
华中科技大学计算机学院

《计算机通信与网络》实验报告

班级 CS2003 班 姓名 汪宇飞 学号 U202015375

项目	Socket 编程 (40%)	数据可靠传输协议设计 (20%)	CPT 组网 (20%)	平时成绩 (20%)	总分
得分					

教师评语：

教师签名：

给分日期：

目 录

实验二 数据可靠传输协议设计实验	1
1.1 环境	1
1.2 系统功能需求	1
1.3 系统设计	2
1.4 系统实现	3
1.5 系统测试及结果说明	7
1.6 其它需要说明的问题	13
1.7 参考文献	14
心得体会与建议	15
2.1 心得体会	15
2.2 建议	15

实验二 数据可靠传输协议设计实验

1.1 环境

1.1.1 开发平台

设备型号: XiaoXinPro 14ACH 2021

系统类型: 64 位操作系统, 基于 x64 的处理器

处理器: AMD Ryzen 7 5800H with Radeon Graphics 3.20GHz

内存: 16.0G

开发平台: Visual Studio 2019

开发语言: C++

1.1.2 运行平台

系统: Microsoft Windows 10

处理器: AMD Ryzen 7 5800H with Radeon Graphics 3.20GHz

开发平台: Visual Studio 2019

1.2 系统功能需求

1. 可靠运输层协议实验只考虑单向传输, 即: 只有发送方发生数据报文, 接收方仅仅接收报文并给出确认报文。
2. 要求实现具体协议时, 指定编码报文序号的二进制位数 (例如 3 位二进制编码报文序号) 以及窗口大小 (例如大小为 4), 报文段序号必须按照指定的二进制位数进行编码。
3. 代码实现不需要基于 Socket API, 不需要利用多线程, 不需要任何 UI 界面。
4. 分别实现 GBN、SR 的可靠传输协议, 在实现 GBN 的基础上根据 TCP 的可靠数据传输机制实现一个简化版的 TCP 协议, 要求:
 - ✓ 报文段格式、接收方缓冲区大小和 GBN 协议一样保持不变;
 - ✓ 报文段序号按照报文段为单位进行编号;
 - ✓ 单一的超时计时器, 不需要估算 RTT 动态调整定时器 Timeout 参数;
 - ✓ 支持快速重传和超时重传, 重传时只重传最早发送且没被确认的报文段;
 - ✓ 确认号为收到的最后一个报文段序号;

- ✓ 不考虑流量控制、拥塞控制。

1.3 系统设计

按照任务书要求，本次实验所实现的是位于模拟网络环境所提供的应用层和网络层之间的运输层 RDT 协议内容，所实现的协议内容与实验提供的模拟网络环境一同实现数据的可靠传输。实验所提供的模拟网络环境如图 1.1 所示。

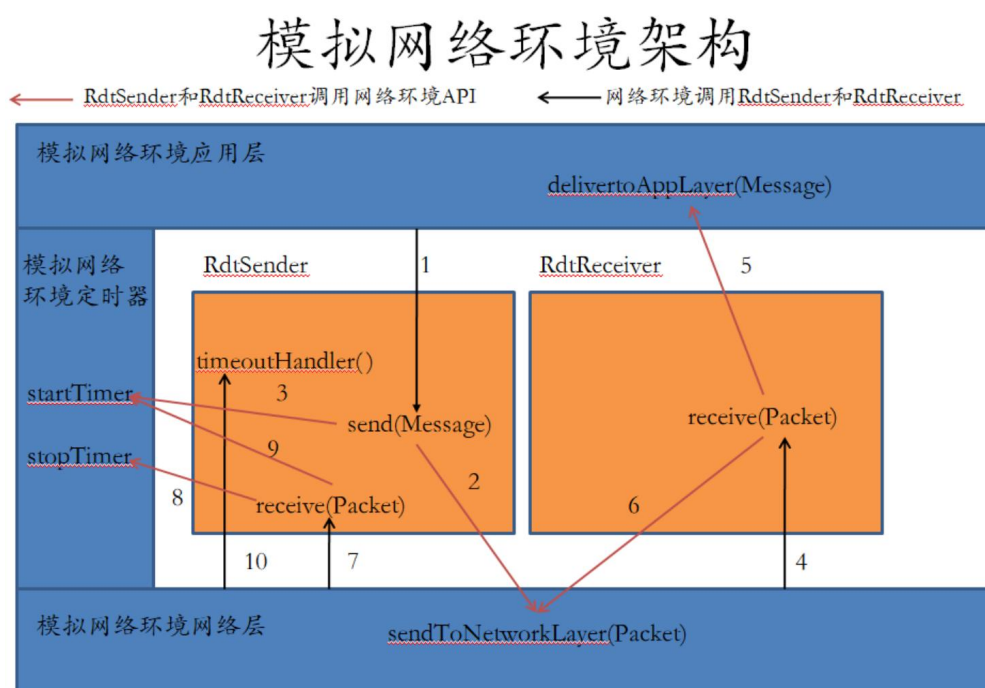


图 1.1 模拟网络环境

该模拟网络环境实现了以下功能：

- (1) 应用层的数据向下递交给发送方运输层 Rdt 协议；
- (2) 接收方运输层 Rdt 协议收到差错检测无误的报文后向上层应用层递交；
- (3) 将发送方运输层 Rdt 协议准备好的报文通过网络层递交给接收方，在递交过程中按一定的概率会产生丢包、报文损坏；
- (4) 定时器的启动、关闭、定时器 Timeout 后通知发送方运输层 Rdt 协议。

