计算机网络实验报告

实验3-4：性能对比实验

姓名：谢雯菲 学号：2110803

实验要求

基于给定的实验测试环境，通过改变延时和丢包率，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（累积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，**累积确认和选择确认**的性能比较。

实验环境

Windows 11

实验所用程序如`3-1`、`3-2`和`3-3`所写，每次传输数据包大小为1024字节，超时等待时间全部设置为200ms，在实验过程中有根据情况进行修改，具体可见实验过程。

吞吐率计算公式：`(float)fileSize / ((end - start) * 1000 / CLOCKS_PER_SEC)`

实验过程

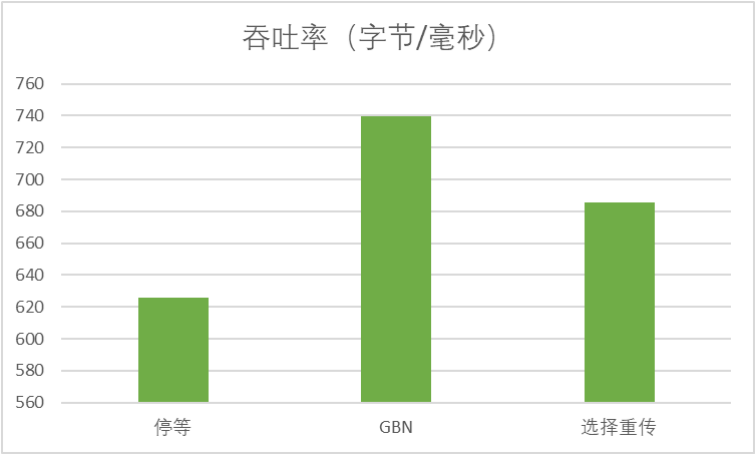
预实验

无丢包无延时初步比较

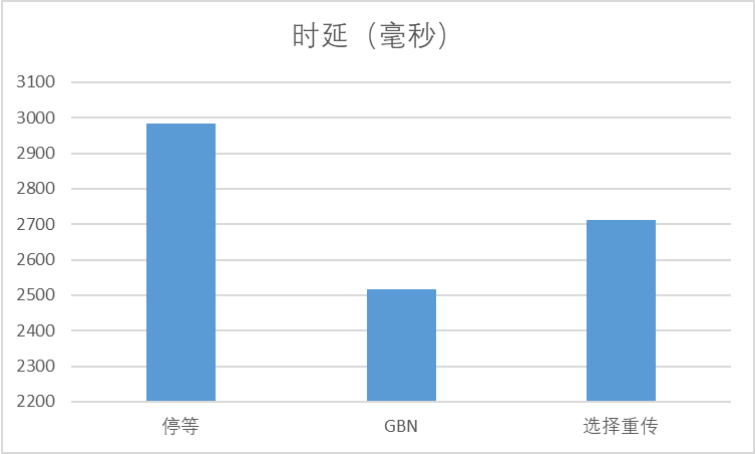
先在丢包率为0，延时为0，传输`1.jpg`，发送端超时等待时间为200ms的情况下进行实验，对停等和滑动窗口协议下的吞吐率和传输时间进行一个初步的比较，滑动窗口大小设为10，共进行三次实验取平均避免实验的偶然性。比较结果如下表所示：

实验次数	吞吐率（字节/毫秒）			传输时间（毫秒）		
	停等	GBN	选择重传	停等	GBN	选择重传
1	568	703	682	3272	2641	2724
2	633	740	671	2932	2511	2766
3	676	775	703	2747	2398	2643
平均	625.7	739.3	685.3	2983.7	2516.7	2711

