

**《计算机网络实践》课程报告**



**基于UDP的TCP协议设计与实现**

**学 号 3022206045 3022232100**

**姓 名 陆子毅 李国鸿**

**学 院 智能与计算学部**

**专 业 计算机科学与技术**

**年 级 2022级**

**任课教师 石高涛**

**2024年08月28日**

# 一、报告摘要

基于系统已有的UDP协议和接口，实现自定义TCP协议。协议设计主要包含连接管理、可靠数据传输、流量控制和拥塞管理四个部分。实验得到一个功能较为完善的TCP协议，可以用构建的TCP接口编写应用程序。

# 二、任务要求

## **2.1 连接管理**

需要实现TCP连接中建立连接的三次握手和TCP关闭连接时的四次挥手。根据TCP的状态转换图正确实现TCP连接建立过程，和TCP连接的关闭过程。

## **2.2 可靠数据传输**

TCP是在IP层提供的不可靠、尽力而为服务基础上建立的一种可靠数据传输服务。实验中需要实现超时重传，报文序列管理等功能。达到简易的可靠数据传输功能。

## **2.3 流量控制**

TCP的流量控制是指在已经建立好连接的双方之间，为了避免发送速率过快和过慢锁带来的过高丢包率，高延迟，网络拥塞和网络链路利用率不高的情况。主要利用滑动收发窗口来实现。

## **2.4 拥塞管理**

TCP的拥塞控制下，连接有四种状态，分别是慢启动，拥塞避免，快速重传和快速回复。

根据TCP的RFC和实践要求，详细描述实践任务需要解决的具体问题。主要解决在陌生网络环境中提高链路利用率，使TCP能够动态调节网络性能。

# 三、协议设计

**注意：协议设计的内容要涵盖“计算机网络课程实践说明书”中“二、TCP功能需求”所要求的所有内容。这是评分的依据。**

下面给出主要的章节。

## **3.1 总体设计**

分为四个板块，分别是，连接管理模块，可靠传输模块，流量控制模块，拥塞管理模块。

## **3.2 连接建立的设计**

TCP在建立连接过程中采用三次握手的方式，参考RFC 793文档中TCP建立连接的描述。“三次握手”是用于建立连接的过程。此过程通常由一个 TCP 启动，并由另一个 TCP 响应。如果两个 TCP 同时启动该过程，则该过程也有效。有丢包和延迟的情况下建立连接的功能，将会在可靠数据传输部分实现。

