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	李劲光		院系	计算机学与技术学院		
班级	1903201		学号	L190202102		
任课教师	聂兰顺		指导教师	聂兰顺		
实验地点			实验时间			
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：

（注：实验报告模板中的各项内容仅供参考，可依照实际实验情况进行修改。）
本次实验的主要目的。

实验内容：

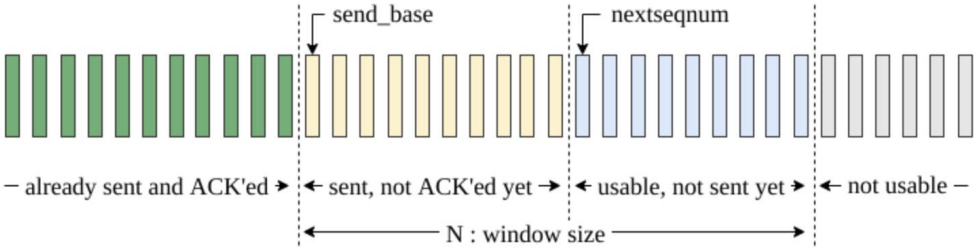
概述本次实验的主要内容，包含的实验项等。

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。
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 GBN 协议，支持双向数据传输；（选作内容，加分项目，可以当堂完成或课下完成）
4）将所设计的 GBN 协议改进为 SR 协议。（选作内容，加分项目，可以当堂完成或课下完成）

实验过程：

以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。

滑动窗口（流水线）协议通过不需要发送方在发送另一个帧之前等待确认来实现网络带宽的利用。
在 Go-Back-N 中，发送方控制数据包的流动，这意味着我们有一个简单的虚拟接收方。因此，我们将首先讨论服务器如何处理数据包。
发送方有一系列要发送的帧。 我们假设窗口大小为 N。此外，存在两个指针来跟踪发送基址 (send_base) 和下一个要发送的数据包 (nextseqnum)。



首先，发送方从发送第一帧开始。 最初，send_base = 0 和 nextseqnum = 0。虽然有更多的数据包要发送并且 nextseqnum 小于 send_base + N； 发送方发送 nextseqnum 指针指向

的数据包，然后递增 nextseqnum。同时，send_base 在收到接收方的确认包后递增。 接收重复的 ACK 消息不会触发任何机制。整个发送窗口有一个计时器，用于测量 send_base 中数据包的超时时间。 因此，如果发生超时，发送方将重新启动定时器，并重新发送从 send_base 开始的发送窗口中的所有数据包。

Algorithm 1: Go-Back-N Sender

```
Function Sender is
    send_base ← 0;
    nextseqnum ← 0;
    while True do
        if nextseqnum < send_base + N then
            send packet nextseqnum;
            nextseqnum ← nextseqnum + 1;
        end
        if receive ACK n then
            send_base ← n + 1;
            if send_base == nextseqnum then
                stop timer;
            else
                start timer;
            end
        end
        if timeout then
            start timer;
            send packet send_base;
            send packet send_base + 1;
            ...
            send packet nextseqnum − 1;
        end
    end
end
```

Go-Back-N 协议采用累积确认。 也就是说，接收到帧 n 的确认意味着帧 n-1、n-2 等也被确认。 我们将这种确认表示为 ACK n。
我们有一个停止等待的实现。

Algorithm 2: Go-Back-N Receiver

```
function Receiver is
    nextseqnum ← 0;
    while True do
        if A packet is received then
            if The received packet is not corrupted and
             sequence_number == nextseqnum then
                deliver the data to the upper layer;
                send ACK nextseqnum;
                nextseqnum ← nextseqnum + 1;
            else
                /* If the packet is corrupted or out of
                   order, simply drop it                  */
                send ACK nextseqnum − 1;
            end
        else
    end
end
```

Go-Back-N 协议适用于发送方和接收方，以确保可靠的数据传输。我们还讨论了累积确认如何强制我们使用大于 N 的 S 值以及如何选择合理的窗口大小 N 以实现更好的利用

