南開大學
Nankai University

南　　开　　大　　学

计　算　机　学　院

计算机网络实验 3-4

基于 UDP 服务设计可靠传输协议并编程实现

杨浩甫　2113824

年级：2021 级

专业：计算机科学与技术

指导教师：张建忠、徐敬东

2023 年 12 月 23 日

目录

一、 实验准备

(一) 实验背景

在之前的实验中，我们探索过三种不同的机制来实现 UDP 协议的可靠传输：

停等机制（Stop-and-Wait）：这是一种基本的机制，其中发送方在发送一个数据包后等待接收方的确认，然后再发送下一个包。这种方法的缺点是如果网络延迟较大，会导致效率低下，因为发送方需要花费大量时间等待确认。

GBN（Go-Back-N）滑动窗口机制：在这个机制中，发送方可以发送多个数据包，而不需要等待每个包的确认。这提高了效率，但是如果发生丢包，发送方需要重新传输整个窗口内的所有数据包，这可能导致带宽的浪费。

SR（Selective Repeat）滑动窗口机制：这种机制允许发送方和接收方各自维护一个窗口，优化了数据传输。在这种情况下，只有丢失的数据包需要被重新传输，而不是整个窗口内的所有包，这降低了不必要的重传和带宽浪费。

(二) 实验内容

接下来我将整合之前三次作业的测试结果，并尝试改变更多的网络环境（如延迟和丢包率）来比较这些机制的性能。我将完成以下三组实验：

- 停等机制与滑动窗口机制性能对比：这将涉及对基础的停等机制和更复杂的滑动窗口机制（GBN 或 SR）在不同网络条件下的性能进行比较。
- 滑动窗口机制中不同窗口大小对性能的影响：在这里，您将比较在累计确认（可能指的是 GBN）和选择确认（SR）两种情况下，窗口大小如何影响性能。
- 相同窗口大小下的累计确认和选择确认的性能比较：这一实验将专注于在固定窗口大小的条件下，GBN 与 SR 机制的性能对比。

为了获得准确的数据，对每组实验进行三次测试并取平均值。这种方法将有助于确保结果的可靠性，特别是在面对复杂的网络环境时。

二、 实验设置

(一) 实验文件

使用统一的测试文件 1.jpg，确保文件大小在所有测试中保持一致。

(二) 传输机制

- 停等机制（Stop-and-Wait）。
- GBN 滑动窗口机制（Go-Back-N）。
- SR 滑动窗口机制（Selective Repeat）。

(三) 性能评价指标

- 传输时间:

  从发送第一个数据包到接收方确认接收最后一个数据包的总时间。测量在不同网络条件下，每种机制完成文件传输所需的时间。

- 吞吐率:

  以比特或字节为单位，计算单位时间内成功传输的数据量。吞吐率 = 总传输数据量（比特或字节）/ 总传输时间（秒）。

三、 停等机制与滑动窗口比较

(一) 不同丢包率

这部分实验我将在时延为 10ms，窗口大小为 8 的情况下，对不同丢包率下，三种流量控制机制的性能进行比较分析。

1. 传输结果

	0%	2%	5%	10%
停等	1.707	9.552	21.392	43.701
GBN	2.091	4.319	7.276	9.587
SR	2.71	4.043	6.979	7.358

表 1: 传输时间比较 (s)

	0%	2%	5%	10%
停等	8650.9	1545.97	690.309	337.912
GBN	8734.41	6367.87	3488.18	2708.19
SR	8450.45	6431.35	3435.02	3032.48

表 2: 吞吐率比较 (kbps)

在上面的表格中我们记录了不同丢包率下，三种流量控制机制的传输性能，下面我们将通过折线图来直观比较一下:

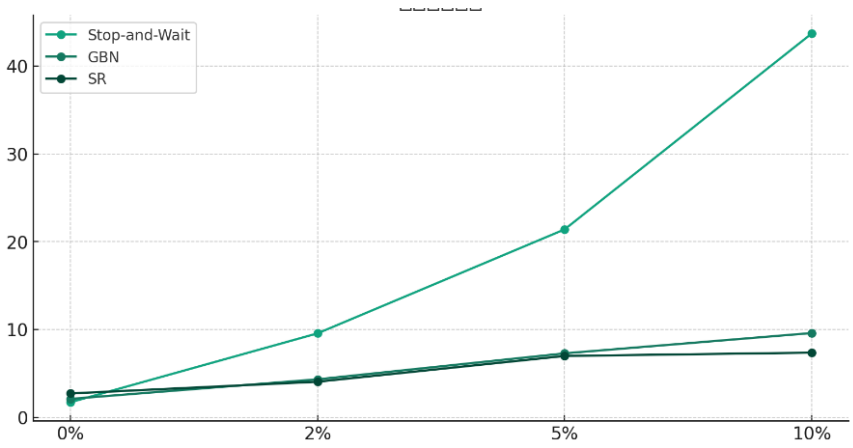

图 1: 传输时间变化

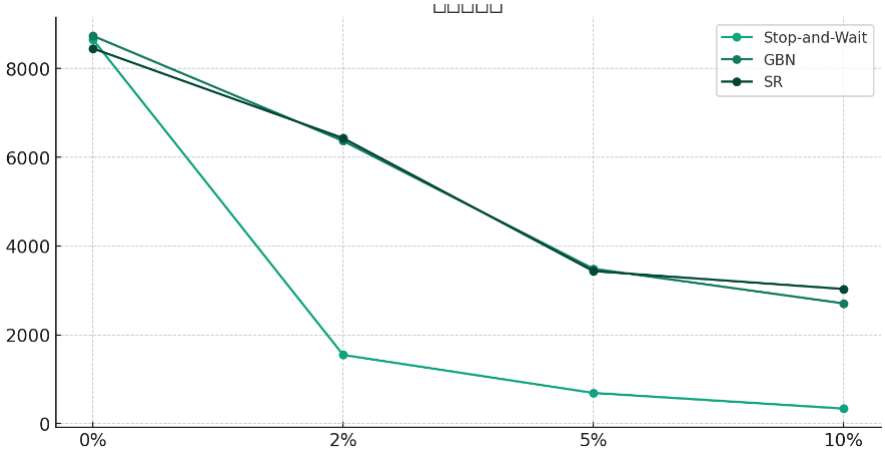

图 2: 吞吐率变化

2. 结果分析

传输时间分析

- 停等机制（Stop-and-Wait）：在所有机制中，停等机制在增加丢包率的情况下表现出最大的传输时间增长。特别是在 10% 的丢包率下，其传输时间显著高于其他两种机制。其丢包率也有相同的趋势。这反映了停等机制在面对高丢包率时的低效率。
- GBN 滑动窗口机制：GBN 机制在低丢包率时的传输时间略高于 SR 机制，但远低于停等机制。然而，在丢包率增加时，其传输时间增长速度不如停等机制显著。GBN 在低丢包率下表现出良好的吞吐率，但随着丢包率的增加，吞吐率有所下降。说明 GBN 在处理中等丢包率的情况下较为有效。
- SR 滑动窗口机制：SR 机制在所有丢包率下均展现出最佳的传输时间表现，特别是在高丢包率（如 10%）下，其效率优势更为明显。SR 机制在所有丢包率下也均维持了相对较高的吞吐率，特别是在中高丢包率下，其表现超过了 GBN 和停等机制。这表明 SR 机制在处理丢包时更加高效。

在窗口大小和延时时长固定的情况下，综合考虑传输时间和吞吐率，SR 机制在各种丢包环境下均表现出最优秀的性能，特别是在高丢包率环境下。GBN 机制在中等丢包率下表现良好，但

在高丢包率下性能下降。相比之下，停等机制在所有测试环境下的表现均较差，特别是在高丢包率环境下，其性能大幅下降。

3. 原因探究

在低延时的情况下，我们的实验结果与理论相符，我也对原因进行探究解释：

1. **停等协议耐丢包性能差的原因**：停等协议在发送一个数据包后需要等待对方的确认应答（ACK）。在此等待期间，不能发送其他数据包。如果数据包或 ACK 丢失，发送方必须等待一个超时周期后才能重传数据包，导致传输效率低下。由于同一时间只有一个数据包在传输，任何丢包都会直接导致整个传输进程暂停，直到该数据包成功传输并得到确认。因此，随着丢包率的上升，停等协议的吞吐率受到极大影响，下降速度最快。

2. **GBN 协议下降速度次之的原因**：GBN 协议允许在等待 ACK 的同时发送多个数据包，提高了传输效率。然而，如果一个数据包丢失，按协议规定，所有后续数据包都必须重新传输，哪怕它们已被接收方正确接收。随着丢包率增加，需要重传的数据包数量也增加，这降低了有效吞吐率。尽管如此，由于 GBN 协议可以发送多个数据包，它比停等协议更能容忍丢包，因此其吞吐率下降速度稍慢。