吞吐率的比较如下图所示：



传输时间的比较如下图所示：



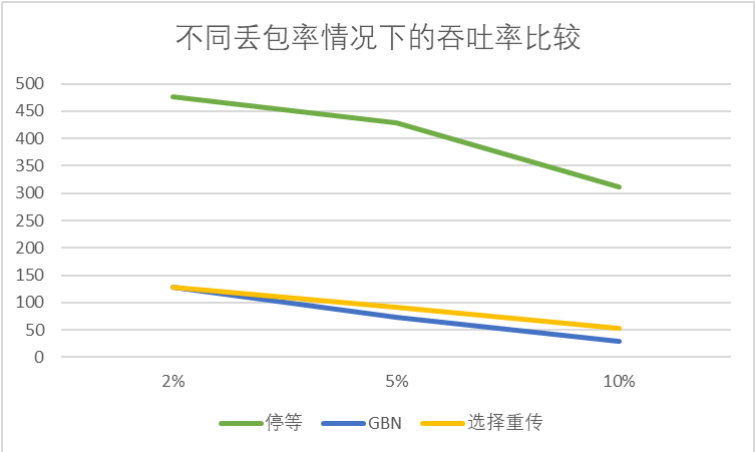
可见，使用`3-1`、`3-2`和`3-3`写的程序进行实验，在没有丢包的情况下，传输效率最高的是停等机制。这似乎与理论有些不符，但是由于吞吐率和传输时间相差不大，且没有丢包和延时，无法得出确定的结论。接下来，对有丢包的情况进行一个初步的实验进一步查看。

有丢包初步比较

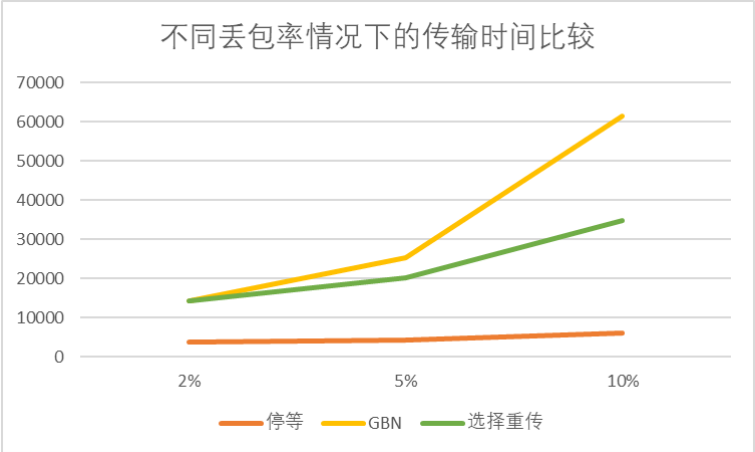
仍然设置最大等待时间为200ms，设置滑动窗口为10，传输`1.jpg`，在不同丢包率的情况下的传输效率如下表所示：

丢包率	吞吐率（字节/毫秒）			传输时间（毫秒）		
	停等	GBN	选择重传	停等	GBN	选择重传
2%	476	129	129	3906	14396	14408
5%	428	74	92	4344	25248	20186
10%	311	30	53	5981	61590	34731

吞吐率比较如下图所示：



传输时间比较如下图所示：



可见，在设定丢包率的情况下，仍然是停等机制的吞吐率较大，传输时间较少，且相比没有丢包的情形差异更大。

强行压制发送端速度

这种结果显然与理论是相悖的，观察文件传输过程可以发现，在实验条件下传输速度主要取决于接收端的接收速度，即使滑动窗口机制下的发送端可以加速发送，但由于接收端仍然需要等待和停等协议接近相同的超时时间接收丢失的包，因此传输速度并没有相差多少，反而由于滑动窗口使得接收方不得不处理失序的包，从而耽误了更多的一些时间。

因此，由于发送和接收速率相差太大，在滑动窗口机制下会出现发送方越发越多，接收方处理时间越来越多的恶性循环的情况。尝试强行压制发送端的速度，在三种机制下每次循环发送文件之后`sleep()`，模拟上层交付流较缓慢的情况。

分别尝试了`sleep(1)`、`sleep(2)`、`sleep(5)`、`sleep(10)`和`sleep(15)`的情况，在`sleep(10)`情况下进行实验的结果与实际情况较吻合，因此，后续实验都将在此基础上进行。

停等机制与滑动窗口机制的性能比较

无丢包无延时

在预实验的基础上，不设置丢包率，延时响应时长为200ms，滑动窗口设为10，发送`1.jpg`的情况如下表所示：

吞吐率	吞吐率（字节/毫秒）			传输时间（毫秒）		
	停等	GBN	选择重传	停等	GBN	选择重传
1	62	58	62	29764	31978	30001