率。
Go-Back-N 的接收器实现尽可能简单:
接收方只跟踪下一个接收的预期序列号：nextseqnum。没有接收缓冲区； 无序数据包被简单地丢弃。 同样，损坏的数据包也会被悄悄丢弃。它总是在接收到新数据包（成功或不成功）时发送对收到的最后一个有序数据包的确认。 因此，如果出现问题，它将生成重复的确认消息。
让 S 表示我们用来标记帧的最大可能序列号。 再次假设我们的窗口大小为 N。现在，让我们想象一个简单的场景:
-发送者发送窗口中的帧，从 0 到 S 枚举它们
-作为响应，它接收 ACK S，标记帧 S、S-1，等等，然后发送方发送第二组帧，再次从 0 到 S 枚举它们之后，发送方收到另一个 ACK S 从发送者的角度来看，最后一步的确认代表什么？ 第二批中的数据包是否全部丢失或发送成功？ 如果 S = N，则发送方无法知道真实结果。 这就是为什么我们必须有严格的不等式 N < S。

正如我们之前所说，流水线协议是对停止等待协议的改进。 为了实现更好的网络利用率，我们在给定时间在发送方和接收方之间有多个“传输中”帧。N 表示 Go-Back-N 中的窗口大小，允许发送方在收到确认之前发送。 基本上如果 N = 1，

SR 复协

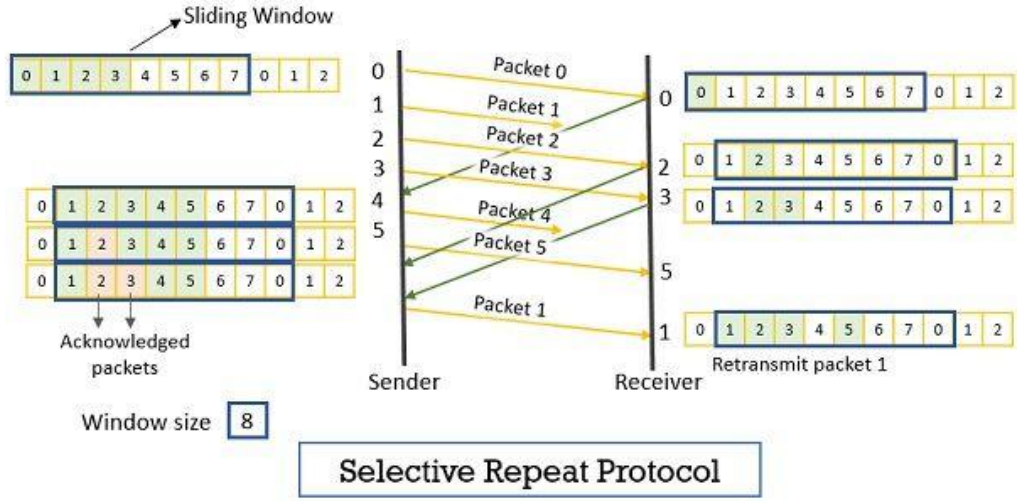
SR 复协议是一种滑动窗口协议，它使用流水线的概念，在发送方等待第一个发送的数据包的确认时，可以发送多个数据包。 选择性重复协议管理发送方和接收方之间的错误和流量控制。
如果错误率和丢包率较低，我们在之前的内容中研究的 go-back-n 协议是有效的。 如果连接不良，将频繁丢失数据包，发送方将不得不重新传输所有未完成的数据包。 这浪费了信道的带宽。
在 go-back-n 中，接收器窗口的大小为 1，因此它仅缓冲一个数据包，以便它必须确认下一个数据包，如果此预期数据包丢失或损坏，则所有接收到的乱序数据包都将被丢弃。 并且发送方必须重新传输所有未完成的数据包，尽管其中一些可能已经安全地到达接收方但出现故障。
数据包的这种重传会增加网络上的流量，从而导致拥塞的累积增加。 另一种方法是选择性重复协议。 在前面的部分中，我们将讨论这个选择性重复协议的工作。
Selective Repeat Protocol 的工作原理，和 go-back-n 一样，选择性重复协议也是一个滑动窗口协议。 与 go-back-n 协议不同，选择性重复协议仅重新发送丢失或损坏的选择性数据包。 让我们首先讨论选择性重复中的窗口。