3. **SR 协议吞吐率下降最慢的原因**：SR 协议允许接收方独立接收和确认每个数据包，提高了传输的灵活性。如果一个数据包丢失，只需重传该数据包，不影响其他数据包的传输。这种机制使得 SR 协议具有最强的丢包耐受性。丢包率的增加会导致一些数据包重传，但与 GBN 协议相比，不会影响到连续的多个数据包。

(二) 不同时延

下面，我将在 8 窗口，丢包率为 0.005 的情况下改变超时时长，观察三种流量控制机制的性能。

1. 传输结果

	0.01	0.05	0.1	0.5
停等	1.707	3.552	4.512	14.361
GBN	2.091	2.746	3.995	22.262
SR	2.71	12.621	12.556	20.858

表 3: 传输时间比较 (s)

	0.01	0.05	0.1	0.5
停等	8650.9	4545.97	3890.309	937.912
GBN	8734.41	5377.67	3696.39	663.332
SR	8450.45	1170.04	1176.1	846.036

表 4: 吞吐率比较 (kbps)

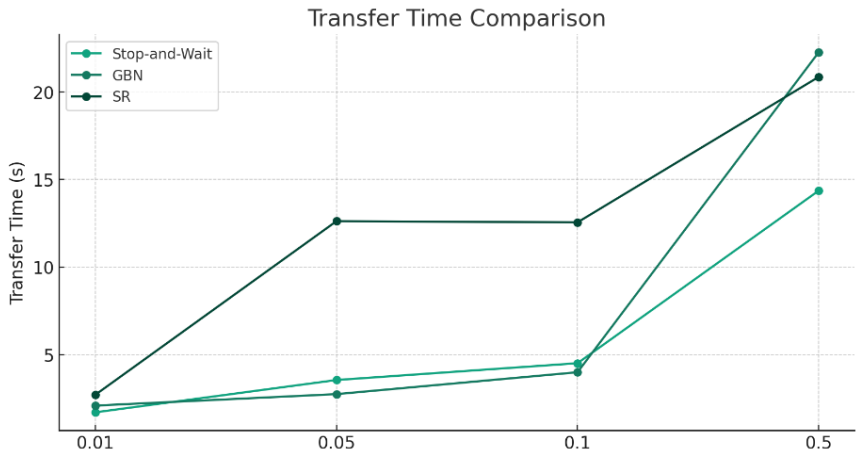

图 3: 传输时间变化

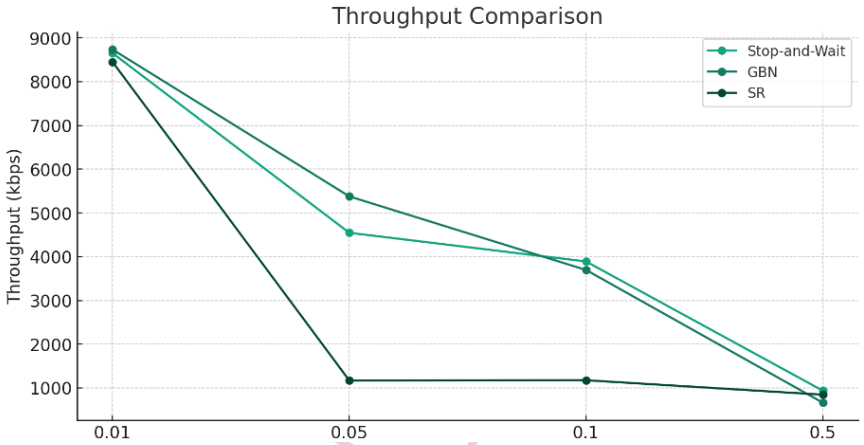

图 4: 吞吐率变化

2. 结果分析

- 停等机制：传输时间随着丢包率的增加而显著增长，特别是在丢包率达到 0.5 时，传输时间急剧上升。吞吐率随着丢包率的增加而显著下降。
- GBN 滑动窗口机制: 在低丢包率下, GBN 的传输时间少于 SR, 但随着丢包率的增加, GBN 的传输时间显著增加，并在 0.5s 时超越了 SR。在低丢包率下吞吐率较高，但随着丢包率的增加，其吞吐率也显著下降。
- SR 滑动窗口机制：SR 在低丢包率下表现较差，但随着丢包率的增加，其传输时间的增加幅度相对较小。尤其在丢包率为 0.5 时，SR 的传输时间低于 GBN。在低丢包率下，SR 的吞吐率与其他两种机制相当。然而，随着延时的增加，其吞吐率急剧下降,，这可能是由于 SR 在处理高丢包率时需要更多的重传。