按照要求，所需要实现完成的部分为图 1.1 中橙色部分内的函数，这些函数共同组成了 RDT 协议。根据不同的具体协议类型，对于相应的类以及函数进行实现与修改，从而实现不同的协议。

对于实验所提供的模拟网络环境中的函数以及按照实验要求所实现的各个函数，其调用关系如下：

- (1) 模拟网络环境模拟产生应用层数据，调用 `RdtSender` 的 `Send(Message)` 方法；
- (2) `RdtSender` 的 `Send(Message)` 方法调用模拟网络环境的 `sendToNetworkLayer(Packet)` 方法，将数据发送到模拟网络环境的网络层；

-
- (3) RtdSender 的 Send(Message)方法调用模拟网络环境的 startTimer()方法启动定时器;
 - (4) 模拟网络环境调用 RdtReceiver 的 receive(Packet)方法将数据交给 RdtReceiver;
 - (5) 如果校验正确, RdtReceiver 调用模拟网络环境的 deliverToAppLayer(Message)方法将数据向上递交给应用层;
 - (6) RdtReceiver 调用模拟网络环境的 sendToNetworkLayer(Packet)方法发送确认;
 - (7) 模拟网络环境调用 RtdSender 的 receive(Packet)方法递交确认给 RtdSender;
 - (8) 如果确认正确, RtdSender 调用模拟网络环境的 stopTimer 方法关闭定时器;
 - (9) 如果确认不正确, RtdSender 调用模拟网络环境的 startTimer 方法重启定时器;
 - (10) 如果定时器超时, 模拟网络环境调用 RtdSender 的 timeoutHandler()方法。

1.4 系统实现

1.4.1 GBN

根据 GBN 协议的实现, 在 GBN 的 Sender 类中加入整型变量基序号 base 以表示已发送但尚未收到确认的第一个数据。同时使用数组 packetWaitingAck[seq_num]表示已发送并等待接收相应 Ack 报文的数据包, 来实现 GBN 发送方窗口。

对于 GBN 核心技术的实现, 在接收方收到报文时, 检查校验和的同时会检查收到的报文序号是否与接收方期待收到的报文序号一致, 如果均正确则向上递交给应用层并发送这一接收到的 Ack 报文同时将接收方期待收到的报文序号加一; 否则会发送期待收到的序号的 Ack 报文。

而在发送方收到 Ack 报文时会检查校验和, 如果正确则改变等待状态为 false 即可以发送新的分组并关闭当前序号的定时器, 并改变 base 值; 若不正确则一直等待直到超时重传。

通过上述设置, 在接收方正确接收报文时, 接收方和发送方均正常工作; 在报文传输过程中出错时, 通过使接收方发送上一次正确接收的报文序号, 使得发送方的发送窗口不滑动, 从而通过超时重传重新发送出问题的报文, 实现了 GBN 协议的核心内容。

