

天津大学

《计算机网络》课程设计 周进度报告



题目：TCP 连接的建立

学 号：	3019244160 3020202184
姓 名：	程子姝 刘锦帆
学 院：	智能与计算学部
专 业：	计算机科学与技术
年 级：	2020 级
任课教师：	石高涛

2022 年 9 月 18 日

目 录

第一章	任务要求	1
1.1	UDP 协议	1
1.2	环境和语言	1
1.3	主要功能	2
1.3.1	连接管理与可靠数据传输	2
1.3.2	流量控制与拥塞控制	2
第二章	总体设计	3
2.1	模块设计	3
2.2	数据结构设计	4
2.3	连接管理	4
第三章	连接建立的设计	5
第四章	连接建立的实现	6
4.1	实现原理	6
4.2	函数具体实现	6
第五章	连接建立的功能测试与结果分析	10
5.1	人员分工	10

一 任务要求

进行实验可以提高我们对 TCP 协议的理解，进一步了解网络协议，尤其是传输层协议的基本功能和实现方法，同时提高编程的能力。

1.1 UDP 协议

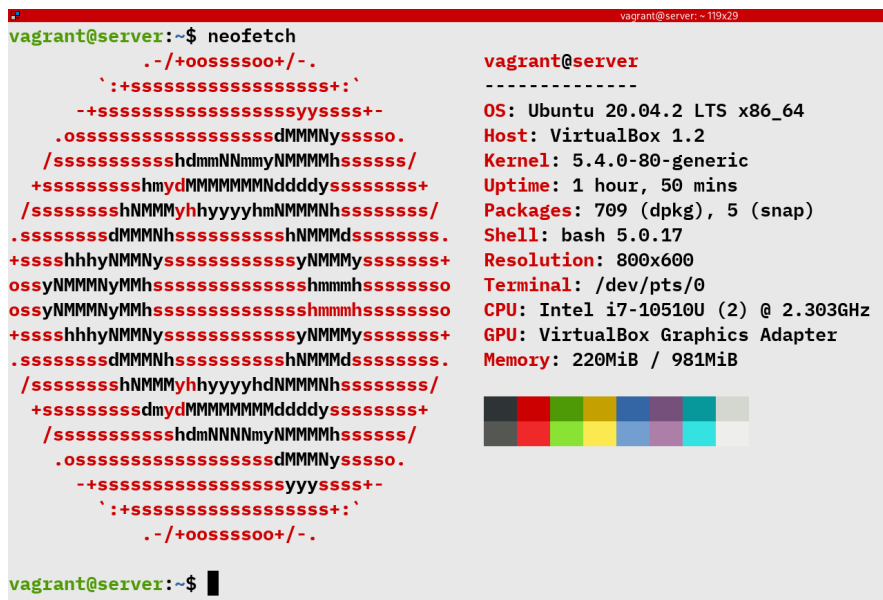
UDP 协议是基本的网络层协议，只负责将报文交给下方协议实体，不支持可靠性等 TCP 所支持的特性。但也因此会有更高的平均传输速率，适合流媒体等对可靠性要求不高，但对实时性要求高的服务。

实验中，我们采用 UDP 作为基本协议实体，在其之上参照 TCP 的实现标准，进行应用层层次的 TJU_TCP 的开发，模拟 TCP 协议实现的具体流程。

该程序可以完成简易的 TCP 协议功能，包括有 UDP 传输反映网络丢包情况、在丢包、乱序的情况下正确传输文件内容并还原序列顺序。在时间充足的情况下能进一步提升 TJU_TCP 功能如高并发等。