总体来说，当延时较低时，GBN 机制在传输时间和吞吐率方面表现相对较好。但在高丢包率环境下，GBN 的性能急剧下降。SR 机制在低延时下表现不如 GB，但在高延时环境下，其传输时间和吞吐率的表现优于 GBN，所以在我的代码中 SR 更适合高延时的环境，而 GBN 在低延时环境下表现更好。出人意料的是，在 0.01 丢包率的情况下，停等机制是表现最好的一种机

制，同时它随超时时间改变而改变的幅度也没有其他两种机制大。说明在低丢包率下，延时时长对于停等机制的影响最小。

3. 原因探究

与理论相比，我的 SR 机制的性能出现异常，回顾代码，我认为主要出现在这部分：

超时重传

```
void timeoutResend() {
    while (isRunning) {
        // 超时重传检查逻辑
        for (int i = base; i < nextSeqNum; ++i) {
            int index = i % WINDOW_SIZE;
            // 判断单个数据包是否超时
            if (((clock() - packetTimes[index]) > timeout.count()) &&
                timeused[i]) {
                std::lock_guard<std::mutex> lock(mtx); // 在代码块中自动上
                    锁，出作用域时自动释放
                SetColor(11, 0); // 红色文本，蓝色背景
                cout << "Client:␣超时重传␣" << window[index].seq << "␣号数据
                    包" << endl;
                sendmessage(client, serveraddr, window[index]);
                packetTimes[index] = clock(); // 重新记录发送时间
            }
        }
        // 延时一段时间再次检查
        std::this_thread::sleep_for(std::chrono::milliseconds(500)); // 假设
            延时100毫秒
    }
}
```

我是启动了一个新的线程来检查是否超时，但是这部分检查时间也包含在总传输时间中。所以在延时较小的时候，这部分时间对总传输时间的影响很大，而在延时较长的时候，这部分时间的影响被削弱，最终造成了在延时较低时 SR 性能不如 GBN，在延时较高的情况下，SR 性能超过 GBN。

四、 窗口大小对性能影响

下面的实验中，我将模拟两种网络环境：

- 拥挤的网络环境：5% 丢包率，0.5s 延时。
- 畅通的网络环境：1% 丢包率，0.05s 延时。

在这两种情况下，我改变 GBN 和 SR 的窗口大小，来探究窗口大小对于两种机制的影响。

NKU

(一) 拥挤的网络环境

1. 实验结果

传输机制	4	8	16	32
GBN 机制	50.662	39.537	32.445	30.462
SR 机制	43.226	36.971	24.594	20.334

表 5: 拥挤环境下传输时间 (s)

传输机制	4	8	16	32
GBN 机制	291.483	362.204	392.737	484.771
SR 机制	341.625	399.424	652.823	726.227

表 6: 拥挤环境下吞吐率 (kbs)

