南開大學
Nankai University

计算机学院

计算机网络实验

基于 UDP 服务设计可靠传输协议并编程实现

姓名：岳一名

学号：2212472

专业：计算机科学与技术

2024 年 11 月 29 日

目录

1 实验简述

本次实验利用数据报套接字在用户空间实现面向连接的可靠数据传输，功能包括：建立连接、差错检测、确认重传等。流量控制采用停等机制，完成给定测试文件的传输。

本次实验就是基础要求，之后的三次实验都是在这个实验的基础上进行修改，所以本次的实验要求就是实现基本的文件传输功能，同时进行性能的计算，比如时延以及吞吐率的计算。

2 实验操作流程

2.1 设置 Router.exe

首先我们设置路由器和服务器之间的通信，结果如下：通过上面的配置我们能够正确的进行我们

图 2.1: 路由器设置

的实验。

2.2 开始执行程序

我们首先执行 myself_server.exe，之后执行 myself_client.exe，开始执行之后程序会进行三次握手链接，之后会输出此次需要传输的包的数量，之后开始进行传输，首先会传输一张图片，之后就会传输一个 txt 文档，每当一个文件传输结束之后，就会在终端进行确认，是不是要继续传输，如果在传输完两个文件之后，我们选择终结传输，那么就会进行四次挥手，与服务器断开连接。

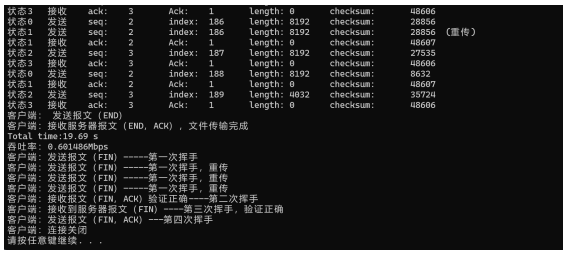

图 2.2: 客户端断开连接

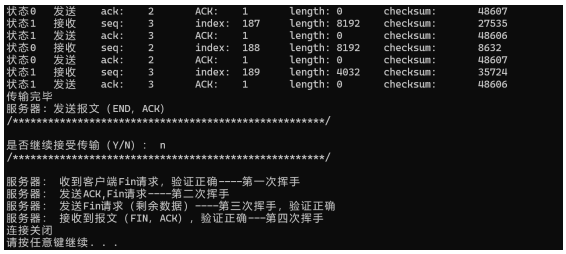

图 2.3: 服务端断开连接

上面两幅图片中，在客户端里面有着吞吐率的计算，有着四次挥手的标志，在服务端的截图里面，

我们能够看到四次挥手的输出，以及选择的操作，是继续接收文件，还是说终止输出，执行四次挥手的操作。

2.3 实验结果

我们的程序在写的过程中，设置了服务端将输出文件输出到文件夹 receive_file 里面，最后能够看到有一张图片以及一个 txt 文件在这个文件夹中，说明我们传输的结果是正确的。

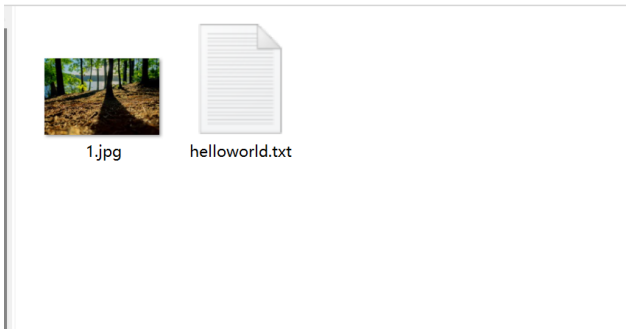

图 2.4: 最终结果

上面就是我们实验的操作流程以及效果下面我们将详细介绍实现细节以及所采用的协议。

3 详细分析

3.1 协议介绍

3.1.1 数据包的构建

下面我们可以来看报文的具体结构：在传输消息的时候，我们使用 message 结构题来构建我们的报文，具体的形式如下：

Listing 1: message 报文定义

```
struct message {
    WORD source_port = 0, dest_port = 0;  // port
    DWORD seq_num = 0, ack_num = 0;       // seq, ack
    WORD length = 0;                      // length
    BYTE flag = 0;                        // flag
    WORD checksum = 0;                    // 校验和
    char msg[Max_Size] = { 0 };           // 消息
};  // 报文
```

上面就是我们报文的结构，首先就是输入源端口的端口号，同时输入目标端口的端口号，同时设置我们的序列号 seq_num 以及 ack_num, 用于我们后续的判断（比如是不是接收的消息，或者序列号是不是确定的）。在上述基础消息设置清楚的时候，我们还需要调用函数来构建我们的校验和：

