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 姓名 王梓熙 学号 U202015369

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 环境

1.1.1 开发平台

开发机器的硬件配置：

处理器：Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

机带 RAM：16.0 GB

系统类型：64 位操作系统，基于 x64 的处理器

操作系统：Windows10

开发平台：Visual Studio 2022

1.1.2 运行平台

（软件运行机器的硬件配置、系统软件组件、第三方组件）

软件运行机器的硬件配置：

处理器：Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

机带 RAM：16.0 GB

系统类型：64 位操作系统，基于 x64 的处理器

操作系统：Windows10

开发平台：Visual Studio 2022

1.2 系统功能需求

实现基于 GBN 的可靠传输协议；实现基于 SR 的可靠传输协议；最后，在实现 GBN 协议的基础上， 根据 TCP 的可靠数据传输机制实现一个简化版的 TCP 协议。

其中，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。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。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。

对于 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

本次实验主要分为三大部分：实现基于 GBN 的可靠传输协议，实现基于 SR 的可靠传输协议，实现简化版的 TCP 协议。每个部分又由发送方和接收方两大模块组合而成。下面分成三个部分来介绍我的系统设计。

1.3.1 GBN 协议

对于 GBN 协议的设计，首先可以绘制发送方与接收方的 FSM 图，如图 1-1 和图 1-2 所示。

根据 GBN 协议，接收方的滑动窗口大小设定为 1，接收方对序号 n 之前的所有分组进行确认，并对所有已发送但未确认的分组统一设置定时器，若存在某个分组超时发生则向发送方发送一个 Ack n，之后发送方重传 n 号分组以及发送窗口中任意序号大于 n 的分组。对于失序分组，接收方进行丢弃处理，并重发按序到达的最高序号分组的 Ack。而发送方收到 Ack 后，如果和 base 一致，则令 base+1，定时器置 0，若滑动窗口里面还有已经发送没确认的元素,继续计时，若滑动窗口里面已经没有已经发送没确认的元素,就不计时了，如果和 base 不一致，则不进行处理。

