哈爾濱工業大學
Harbin Institute of Technology

计算机网络
课程实验报告

实验名称	可靠数据传输协议-GBN 协议的设计与实现					
姓名	王子奕		院系	计算学部		
班级	1937101		学号	1190200121		
任课教师	李全龙		指导教师	李全龙		
实验地点	G207		实验时间	2021.11.5		
实验课表现	出勤、表现得分(10)		实验报告得分(40)		实验总分	
	操作结果得分(50)					
教师评语						

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

实验目的：

理解可靠数据传输的基本原理；掌握停等协议的工作原理；掌握基于 UDP 设计并实现一个停等协议的过程与技术。

理解滑动窗口协议的基本原理；掌握 GBN 的工作原理；掌握基于 UDP 设计并实现一个 GBN 协议的过程与技术。

实验内容：

1. 基于 UDP 设计一个简单的 GBN 协议，实现单向可靠数据传输（服务器到客户的数据传输）。
2. 模拟引入数据包的丢失，验证所设计协议的有效性。
3. 改进所设计的停等协议，支持双向数据传输。
4. 基于所设计的停等协议，实现一个 C/S 结构的文件传输应用。
5. 改进所设计的 GBN 协议，支持双向数据传输。
6. 将所设计的 GBN 协议改进为 SR 协议。

实验过程：

一、数据报文格式设计

数据报文由发送端的原始数据报文和接收端的相应确认报文构成构成

数据报文的格式如下：

MSG	编号	数据	EOF

数据报文以 MSG 字符串开始，EOF 字符串结束，中间用空格隔开。

确认报文的格式如下：

ACK	编号	EOF

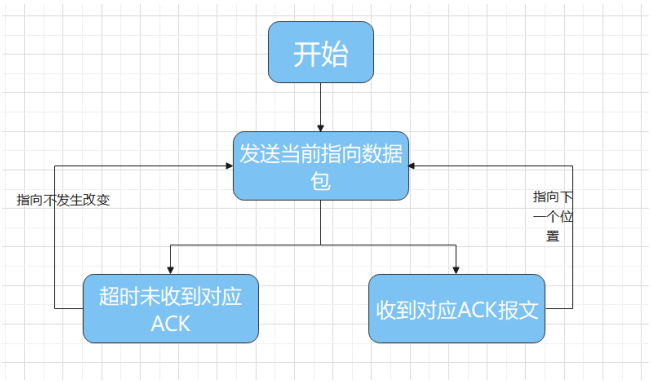
响应确认报文以ACK开始，EOF字符串结束，中间用空格隔开。

通过统一的数据报文格式我们可以高效地处理各种消息。

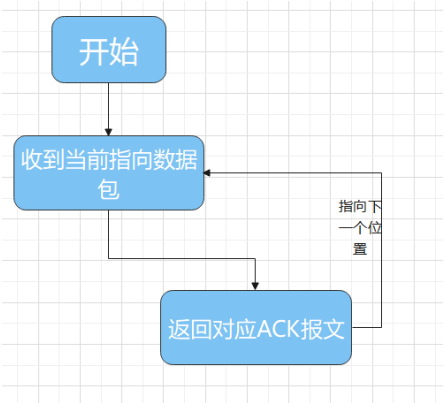
二、协议接收端发送端设计

（1） 停等协议

发送端流程图：



接收端流程图



例如当发送0数据包之后开始计时，只有接收到ack0才能继续发送1数据包，

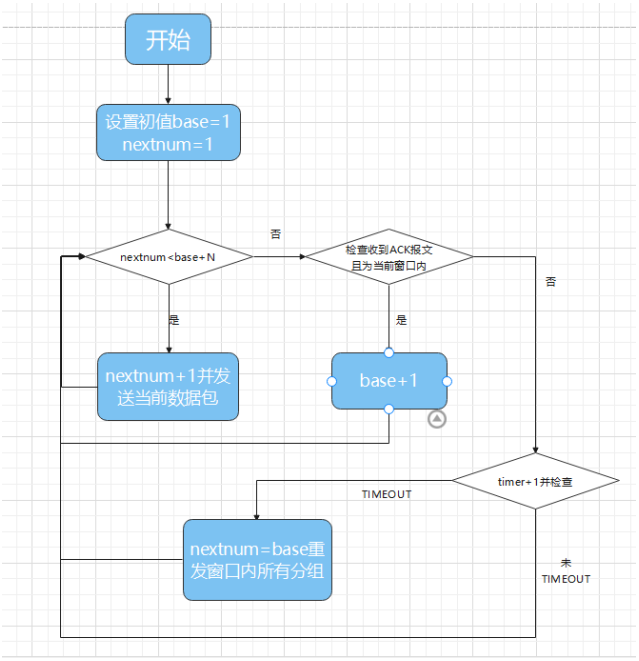
如果在没有接收到ack0的时候已经超时，这时候需要重传数据包0；接收方按照所收到的数据包的编号返回相应的ack，当上一个收到的是数据包0后下一个只有是数据包1才能接收放到接收文件中，如果下一个还是数据包0那么就要丢弃。