3.1.2 校验和的构建

下面是我们如何构建校验和的操作：

Listing 2: 校验和的构建

```
void setChecksum(message* messages, pseudoHead* ph) {
  //设为0
  messages->checksum = 0;
  int sum = 0;
  int len_pseudo = sizeof(pseudoHead);
  int len_msg = sizeof(message);
  for (int i = 0; i < len_pseudo / 2; i++) {
    sum += ((WORD*)ph)[i];
  }
  for (int i = 0; i < len_msg / 2; i++) {
    sum += ((WORD*)messages)[i];
  }
  while (sum >> 16) {
    sum = (sum & 0xffff) + (sum >> 16);
  }
  //就是将所有的数据按WORD(十六位)相加，超过十六位的数字平移到末尾再加上。
  //最后对结果取反就成为校验位。
  messages->checksum = ~sum;
};
```

上面就是我们设置校验和的函数，步骤就是将伪首部和报文分成十六位的数字，并且进行十六位反码加法（就是如果最后的结果超过十六位，九江超过十六位的部位右移，与第十六位相加，重复上面的操作，知道最后的结果在十六位一下），最后对相加的结果进行取反。这样能够构造出校验和。

同时校验校验和操作和构造校验和的操作一样，最后就是看结果是不是为十六位全是 1，即 0xffff，如果是这个结果的话，就说明我们传输的报文是没有 01 置换的，是正确的报文，这样就能继续下面的操作了。字符串数组 msg 就是我们存储发送文件数据的地方，每次大小在 8192 个字节。

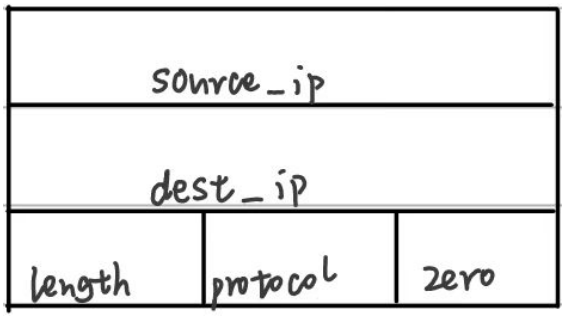

图 3.5: 伪首部结构

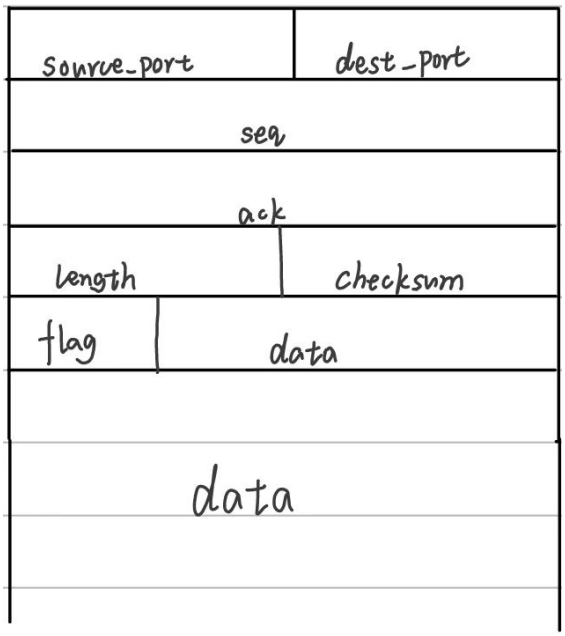

图 3.6: 报文结构

3.2 协议使用

3.2.1 发送端协议分析

此次实验使用的是可靠传输协议 tdt3.0，在客户端发送消息的时候会有 ack 的标识，同时对于 ack 我们还有特殊的标号 seq 来进行区分，这样可以使服务端能够区分究竟是当前传输的新的数据包，还是一个重新传入的数据包

