南開大學

计算机网络课程实验报告

UDP 可靠传输 Part 04

学　　院：网络安全学院

专　　业：信息安全

学　　号：2111252

姓　　名：李佳豪

班　　级：信安一班

1 实验背景与目的

在之前的工作中，完成了 c++ `socket` 编程中，使用三种机制实现 UDP 的可靠传输。这种可靠体现在传输通道的确认和数据的正确、有序。对于这三种机制（停等、GBN、SR）他们在实现难度、数据传输效率上各有优劣，停等机制虽然实现简单，但是若网络时延大时，导致大量双方大量**等待时间的浪费**；GBN 机制下，发送端增加窗口概念，更加复杂，但是出现丢包时导致多传输窗口内所有的数据包，导致**浪费过多宽带**；而对于 SR 机制，通过在发送端和接受端均增加窗口概念优化了上述问题，但是在传输通道紧张时，**无法解决拥塞问题**，因此希望能引进一定的拥塞控制算法...

可以看到前三部分的实验，我们一步步扩增算法使得我们的可靠传输更具效率、更能适应复杂的网络环境；而 Part4 中，我们重点不再是代码结构的设计和算法的利用，而是将重心放在在不同网络参数–丢包率以及延迟时间，**以他们为自变量，以文件传输的吞吐率和时延作为因变量，对不同机制的性能对比分析**；此外进行三种传输方式各自出现的问题进行进一步讨论。

2 实验内容

本次实验完成下面 3 组性能对比实验：（1）停等机制与滑动窗口机制性能对比；（2）滑动窗口机制中不同窗口大小对性能的影响（累计确认和选择确认两种情形）；（3）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。每一次记录的数据采取进行三次取平均；

3 实验前准备

3.1 实现丢包、延迟

使用 rooter.exe 无法很好的决定对网络延时的模拟，因为实际情况下的网络延时是时大时小，不能一概以一个时间讨论，否则会对这些机制传输的认识不全面；

因此丢包和延时在代码中简单模拟;

- **模拟丢包**，可以简单使用一个概率种子对每一次调用 `sendto` 函数时进行概率生成，判定是否发生丢包
- **模拟时延**，特别注意的是不同数据包延时时间不同，因此需要使用 `thread` 封装 `sendto`，之后利用概率使得对每一个数据包随机一个延迟时间，这个延时时间最小值适当即可，而最大值需要大于发生端超时时间，以便可以模拟现实中所有情况。

具体的代码实现已经在 *Part1* 中详细介绍过，这里不再赘述。

3.2 不同算法复杂度导致的时间不平等

由于三种算法的实现复杂度不同（尤其平等机制对数据打包、发送的处理下比后两者简单许多），而各自实现复杂度的差异，将影响后续数据的对比，因此引入新的参数`#define TWICE_GAP 32`，意思为相邻两次 sendto 函数调用的间隙固定为 32 个时钟周期，虽然这种操作使得原有传输速率得到限制，但却掩盖了不同算法之间复杂度的差异，有利于更好比较算法本身。

3.3 根据指标衡量性能

吞吐率直观表示了更高的数据传输效率，但也可能伴随着更高的丢包率或重传率；这里的性能主要体现在如何在更短时间内传输完成文件且实际发送数据尽可能少，即有较高吞吐率和较低的传输总量。

此外，不同机制下的传输机制在不同网络环境中的性能差距可能大相径庭，不同机制在不同参数下的性能也不尽相同；

因此后续对性能的对比需要综合考虑三种指标、如有需要还需要综合网络负载程度。

4 停等与滑动窗口对比

滑动窗口机制采取 GBN、SR，窗口大小为 8；

4.1 控制时延，改变丢包率

时延：*0ms*，两次发送间隙*:10ms*

通过对比，可以看到在不同丢包率下，这三种模式的表现差异。

- 首先关注传输总字节，随着丢包率的增加，每种模式的总字节数有所增加，但 SR 增长迅速，这也跟他每次重发窗口内所有包有关；
- 关注吞吐率，可以看到 GBN 较为稳定，这是因为他总是一个稳定速度进行发送新的数据报或者重传窗口缓冲区内的数据包，而 SR 反而有所增长，是因为他的传输时间增幅较小，而传输总量随着丢包率上升而上升，增幅大，导致吞吐率升高；而停等机制，吞吐率迅速降低...
- 总的说，相比停等机制，窗口机制的 SR 可以稳定使得传输效率提升，而 GBN 却反而降低传输效率。