（2） GBN协议

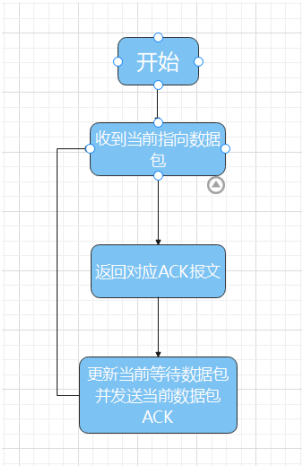
原理：GBN是属于传输层的协议,它负责接收应用层传来的数据,将应用层的数据报发送到目标IP和端口

滑动窗口: 分为接收方窗口和发送方窗口。假设在序号空间内，划分一个长度为N的子区间，这个区间内包含了已经被发送但未收到确认的分组的序号以及可以被立即发送的分组的序号，这个区间的长度就被称为窗口长度。

发送端流程图：



接收端流程图：



Nextnum为当前发送分组编号，base为当前窗口第一个分组，N为窗口大小，timer为计时器。当nextnum与base+N相等时为该窗口已全部发送。GBN中的计时器是整个窗口共享一个计时器，若出现超时的情况需要重发整个窗口内的分组。

GBN检查当前收到ACK报文信息：

```python
def __check(self, addr):
    """接收 ACK 报文并更新滑动窗口"""
    rlist, _, _ = select.select([self.__socket], [], [], 1)
    if rlist:
        msg_bytes, _ = self.__socket.recvfrom(BUFFER_SIZE)
        message: str = msg_bytes.decode()
        # 对确认报文处理
        if message.startswith("ACK "):
            messages = message.split()
            ack_number = int(messages[1])
            # 模拟 ACK 报文丢失
            if DEBUG_RANDOM_THROW_ON and random.random() < DEBUG_LOST_RATE:
                print("{} lose ack packet {}.".format(
                    self.__name, ack_number))
                return
            print("{} receive ACK message {}".format(self.__name, message))
            self.__base = (ack_number + 1) % SEQUENCE_LENGTH
            if self.__base == self.__next_seq_num:
                # 特殊值，表示停止计时
                self.__timer = -1
            else:
                self.__timer = time.time()
    elif self.__timer != -1 and time.time() - self.__timer > TIME_LIMIT:
        # 超时处理
        print("{} timeout on pkt {}.".format(self.__name, self.__base))
        self.__timeout(addr)
```

（3） SR协议

SR协议是在GBN协议的基础上进行的改进。对于SR协议来说，发送方需要做到：为每一个已发送但未被确认的分组都需要设置一个定时器，当定时器超时的时候只发送它对应的分组。当发送方收到ACK的时候，如果是窗口内的第一个分组，则窗口需要一直移动到已发送但未未确认的分组序号。对于接收方,需要设置一个窗口大小的缓存,即使是乱序到达的数据帧也进行缓存,并发送相应序号的ACK, 并及时更新窗口的位置,窗口的更新原则同发送方。此外SR协议只发送未收到的包，因此对于乱序收到的包要做一个缓存操作，在代码实现中利用一个bool数组判断该包是否已经收到过。

SR协议超时处理如下

```python
    else:  # 未收到 ACK 报文
        i = self.__base
        while i != self.__next_seq_num:
            if not self.__ack_flags[i] and self.__timer[i] != -1:
                if time.time() - self.__timer[i] > TIME_LIMIT:
                    # 超时处理
                    print("{} timeout on pkt {}.".format(self.__name, i))
                    self.__socket.sendto(self.__data_seq[i], addr)
                    print(
                        "{} resend message {} to {}.\nMessage: {}".format(
                            self.__name, i, addr,
                            self.__data_seq[i].decode()))
                    self.__timer[i] = time.time()
            i = (i + 1) % SEQUENCE_LENGTH
```

SR协议缓存所有收到的数据：

```python
"""SR 协议的发送端"""
def __init__(self, name: str, addr):
    self.__name = name
    self.__base = 0
    self.__next_seq_num = 0
    self.__timer = [-1] * SEQUENCE_LENGTH
    self.__data_seq = [b'0'] * SEQUENCE_LENGTH
    self.__ack_flags = [False] * SEQUENCE_LENGTH
```