Window，在选择性重复中，发送方和接收方都有一个滑动窗口。 在发送方，该窗口涵盖了已发送或可以发送的数据包序列。 在接收方，滑动窗口覆盖了已接收或预期接收的数据包的序列号。 在选择性重复中，发送方和接收方窗口的大小相同。 让我们简要地研究发送方窗口和接收方窗口。

Sender Window，与 go-back-n 协议相比，selective repeat 中的 sender window 要小得多。 这里发送窗口的大小是 2m-1。 这里 m 是包头用来表示对应包的序号的比特数。
发送方窗口涵盖已发送但尚未确认的数据包、无序确认的数据包和发送方应用层收到相应

数据后即可发送的数据包。

接收窗口，接收器窗口的最大大小为 2m-1，与发送器窗口相同。 接收器窗口涵盖了无序接收的数据包的序列号，并且正在等待较早发送但尚未接收的数据包。
接收者传输层不会乱序向应用层传送数据包。 它等待直到接收到一组连续的数据包，以便将它们传送到应用层。在选择性重复中，在发送端，每个发送的数据包都附加一个计时器，如果在计时器到期之前没有收到确认，则 相应的数据包被重新发送。



发送端操作，如上图所示，由于窗口大小为 8，发送方窗口覆盖了从 0 到 7 的数据包序列号。 发送方已发送数据包 0、1、2 和 3，正在等待接收来自 数据包 4、5、6 和 7 用于其应用层。

接收端操作，接收器窗口的大小为 8，因此接收器窗口覆盖从 0 到 7 的序列号。如果首先收到数据包 0 并立即发送确认。 稍后它接收到乱序的数据包 2 和 3 并发送对这些接收到的数据包的确认。 迟到它再次收到一个乱序的数据包 5 和 1。