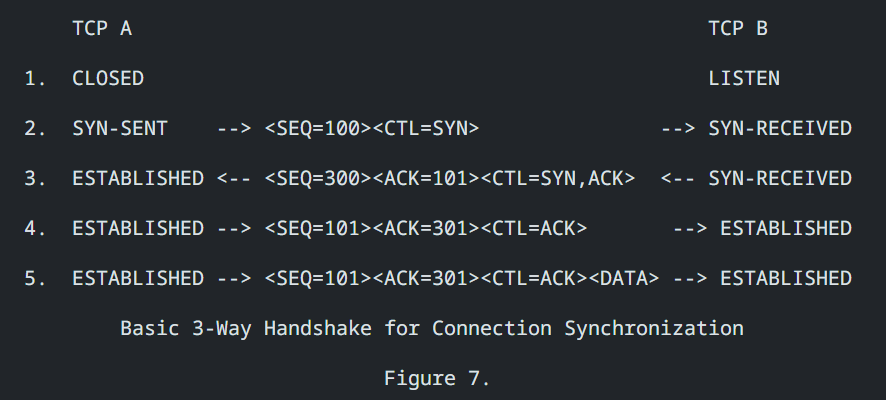


图1 RFC 793对于“三次握手”的图解

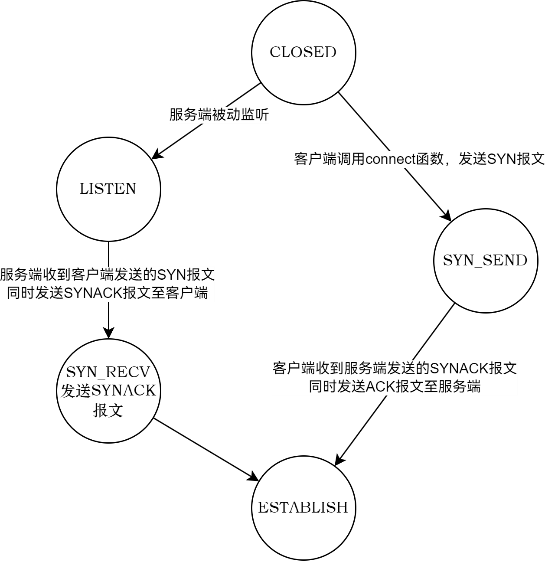


图2 三次握手的FSM图解

控制三次握手最主要的数据就是客户端和服务端的状态（state），需要根据客户端和服务端收到的不同的报文flag来进行不同的状态转换。同时在建立连接的过程中需要维护一个全连接队列，将已经完成三次握手的sock放入队列中，等待tju\_accept()函数的调用。建立完成时，需要服务端新建一个sock来建立与客户端的连接而不是直接使用listen sock作为建立的连接进行通信。

根据设计的“三次握手”协议，双方在建立连接的通信过程如下：

1. client调用tju\_connect函数，向server发送SYN报文请求连接。client状态转变为SYN\_SENT。server收到client发送的SYN报文，将server的连接状态转变为SYN\_RECV。
2. server发送SYNACK报文给client，client收到server发送的SYN\_ACK报文之后，将连接状态转变为ESTABLISH。
3. client向server发送ACK报文，确认连接建立，server收到ACK报文之后，新建一个sock并将连接状态变为ESTABLISH，并将新建立的sock放入全连接队列中

## **3.3 可靠数据传输的设计**

说明TCP可靠数据传输的主要数据结构和协议规则。用FSM图表示主要工作流程。

说明发送端和接收端缓冲区的管理，滑动窗口的设计方法。

## **3.4 流量控制的设计**

说明流量控制的原理和设计方法。

## **3.5 连接关闭的设计**

TCP连接的关闭采用“四次挥手”协议。TCP 是全双工通信，可以双向传输数据。任何一方都可以在数据传送结束后发出连接释放的通知，待对方确认后进入半关闭状态。当另一方也没有数据再发送的时候，则发出连接释放通知，对方确认后就完全关闭了 TCP 连接。

（1）第一次挥手：客户端发送一个 FIN（SEQ=x） 标志的数据包->服务端，用来关闭客户端到服务端的数据传送。然后客户端进入 FIN-WAIT-1 状态。

（2）第二次挥手：服务端收到这个 FIN（SEQ=X） 标志的数据包，它发送一个 ACK （ACK=x+1）标志的数据包->客户端 。然后服务端进入 CLOSE-WAIT 状态，客户端进入 FIN-WAIT-2 状态。

（3）第三次挥手：服务端发送一个 FIN (SEQ=y)标志的数据包->客户端，请求关闭连接，然后服务端进入 LAST-ACK 状态。

（4）第四次挥手：客户端发送 ACK (ACK=y+1)标志的数据包->服务端，然后客户端进入TIME-WAIT状态，服务端在收到 ACK (ACK=y+1)标志的数据包后进入 CLOSE 状态。此时如果客户端等待 2MSL 后依然没有收到回复，就证明服务端已正常关闭，随后客户端也可以关闭连接了。