2	53	63	62	35012	29695	29988
3	56	58	63	33465	32161	29873
平均	57	59.7	62	32747	31278	29873

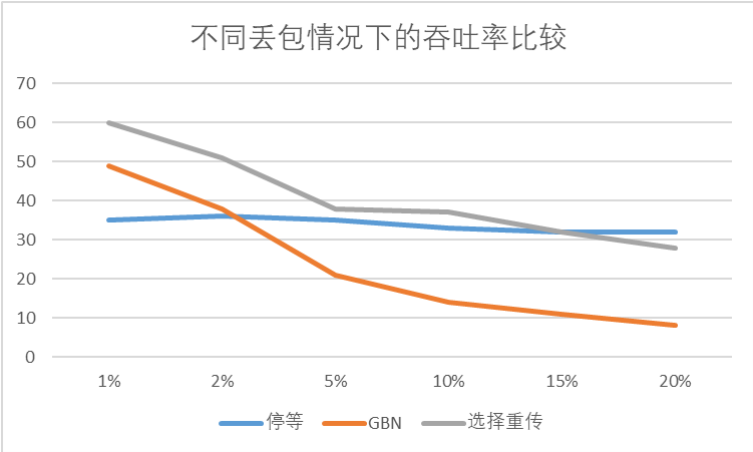
由上表可见，在不设置丢包的情况下，三种机制的传输效率差不多，传输时间也差不多，总体上吞吐率：选择重传>GBN>停等，传输时间：选择重传<GBN<停等。

不同丢包率情况的比较

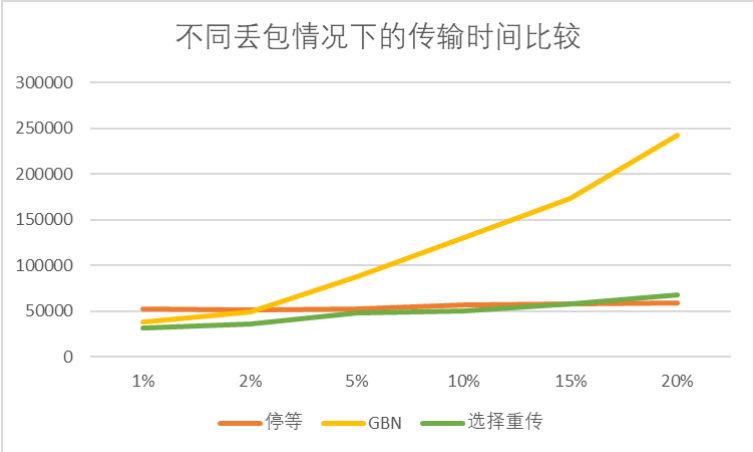
无延时，测试不同丢包率情况下的传输效率，其他条件与无丢包无延时时相同，结果如下表所示：

丢包率	吞吐率（字节/毫秒）			传输时间（毫秒）		
	停等	GBN	选择重传	停等	GBN	选择重传
1%	35	49	60	52922	37753	31097
2%	36	38	51	51421	48756	36190
5%	35	21	38	51421	48756	36190
10%	33	14	37	56613	130454	50201
15%	32	11	32	58342	173275	58220
20%	32	8	28	58445	242220	67398