关键功能的实现流程如图 1.2 所示。

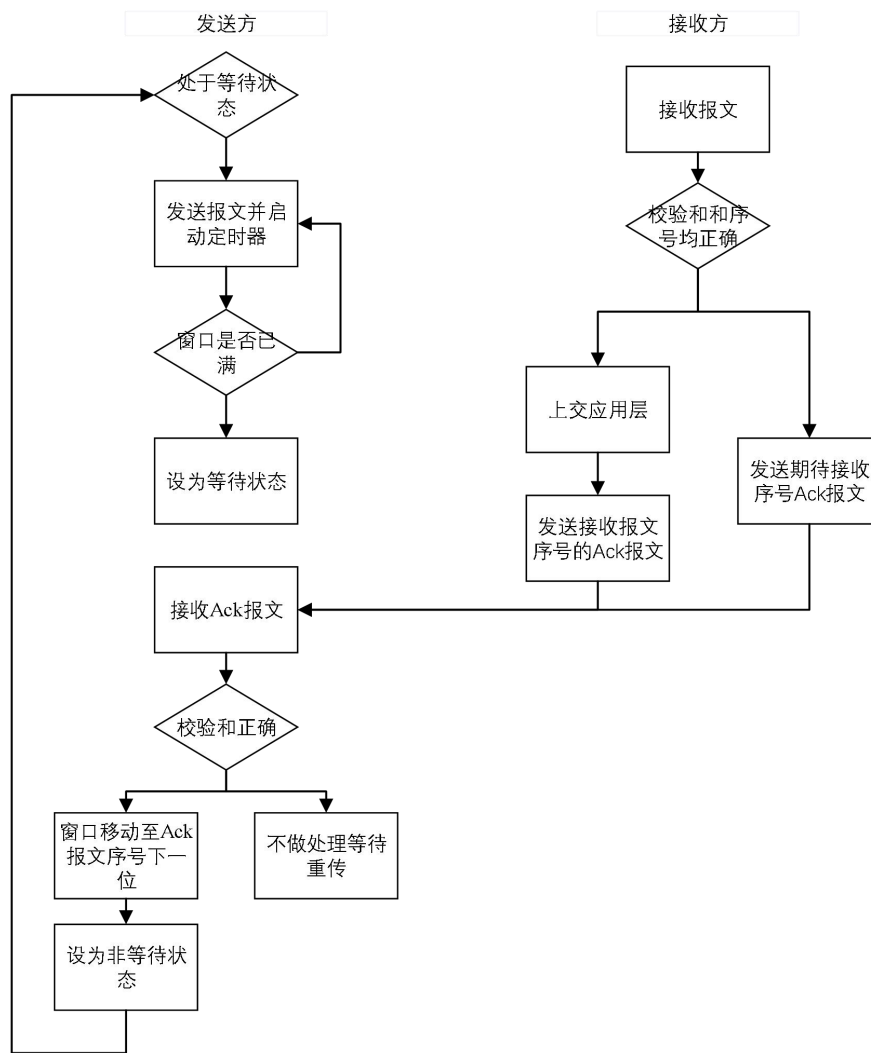


图 1.2 GBN 关键功能

1.4.2 SR

根据 SR 协议的实现，在 SR 的 Sender 类中加入整型变量基序号 `send_base` 以表示已发送但尚未收到确认的第一个数据。同时使用数组 `packetWaitingAck[seq_num]` 表示已发送并等待接收相应 Ack 报文的数据包，来实现 SR 发送方窗口。使用数组 `packetAacked[seq_num]` 表示 SR 发送方窗口已经收到 Ack 的报文，从而避免不必要的重传，仅重传未收到 Ack 的报文而不是和 GBN 中一样全部重传。

在 SR 的 Receive 类中加入数组 `packetAacked[seq_num]` 表示接收方窗口已发送 Ack 报文的序号，从而实现接收方窗口，并使用 `rcv_base` 表示窗口基序号。数组 `buffered[seq_num]` 表示分组缓存区用于缓存所受到不连续的报文，在收到正确分组时一并向上交付。

对于 SR 核心技术的实现，在接收方收到报文时，检查校验和的同时会检查收到的报文序号是否位于接收方窗口之间，若均符合则根据 `packetAacked[seq_num]` 数组的对应值判断是否接收过该分组并回应 Ack 过，若未收到过则改变对应标志并存入缓存区 `buffered[seq_num]` 数组，然后判断所受到报文的序号是否等于接收方窗口基序号 `rcv_base`，若相等则说明可以将缓存区

的分组报文一并向上递交；若校验和正确但收到报文的序号位于 $[rcv_base-N;rcv_base-1]$ 之间，说明该分组的 Ack 报文已经发送过，则再次发送对应的 Ack 报文；若不符合上述两种情况则会发送上一次正确接收的 Ack 报文。

而在发送方收到 Ack 报文时会检查校验和，如果正确则进一步根据 `packetAked[seq_num]` 判断该分组是否已收到过 Ack 报文以及是否处于发送方窗口内，若均符合改变等待状态为 `false` 即可以发送新的分组并关闭当前序号的定时器，并改变 `send_base` 值；若不正确则一直等待直到超时重传。

通过上述设置，在接收方正确接收报文时，接收方和发送方均正常工作；在接收方收到序号不连续的报文时将其缓存起来并发送该序号的 Ack 报文而不是期待接收的序号的 Ack 报文，使得发送方避免对于已成功发送的报文进行不必要的再次发送而是只选择发送失败的报文进行重传，即选择重传。