表 1: 不同丢包率下停等机制、GBN 和 SR 的吞吐率对比

丢包率 (%)	停等机制吞吐率 (bps)	GBN 吞吐率 (bps)	SR 吞吐率 (Kbps)
0	347,315	370,029	370,029
5	296,779	352,275	362,733
10	285,526	356,584	395,533
15	251,480	353,652	413,862
20	209,936	348,337	425,439

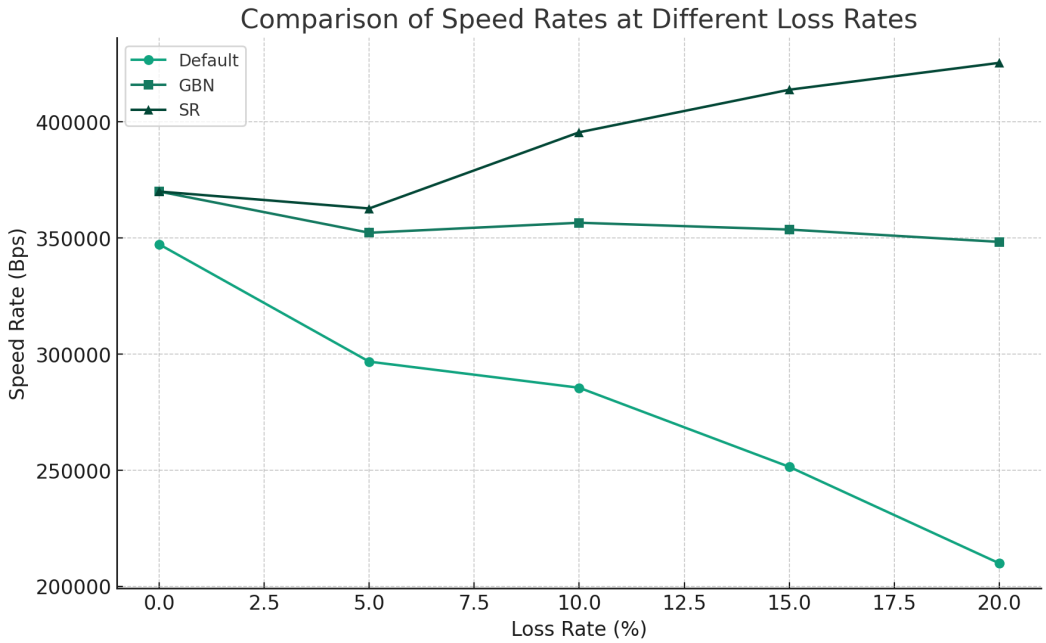

图 4.1: 不同丢包率下速率对比

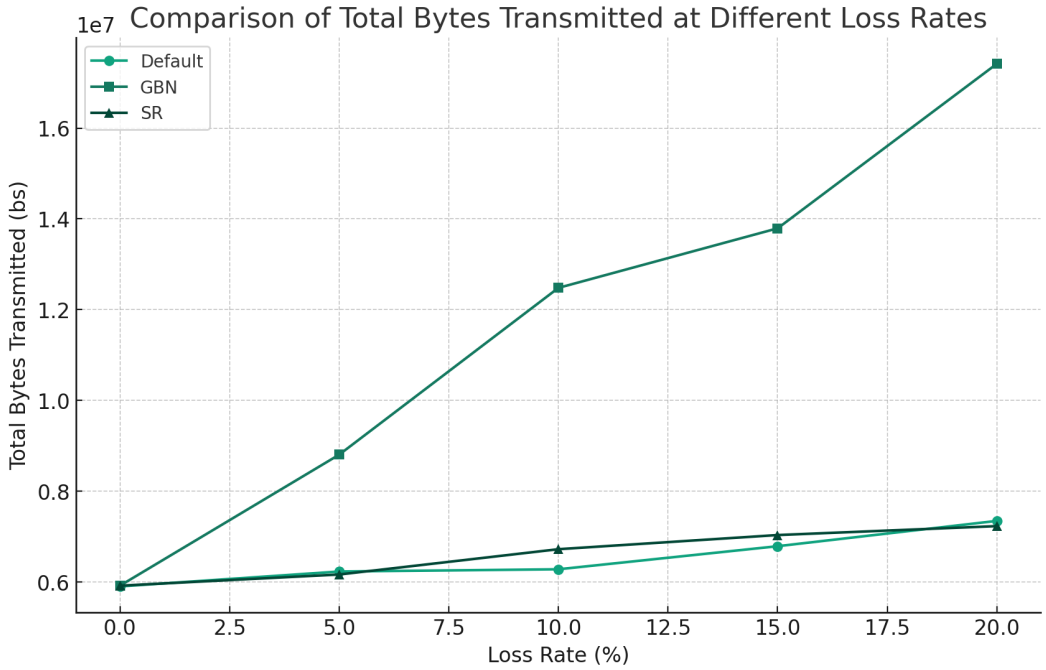

图 4.2: 不同丢包率下传输总量对比

4.2 控制丢包，改变时延

- SR、GBN 的曲线几乎全部一致，在传输总字节有所增加是因为随机延迟导致的失序，使得他们产生重传现象。
- 随着时延增加，停等机制传输时间明显增加、吞吐率明显降低，传输效率差；而在这种较低延迟和丢包情景中，GBN 和 SR 表现几乎一致；但停等机制性能下降明显

表 2: 不同时延下停等机制、GBN 和 SR 的吞吐率对比

时延 (ms)	停等机制吞吐率 (Kbps)	GBN 吞吐率 (Kbps)	SR 吞吐率 (Kbps)
10	347315	348262	5920720
30	347318	348262	5920480
50	268380	349227	5936864
100	147609	350192	5953264
150	101799	351156	5969664
190	86828	352121	6002464