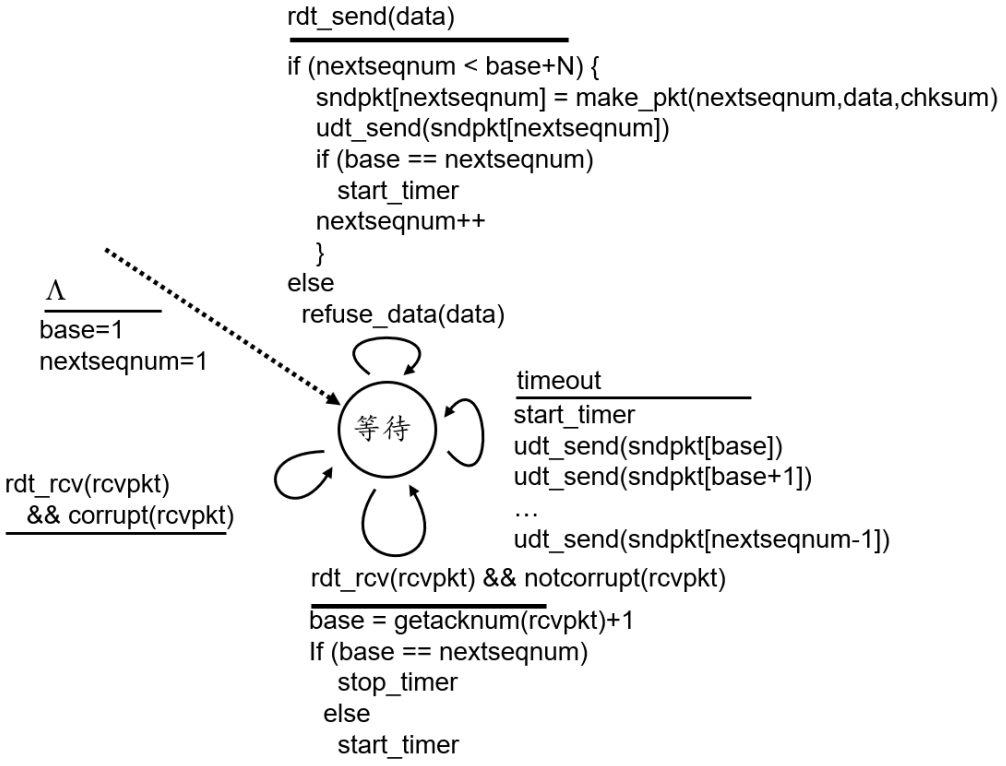

图 1-1 GBN 协议发送方 FSM 图

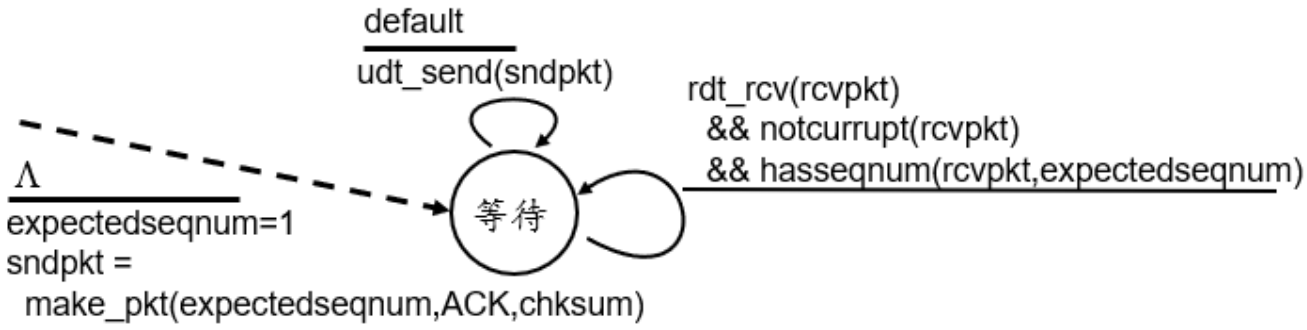

图 1-2 GBN 协议接收方 FSM 图

1.3.2 SR 协议

对于 SR 协议，首先我们可以做出协议示意图，如图 1-3 所示。

根据 SR 协议 ，当发送方从上层收到数据，如果下一个可用于该分组的序号在窗口内，则将数据打包并发送；如果 n 超时，则重传分组 n，重置定时器；如果收到 n 的确认在[sendbase,sendbase+N-1]范围内，则标记分组 n 为已接收，如果 n 是发送窗口基序号 sendbase，则将窗口基序号前推到下一个未确认序号。对于接收方，若分组序号 n 在[rcvbase, rcvbase+N-1]范围内，则发送 n 的确认 ACK(n)，如果分组序号不连续(失序)，则将其缓存，将该分组以及以前缓存的序号连续的分组一起交付给上层，将窗口前推到下一个未收到的分组；若分组序号 n 在 [rcvbase-N,rcvbase-1]范围内，则再次发送 n 的确认 ACK(n)；对于其他情况，则忽略该分组。