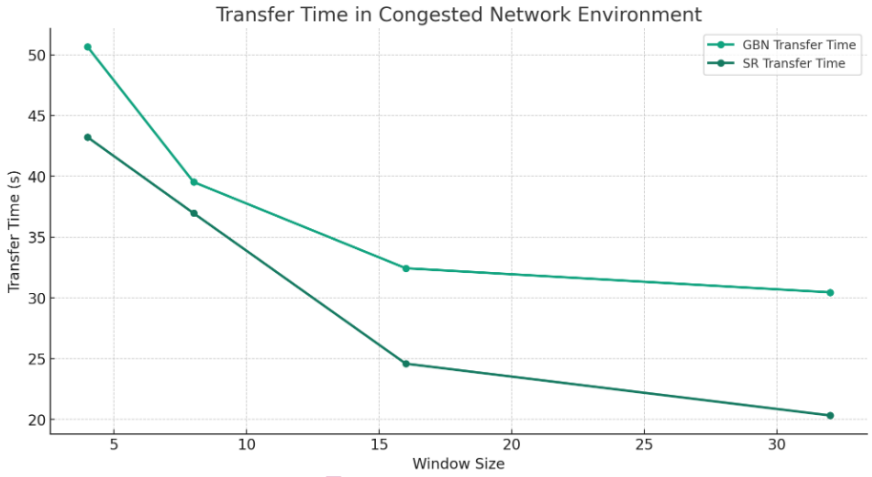

图 5: 传输时间变化

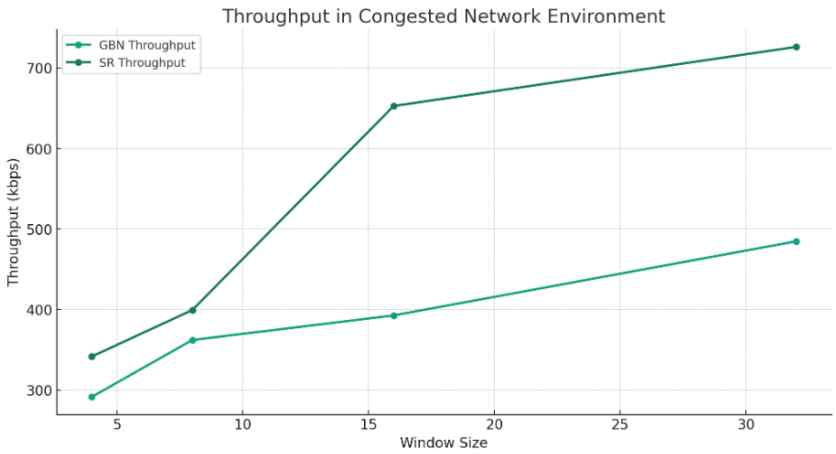

图 6: 吞吐率变化

2. 结果分析

在拥挤的网络环境下，我们可以观察到以下趋势：

- GBN 机制的传输时间随着窗口大小的增加而降低，但在窗口大小达到一定值后，降低的速度减缓。GBN 机制的吞吐率随着窗口大小的增加而增加，显示出随着窗口增大，可以在同一时间内发送更多的数据，从而提高吞吐率。
- SR 机制的传输时间同样随着窗口大小的增加而显著降低，降低的速度似乎比 GBN 更加显著。SR 机制的吞吐率随着窗口大小的增加也有所增加，并且在窗口较大时，SR 的吞吐率显著高于 GBN。

总的来说，在拥挤的网络环境下，增加窗口大小对于提高 GBN 和 SR 机制的性能是有益的，但 SR 机制似乎更能利用较大的窗口大小来显著提高其性能。这可能是因为其更高效的数据包管理，在面对高延时和丢包率时，能够更好地维持数据传输的连续性和吞吐率。

(二) 畅通的网络环境

1. 测试结果

	4	8	16	32
GBN 机制	3.23	3.042	2.992	2.774
SR 机制	4.453	3.998	3.937	3.057

表 7: 畅通下传输时间 (s)

	4	8	16	32
GBN 机制	4571.86	4854.4	4935.53	5436.4
SR 机制	4292.69	4126.49	4289.909	4825.35

表 8: 畅通环境下吞吐率 (kbs)