关键功能的实现流程如图 1.3 所示。

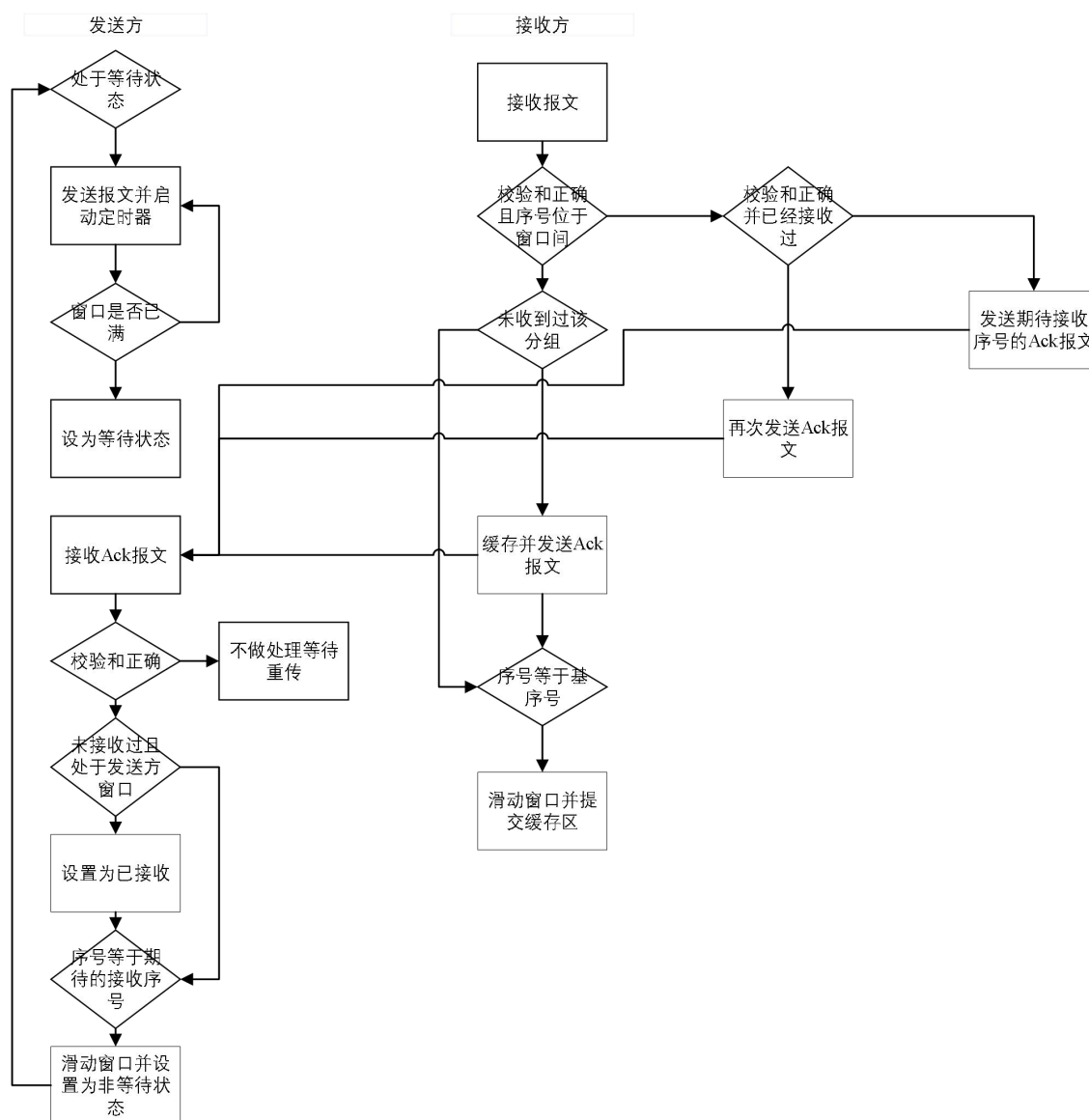


图 1.3 SR 关键功能

1.4.3 TCP

按照任务书的要求，TCP 协议的实现为基于 GBN 协议的简化版。

根据 TCP 协议的实现，在 TCP 的 `Sender` 类中加入整型变量基序号 `base` 以表示已发送但尚未收到确认的第一个数据；使用数组 `packetWaitingAck[seq_num]` 表示已发送并等待接收相应 `Ack` 报文的数据包，来实现 TCP 发送方窗口；使用整型变量 `y` 来表示收到的 `Ack` 报文的序号即接收方期待收到的下一个报文的序号；使用整型变量 `dupAck` 表示冗余 `Ack` 的个数，从而判断是否需要快速重传。

对于 TCP 核心技术的实现，在接收方收到报文时，在 GBN 接收方实现的基础上进行修改，将 `Ack` 报文的序号设置为所期待的下一个报文的序号。

而在发送方收到 `Ack` 报文时会检查校验和，如果正确则将 `y` 设置为所受到的 `Ack` 报文的序号，并检查 `y` 是否位于发送方窗口内并不等于基序号 `base`，若符合则设置为非等待状态并滑动窗口，将基序号 `base` 设置为 `y`，将 `dupAck` 设置为 0 并重新设置计时器，若不符合则说明收到了冗余的 `Ack` 报文，将 `dupAck` 加一，若 `dupAck` 值为 3 则需要快速重传；若不正确则一直等待。

通过上述设置，在接收方正确接收报文时，接收方和发送方均正常工作；在报文传输过程中出现问题时，接收方会发送序号为所期待收到的下一个的序号的 `Ack` 报文，当发送方收到三个冗余的 `Ack` 报文时即说明出现了问题需要进行快速重传，从而实现了 TCP 协议的核心功能。