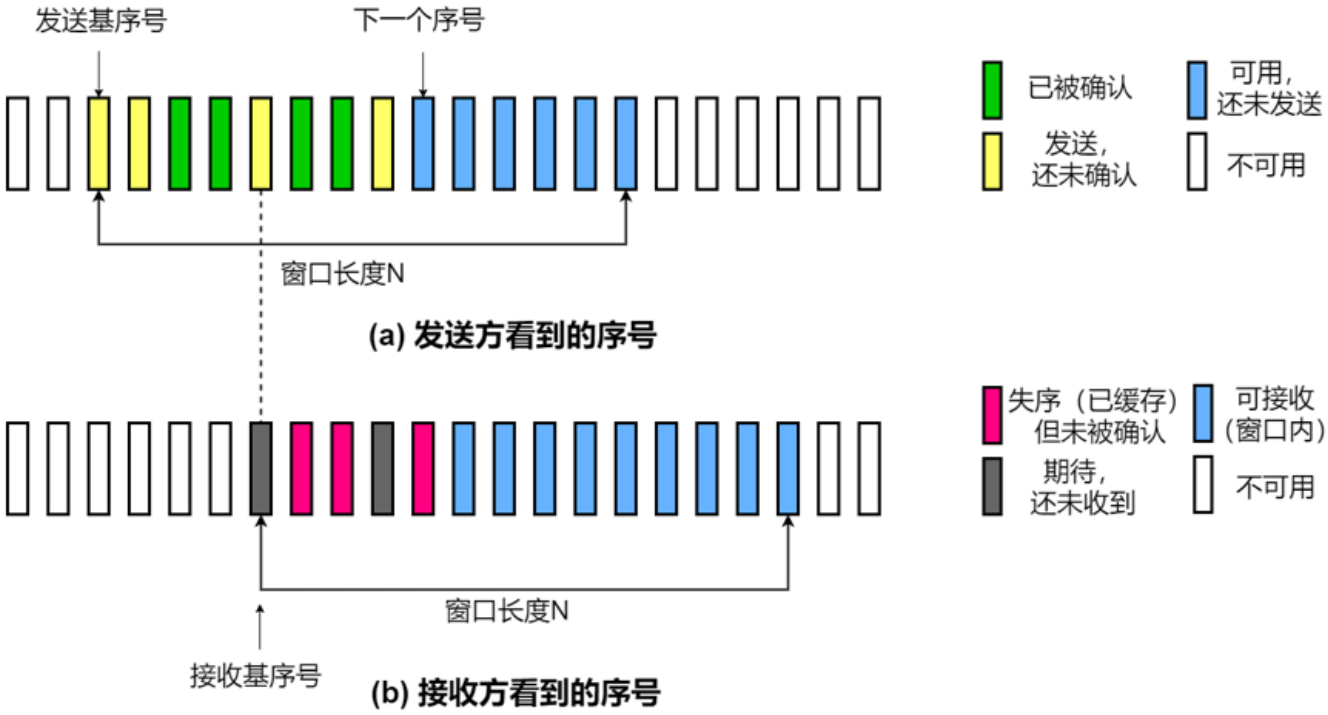

图 1-3 SR 协议示意图

1.3.3 TCP 协议

根据实验要求，我们需要在实现 GBN 协议的基础上， 根据 TCP 的可靠数据传输机制实现一个简化版的 TCP 协议。

根据 TCP 协议的内容，当发送方收到应用层数据，则将数据封装入报文段中，每个报文段都包含一个序号，序号是该报文段第一个数据字节的字节流编号，同时启动定时器，一个报文段一个定时器；当发送方收到超时信号，则重传认为超时的报文段.就是重传滑动窗口里面第一个已发送未确认的元素，重启定时器；当发送方收到 ACK，如果是对以前的未确认报文段的确认，更新 SendBase，如果当前有未被确认的报文段，TCP 还要重启定时器。当接收方收到一个一个正常的分组(按序分组)，等待 500ms,等待收取第二个按序分组，如果 500ms 没收到,那就只发这个，如果 500ms 内收到了,就按照第二个来的分组发送 ACK；当接收方收到一个乱序分组，立即发送冗余 ACK，指明下一个期待字节的序号。同时我们需要增加快速重传的操作，当收到三个冗余 ACK，则重传滑动窗口里面第一个已发送未确认的元素。即我们需要在 GBN 发送方的基础设计上增加快速重传功能，因此需要修改 receive 函数。

1.4 系统实现

1.4.1 GBN 协议实现

对于发送方，发送方收到 Ack 后，如果和 base 一致，则令 base+1，定时器置 0，若滑动窗口里面还有已经发送没确认的元素,继续计时，若滑动窗口里面已经没有已经发送没确认的元素,就不计时了，如果和 base 不一致，则不进行处理。

对于接收方，接收方的滑动窗口大小设定为 1，接收方对序号 n 之前的所有分组进行确认，并对所有已发送但未确认的分组统一设置定时器，若存在某个分组超时发生则向发送方发送一个 Ack n，之后发送方重传 n 号分组以及发送窗口中任意序号大于 n 的分组。对于失序分组，接收方进行丢弃处理，并重发按序到达的最高序号分组的 Ack。

GBN 协议相关函数设计如表 1-1 所示。

表 1-1 GBN 协议相关函数设计

	函数名称	函数功能
发送方	getWaitingState()	获取发送方滑动窗口等待状态
	timeoutHandler()	处理超时情况
	send()	检验发送窗口，打包上层数据并通过网络层发送
	reveive()	接收并检验 ack，管理计时器与滑动窗口
接收方	Receive()	接收、检验并处理报文，发送相应 ack

发送方函数 getWaitingState()通过返回 waitingState 值来确认窗口空闲情况。

发送方函数 send()用以检验发送窗口以及从网络层发送数据。上层调用 Send 函数时,Send

函数首先确认发送窗口情况，发送窗口已若满表示无法发送数据，返回 false；若发送窗口未满，则将待发送数据打包，设置相关参数，设置检查并加入双端队列容器，最后返回 true。

发送方函数 receive()用以接收检验 Ack、管理计时器与移动滑动窗口。首先检查校验并确认 ack 是否损坏，一旦损坏则不予处理；通过检验后，将接受情况与更新的滑动窗口输出在控制台上。

发送方函数 timeoutHandler()用于处理超时情况进行处理。超时情况发生时，本函数将关闭并重启计时器，继而重发所有已发送未确认分组，结果在控制台显示。

接收方函数 Receive()函数内实现了两大功能，分别为接受、检验、处理数据包和发送对应 ack。接收方缓冲区有空余空间时，函数将检查校验和，检查结果与接受结果正确且数据包序号与接收方期望序号相同时则通过检查，取出包内信息，将其递交上层，并在控制台中输出，如果校验和有误未通过检查，则丢弃数据包，并重新发送待接受的数据包。

1.4.2 SR 协议实现

对于发送方，当发送方从上层收到数据，如果下一个可用于该分组的序号在窗口内，则将数据打包并发送；如果 n 超时，则重传分组 n, 重置定时器；如果收到 n 的确认在[sendbase,sendbase+N-1]范围内，则标记分组 n 为已接收，如果 n 是发送窗口基序号 sendbase，则将窗口基序号前推到下一个未确认序号。