1.2 环境和语言

使用指定的 Linux 操作系统，C 语言实现 TJU_TCP 协议。系统配置如图1.2所示。



```

vagrant@server:~$ neofetch
      .-/+oosssso+/-.
      `:+ssssssssssssss++:`
      -+ssssssssssssssyyss+-
      .ossssssssssssssdMMMMyssso.
      /ssssssssshdmmNNmmyNMMMhsssss/
      +ssssssshmydMMMMMMNdddyssss+
      /ssssssshNMMMyhhyyyhmNMMNhsssss/
      .sssssssdMMMNhssssssshNMMdssssss.
      +sssshhhyNMMNysssssssssyNMMMyssss+
      ossyNMMMNyMMhssssssssssshmmhssssso
      ossyNMMMNyMMhssssssssssshmmhssssso
      +sssshhhyNMMNysssssssssyNMMMyssss+
      .sssssssdMMMNhssssssshNMMdssssss.
      /ssssssshNMMMyhhyyyhdNMMNhsssss/
      +sssssssdmydMMMMMMNdddyssss+
      /ssssssssshdmmNNNmyNMMMhsssss/
      .ossssssssssssssdMMMMyssso.
      -+ssssssssssssssyyss+-
      `:+ssssssssssssss++:`
      .-/+oosssso+/-.

vagrant@server:~$
  
```

```

vagrant@server
-----
OS: Ubuntu 20.04.2 LTS x86_64
Host: VirtualBox 1.2
Kernel: 5.4.0-80-generic
Uptime: 1 hour, 50 mins
Packages: 709 (dpkg), 5 (snap)
Shell: bash 5.0.17
Resolution: 800x600
Terminal: /dev/pts/0
CPU: Intel i7-10510U (2) @ 2.303GHz
GPU: VirtualBox Graphics Adapter
Memory: 220MiB / 981MiB
  
```

图 1-1 测试服务器配置信息

1.3 主要功能

实验内容可以大致划分为四个部分：连接管理、可靠数据传输、流量控制和拥塞控制。

1.3.1 连接管理与可靠数据传输

TCP 提供客户与服务器之间的连接。TCP 客户先与某个给定服务器建立一个连接，然后通过该连接与服务器交换数据，最后终止该连接。因此该部分包括如下功能：

- 1) 三次握手：建立连接；
- 2) 四次挥手：断开连接。

1.3.2 流量控制与拥塞控制

流量控制：TCP 匹配发送端与接收端的速度，防止缓冲区溢出；拥塞控制：防止过多的数据注入到网络中。需要实现的部分如下：

- 1) 流量控制；
- 2) 慢启动；
- 3) 拥塞避免；
- 4) 快速重传。

二 总体设计

根据 TCP 协议具有的功能和实践内容结合，本次实验的协议设计分为以下三个部分，分别是连接管理和流量控制、可靠数据传输和拥塞控制。

2.1 模块设计

根据指导书要求的主要功能，可将 TJU_TCP 分为如图2.1四个功能模块：

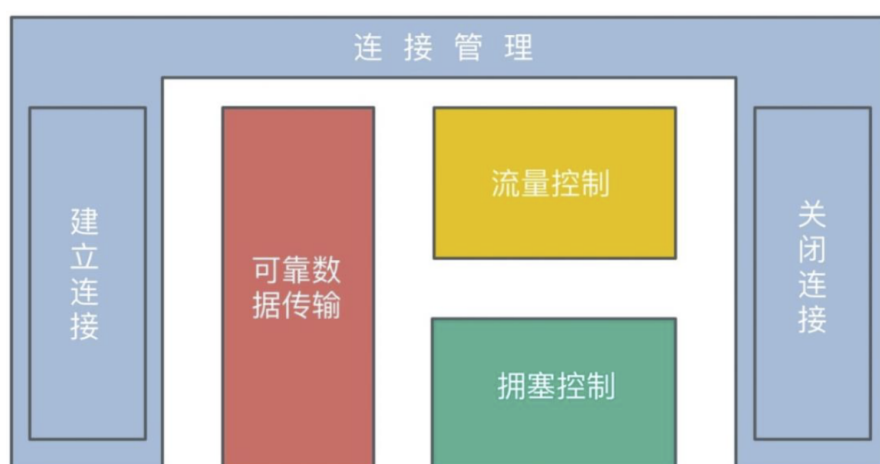


图 2-1 TCP 连接的功能模块

- 1) **连接管理模块** 分为建立连接模块与关闭连接模块，分别用于实现 TCP 连接的建立与拆除。
- 2) **可靠数据传输模块** TCP 的在 IP 层的不可靠、尽力而为服务的基础上建立的一种可靠数据传输服务。本实验中我们可以通多数个头部数据的维护做单简易的数据可靠性能，包括但不限于使用序列号和 ACK 信息、计时机制、重传机制、检验。
- 3) **流量控制模块** TCP 流量控制是在已经建立好连接的双方之间通信过程中避免让发送速率过慢浪费网络资源以及发送方发送速率过快导致接收方来不及计时接收导致较高延迟和网络拥塞的情况。我们利用滑动窗口机制在 TCP 连接上实现对发送方流量的控制。
- 4) **拥塞控制模块** 拥塞控制算法有四种状态，分别是慢启动、拥塞避免、快速重传和快速恢复。其针对的是如何在陌生或发现阻塞的网络中在规避死锁的情