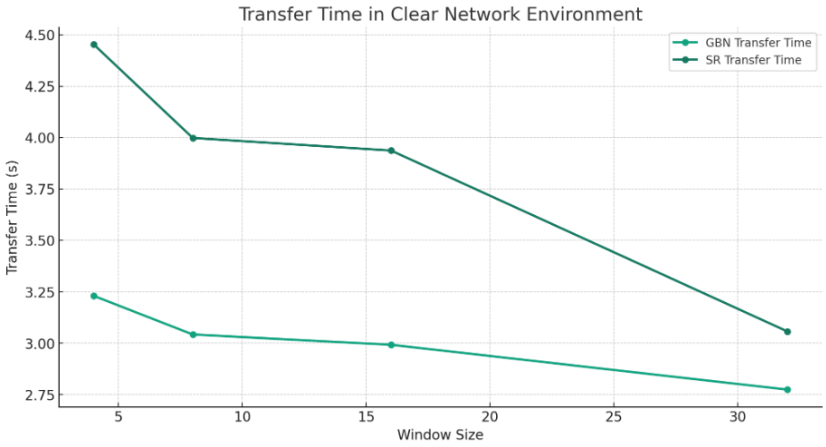

图 7: 传输时间变化

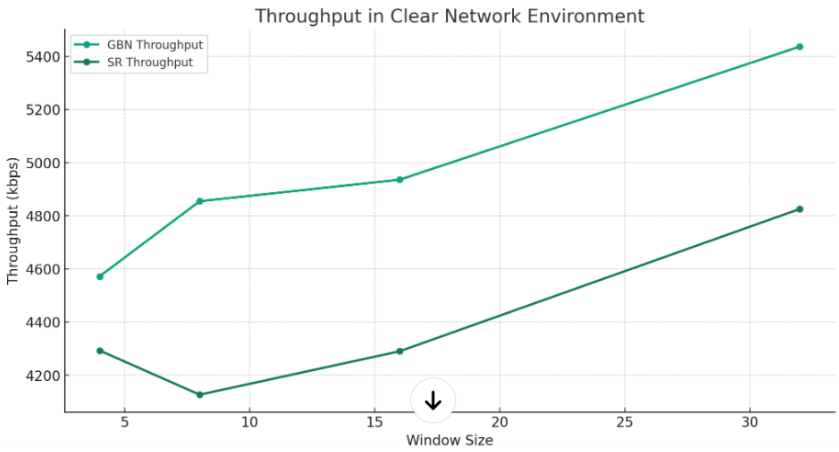

图 8: 吞吐率变化

2. 结果分析

在畅通的网络环境下，以下是观察到的趋势和分析：

- GBN 机制的传输时间随着窗口大小的增加而略微降低，但整体变化不大，说明在低延迟和低丢包率的网络条件下，窗口大小对传输时间的影响较小。GBN 机制的吞吐率随着窗口大小的增加而增加，表明在畅通的网络环境下，增大窗口大小有利于提高数据传输的效率。
- SR 机制的传输时间高于 GBN 机制，但随着窗口大小的增加，传输时间有所降低，尤其是当窗口大小增加到 32 时，传输时间显著降低。SR 机制的吞吐率在窗口大小较小时低于 GBN 机制，但随着窗口大小的增加，吞吐率有所接近，并有更大的增长趋势。

总的来说，尽管在畅通的网络环境下，增加窗口大小对于提高吞吐率有一定的帮助，但这种影响不如在拥挤的网络环境下显著。GBN 机制在小窗口大小时表现较好，但 SR 机制随着窗口大小的增加，性能提升更明显，尤其是在传输时间上的降低和吞吐率的提升上。

(三) 原因分析

通过拥挤网络环境和畅通网络环境下 SR 的性能对比，也验证了我之前对于 SR 代码的猜想。经过查阅资料，我也大概明白了 GBN 性能变化的原因：窗口的增大使得其更容易产生碰撞与等待，导致传输总量、总时间的增加，在网络拥挤情况下增加更为严重，性能不一定提高。

五、 确认方式对性能影响

接下来我将进行滑动窗口机制中相同窗口大小情况下，累计确认和选择确认的性能比较。我将采用窗口大小为 16，改变丢包率和时延，测量 GBN 协议和 SR 协议的传输时间与吞吐率

(一) 丢包率为 0.05，改变延时

1. 测试结果

	10ms	50ms	100ms	200ms
GBN 机制	2.108	6.07	10.986	20.902
SR 机制	3.049	9.476	13.937	18.057

表 9: 改变延时传输时间 (s)

	10ms	50ms	100ms	200ms
GBN 机制	7005.26	2432.8	1344.17	706.492
SR 机制	6948.326	2484.548	1289.909	825.35

表 10: 改变延时吞吐率 (kbs)