对于接收方，若分组序号 n 在[rcvbase, rcvbase+N-1]范围内，则发送 n 的确认 ACK(n)，如果分组序号不连续(失序)，则将其缓存，将该分组以及以前缓存的序号连续的分组一起交付给上层，将窗口前推到下一个未收到的分组；若分组序号 n 在 [rcvbase-N,rcvbase-1]范围内，则再次发送 n 的确认 ACK(n)；对于其他情况，则忽略该分组。

SR 协议相关函数设计如表 1-2 所示。

表 1-2 SR 协议相关函数设计

	函数名称	函数功能
发送方	getWaitingState()	获取发送方滑动窗口等待状态
	timeoutHandler()	处理超时情况
	send()	检验发送窗口，打包上层数据并通过网络层发送
	reveive()	接收并检验 ack，管理计时器与滑动窗口
接收方	Receive()	接收、检验并处理报文，发送相应 ack

发送方函数 getWaitingState()通过返回 waitingState 值来确认窗口空闲情况。

发送方函数 send()用以检验发送窗口以及从网络层发送数据。上层调用 Send 函数时，Send 函数首先确认发送窗口情况，发送窗口已若满表示无法发送数据，返回 false；若发送窗口未满，则将待发送数据打包，设置相关参数，设置检查并加入双端队列容器，最后返回 true。

发送方函数 receive()用以接收检验 Ack、管理计时器与移动滑动窗口。首先检查校验并确认 ack 是否损坏，一旦损坏则不予处理；通过检验后，将接受情况与更新的滑动窗口输出在

控制台上。

发送方函数 timeoutHandler()用于处理超时情况进行处理。超时情况发生时，本函数将关闭并重启计时器，继而重发所有已发送未确认分组，结果在控制台显示。

接收方函数 Receive()函数内实现了两大功能，分别为接受、检验、处理数据包和发送对应 ack。接收方缓冲区有空余空间时，函数将检查校验和，检查结果与接受结果正确且数据包序号与接收方期望序号相同时则通过检查，取出包内信息，将其递交上层，并在控制台中输出，如果校验和有误未通过检查，则丢弃数据包，并重新发送待接受的数据包。

1.4.3 TCP 协议实现

对于发送方，当发送方收到应用层数据，则将数据封装入报文段中，每个报文段都包含一个序号，序号是该报文段第一个数据字节的字节流编号，同时启动定时器，一个报文段一个定时器；当发送方收到超时信号，则重传认为超时的报文段.就是重传滑动窗口里面第一个已发送未确认的元素，重启定时器；当发送方收到 ACK，如果是对以前的未确认报文段的确认，更新 SendBase，如果当前有未被确认的报文段，TCP 还要重启定时器。

对于接收方，当接收方收到一个一个正常的分组(按序分组)，等待 500ms,等待收取第二个按序分组，如果 500ms 没收到,那就只发这个，如果 500ms 内收到了,就按照第二个来的分组发送 ACK；当接收方收到一个乱序分组，立即发送冗余 ACK，指明下一个期待字节的序号。

本次实验中，TCP 协议是以 GBN 协议实现为基础实现的，我们需要在 GBN 协议的基础上增加快速重传的操作，当收到三个冗余 ACK，则重传滑动窗口里面第一个已发送未确认的元素，即我们需要在 GBN 发送方的基础设计上增加快速重传功能，因此需要修改 GBN 中的 receive 函数。

TCP 协议相关函数设计如表 1-3 所示。

表 1-3 TCP 协议相关函数设计

	函数名称	函数功能
发送方	getWaitingState()	获取发送方滑动窗口等待状态
	timeoutHandler()	处理超时情况
	send()	检验发送窗口，打包上层数据并通过网络层发送
	reveive()	接收并检验 ack，管理计时器与滑动窗口，当收到三个冗余 ACK，则重传滑动窗口里面第一个已发送未确认的元素
接收方	Receive()	接收、检验并处理报文，发送相应 ack

发送方函数 getWaitingState()通过返回 waitingState 值来确认窗口空闲情况。

发送方函数 send()用以检验发送窗口以及从网络层发送数据。采用单个确认方法，所以为每个发送分组单独设置计时器，上层调用 Send 函数时，Send 函数首先确认发送窗口情况，发送窗口已若满表示无法发送数据，返回 false；若发送窗口未满，则将待发送数据打包，设置相关参数，设置检查并加入双端队列容器，最后返回 true。

发送方函数 receive()用以接收检验 Ack、管理计时器与移动滑动窗口。首先检查校验并确认 ack 是否损坏，一旦损坏则不予处理；通过检验后，将接受情况与更新的滑动窗口输出在控制台上。在更新基序号后对冗余 ack 加以计数与判断。当达到三个冗余 ack 时，对相关数据包予以快速重传。