吞吐率比较如下图所示：



传输速度比较如下图所示：



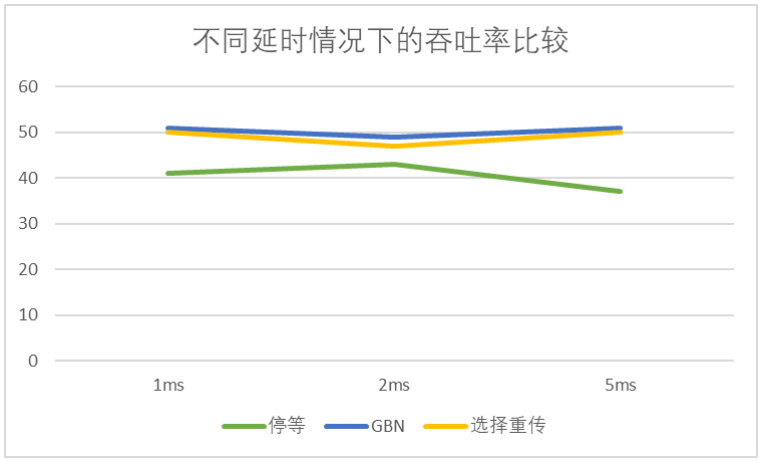
由上图可知，随着丢包率增大，三种机制的传输效率均有所下降，但是停等机制的传输效率变化较小。因此，可以总结出在丢包率较小（小于2%）时，滑动窗口机制的传输效率较高，在丢包率较大时，停等机制的传输效率较好。在实际中，丢包率一般不会超过5%，因此还是滑动窗口机制的性能较优秀。

不同延时情况的比较

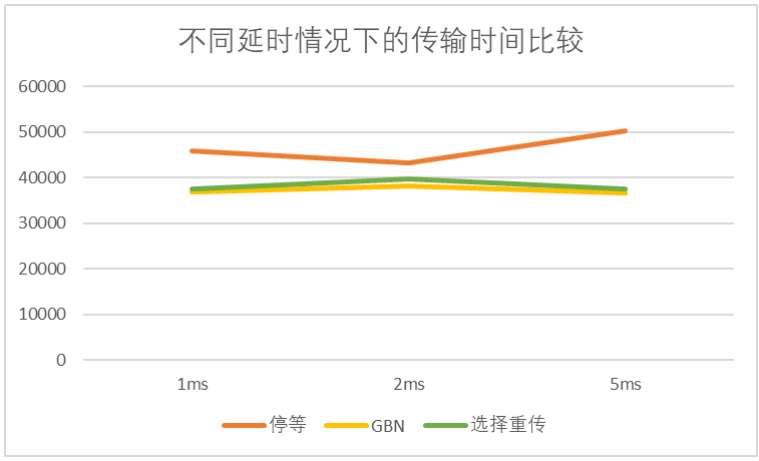
无丢包，在延时时间不同，其他条件与无延时无丢包时相同的情况下进行比较，结果如下表所示（最大延时仅测试到5ms，超过5ms后停等机制传输时间过长，本次实验未纳入测试）：

延时	吞吐率（字节/毫秒）			传输时间（毫秒）		
	停等	GBN	选择重传	停等	GBN	选择重传
1ms	41	51	50	45776	36747	37428
2ms	43	49	47	43158	38171	39734
5ms	37	51	50	50335	36619	37447

吞吐率比较如下图所示：



传输时间比较如下图所示：



由上图可知，在有时延的情况下，停等机制的传输效率不如滑动窗口机制，吞吐率总体上偏小。且在实验过程中，测试超过5ms延时的传输效率时，停等机制的传输效率会大幅度减少，猜测是由于超时等待时间设置得不匹配得问题，故在此没有纳入测试范围。

综上，在无丢包无延时的情况下，停等机制与滑动窗口机制的传输效率相差不大；在丢包率较小的情况下，滑动窗口机制的性能较好；在有延时的情况下，滑动窗口的性能较好。因此，在日常环境中，采用滑动窗口机制优于停等机制。