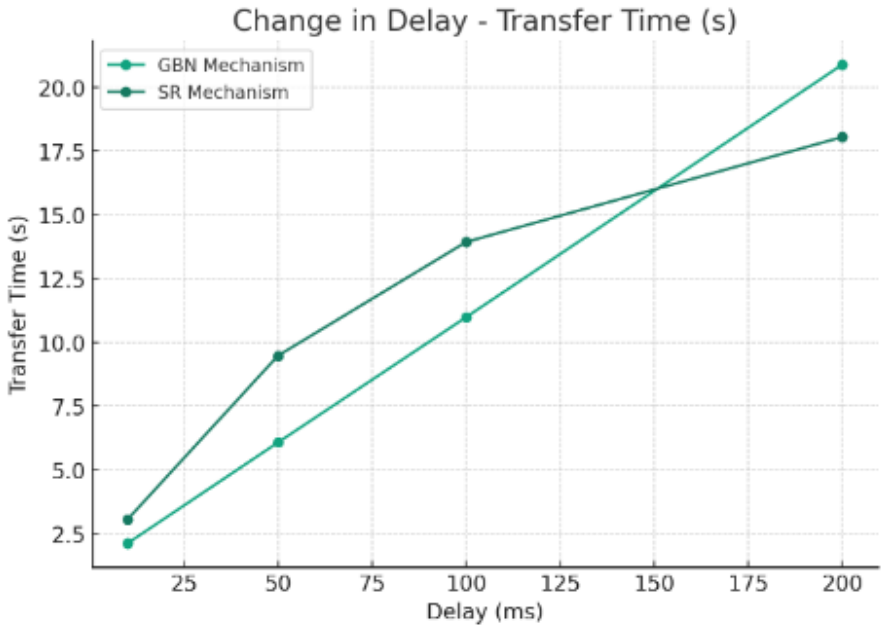

图 9: 传输时间变化

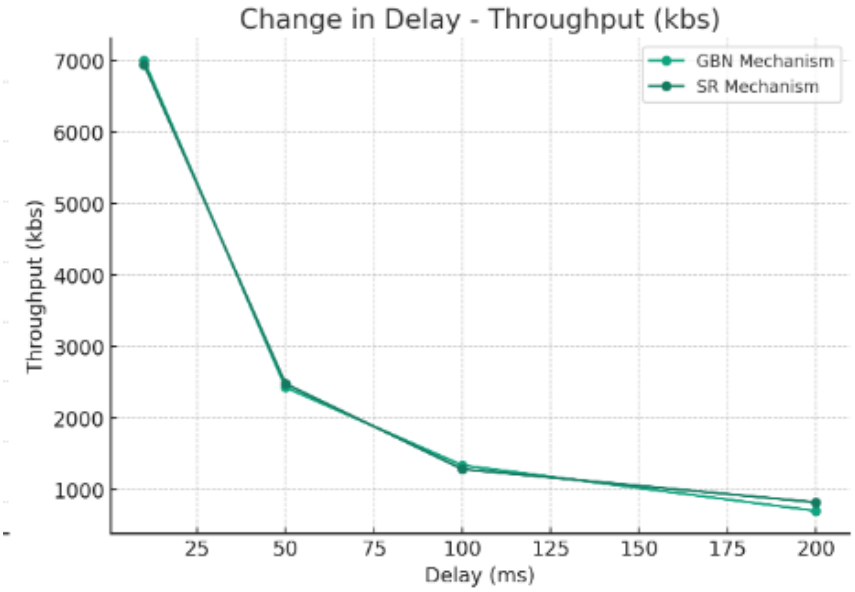

图 10: 吞吐率变化

2. 结果分析

改变延时 - 传输时间：随着延时的增加，GBN 机制和 SR 机制的传输时间都有所增加。GBN 机制在低延时下的传输时间普遍低于 SR 机制，表明 GBN 机制在处理低延时方面可能更加高效。在高延时中，SR 和 GBN 性能相近。

改变延时 - 吞吐率：随着延时的增加，两种机制的吞吐率都呈下降趋势。GBN 机制在大多数测试的延时中的吞吐率都高于 SR 机制，这可能表明 GBN 机制在数据传输效率上具有优势。这也主要是因为 GBN 在同等时间里重传了更多的数据包。

(二) 延时为 10ms，改变丢包率

1. 测试结果

	0.01	0.03	0.05	0.1
GBN 机制	2.619	7.158	10.991	21.967
SR 机制	4.535	10.998	14.937	17.057

表 11: 改变丢包率传输时间 (s)

	0.01	0.03	0.05	0.1
GBN 机制	5638.45	2063.02	1343.56	672.24
SR 机制	4280.2	1526.49	1289.909	825.35

表 12: 改变丢包率吞吐率 (kbs)