发送方函数 timeoutHandler()用于处理超时情况进行处理。超时情况发生时，本函数将关闭并重启计时器，继而重发所有已发送未确认分组，结果在控制台显示。

接收方函数 Receive()函数内实现了两大功能，分别为接受、检验、处理数据包和发送对应 ack。接收方缓冲区有空余空间时，函数将检查校验和，检查结果与接受结果正确且数据包序号与接收方期望序号相同时则通过检查，取出包内信息，将其递交上层，并在控制台中输出，如果校验和有误未通过检查，则丢弃数据包，并重新发送待接受的数据包。

1.5 系统测试及结果说明

1.5.1 系统测试的硬件环境

处理器：Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

机带 RAM：16.0 GB

系统类型：64 位操作系统，基于 x64 的处理器

1.5.2 GBN 协议测试及结果分析

首先通过 check_win.bat 脚本比较输入输出结果，发现结果一致，如图 1-4 所示。然后在 VS2022 环境下手动运行 GBN 协议，结果如图 1-5，图 1-6 所示。

根据测试结果，可知 GBN 协议完成了滑动窗口移动，实现了超时重传等操作，满足了实验要求的功能。

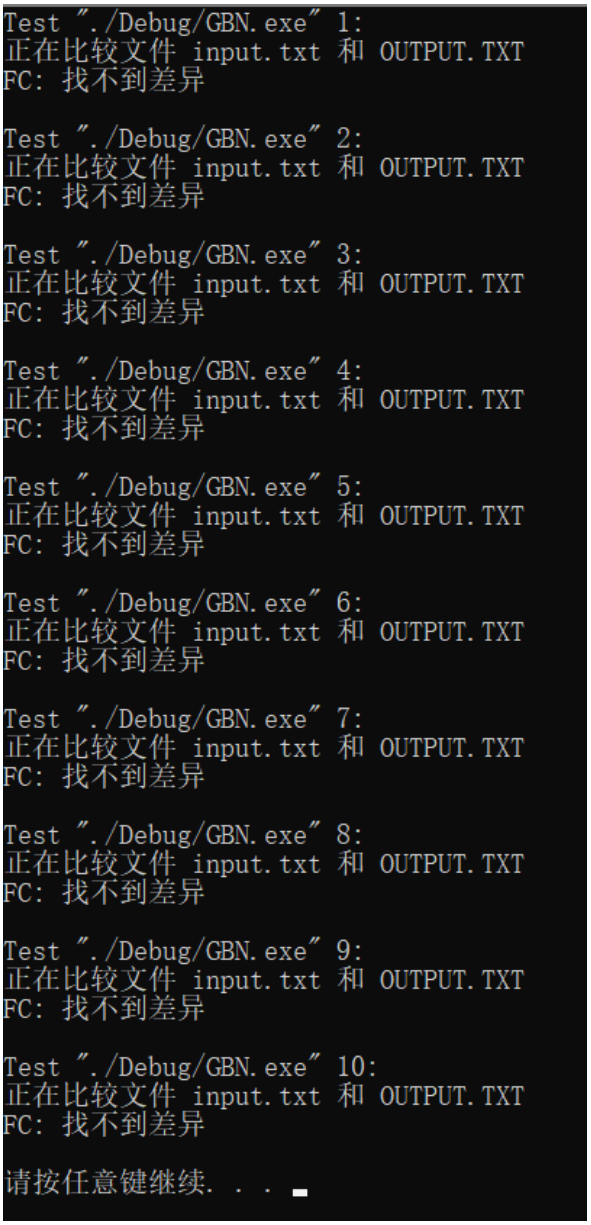

图 1-4 check_win.bat 脚本运行结果

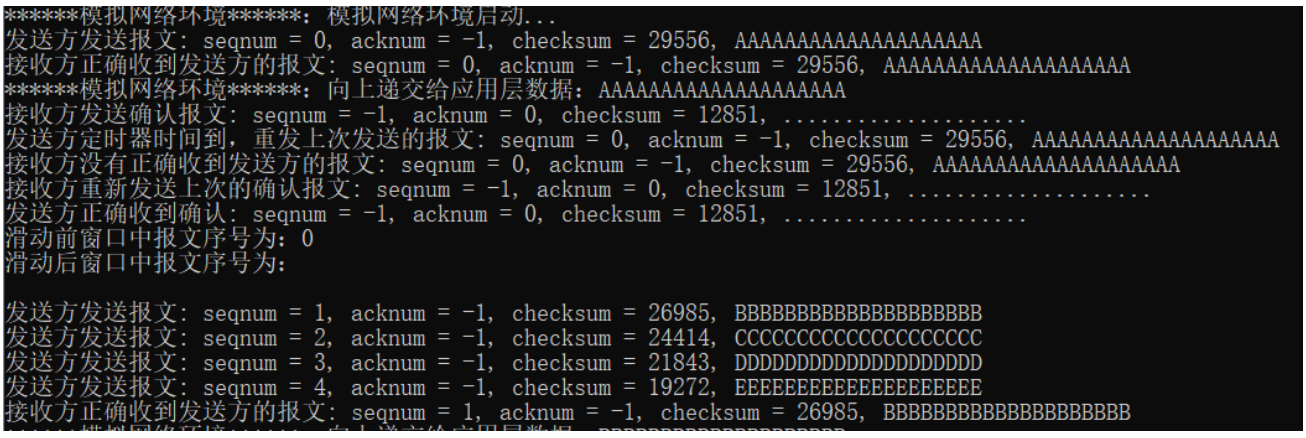

图 1-5 GBN 协议正常接收

```
******模拟网络环境******：向上递交给应用层数据：EEEEEEEEEEEEEEEEEEE
接收方发送确认报文：seqnum = -1, acknum = 4, checksum = 12847, .....................
发送方正确收到确认：seqnum = -1, acknum = 2, checksum = 12849, .....................
滑动前窗口中报文序号为：2 3 4
滑动后窗口中报文序号为：3 4

发送方定时器时间到，重发上次发送的报文：seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDD
发送方定时器时间到，重发上次发送的报文：seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEEE
接收方没有正确收到发送方的报文：seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDD
接收方重新发送上次的确认报文：seqnum = -1, acknum = 4, checksum = 12847, .....................
发送方正确收到确认：seqnum = -1, acknum = 4, checksum = 12847, .....................
滑动前窗口中报文序号为：3 4
滑动后窗口中报文序号为：4

滑动前窗口中报文序号为：4
滑动后窗口中报文序号为：
```

图 1-6 GBN 协议超时重传

1.5.3 SR 协议测试及结果分析

首先通过 check_win.bat 脚本比较输入输出结果，发现结果一致，如图 1-7 所示。然后在 VS2022 环境下手动运行 SR 协议，结果如图 1-8，图 1-9 所示。