关键功能的实现流程如图 1.4 所示。

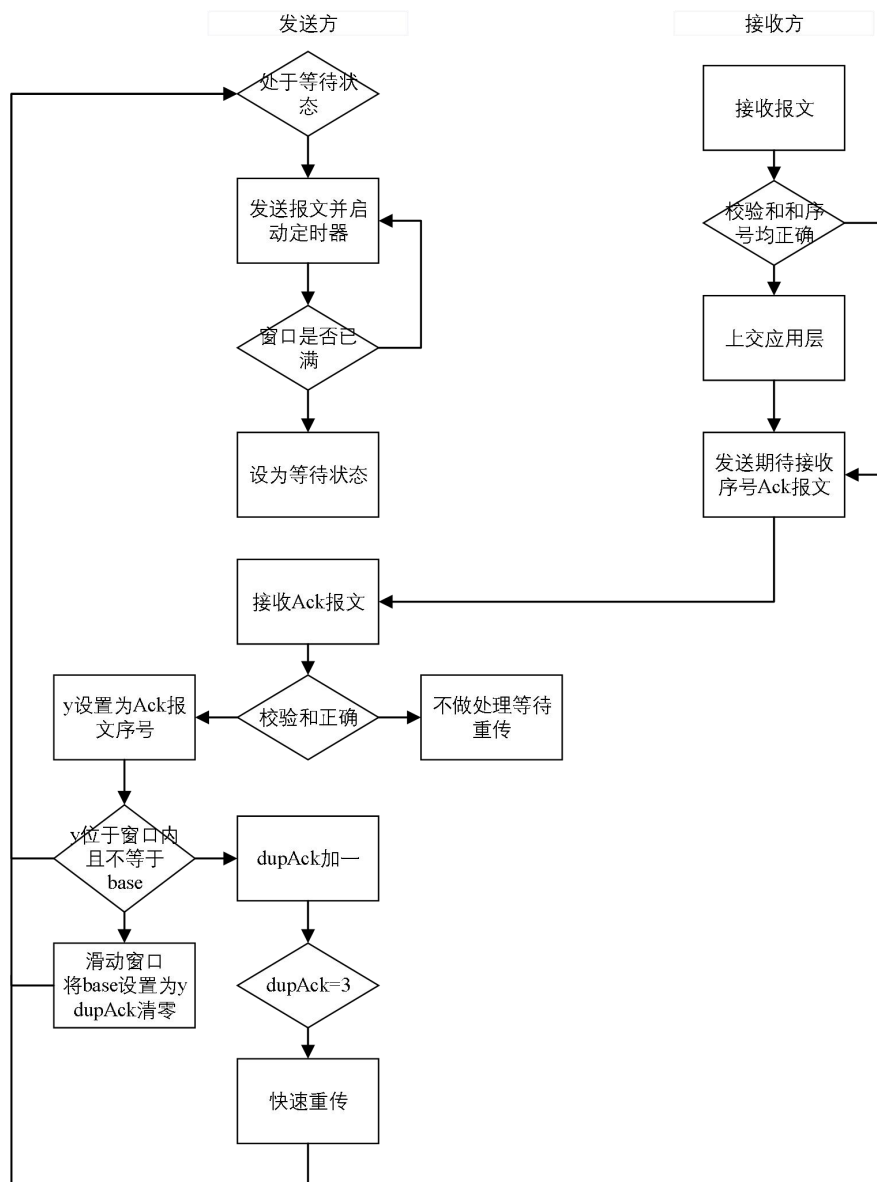


图 1.4 TCP 关键功能

1.5 系统测试及结果说明

系统测试硬件环境如下：

设备型号：XiaoXinPro 14ACH 2021

系统类型：64 位操作系统, 基于 x64 的处理器

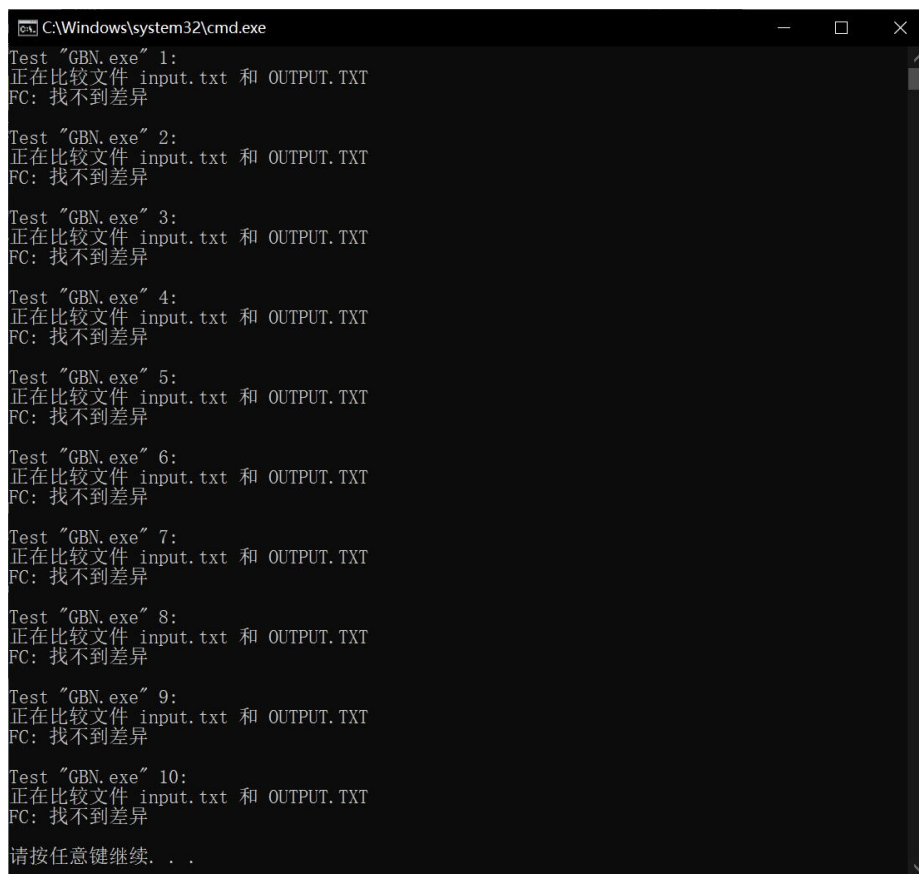
处理器：AMD Ryzen 7 5800H with Radeon Graphics 3.20GHz