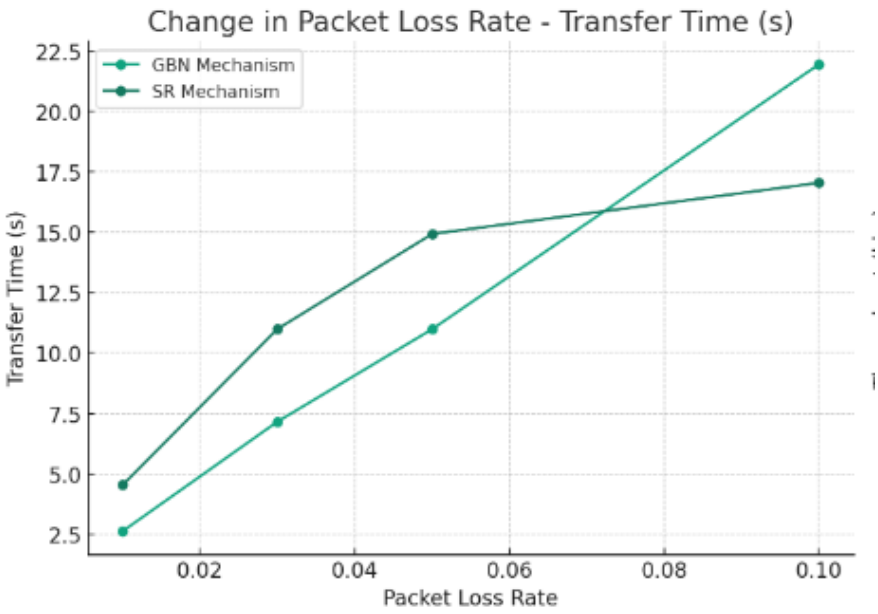

图 11: 传输时间变化

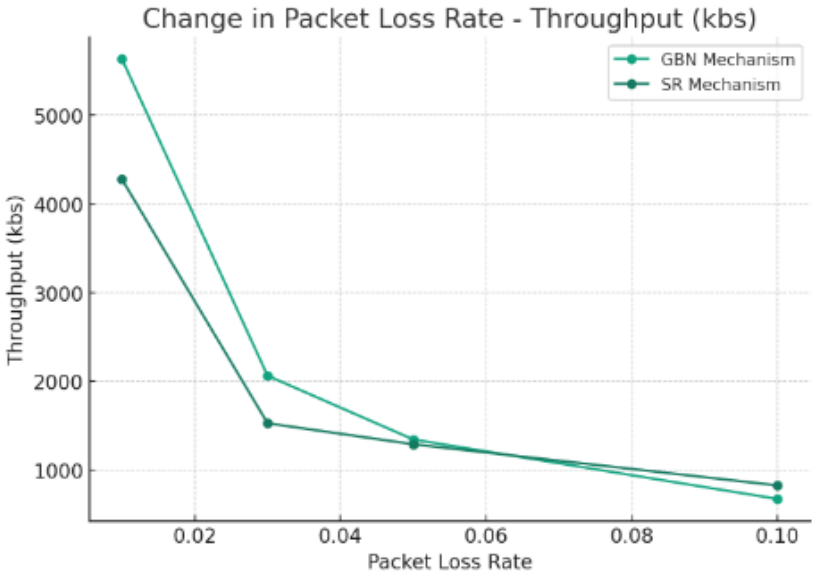

图 12: 吞吐率变化

2. 结果分析

改变丢包率 - 传输时间：随着丢包率的增加，两种机制的传输时间都有所增加。GBN 机制在各个丢包率下的传输时间普遍低于 SR 机制，但是在高丢包率的时候，GBN 稍逊于 SR。

改变丢包率 - 吞吐率：随着丢包率的增加，两种机制的吞吐率都有所下降。GBN 机制在大多数测试的丢包率中的吞吐率都高于 SR 机制，这可能表明 GBN 机制在较低丢包环境下依然能保持较好的性能。但是 SR 在较高丢包环境下优于 GBN。

(三) 原因分析

在这个实验中我发现，SR 的选择确认机制整体来说性能不如 GBN 的累计确认，这主要和我的代码编写有关。除了上文提到的检查超市花费时间以外，还有一个主要原因：**我使用了三个线程，每个线程都是不停地 while 循环，每个循环中我都加了互斥锁**。这就导致发送端每次的接受、检查、发送次序不固定，我在后续验证中也发现，即使接受不到发送端消息，客户端也会多次运行接受线程，这就导致其他线程无法运行。这也是 SR 性能不如 GBN 的重要原因。但是随着丢包率和延时增加，**由于互斥锁导致的延时影响被削弱,SR 超时无需重传缓冲区内所有数据包**，这也是后期 SR 的性能会超过 GBN 的原因。

整体来说 GBN 的吞吐率较大，这是因为 **GBN 每次重传的数据包要远多于 SR**。所以每个时间段吞吐的数据包也较多。

六、 总结与反思

在最后一次实验中，完成了对前面三次实验的总结与归纳，本此实验主要完成以下工作：

- 在不同丢包率和时延下比较三种流量控制机制。
- 在拥挤网络环境和畅通网络环境下比较两种滑动窗口的性能。
- 在固定窗口大小的情况下比较两种确认机制。

主要得出以下结论：

- 在丢包率很小的情况下，停等机制要优于其他两种机制，因为它不需要在代码中检查窗口的情况，所以传输时间也较少。其他情况下，两种滑动窗口机制都由于停等机制的性能。
- 在时延很少的情况下，两种滑动窗口的性能相近，并高于停等机制。
- 随着时延增加，SR 机制的性能先差于 GBN，之后会超过 GBN。这是由于 SR 采用了选择重传，比 GBN 全部重传节省了时间。

由于对不同传输机制的性能影响因素过多，包括机制自己的复杂度实现、网络环境的参数等，导致对性能的对比结果并不充分，也和理论效果有所差距。在分析原因的过程中，我也加深了对三种流量控制机制、两种重传机制、两种确认机制的了解。同时，我也找到了我的程序改进方向。