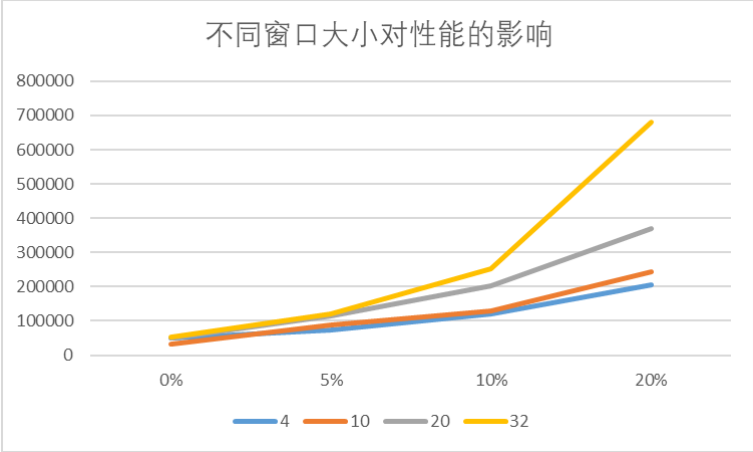
滑动窗口机制中不同窗口大小对性能的影响

累积确认

控制变量传输`1.jpg`，超时等待时间为200ms，在不同丢包率和不同窗口大小的情况下实验，所得结果如下表所示（传输时间（毫秒））：

丢包率	窗口大小			
	4	10	20	32
0%	50452	31278	50909	51846
5%	73762	87271	113244	119607
10%	119896	130454	203615	251278
20%	204415	242220	370336	680741

画出折线图如下图所示（纵坐标为传输时间（毫秒））：



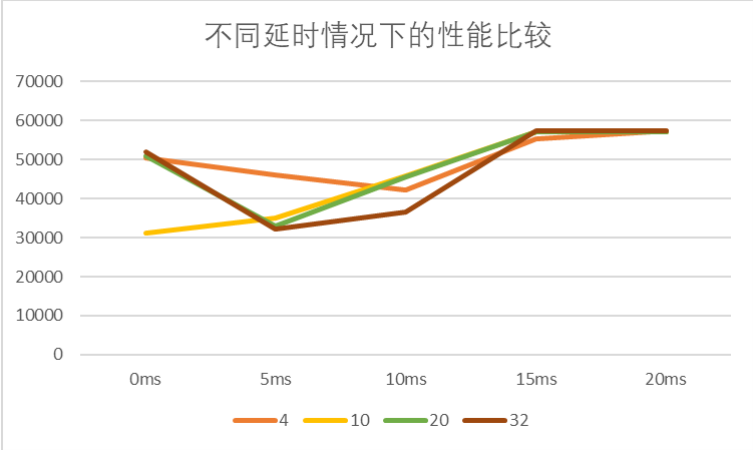
由上图可见，在丢包率较小时，不同窗口大小的性能相差不大；而在丢包率较大时，窗口越大，累积确认的传输效率越低。

这是由于累积确认每次重传时都需要将窗口中的数据包全部重传，而窗口越大，需要重传的包就越多，一旦某个包丢失或者延时，重传的时间就会大幅增加，传输效率就会降低。

在不同延时和不同窗口大小的情况下实验，所得结果如下表所示（传输时间（毫秒））：

延时	窗口大小			
	4	10	20	32
0ms	50452	31278	50909	51846
5ms	45987	35053	32952	32219
10ms	42111	45706	45623	36652
15ms	55195	57096	57113	57297
20ms	57226	57070	57165	57276

画出折线图如下图所示（纵坐标为传输时间（毫秒））：



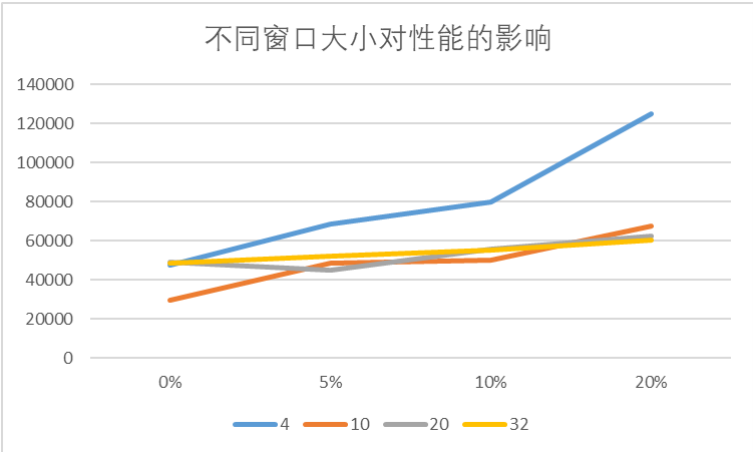
由上图可见，总体上来说，在延时时间较小时，较大的窗口传输性能较好；而在延时时间较大时，不管窗口大小，传输性能都有明显的下降，这是由于较大的延时极其容易引起累积确认GBN的恶性循环。