（4） 三种协议比较

停等协议：接收窗口=发送窗口=1

GBN协议：接收窗口=1 发送窗口=N>1

SR协议：接收窗口=发送窗口=N>1

三、模拟数据包丢失

对于数据报文，我们在协议的接收端模拟丢失；对于确认报文，我们在协议发送端模拟丢失。利用一个随机数，来调控数据的丢失率，这样做方便我们调试信息。并且在丢包时打印丢失信息。

四、双向传输设计

对于一个全双工通信的系统，其客户端和服务器其实是对称的，其逻辑可以拆分就是一个发送端和一个接收方。对于发送方，其核心逻辑就是将分组后的数据根据滑动窗口发送给接收方，直到所有的数据都成功发送并接受到对应的确定报文。设计两个用户互相通信，每人分配两个端口作为发送端和接收端。

五、C/S结构文件传输应用设计

利用命令行窗口输入所需数据，发送并缓存在本地，详细内容见实验结果。

实验结果：

一、停等协议实现

```
Bob send message 0 to ('127.0.0.1', 49375).
Message: Bob1
Alice send ack message 0 to ('127.0.0.1', 49378).
Bob lose ack packet 0.
Bob timeout on pkt 0.
Bob resend message 0 to ('127.0.0.1', 49375).
Message: MSG 0 Bob1
Alice received unexpected message 0, resend ack message 1 to ('127.0.0.1', 49378).
Bob lose ack packet 0.
Bob timeout on pkt 0.
Bob resend message 0 to ('127.0.0.1', 49375).
Message: MSG 0 Bob1
Alice received unexpected message 0, resend ack message 1 to ('127.0.0.1', 49378).
Bob receive ACK message ACK 0
Bob send message 1 to ('127.0.0.1', 49375).
Message: Bob2
Alice send ack message 1 to ('127.0.0.1', 49378).
Bob receive ACK message ACK 1
Bob send message 2 to ('127.0.0.1', 49375).
Message: Bob3
Alice send ack message 2 to ('127.0.0.1', 49378).
Bob receive ACK message ACK 2
Bob send message 3 to ('127.0.0.1', 49375).
Message: EOF
Alice send ack message 3 to ('127.0.0.1', 49378).
Bob receive ACK message ACK 3
Sender Bob end.

##############
Finally Alice recevied message:
Bob1 Bob2 Bob3
##############
```

传输过程中每遇到当前发送未收到ACK都会在超时后重发，实现“一传一收”

二、GBN协议实现

```
D:\anaconda\envs\wzy\python.exe D:/大三/WebLab/Lab2_GBN&SR/gbn_test.py
Bob send message 0 to ('127.0.0.1', 49675).
Message: Bob0
Alice send ack message 0 to ('127.0.0.1', 49678).
Bob receive ACK message ACK 0
Bob send message 1 to ('127.0.0.1', 49675).
Message: Bob1
Alice send ack message 1 to ('127.0.0.1', 49678).
Bob lose ack packet 1.
Bob send message 2 to ('127.0.0.1', 49675).
Message: Bob2
Alice lose packet 2.
Bob send message 3 to ('127.0.0.1', 49675).
Message: Bob3
Alice received unexpected message 3, resend ack message 2 to ('127.0.0.1', 49678).
Bob receive ACK message ACK 1
Bob send message 4 to ('127.0.0.1', 49675).
Message: Bob4
Alice received unexpected message 4, resend ack message 2 to ('127.0.0.1', 49678).
Bob lose ack packet 1.
Bob timeout on pkt 2.
Bob resend message 2 to ('127.0.0.1', 49675).
Message: MSG 2 Bob2
Bob resend message 3 to ('127.0.0.1', 49675).
Message: MSG 3 Bob3
Bob resend message 4 to ('127.0.0.1', 49675).
Message: MSG 4 Bob4
Alice send ack message 2 to ('127.0.0.1', 49678).
Bob receive ACK message ACK 2
Bob send message 5 to ('127.0.0.1', 49675).
Message: EOF
Alice send ack message 3 to ('127.0.0.1', 49678).
Bob receive ACK message ACK 3
Alice send ack message 4 to ('127.0.0.1', 49678).
Bob receive ACK message ACK 4
Alice send ack message 5 to ('127.0.0.1', 49678).

#############
Finally Alice recevied message:
Bob0 Bob1 Bob2 Bob3 Bob4
#############

Bob lose ack packet 5.
Bob timeout on pkt 5.
Bob resend message 5 to ('127.0.0.1', 49675).
Message: MSG 5 EOF
Alice lose packet 5.
Bob timeout on pkt 5.
Bob resend message 5 to ('127.0.0.1', 49675).
Message: MSG 5 EOF
Alice lose packet 5.
Bob timeout on pkt 5.
Bob resend message 5 to ('127.0.0.1', 49675).
Message: MSG 5 EOF
Alice received unexpected message 5, resend ack message 6 to ('127.0.0.1', 49678).
Bob receive ACK message ACK 5
Sender Bob end.
```