况下尽可能地提高网络链路利用效率，使网络拥有动态地调节网络性能的能力。

2.2 数据结构设计

TJU_TCP 的包头是标准 TCP 包头的简化，共 20 个字节，如图2.2所示。

8bit	8bit	8bit	8bit
Source Port Number		Destination Port Number	
Sequence Number			
Acknowledgement Number			
Header Length		Packet Length	
Flags	Advertised Window		Extension

图 2-2 TCP 包头结构

2.3 连接管理

在实现三次握手时，除了框架代码中已提供的数据结构还增加了全连接表与半连接表。

```
#define MAX_SOCKET 32
tju_tcp_t* listen_socks[MAX_SOCKET];
tju_tcp_t* established_socks[MAX_SOCKET];
tju_tcp_t* syn_queue[MAX_SOCKET]; // 半连接表
tju_tcp_t* accept_queue[MAX_SOCKET]; // 全连接表
tju_tcp_t* bind_socks[MAX_SOCKET]; // 监听socket表
//tju_sock_q* sock_queue[MAX_SOCKET];
int accept_queue_n;
```

图 2-3 TCP 管理连接的结构

一个连接来到时，会先将它放到半连接的表中，暂存。等到其后的连接报文到来，又再将它从半连接提到全连接，并将总个数 +1。

三 连接建立的设计

该部分涉及到的 TCP 的状态转变如下图3-1所示：

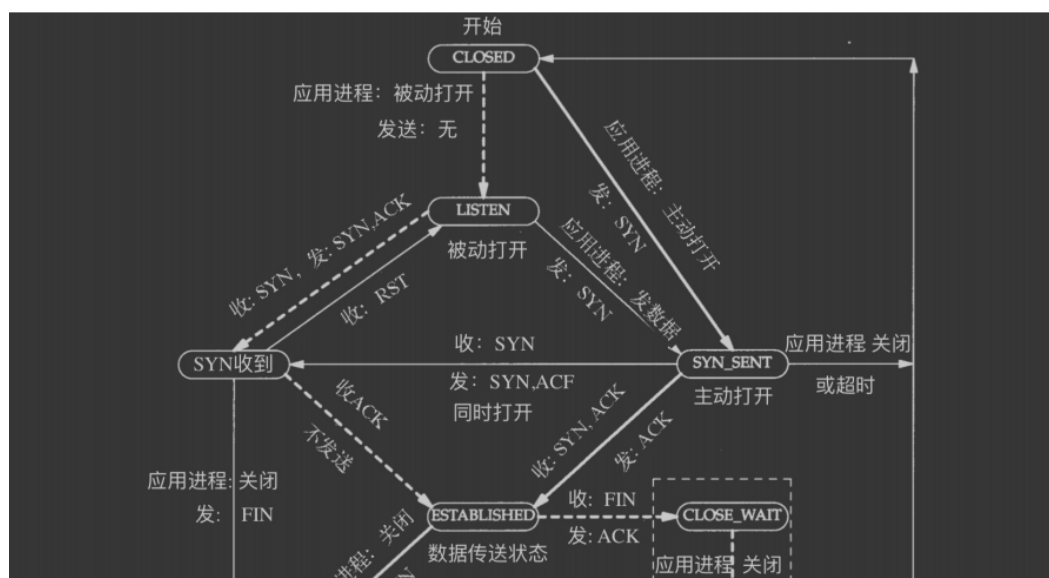


图 3-1 状态机

TCP 连接的建立采用的是 CS，即客户服务器方式。客户 A 主动发起连接建立的一方作为客户端，服务器 B 被动等待连接建立的一方是服务器端。TCP 连接建立所交换的三个 TCP 报文也称为三次握手过程。

