报文请求释放连接，server发送ACK报文以及FIN+ACK报文请求释放连接，client发送ACK报文响应，完成四次挥手过程，如图7所示。

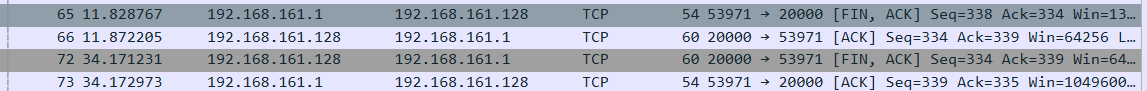


图7 四次挥手

1. 实现关键点和对应代码解决方案

① 如何对报文进行封装：使用python中struct库的pack和unpack函数可以将数据封装成字节序列便于进行发送。

② 如何将文件随机分成不同大小的块：使用random库中的randint函数随机生成位于[LMIN,LMAX]范围内的长度，定义记录位置的指针，通过切片操作将每一次随机生成的长度的数据存放在列表中，位置指针向后移动，文件最后位置的长度小于生成的随机长度则将剩余所有数据放入列表。

③ 如何将反转后的字符串写入文件，最后输出的文件是原文件的全部反转：判断是否第一次写入文件，若是第一次，则直接写入，若不是，则需要将文件内容读出，将文件指针指向开头位置，先写入新内容，再写入读出的旧的文件内容。

④ 服务器如何反转字符串并判断何时结束：使用切片操作反转字符串；使用一个变量记录当前处理的块数，一个变量记录总块数，当当前处理的块数和总块数相同，结束和该客户端的连接。

⑤ 如何处理多个客户端：传统方式：同步阻塞IO即BIO

同步非阻塞IO即NIO(使用select)——对应server\_nio.py文件：一个单独的线程 管理多个客户端，减少了阻塞和资源占用，可扩展性高。使用select函数监控不同socket，对列表中每个元素进行轮询，在可读状态中处理新的连接和客户端通信，在异常状态中移除错误套接字，同时需要为select设置超时时间，若30s内没有客户端连接，select超时返回三个空列表，关闭连接。缺点：不停轮询，占用较多的CPU资源。

异步阻塞IO即IO多路复用(使用epoll)——对应server\_io-multiplexing.py文件：epoll实现并发服务器，处理多个客户端，生成一个epllo选择器实例 I/O多路复用，监控多个socket连接，使用selectors.DefaultSelector()实现多路复用，在事件循环不断地调用select获取被激活的socket，同时需要为select设置超时时间，若30s内没有客户端连接，select超时返回一个空列表，关闭连接。

异步非阻塞IO即AIO(线程池伪异步)——对应server\_aio.py文件：使用accept接收客户端连接，并设置超时时间，若30s内没有客户端连接，则捕捉超时异常关闭连接。将每一个连接的客户端通过线程池异步提交，进行各自的数据交互。

实现效果：NIO与IO多路复用当两个客户端同时连接后，后与服务器建立连接的客户端需要等待前一个连接的客户端与服务器交互完成（因为前一个客户端一直有数据，因此轮询会一直在前一个客户端）。AIO中两个客户端可同时与服务器进行数据交互。

1. 用到并掌握的知识点

① 需要通过读取文件，将文件指针位置移动至开头才可以在文件开头追加数据。

② struct库中pack和unpack函数的用法，可以将不同字节的数据封装和解封装。

③ 可以通过设置timeout结束accept、select的阻塞状态。

④ 传统的IO模型为BIO同步阻塞IO，主线程不断轮询客户端有没有连接上来，然后建立线程池子，利用子线程处理数据读写。NIO同步非阻塞IO，利用select处理+socket(配置成非阻塞模式)实现，select在多个socket之间轮询，哪路有数据就处理哪路，某个连接读取数据时，可能也会阻塞，导致全局都被拖延。IO多路复用异步阻塞IO，在一个线程内同时处理多个socket的IO请求，可以注册多个socket，然后不断地调用select读取被激活的socket。AIO异步非阻塞IO，应用程序把数据读取任务提交给操作系统，等系统拿到数据后之间返回给应用程序，socket异步调用accept等函数。

1. Git

https://github.com/npcccccccccccccc/UDP-Socket-Programming