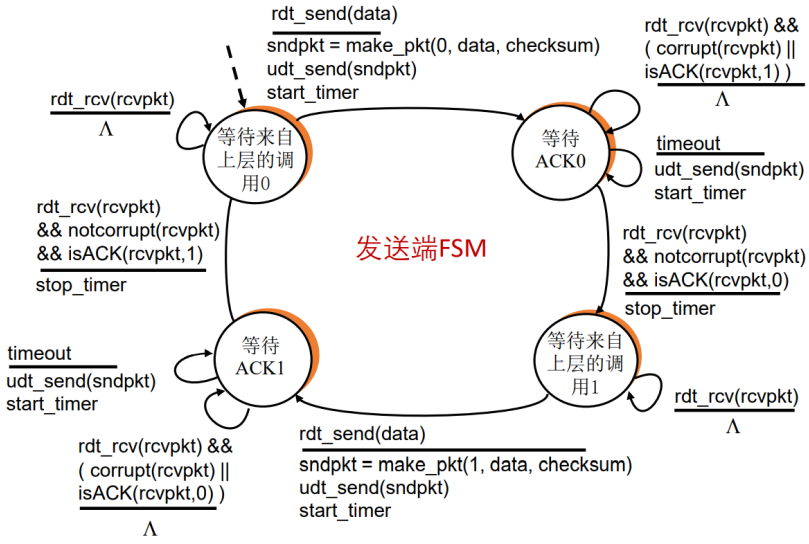

图 3.7: 发送端状态转移

借助上图我们分析发送端的状态转移，并且分析我们如何进行信息校验以及发起重传的。

首先在等待来自上层调用 0 状态开始传输，我们发送端首先发送一个数据包（我对于等待来自上层调用 0 状态的 ack 序列号设置为 2，同时等待来自上层调用 1 状态的 seq 设置为 3），并将其序列号设置为 2，之后设置状态为之后将状态设置为 1，即等待 ACK0，在这个状态中，我们设置时钟为 0.1s，如果超过这个限定时间没有收到来自接收端的正确的确认消息（即检验和正确并且序列号和上次发送的报文的序列号一致）我们就执行重传操作（虽然在程序里这里已经是 1 状态了，但是起始还是在执行 0 状态的操作）。如果我们在 1 状态接收到了正确的报文，我们就进入下一个状态，开始发送下一个数据包，和上一个报文的构造方式一样，只不过这次的序列号为 3，之后进入相同的等待状态。等待正确的报文发送回来，又进入状态 0，这样就是一个循环。

3.2.2 接收端协议分析

之后是接收端协议。在该协议中，接收端有两个主要状态：状态 0 和状态 1。初始时，接收端处于状态 0，并等待从发送端接收到报文。在状态 0 时，接收端接收到的是序列号为 2 的报文。为了确保数据传输的正确性，接收端会检查报文的检验和以及序列号。如果报文的检验和和序列号都正确，接收端将进入状态 1，并发送一个包含确认号的 ACK 报文，告知发送端接收到序列号为 2 的报文。

在状态 1 中，接收端继续等待接收到序列号为 3 的报文。在成功接收到序列号为 3 的报文并确认其检验和无误后，接收端再次进入状态 0，并发送相应的 ACK 报文。接收端会持续循环这一过程，直到整个文件的所有数据都传输完毕。这种状态的转换过程保证了文件数据的可靠接收。

除了基本的数据传输功能，协议中还引入了一个标志位 flag，用于标识报文的类型。flag 可以表示不同的报文类型，例如：SYN 报文用于建立连接，ACK 报文用于确认接收到的数据包，而 END 报文

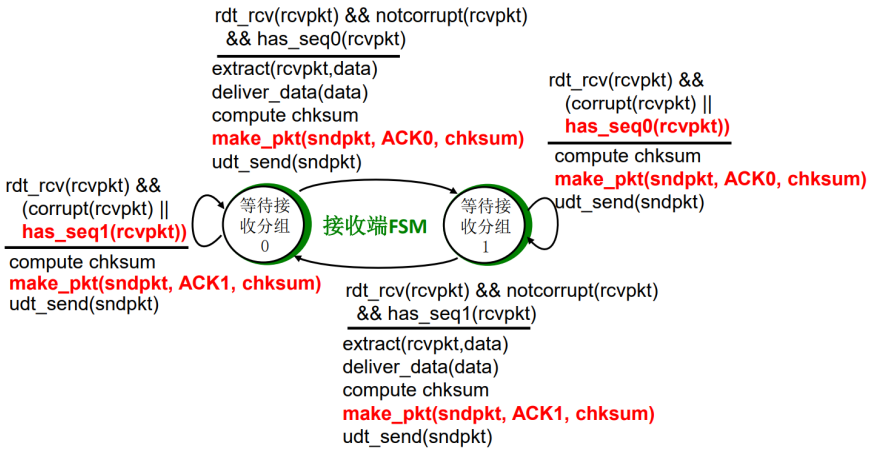

图 3.8: 接收端状态转移

则用于指示文件传输的结束。带有 END 标志的报文会在文件数据传输完毕后发送，用于告知接收端文件已经传输完毕，并传输文件的名称和格式信息。END 报文的传输标志着文件传输的完成。

通过这种基于状态机的方式，接收端能够准确地接收每个数据包，并通过适当的 ACK 和 END 报文来确保文件的完整传输，最终构建出整个文件。

4 实验总结

本次实验通过实现基于数据报套接字的可靠数据传输协议，成功完成了连接建立、差错检测、确认重传等功能，并通过停等机制实现了流量控制。实验中，我们测试了时延和吞吐率，验证了协议的稳定性与可靠性。通过对协议中序列号、确认机制、校验和等关键技术的分析，我们加深了对 TCP 协议的理解。在调试过程中，解决了数据丢失、重传延时、同步问题等挑战，并优化了协议性能。实验使我对网络协议的实现与优化有了更深入的认识，未来可以进一步改进为多线程支持、优化流量控制机制，提升协议的适应性和性能。

以上就是本次实验的解析，其他详细内容在压缩包里面有源代码以及相应的截图。