内存：16.0G

开发平台：Visual Studio 2019

1.5.1 GBN

使用所提供的测试脚本测试 GBN.exe，重复运行十次均运行成功且得到了正确输出，运行结果如图 1.5 所示。



```
C:\Windows\system32\cmd.exe
Test "GBN.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 2:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 3:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

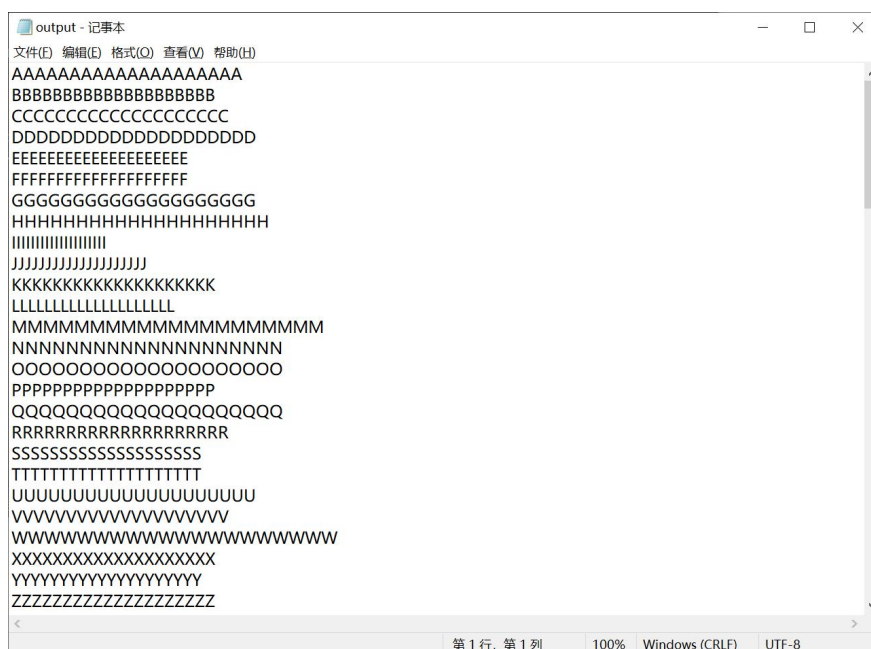
Test "GBN.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "GBN.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

请按任意键继续. . .
```

图 1.5 运行脚本测试 GBN

运行所得到的 out 文件如图 1.6 所示。



```
output - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
AAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFF
GGGGGGGGGGGGGGGGGGGG
HHHHHHHHHHHHHHHHHHHH
IIIIIIIIIIIIIIIIIIII
JJJJJJJJJJJJJJJJJJJJ
KKKKKKKKKKKKKKKKKKKK
LLLLLLLLLLLLLLLLLLLL
MMMMMMMMMMMMMMMMMMMM
NNNNNNNNNNNNNNNNNNNN
OOOOOOOOOOOOOOOOOOOO
PPPPPPPPPPPPPPPPPPPP
QQQQQQQQQQQQQQQQQQQQ
RRRRRRRRRRRRRRRRRRRR
SSSSSSSSSSSSSSSSSSSS
TTTTTTTTTTTTTTTTTTTT
UUUUUUUUUUUUUUUUUUUU
VVVVVVVVVVVVVVVVVVVV
WWWWWWWWWWWWWWWWWWWW
XXXXXXXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYYYYYYYY
ZZZZZZZZZZZZZZZZZZZZ
第 1 行, 第 1 列 100% Windows (CRLF) UTF-8
```

图 1.6 out 文件

运行过程中的输出重定向至 log 文件，如图 1.7 所示。



```
log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
*****模拟网络环境*****: 模拟网络环境启动...
发送方发送报文: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAA
接收方正确收到发送方的报文: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAA
*****模拟网络环境*****: 向上递交给应用层数据: AAAAAAAAAAAAAAAAAAAAA
接收方发送确认报文: seqnum = -1, acknum = 0, checksum = 12851, .....
发送方正确收到确认: seqnum = -1, acknum = 0, checksum = 12851, .....
滑动窗口前base=0, nextseqnum=1
在窗口中的序号为: 0
滑动窗口后base=1, nextseqnum=1
在窗口中的序号为:
发送方发送报文: seqnum = 1, acknum = -1, checksum = 26985, BBBB BBBB BBBB BBBB BBBB
发送方发送报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCCCCC
发送方发送报文: seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDDD
发送方发送报文: seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEE
接收方正确收到发送方的报文: seqnum = 1, acknum = -1, checksum = 26985, BBBB BBBB BBBB BBBB BBBB
*****模拟网络环境*****: 向上递交给应用层数据: BBBB BBBB BBBB BBBB BBBB
接收方发送确认报文: seqnum = -1, acknum = 1, checksum = 12850, .....
接收方正确收到发送方的报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCCCCC
*****模拟网络环境*****: 向上递交给应用层数据: CCCCCCCCCCCCCCCCCCCCC
接收方发送确认报文: seqnum = -1, acknum = 2, checksum = 12849, .....
发送方正确收到确认: seqnum = -1, acknum = 1, checksum = 12850, .....
滑动窗口前base=1, nextseqnum=5
在窗口中的序号为: 1 2 3 4
滑动窗口后base=2, nextseqnum=5
在窗口中的序号为: 2 3 4
接收方没有正确收到发送方的报文,数据校验错误: seqnum = 3, acknum = -1, checksum = 21843, EDDDDDDDDDD
第 1 行, 第 1 列 100% Windows (CRLF) ANSI
```

图 1.7 log 文件

窗口滑动示例如图 1.8 所示。

```
发送方正确收到确认: seqnum = -1, acknum = 2, checksum = 12849, .....
滑动窗口前base=2, nextseqnum=5
在窗口中的序号为: 2 3 4
滑动窗口后base=3, nextseqnum=5
在窗口中的序号为: 3 4
```