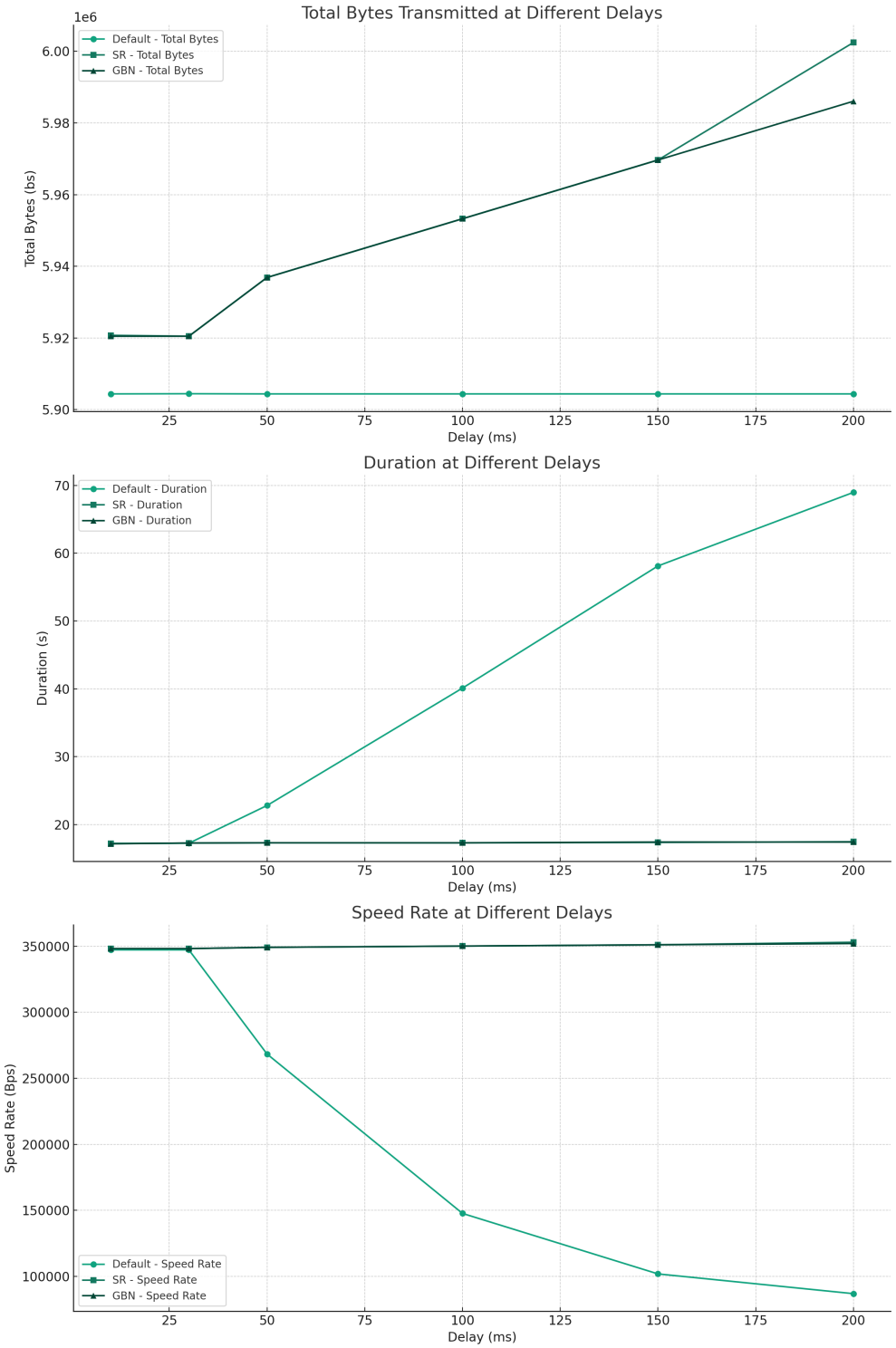

图 4.3: 不同延迟下传输总量对比

4.3 总结

总的来看，滑动窗口机制中 SR 不论如何修改条件，其性能总是优于停等，而 GBN 在丢包率较高 (或者说导致超时重传因素过多时)，效率将低于停等。

5 滑动窗口机制中不同窗口大小对性能的影响

需要考虑不同的网络负载下窗口的影响，因为在低负荷的网络中，数据包总能得到即使确认，因此不同窗口导致的性能差别可能无法体现。

情景 1 模拟网络通畅、超时重传和失序均很少出现的情景: 时延采用 50s，丢包率 1%,

情景 2 模拟较为复杂的网络环境，延迟的随机会导致超时和失序；时延 50-400 随机延迟，丢包率 5%

表 3: GBN 和 SR 在不同网络载荷下不同窗口的吞吐率对比

时延 (ms)	丢包率 (%)	窗口大小	GBN 吞吐率 (Kbps)	SR 吞吐率 (Kbps)
50	1	2	302657	254833
50	1	8	351236	266817
50	1	16	349495	252940
50	1	32	346330	234461
50-400	5	2	351156	351156
50-400	5	8	356944	338025
50-400	5	16	346640	317645
50-400	5	32	348995	300123

5.1 GBN 机制

- 对于吞吐率，不论什么场景，吞吐率变化很小，几乎稳定。但是传输时间、传输总字节数均增长迅速；在这种情况下，一旦丢包，窗口越大重新发送的数据包就越多，浪费的时间和传输总量也自然变大。
- 尤其关注传输的总字节数，可以看到随着窗口增大，负载低的网络其增长并不明显，但是当负载高的网络下增长尤其严重；

5.2 SR 机制

- 对于传输总时间，随着窗口增大，SR 在不同场景均能稳定在一个较小范围，具体来说，在负荷低的环境时间先减少再升高；而对于传输吞吐率在两个场景下均随着窗口的变大而有下降趋势
- 同样关注传输的总字节数，在低负载网络随着窗口增大，总字节几乎不变，但是高负载网络下大窗口明显降低了