具体的 TCP 连接建立过程是：

1. 初始时客户端和服务端状态都是链接关闭状态（CLOSED），客户端主动发送带有 SYN=1 和某一序列号 x 的报文交由服务器端，自身状态由关闭（CLOSED）转变成 SYN-SENT。
2. 客户端本身程序使其由不可连接的关闭（CLOSED）状态转变为可被连接的监听状态（LISTEN），在收到某一客户端带有 SYN=1 的报文后进入 SYN-RCVD 状态，同时往该客户端发送带有 SYN=1，ACK=1，确认号 $ack=x+1$ 和另一序列号 y 的报文。
3. 在接受到该报文后，原本处于 SYN-SENT 状态的客户端转变为连接已建立（ESTABLISHED），并且向服务端确认上一报文内容。服务器端接受到该确认后连接状态也随之转变成连接已建立状态（ESTABLISHED）。

四 连接建立的实现

4.1 实现原理

首先服务器需要做好接受外来连接的准备，通过调用函数 `tju_socket`、`tju_bind`、`tju_listen` 来实现，成为被动打开。在服务器 `socket` 进入监听状态后，客户端就可以发起三次握手：

1. 客户端调用 `tju_connect` 函数发起连接，向服务端发送 SYN 报文；
2. 服务端收到客户端发来的 SYN 报文后，向客户端发送 SYN_ACK；
3. 客户端收到 SYN_ACK 后向服务端发送 ACK 报文。

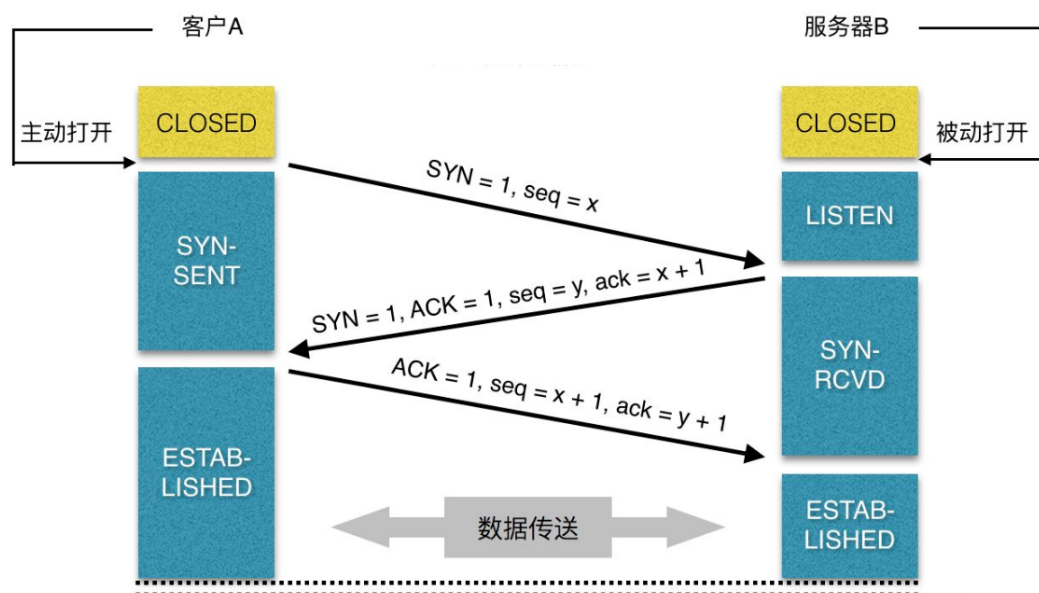


图 4-1 三次握手

4.2 函数具体实现

通过分析我们可知连接建立的实现主要集中在 `connect-accept` 阶段，在本次实验中就是 `tju_connect`、`tju_handle_packet`、`tju_accept` 三个函数。

tju_connect() `tju_connect()` 函数是三次握手的开始，需要完成如下操作：

1. 确定其 `socket` 使用的 ip 地址和端口号；
2. 向服务端发送 SYN 报文，将状态改为 `SYN_SENT`；
3. 阻塞等待，直到收到 `SYN_ACK`；
4. 收到 `SYN_ACK` 后向服务端发送 ACK 报文；