图 1.8 窗口滑动

超时重传如图 1.9 所示。

```
发送方定时器时间到, 重发[5,0]的报文
重新发送数据包: seqnum = 5, acknum = -1, checksum = 41116, VVVVVVVVVVVVVVVVVVVV
重新发送数据包: seqnum = 6, acknum = -1, checksum = 38545, WWWWWWWWWWWWWWWWWWWWWW
重新发送数据包: seqnum = 7, acknum = -1, checksum = 35974, XXXXXXXXXXXXXXXXXXXXX
重新发送数据包: seqnum = 0, acknum = -1, checksum = 33411, YYYYYYYYYYYYYYYYYYYY
```

图 1.9 超时重传

1.5.2 SR

使用所提供的测试脚本测试 SR.exe，重复运行十次均运行成功且得到了正确输出，运行结果如图 1.10 所示。

```
C:\Windows\system32\cmd.exe
Test "SR.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 2:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 3:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "SR.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

请按任意键继续. . .
```

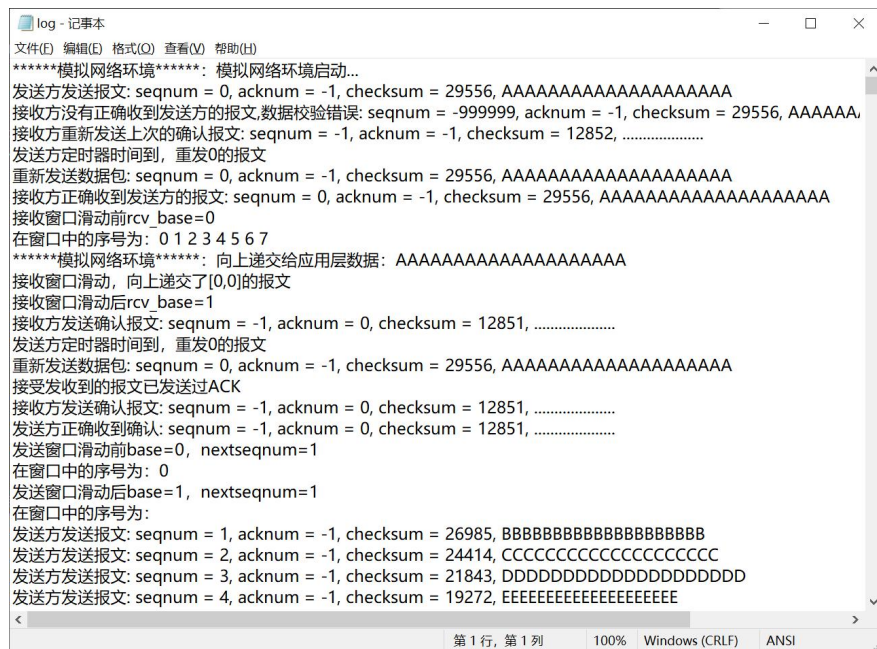
图 1.10 运行脚本测试 SR

运行所得到的 out 文件如图 1.11 所示。

```
output - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
AAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFF
GGGGGGGGGGGGGGGGGGGG
HHHHHHHHHHHHHHHHHHHH
IIIIIIIIIIIIIIIIIIII
JJJJJJJJJJJJJJJJJJJJ
KKKKKKKKKKKKKKKKKKKK
LLLLLLLLLLLLLLLLLLLL
MMMMMMMMMMMMMMMMMMMM
NNNNNNNNNNNNNNNNNNNN
OOOOOOOOOOOOOOOOOOOO
PPPPPPPPPPPPPPPPPPPP
QQQQQQQQQQQQQQQQQQQQ
RRRRRRRRRRRRRRRRRRRR
SSSSSSSSSSSSSSSSSSSS
TTTTTTTTTTTTTTTTTTTT
UUUUUUUUUUUUUUUUUUUU
VVVVVVVVVVVVVVVVVVVV
WWWWWWWWWWWWWWWWWWWW
XXXXXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYYYYYYYY
ZZZZZZZZZZZZZZZZZZZZ
```

图 1.11 out 文件

运行过程中的输出重定向至 log 文件，如图 1.12 所示。



```
log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
*****模拟网络环境*****: 模拟网络环境启动...
发送方发送报文: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAAA
接收方没有正确收到发送方的报文,数据校验错误: seqnum = -999999, acknum = -1, checksum = 29556, AAAAAA,
接收方重新发送上次的确认报文: seqnum = -1, acknum = -1, checksum = 12852, .....
发送方定时器时间到, 重发0的报文
重新发送数据包: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAAA
接收方正确收到发送方的报文: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAAA
接收窗口滑动前rcv_base=0
在窗口中的序号为: 0 1 2 3 4 5 6 7
*****模拟网络环境*****: 向上递交给应用层数据: AAAAAAAAAAAAAAAAAAAAAA
接收窗口滑动, 向上递交了[0,0]的报文
接收窗口滑动后rcv_base=1
接收方发送确认报文: seqnum = -1, acknum = 0, checksum = 12851, .....
发送方定时器时间到, 重发0的报文
重新发送数据包: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAAA
接受发收到的报文已发送过ACK
接收方发送确认报文: seqnum = -1, acknum = 0, checksum = 12851, .....
发送方正确收到确认: seqnum = -1, acknum = 0, checksum = 12851, .....
发送窗口滑动前base=0, nextseqnum=1
在窗口中的序号为: 0
发送窗口滑动后base=1, nextseqnum=1
在窗口中的序号为:
发送方发送报文: seqnum = 1, acknum = -1, checksum = 26985, BBBBBBBBBBBBBBBBBBBBBB
发送方发送报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCCCC
发送方发送报文: seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDDD
发送方发送报文: seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEEE
```