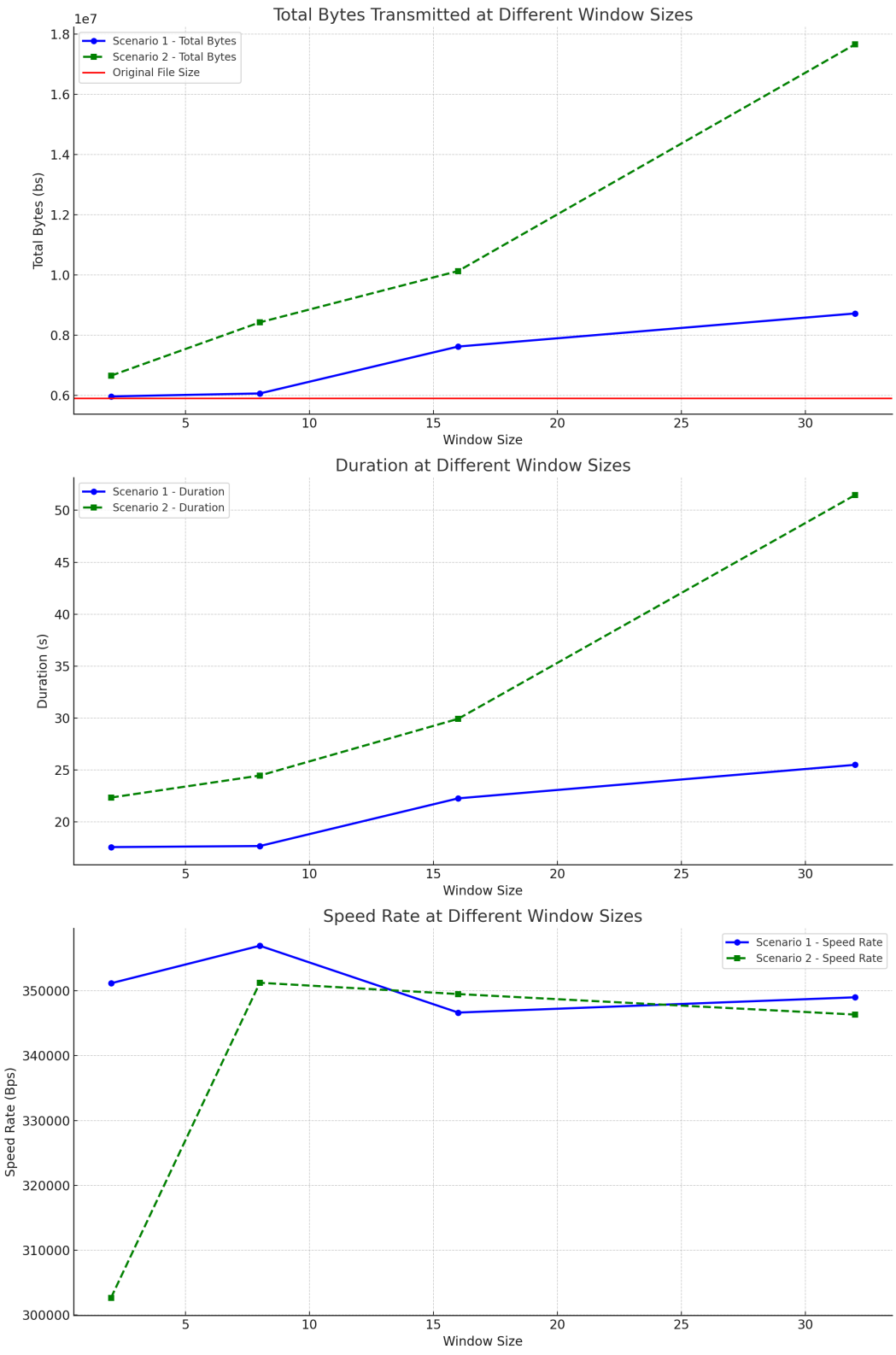

图 5.4: GBN：窗口大小变化下统计图

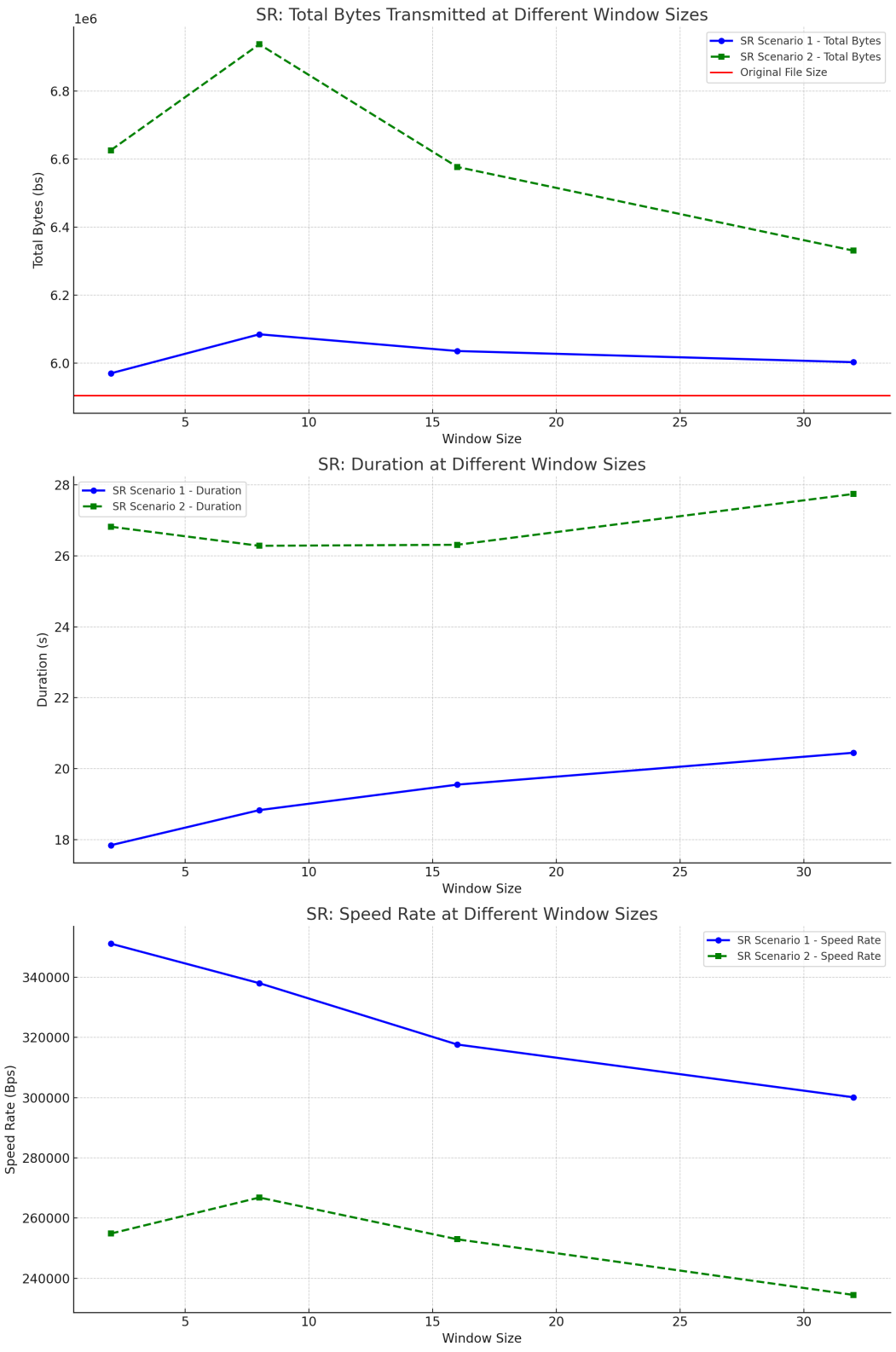

图 5.5: SR：窗口大小变化下统计图

5.3 总结

总的说，对于 GBN 机制，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；对于 SR 机制，选择确认机制保证其传输总量的得到稳定，随着窗口增大，在网络负荷大时性能依旧会降低。

6 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。

采取窗口大小为 8

6.1 固定时延，改变丢包率

- 总字节数：随着丢包率的增加，两种模式的总字节数都有所增加。GBN 模式的增加更显著，特别是在丢包率超过 10% 时。
- 传输时间：GBN 和 SR 模式的持续时间都随着丢包率的增加而增加，其中 GBN 模式的增加更为显著。
- 吞吐率：两种模式的速率都随着丢包率的增加而降低。SR 模式的速率下降比 GBN 模式更快，特别是在丢包率超过 5% 时。

Loss Rate (%)	Total Bytes (bs)	Duration (s)	Speed Rate (Bps)
GBN			
0	5936864	17.309	349227
5	6904480	20.152	345224
10	11840880	34.888	348261
15	12496880	37.089	337753
20	18433712	55.435	335158
SR			
0	5936864	17.29	349227
5	6396064	22.167	290730
10	7052048	27.132	261186
15	7216048	28.318	257716
20	7954080	35.745	227259