选择确认

控制变量传输`1.jpg`，超时等待时间为200ms，在不同丢包率和不同窗口大小的情况下实验，所得结果如下表所示（传输时间（毫秒））：

丢包率	窗口大小			
	4	10	20	32
0%	47535	29873	49279	48850
5%	68567	48527	44924	52470
10%	80016	50201	55952	55345
20%	124927	67398	62624	60690

比较的折线图如下所示（纵坐标为传输时间（毫秒））：



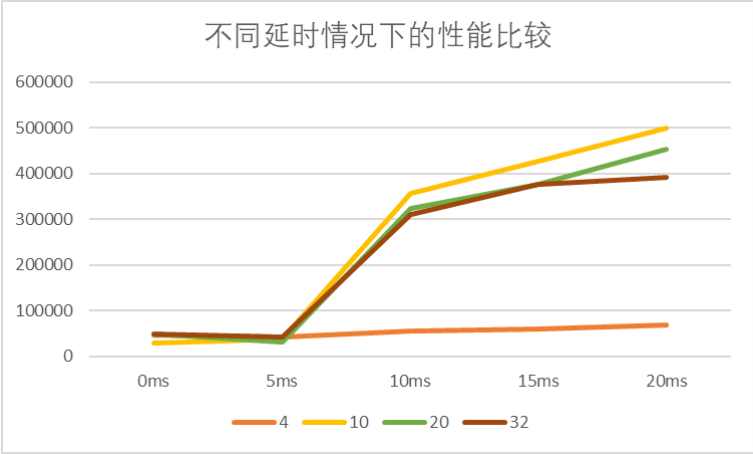
由上图可知，仍然是当丢包率较小时，窗口大小对性能的影响不大；而随着丢包率的增大，总体上来说窗口较大时，传输性能较好。而在窗口都比较大（20~32）时，性能的差异是非常细微的；且窗口较大时，丢包率对性能的影响更小。

在不同延时和不同窗口大小的情况下实验，所得结果如下表所示（传输时间（毫秒））：

延时	窗口大小			
	4	10	20	32

0ms	47535	29873	49279	48850
5ms	41316	38533	31744	41902
10ms	54946	355816	323622	311484
15ms	59115	426846	376330	377096
20ms	68750	500063	452620	391303

比较的折线图如下所示（纵坐标为传输时间（毫秒））：



由上图可知，在延时较小时，更大的窗口会有略微的优势，而在延时较大时，小窗口的重传时间更少，传输性能更高。但这也与发送端设置的超时判断时间有关，在高延时时适当增大发送端的超时判断时间会增加大窗口情况下的传输性能。

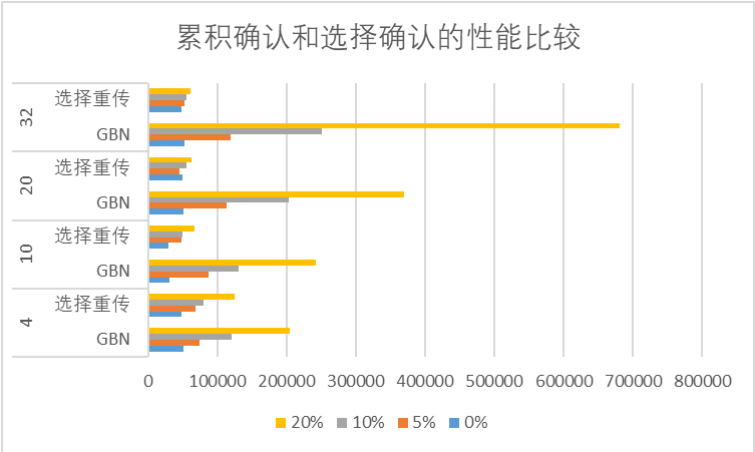
综上，在实际的存在丢包和延时的情况时，会更多地选择较大窗口的选择重传机制进行传输。