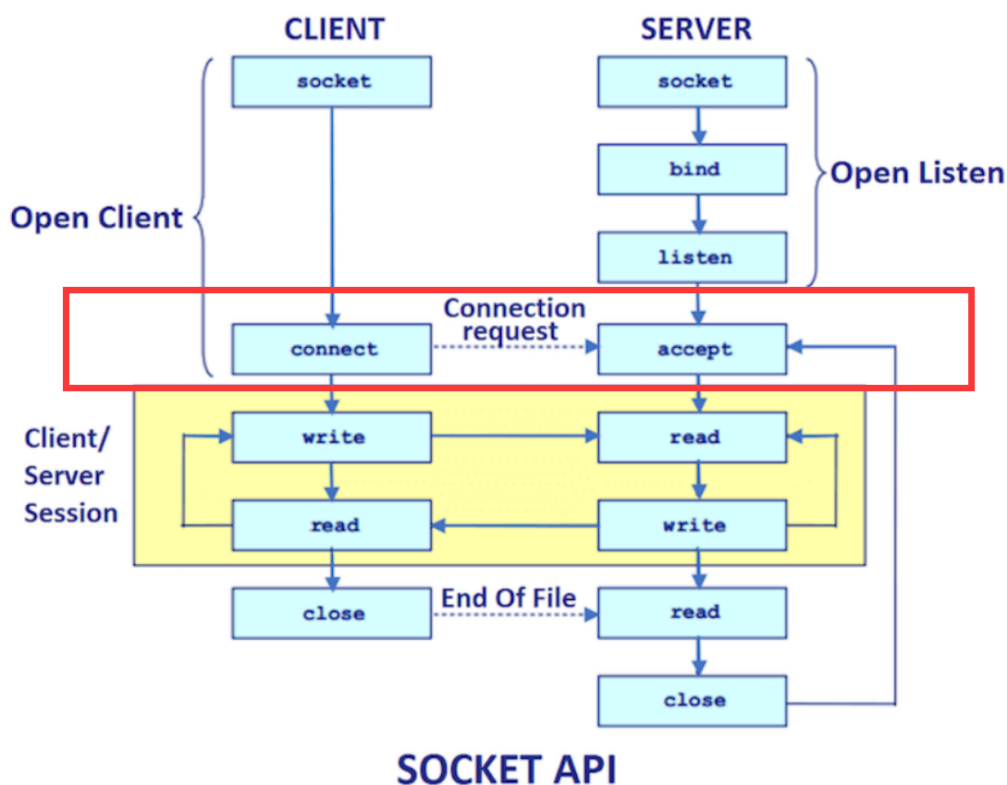


图 4-2 建立连接

5. 将 socket 的状态改为 ESTABLISH，并放入 Ehash 中；

6. 完成三次握手，返回。

具体实现中，由于服务端的 `tju_accept()` 函数需要返回的是一个完成三次握手的 socket，因而三次握手的过程不能放在 `tju_accept()` 函数中而是应该在 `tju_handle_packet()` 函数的处理 TCP 报文中完成。

流程图如图4-3 所示



图 4-3 流程图

tju_accept() tju_accept() 是服务端完成三次握手的主要函数，需要完成的任务如下：

1. 等待客户端的 SYN 报文；
2. 收到 SYN 报文后，向客户端返回 SYN_ACK 报文；
3. 将 socket 的状态改为 SYN_SENT 并放入半连接队列中；
4. 等待客户端的 ACK 报文；
5. 收到 SYN 报文后，取出半连接队列中的 socket，将其状态改为 ESTABLISH 并放入全连接队列；
6. 若全连接队列中有 socket，取出该 socket，记录到 Ehash 中并返回。

流程图如图4-4 所示



图 4-4 流程图

tju_handle_packet() 根据收到报文的类型来区分函数的实现：

1. 收到 SYN 报文
2. 判断收到报文的 socket 状态是否为 LISTEN，如果是，则说明目前还处于三次握手的环节，且收到该报文的是服务端；
3. 判断收到报文的类型，如果是 SYN 报文，说明还处于三次握手的第一阶段，执行 accpet 函数中的 (2)、(3) 步；
4. 将 socket 存入半连接队列中