图 1.12 log 文件

发送窗口滑动示例如图 1.13 所示。

```
发送方正确收到确认: seqnum = -1, acknum = 1, checksum = 12850, .....
发送窗口滑动前base=1, nextseqnum=5
在窗口中的序号为: 1 2 3 4
发送窗口滑动后base=2, nextseqnum=5
在窗口中的序号为: 2 3 4
```

图 1.13 发送窗口滑动

接收窗口滑动示例如图 1.14 所示。

```
接收方正确收到发送方的报文: seqnum = 5, acknum = -1, checksum = 16701, FFFFFFFFFFFFFFFFFFFF
接收窗口滑动前rcv_base=5
在窗口中的序号为: 5 6 7 0 1 2 3 4
*****模拟网络环境*****: 向上递交给应用层数据: FFFFFFFFFFFFFFFFFFFF
接收窗口滑动, 向上递交了[5,5]的报文
接收窗口滑动后rcv_base=6
```

图 1.14 接收窗口滑动

超时重传如图 1.15 所示。

```
发送方定时器时间到, 重发7的报文
重新发送数据包: seqnum = 7, acknum = -1, checksum = 11559, HHHHHHHHHHHHHHHHHHHHHH
```

图 1.15 超时重传

1.5.3 TCP

使用所提供的测试脚本测试 TCP.exe，重复运行十次均运行成功且得到了正确输出，运行结果如图 1.16 所示。


```
C:\Windows\system32\cmd.exe
Test "TCP.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 2:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 3:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "TCP.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

请按任意键继续. . .
```

图 1.16 运行脚本测试 TCP

运行所得到的 out 文件如图 1.17 所示。

```
output - 记事本
文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)
AAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFF
GGGGGGGGGGGGGGGGGGGG
HHHHHHHHHHHHHHHHHHHH
IIIIIIIIIIIIIIIIIIII
JJJJJJJJJJJJJJJJJJJJ
KKKKKKKKKKKKKKKKKKKK
LLLLLLLLLLLLLLLLLLLL
MMMMMMMMMMMMMMMMMMMM
NNNNNNNNNNNNNNNNNNNN
OOOOOOOOOOOOOOOOOOOO
PPPPPPPPPPPPPPPPPPPP
QQQQQQQQQQQQQQQQQQQQ
RRRRRRRRRRRRRRRRRRRR
SSSSSSSSSSSSSSSSSSSS
TTTTTTTTTTTTTTTTTTTT
UUUUUUUUUUUUUUUUUUUU
VVVVVVVVVVVVVVVVVVVV
WWWWWWWWWWWWWWWWWWWW
XXXXXXXXXXXXXXXXXXXX
YYYYYYYYYYYYYYYYYYYY
ZZZZZZZZZZZZZZZZZZZZ
```

图 1.17 out 文件

运行过程中的输出重定向至 log 文件，如图 1.18 所示。

1.7 参考文献

[1] 詹姆斯·F·库罗斯、基恩·W·罗斯,《计算机网络:自顶向下方法》[M],北京:机械工业出版社,2018

心得体会与建议

2.1 心得体会

本学期的计算机网络实验共由三次实验组成，分别为 Socket 编程、RDT 协议、CPT 组网实验组成。

对于 Socket 编程实验，由于是第一次实验，对于实验各个方面均不太熟悉。同时由于实验所给出的任务书以及 Demo 程序均体量较大且较为复杂，产生了一些畏难情绪。但在耐心阅读了任务书和 Demo 程序代码并尝试运行 Demo 程序之后，我发现这次实验其实并没有看上去那样复杂，相应的理论知识均在课堂上已经讲述了，实验相关的具体代码也在 Demo 程序中给出了事例。基于这些我顺利完成了本次 Socket 编程实验并较好地实现了所要求的各个功能，并在完成实验的同时进一步理解了 Socket 相关的理论知识。除此之外，出于对于实验要求的实现，我写了两个 html 文件，虽然只是寥寥几行代码，但是也体会到了这一以前尚未接触过的语言的乐趣。

对于 RDT 协议实验，由于有了第一次实验的基础，以及相应理论知识即各种 RDT 协议的具体实现，也顺利地完成了。在实验所提供的 StopWait 协议的实现代码基础上，根据 GBN、SR 和 TCP 协议的原理进行不同的修改与实现。在实现的过程中我也对于 RDT 协议的几种实现有了更深刻的理解，对于理论知识的掌握也更加扎实与透彻。

最后对于 CPT 组网实验，本次实验是要求使用图形化开发工具进行设计，虽然工具有一些不太好用，但是本次实验总体上还是十分有趣的。使用图形化工具对于主机、交换机以及路由器等进行连接与配置，以及最后基于实际应用场景进行自己的设计，都十分有趣并且能加深我们对于相应理论知识的理解。

总体而言，由于实验的设置每次实验都有十分充足的时间去完成，老师和助教们也非常耐心地解答我们的问题，本次计算机网络实验顺利地完成了，对于相应的理论知识我也有了更为透彻的理解。

2.2 建议

对于 Socket 编程和 RDT 协议实验，虽然所提供的任务书以及示例程序已经非常详尽，但是体量较大，阅读理解起来花费时间较长。建议老师可以在实验开始时为我们讲述一下所提供的程序各部分功能并对于程序的运行进行展示，方便我们对于实验总体有具体了解从而快速上手。