共发送五条数据，窗口大小为3. 可观察到数据1正常发送，但数据二丢失后重新发送数据二三四。

三、双向通信SR协议实现

```
Alice send message 0 to ('127.0.0.1', 49577).
Message: Alice
Bob send message 0 to ('127.0.0.1', 49575).
Message: Bob0
Bob send ack message 0 to ('127.0.0.1', 49576).
Alice receive ACK message ACK 0
Alice send message 1 to ('127.0.0.1', 49577).
Message: EOF
Bob send ack message 1 to ('127.0.0.1', 49576).
AliceEOFAlice receive ACK message ACK 1

666

##############
Finally Bob recevied message:
Alice
##############

Alice lose packet 0.
Bob send message 1 to ('127.0.0.1', 49575).
Message: Bob1
Alice send ack message 1 to ('127.0.0.1', 49578).
Bob receive ACK message ACK 1
Bob send message 2 to ('127.0.0.1', 49575).
Message: Bob2
Alice send ack message 2 to ('127.0.0.1', 49578).
Bob receive ACK message ACK 2
Bob send message 3 to ('127.0.0.1', 49575).
Message: Bob3
Alice send ack message 3 to ('127.0.0.1', 49578).
```

部分运行结果如上，全文请查看sr.pdf。可观察到发送端只发送未收到ACK的数据，相比GBN效率更高。

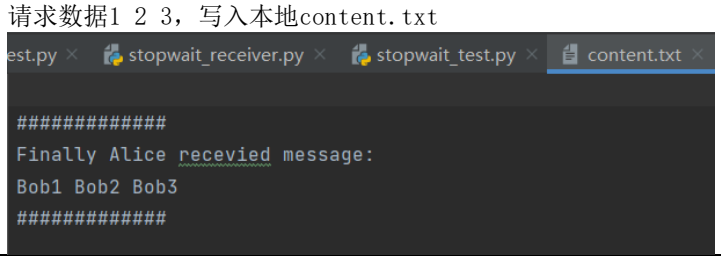
四、C/S结构缓存本地运行结果

```
D:\anaconda\envs\wzy\python.exe D:/大三/WebLab/Lab2_GBN&SR/stopwait_test.py
Please input your expect id:1 2 3
Bob send message 0 to ('127.0.0.1', 49375).
Message: Bob1 Alice send ack message 0 to ('127.0.0.1', 49378).

Bob receive ACK message ACK 0
Bob send message 1 to ('127.0.0.1', 49375).
Message: Bob2
Alice send ack message 1 to ('127.0.0.1', 49378).
Bob receive ACK message ACK 1
Bob send message 2 to ('127.0.0.1', 49375).
Message: Bob3
Alice lose packet 2.
Bob timeout on pkt 2.
Alice send ack message 2 to ('127.0.0.1', 49378).
Bob resend message 2 to ('127.0.0.1', 49375).
Message: MSG 2 Bob3
Bob receive ACK message ACK 2
Bob send message 3 to ('127.0.0.1', 49375).
Message: EOF
Alice send ack message 3 to ('127.0.0.1', 49378).
Bob receive ACK message ACK 3
Sender Bob end.

############
Finally Alice recevied message:
Bob1 Bob2 Bob3
############
```

请求数据1 2 3，写入本地content.txt



问题讨论：
可靠数据传输的三种类型： GBN：回退N(go back N)步,如果某个报文段没有被正确的接收，那么从这个报文段到后面的报文段都要重新发送，返回的ACK采用累计确认的机制，也就是说如果GBN返回的ACK=3，也就是说3报文段和3 之前的报文段都被正确地接收了。 SR：接收方设置缓冲区，为每个报文段设置计时器，如果某个报文段没有被正确接收但是后面的报文段被正确接收了，那么就只需要重发这一个报文段，返回的ACK就是当前接收成功的报文段序号 TCP：和SR类似，但是TCP有快速重传机制，不需要等待某个报文段的计时器超时才能重传，返回的ACK编号是期待接收到的下一个报文的序号
心得体会：
通过本次实验我深刻理解了可靠数据传输协议的三种基本分类、滑动窗口的具体实现流程，加深了对计算机网络这门课的理解。