伪代码如下：

```

1      // 判断报文类型
2  if  收到SYN报文：
3      if  报文socket状态==LISTEN：
4      执行tju_accept()中2、3步骤
  
```

```
5      // 将socket 存入半连接队列
6      tju_tcp_t  *new_sock
7      new_sock->state  =  SYN_SENT
8      syn_queue[hashval]  =  new_sock
9      // 向客户端发送SYN_ACK报文
10     creat_packet_buf()
11     sendTOLayer3()
12  if  收到ACK报文:
13     if  socket->state  ==  SYN_SENT:
14     // 将socket 存入全连接队列
15     sock->established_local_addr赋值
16     sock->established_remote_addr赋值
17     sock->state  =  ESTABLISH
18     accept_queue[hashval]  =  sock
19  return  sock
```

五 连接建立的功能测试与结果分析

结果如图，测试网站：

```
[建立连接测试] 对客户端进行评分 若正确发送第一次握手SYN数据包 +20分 若正确响应SYNACK +40分

[建立连接测试] 客户端建立连接部分的得分为 60 分

[建立连接测试] 建立连接部分的得分为 100 分
=====

===== 各项测试项目得分汇总 =====
{"scores": {"establish_connection": 100}}

Score for this problem: 100.0
```

图 5-1 网站测试结果

本地测试如图5-2 所示：

可以看到我们很快完成了任务，同时为了方便后期任务的顺利进行，我们对项目结构添加了 Log 等 debug 常用信息，以便在后期的挑战中更好地完成。

5.1 人员分工

人员分工如表5-1所示。

人员	项目分工
程子姝	完成大部分代码工作，以及协议实现部分
刘锦帆	完成 client 和 server 端的处理和 debug.h 以及写作

表 5-1 人员分工表

```

===== 建立连接的测试(服务端) =====

[建立连接测试] 开启服务端和客户端 将输出重定向到文件

[数据传输测试] 设置双方的网络通讯速率 丢包率 和延迟

[建立连接测试] 等待8s测试结束 测试服务端时三次握手应该在6s内完成

[建立连接测试] 打印文件里面的日志
=====
[ LOG ][Traceback: src/tju tcp.c:263,tju_handle_packet]服务端收到SYN报文
[ LOG ][Traceback: src/tju tcp.c:285,tju_handle_packet]服务端建立半连接并发送SYN_ACK报文
[客户端] 发送SYN 等待服务端第二次握手的SYN|ACK
[客户端] 收到一个TCP数据包
[客户端] 服务端发送的ACK|SYN报文检验通过，成功建立连接
{{GET SCORE}}
{{TEST SUCCESS}}
=====

[建立连接测试] 对服务端进行评分 正确发送第二次握手SYN|ACK数据包 +40分

[建立连接测试] 服务端建立连接部分的得分为 40 分

===== 建立连接的测试(客户端) =====

[建立连接测试] 开启服务端和客户端 将输出重定向到文件

[数据传输测试] 设置双方的网络通讯速率 丢包率 和延迟

[建立连接测试] 等待8s测试结束 测试客户端时三次握手应该在6s内完成

[建立连接测试] 打印文件里面的日志
=====
[服务端] 收到一个TCP数据包
[服务端] 客户端发送的第一个SYN报文检验通过
{{GET SYN SCORE}}
[服务端] 发送SYNACK 等待客户端第三次握手的ACK
[服务端] 收到一个TCP数据包
[服务端] 客户端发送的ACK报文检验通过，成功建立连接
{{GET ACK SCORE}}
{{TEST SUCCESS}}
[ LOG ][Traceback: src/tju tcp.c:163,tju_connect]客户端发送SYN报文
[ LOG ][Traceback: src/tju tcp.c:292,tju_handle_packet]客户端收到SYN_ACK报文
[ LOG ][Traceback: src/tju tcp.c:300,tju_handle_packet]客户端发送ACK报文
=====

[建立连接测试] 对客户端进行评分 正确发送第一次握手SYN数据包 +20分 正确响应SYNACK +40分

[建立连接测试] 客户端建立连接部分的得分为 60 分

[建立连接测试] 建立连接部分的得分为 100 分
=====
=====

```

图 5-2 本地测试结果