表 4: 修改丢包率

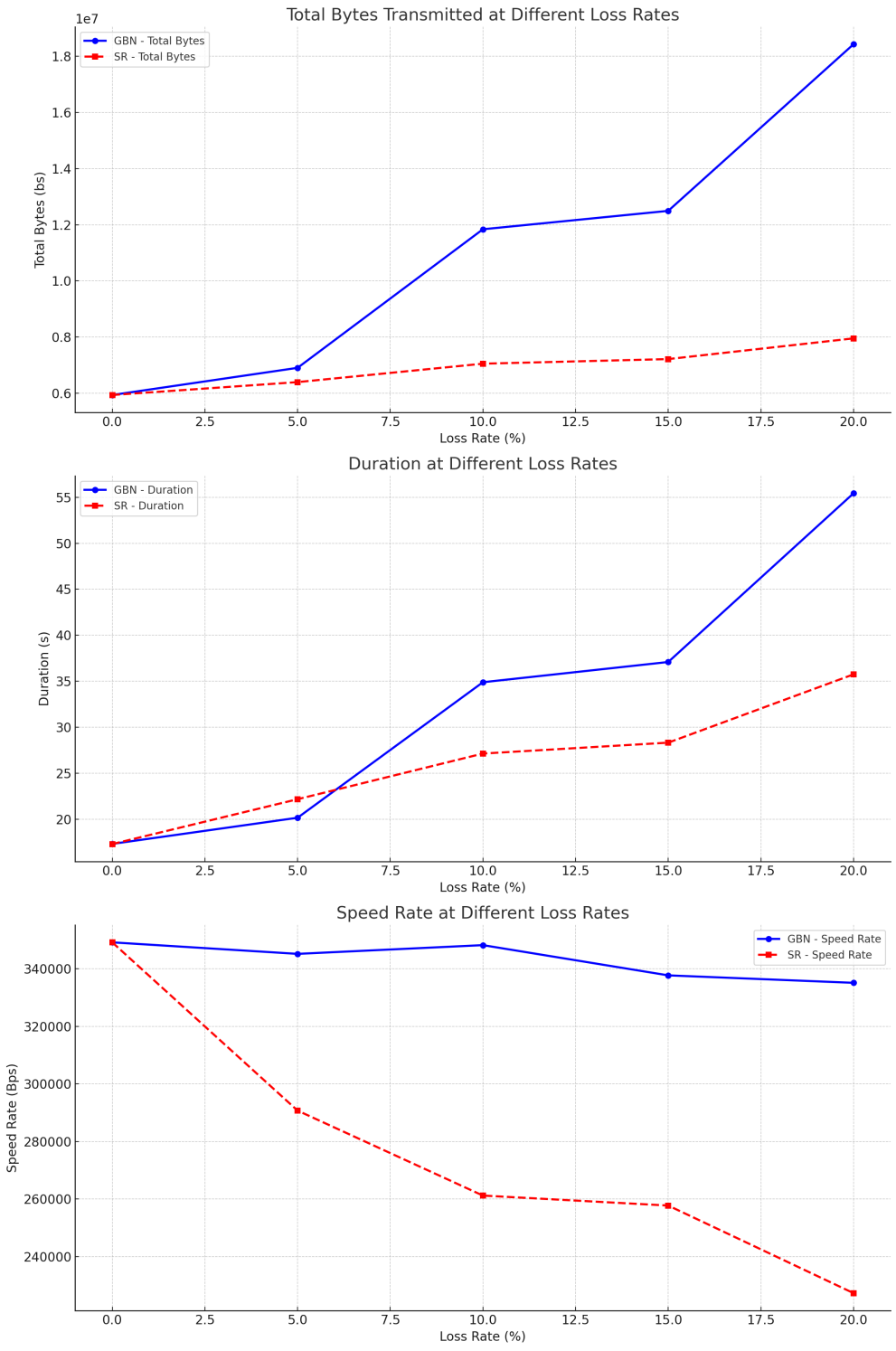
Total Bytes Transmitted at Different Loss Rates
1e7
GBN - Total Bytes
SR - Total Bytes
Total Bytes (bs)
Loss Rate (%)
Duration at Different Loss Rates
GBN - Duration
SR - Duration
Duration (s)
Loss Rate (%)
Speed Rate at Different Loss Rates
GBN - Speed Rate
SR - Speed Rate
Speed Rate (Bps)
Loss Rate (%)

图 6.6: 修改丢包率

6.2 固定丢包率，改变时延

这里引入参数最大延时率，因为在实现随机延时时，采用的随机特定概率产生大延迟、否则产生小延迟实现的，最大延时率则表示产生大延迟（将引起失序和超时），如下结果：

- 总字节数：随着最大延迟率的增加，两种模式的总字节数都有所增加，但 GBN 模式的增加更为显著，而 SR 机制可以稳定。
- 传输时间：两种模式的持续时间随着最大延迟率的增加而增加。但 GBN 稳定增长，SR 则稳定在一个范围，传输时间受时延影响小。
- 吞吐率：随着最大延迟率的增加，两种模式的速率都有所降低，但 SR 机制可以的吞吐率不会影响太多，在一个范围波动，而不是一直降低。

Max Delay Rate (%)	Total Bytes (bs)	Duration (s)	Speed Rate (Bps)
0	-	-	-
1	6068064	17.616	356944
5	6461664	18.823	358981
10	6461664	18.846	358981
20	6592864	19.165	346992
40	6625664	19.157	348719

表 5: GBN 机制

Max Delay Rate (%)	Total Bytes (bs)	Duration (s)	Speed Rate (Bps)
0	5986064	17.604	352121
1	5986064	17.604	352121
5	6018864	18.089	334381
10	6035248	18.042	335291
20	6182864	18.676	343492
40	6002448	17.625	353085