根据测试结果，可知 SR 协议完成了滑动窗口移动，实现了超时重传等操作，满足了实验要求的功能。

```
Test "./Debug/SR.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

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

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

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

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

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

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

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

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

Test "./Debug/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 1-7 check_win.bat 脚本运行结果

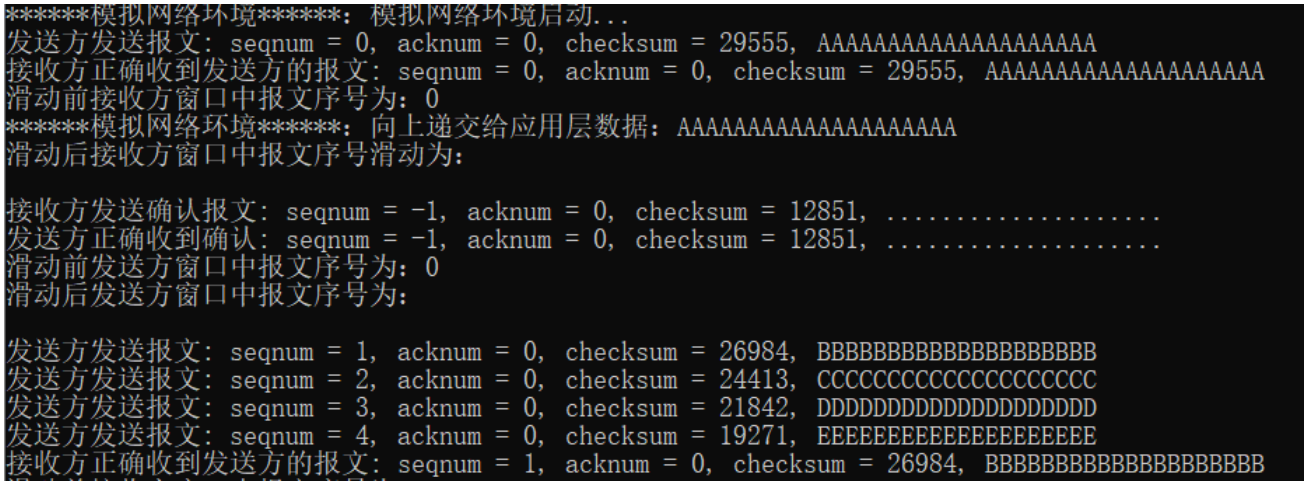

图 1-8 SR 协议正常接收

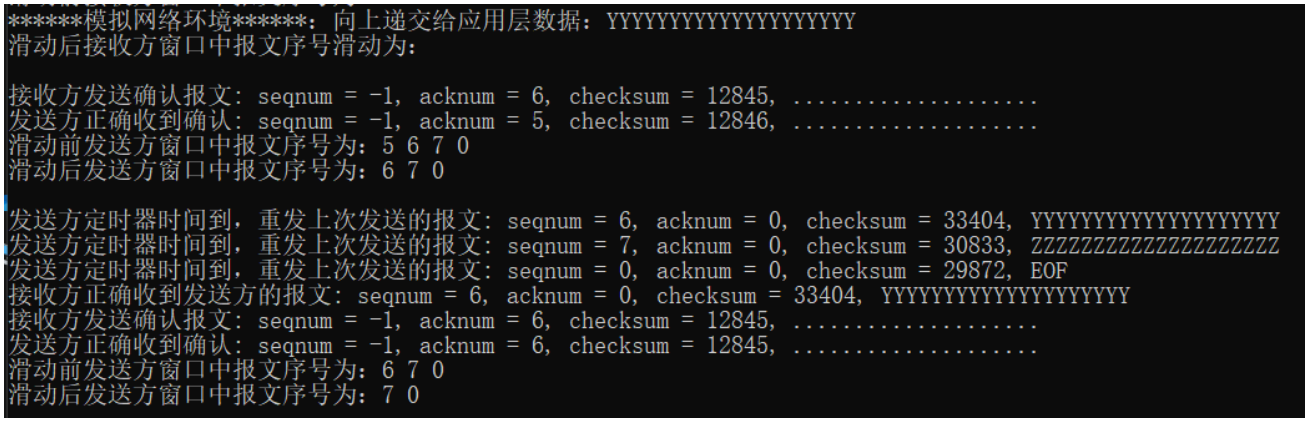

图 1-9 SR 协议超时重传

1.5.4 TCP 协议测试及结果分析

首先通过 check_win.bat 脚本比较输入输出结果，发现结果一致，如图 1-10 所示。然后在 VS2022 环境下手动运行 TCP 协议，结果如图 1-11，图 1-12 所示。

根据测试结果，可知 TCP 协议完成了滑动窗口移动，实现了超时重传，三次冗余 ACK 快速重传等操作，满足了实验要求的功能。

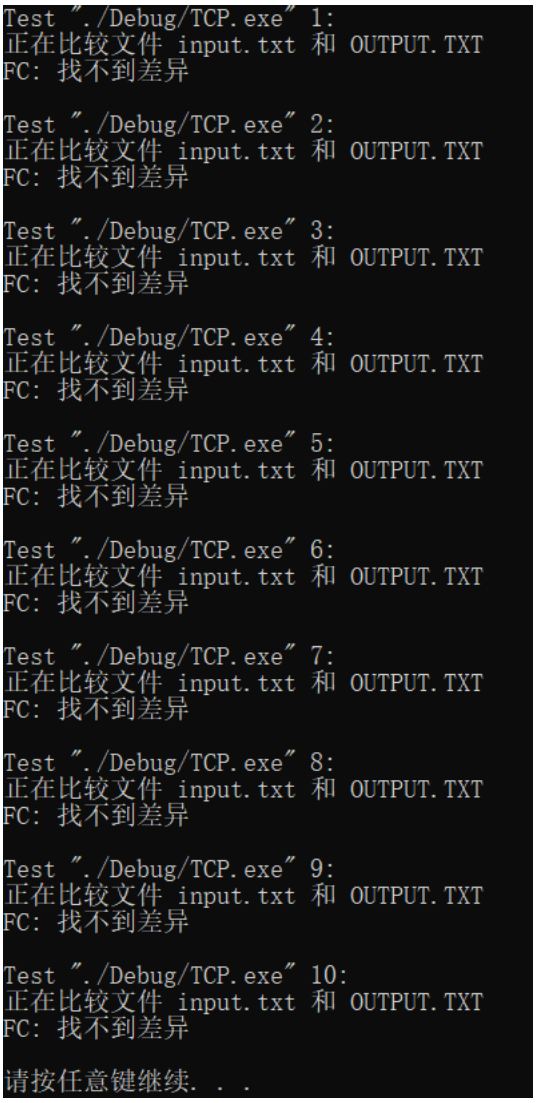

图 1-10 check_win.bat 脚本运行结果

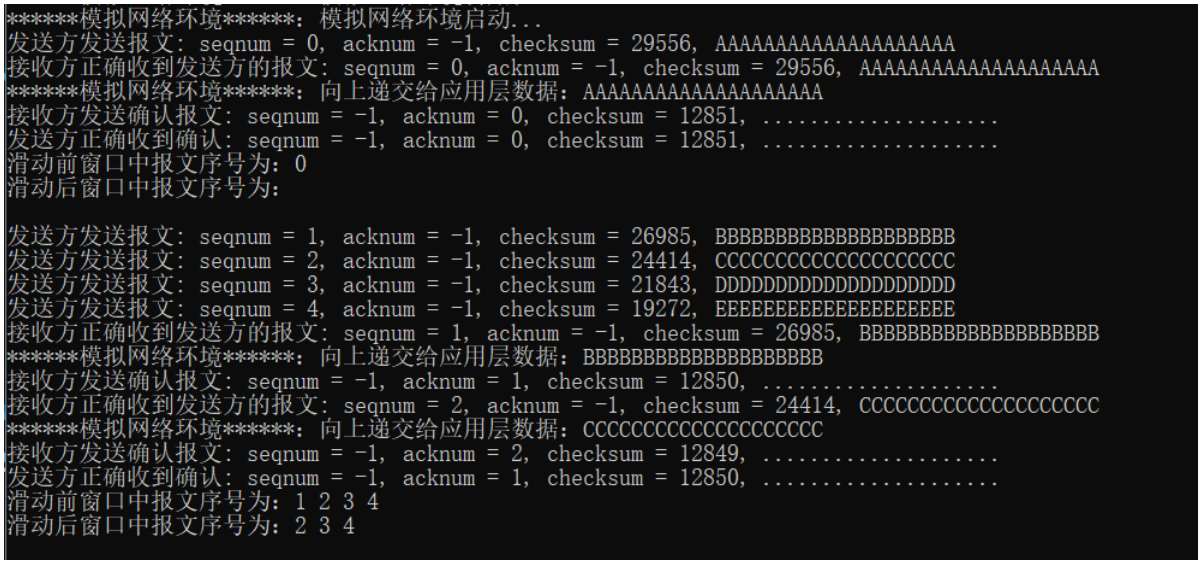

图 1-11 TCP 协议正常接收