## **3.6拥塞控制的设计**

说明拥塞控制的原理、主要数据结构和协议规则。用FSM图表示主要工作流程。

# 四、协议实现

详细描述功能实现的细节。主要功能模块使用流程图或者伪代码来辅助说明。禁止贴源码。

**注意：协议实现的内容要和“协议设计”部分相对应。每项功能设计都要有相应的实现。**

## **4.1 连接建立的实现**

（一）全连接队列

使用数据结构实现了全连接队列，用于存储已经可以用于通信的连接。具体实现参照C++数据结构队列的实现。在tju\_socket函数中完成队列的初始化。

在tju\_handle\_packet中，在server成功建立连接之后，将新建立的连接放入队列中等待tju\_accept函数调用取出。

同时tju\_accept函数中添加了阻塞，当队列中没有新建立的连接的时候会陷入等待，直到连接建立完成，返回可以直接用于通信的sock。

（二）三次握手建立连接

主要在tju\_handle\_packet函数和tju\_connect函数中实现。

首先，在tju\_connect函数中，当函数被调用，就向服务端发送一个SYN报文，请求建立连接。建立连接过程中，需要提前建立sock，并写入established\_socks哈希表中，避免之后因不存在能够处理服务端发送的SYNACK报文的sock而将包丢弃。随机进入阻塞状态直到sock的状态变为ESTABLISH。

然后，需要修改tju\_handle\_packet函数，该函数用于处理收到的报文。通过预值的提取函数，将报文的flag，seq，ack，src，des等字段提取出来，然后根据sock目前的状态，结合tcp连接的状态转换图进行相应的报文发送。

下面是tju\_handle\_packet函数的伪代码：

根据 sock 的状态进行处理

如果状态是 LISTEN

如果 flag 是 SYN\_FLAG\_MASK

更新状态为 SYN\_RECV

创建并发送 SYN-ACK 包

如果状态是 SYN\_SENT

如果 flag 是 SYN\_FLAG\_MASK | ACK\_FLAG\_MASK

创建并发送 ACK 包

更新状态为 ESTABLISHED

如果状态是 SYN\_RECV

如果 flag 是 ACK\_FLAG\_MASK

分配并初始化新连接 new\_conn

设置新连接的本地和远程地址

更新新连接状态为 ESTABLISHED

计算哈希值并存储新连接

将新连接加入接受队列

更新 sock 状态为 LISTEN

## **4.2 可靠传输的实现**

## **4.3 流量控制的实现**

## **4.4 连接关闭的实现**

与连接关闭相关的函数主要是tju\_close和tju\_handle\_packet。

1. tju\_close

由主动提出关闭连接的一方调用，调用时发送FINACK报文，告知对方想要关闭连接。

同时进入阻塞等待状态，等待自己sock的状态变为CLOSED完成连接关闭。

1. tju\_handle\_packet

与建立连接过程的逻辑相同，也是通过flag来控制状态的转变以及做出相应的反应。下面是具体实现的伪代码：

如果状态是 ESTABLISHED

如果 flag 是 FIN\_FLAG\_MASK | ACK\_FLAG\_MASK

创建并发送 ACK 包

更新状态为 CLOSE\_WAIT

创建并发送 FIN-ACK 包

调试信息 "FIN sent! sock state -> LAST\_ACK"

更新状态为 LAST\_ACK

如果状态是 FIN\_WAIT\_1

如果 flag 是 ACK\_FLAG\_MASK

更新状态为 FIN\_WAIT\_2

否则如果 flag 是 FIN\_FLAG\_MASK | ACK\_FLAG\_MASK

调试信息 "FIN received! sock state -> CLOSING"