表 6: SR 机制

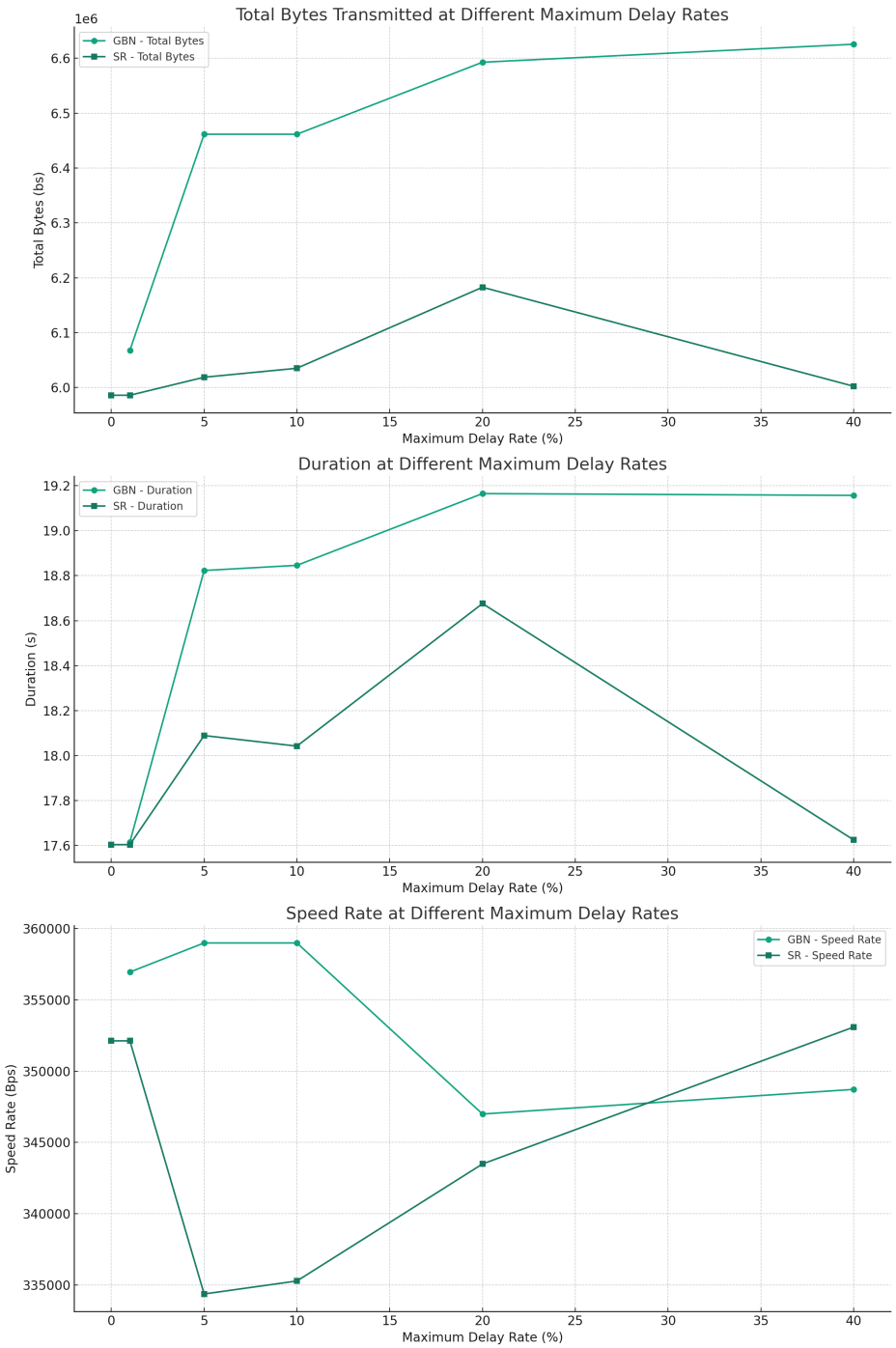

图 6.7: 修改最大延时

6.3 总结

无论是不同丢包率还是延时时间，SR 机制性能都优于 GBN 机制，尤其在延时越大、丢包率越大时，GBN 机制虽然能稳定自己的吞吐率在一个较高水平，但是其传输时间和传输字节数均线性的增长。因此 SR 机制更优。

7 实验结论及心得体会

- 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、网络环境的参数等等，导致对性能的对比结果并不充分
- 不过整体对三个机制有整体认知:
  - **停等机制**，算法实现成本最小，理论上网络极度顺畅时可以把发送端和接收端的性能发挥到极致，但是实际性能受网络时延和丢包影响波动较大。
  - **GBN 机制**，接收端需要维护队列对发送的数据包进行缓存，因此发送一次数据包的机器成本提高；在延迟环境中，传输时间所受影响小；但丢包的环境中每一次重传往往伴随窗口所有数据的重传，导致虽然维持吞吐率但占据很多网络资源；
  - **SR 机制**，接发端均需要维护队列保存发、接的数据包，进一步提高发送一次数据包的机器成本；但更能适应延迟较高、丢包率较高的情况；但窗口大小的选择依旧需要根据情况而定；

所有实验数据、画图程序均在远程仓库: *https://github.com/FondH/cn/*