累积确认和选择确认的性能比较

使用上述实验中的数据，在不同丢包率的情况下的比较的表格如下所示（传输时间（毫秒））：

丢包率	**窗口大小**							
	4		**10**		**20**		**32**	
	GBN	选择重传	GBN	选择重传	GBN	选择重传	GBN	选择重传
0%	50452	47535	31278	29873	50909	49279	51846	48850
5%	73762	68567	87271	48527	113244	44924	119607	52470
10%	119896	80016	130454	50201	203615	55952	251278	55345
20%	204415	124927	242220	67398	370336	62624	680741	60690

比较的柱形图如下所示（横坐标为传输时间（毫秒））：

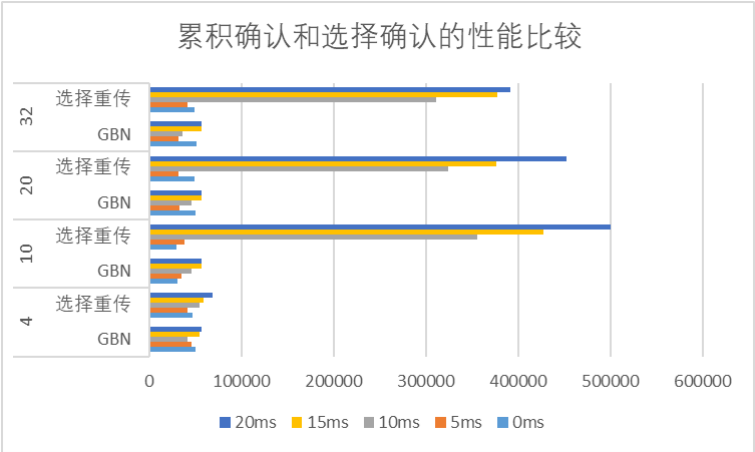


由上图可知，很明显在窗口大小相同时，选择重传的性能普遍更好，且丢包率越大，性能差距更明显。此外，随着丢包率的增大，累积确认的传输性能下降得更快，而选择确认的性能下降较为平缓。并且，就有丢包率的情况而言，随着窗口的增大，累积确认的传输性能反而降低，而选择重传的传输性能有所增加。

使用上述实验中的数据，在不同延时的情况下的比较的表格如下所示（传输时间（毫秒））：

延时	窗口大小							
	4		10		20		32	
	GBN	选择重传	GBN	选择重传	GBN	选择重传	GBN	选择重传
0ms	50452	47535	31278	29873	50909	49279	51846	48850
5ms	45987	41316	35053	38533	32952	31744	32219	41902
10ms	42111	54946	45706	355816	45623	323622	36652	311484
15ms	55195	59115	57096	426846	57113	376330	57297	377096
20ms	57226	68750	57070	500063	57165	452620	57276	391303

比较的柱形图如下所示（横坐标为传输时间（毫秒））：



由上图可知，在延时较小（<10ms）时，选择确认与累积确认的性能差异不大，在延时较大时，选择确认和累积确认的传输效率都有所下降，但是选择确认的效率下降更明显。推测是由于超时等待时间设置的200ms太小的原因，导致选择确认重传次数过多，接收端来不及处理。

接下来把累积确认和选择确认的超时时间调大，此处设为1500ms，再次比较在延时20ms时的传输时间，结果如下表所示：

延时	窗口大小							
	4		10		20		32	
	GBN	选择重传	GBN	选择重传	GBN	选择重传	GBN	选择重传
20ms	56316	56762	56399	56492	56289	56775	56340	56778

上述数据中，累积确认和选择确认的效率差不多，可以验证我们的猜想。也就是说，由延时带来的重传过多的问题是可以通过手动调整超时判断时间进行优化的。可以得知在同时存在丢包和延时的现实情况中，选择确认的性能更优。

综上，总体来说，在窗口大小相同时，选择确认相对累积确认可以适应更多的传输情况，性能更好。

总结

通过本次实验，深入对比分析了停等机制、累积确认机制和选择确认机制在传输性能上的差异，对实验过程中由于实验环境或编程方式导致的不合理之处进行了优化，在此基础上模拟真实环境在不同丢包率和延时的情况下进行实验比较，并对结果进行了较为全面的分析。