创建并发送 ACK 包

更新状态为 CLOSING

如果状态是 FIN\_WAIT\_2

如果 flag 是 FIN\_FLAG\_MASK | ACK\_FLAG\_MASK

更新状态为 TIME\_WAIT

创建并发送 ACK 包

更新状态为 CLOSED

如果状态是 LAST\_ACK

如果 flag 是 ACK\_FLAG\_MASK

更新状态为 CLOSED

如果状态是 CLOSING

如果 flag 是 ACK\_FLAG\_MASK

更新状态为 TIME\_WAIT

更新状态为 CLOSED

## **4.5 拥塞控制的实现**

# 五、实验结果及分析

测试所实现协议的功能和性能，并对性能结果进行分析。需要针对考察点逐一展开。

## **5.1 连接建立的功能测试与结果分析**

线上测试结果和本地测试相同，客户端和服务器能够正常建立连接。连接建立功能基本实现。

图形用户界面, 网站

描述已自动生成

表格

低可信度描述已自动生成

表格

描述已自动生成

## **5.2 可靠传输的功能测试与结果分析**

## **5.3 流量控制的功能测试与结果分析**

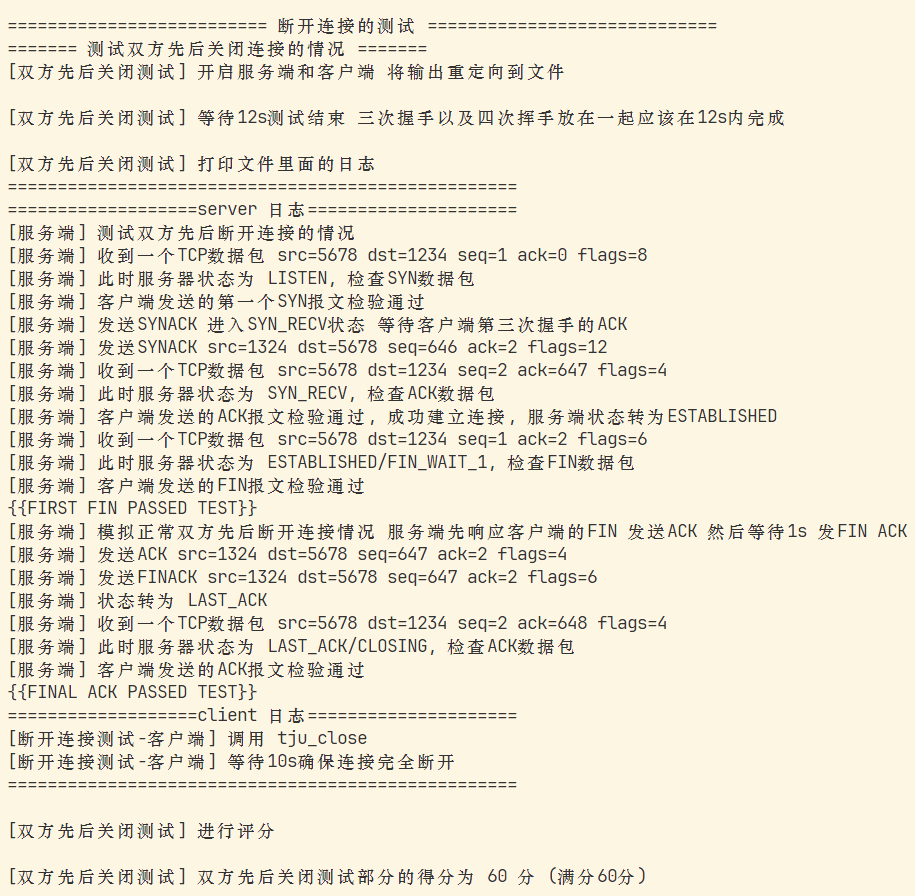
## **5.4 连接关闭的功能测试与结果分析**

客户端与服务端先后关闭连接和同时关闭连接的测试正常。连接关闭功能基本实现。

线上测试的日志基本与本地日志一致，只给出测试结果

图形用户界面, 文本, 应用程序, 电子邮件, 网站

描述已自动生成



文本

描述已自动生成

## **5.5 拥塞控制的功能测试与结果分析**

## **5.6 TCP协议性能测试与结果分析**

# 六、总结

总结在实践过程中遇到的各类问题、困难以及解决过程中的收获，对实践内容等方面的体会与建议。