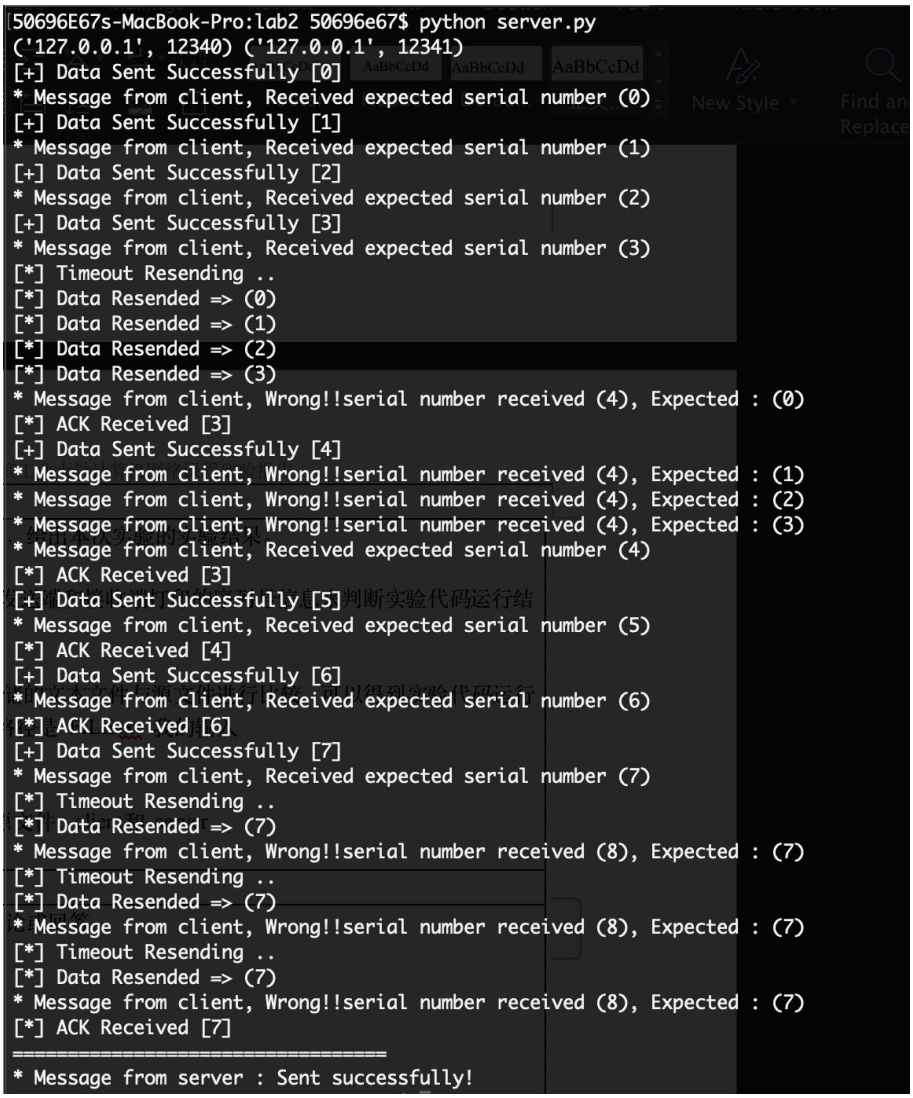
实验结果：

采用演示截图、文字说明等方式，给出本次实验的实验结果。

终端输出：可以通过终端打印的发送端和接收端打印的序列号信息来判断实验代码运行结果的正确性；

接收文件：通过将接收数据中存储的文本文件与源文件进行比较，可以得到实验代码运行结果的正确性。 我的输入文件路径是 FILE.txt 我的输入
文件路径为 SAVED.txt

GBN协议全双工通信测试,有关源文件，client和 server。



对于SR协议全双工通信的测试，client和 server。

```
50696E67s-MacBook-Pro:lab2 50696e67$ python server.py
('127.0.0.1', 12340) ('127.0.0.1', 12341)
[+] Data Sent Successfully [0]
* Message from client : data recieved (0)
* CLIENT: window slided (1)
[*] ACK Received [0]
* SERVER: window slided (1)
[+] Data Sent Successfully [1]
[+] Data Sent Successfully [2]
* Message from client : data recieved (2)
[*] ACK Received [2]
[+] Data Sent Successfully [3]
* Message from client : data recieved (3)
[*] ACK Received [3]
[*] Timeout (1) Resending
* Message from client : data recieved (1)
* CLIENT: window slided (2)
* CLIENT: window slided (3)
* CLIENT: window slided (4)
[*] ACK Received [1]
* SERVER: window slided (2)
* SERVER: window slided (3)
* SERVER: window slided (4)
================================================
* Message from server : Sent successfully!
* Message from client : Sent successfully!
```

问题讨论：

对实验过程中的思考问题进行讨论或回答。

GBN特点：由于网络中的流量控制，需要控制这些未确认报文的数量N，否则会造成网络拥塞。 在GBN协议中，发送方可以在窗口大小N的限制内发送足够多的数据包。接收方收到数据包后，向发送方发送ACK。 当发送方收到连续的 ACK 时，窗口将向前移动。 滑动，发送方可以很容易地传输新的数据包。 在接收端，如果一个数据包丢失，则丢失的数据包编号之后的所有数据包都必须从该数据包中重新传输，但在这种情况下，接收器不需要准备一定的空间缓冲区来存储数据包。

- Go back N 比其他协议更常用。
- SR 协议由于其复杂性而较少使用。
- Stop and Wait ARQ由于其效率低而较少使用。
- 根据上下文和资源可用性，使用返回 N 或选择性重复。
- 选择性重复和停止以及等待 ARQ 在重传方面是相似的。
- 如果发送方窗口大小相同，则返回 N 和选择性重复在效率方面相似。
- SR协议可以认为是Stop and Wait ARQ和Go back N的优点的结合。
- SR 协议优于其他协议，但由于其复杂性，较少使用。

心得体会:

结合实验过程和结果给出实验的体会和收获。

通过此次实验,我对 GBR 和 SR 协议的有了更加深刻的认识,我了解累积确认如何强制我们使用大于 N 的 S 值以及如何选择合理的窗口大小 N 以实现更好的利用。