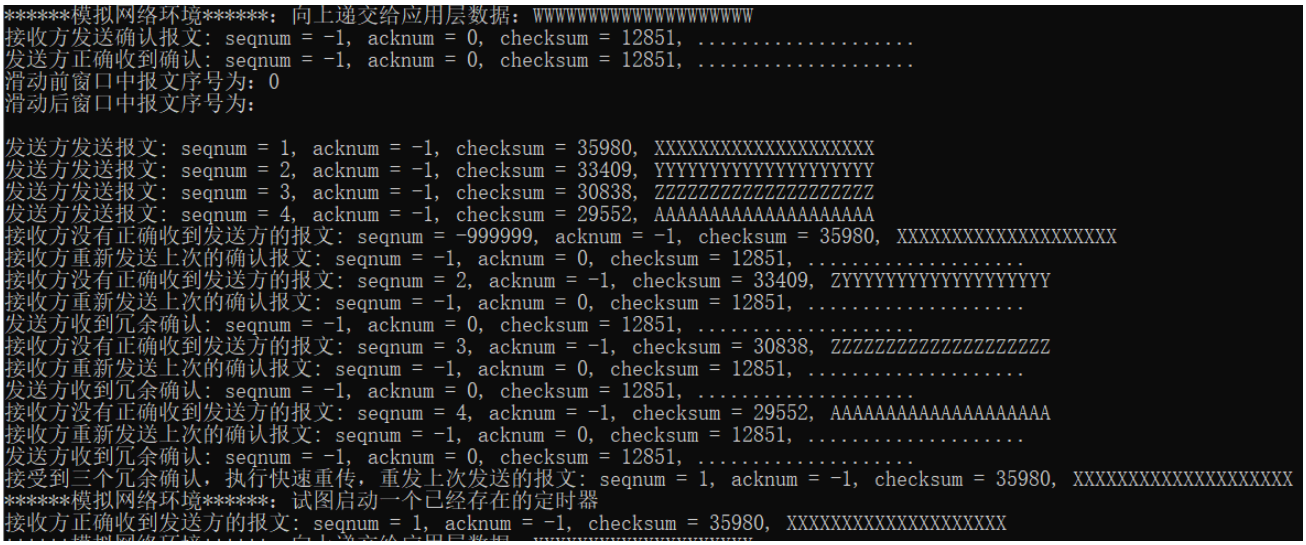

图 1-12 TCP 协议三次冗余 ACK 快速重传

1.6 其它需要说明的问题

无其他问题。

1.7 参考文献

实验二任务书。

心得体会与建议

2.1 心得体会

通过计算机网络的三次实验，我学到了很多知识，有很多的收获。通过第一次 socket 编程实验，让我对 socket 套接字编程有了一定的了解，通过第二次可靠数据传输协议实验，让我对计算机网络运输层的 GBN、SR、TCP 协议都有了更深刻的理解，通过第三次基于 CPT 的组网实验，更是让我对网络层中 IP 地址的分配、路由协议的配置等内容有了更为深刻的认识。这三次计网实验，从不同方面，循序渐进的让我们学习、理解有关计算机网络的相关知识。

总的来说，这三次计网实验时间安排合理，难度适中，与理论课所学知识关联性高，同时实验任务书较为明确，指导清晰。通过这三次实验课，让我们对理论上学到的计网知识理解更加深刻，让我们对理论知识的具体应用也更加熟练，这是我们实验课最大的收获！

2.2 建议

这三次计网实验时间安排合理，难度适中，与理论课所学知识关联性高，同时实验任务书较为明确，指导清晰，实验内容也比较有趣。可能就是任务书略微有一点繁琐，如果能更加精简一点就更好了。