性能对比实验

实验探究内容：

1. 停等机制与滑动窗口性能对比；
2. 滑动窗口机制不同窗口大小对性能的影响；
3. 有拥塞控制和无拥塞控制的性能对比。

实验设计思路

1. 为了保证实验数据的可靠性，每次测量程序运行时间取五次运行时间的平均值；
2. 使用`C++`中的`chrono`计时函数，更加准确，并且在测试性能时取消日志输出，增加实验结果准确性；
3. 采用控制变量的方法，分别测试传输文件大小、路由器丢包率、路由器延迟三个变量对程序的性能影响。

实验结果

（1）停等机制与滑动窗口性能的对比

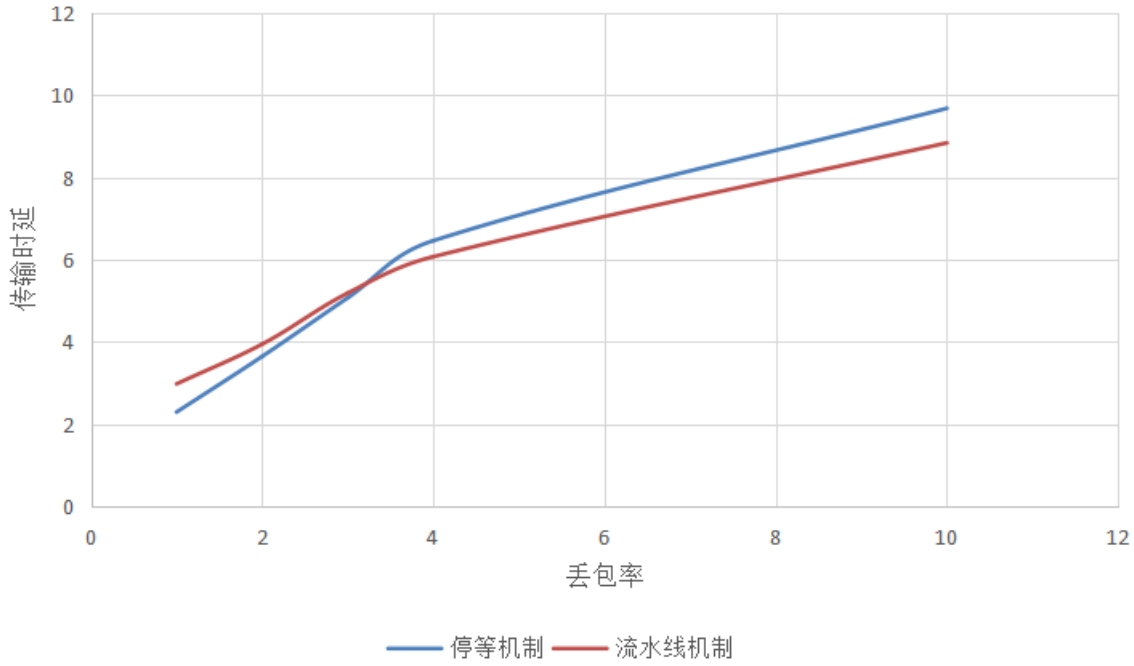
传输时延的对比：

Miss Rate	1%	2%	3%	4%	10%
Latency/ms	0	0	0	0	0
停等机制	2.293	3.646	5.072	6.462	9.682
滑动窗口机制	2.978	3.945	5.188	6.073	8.841

吞吐量的对比：

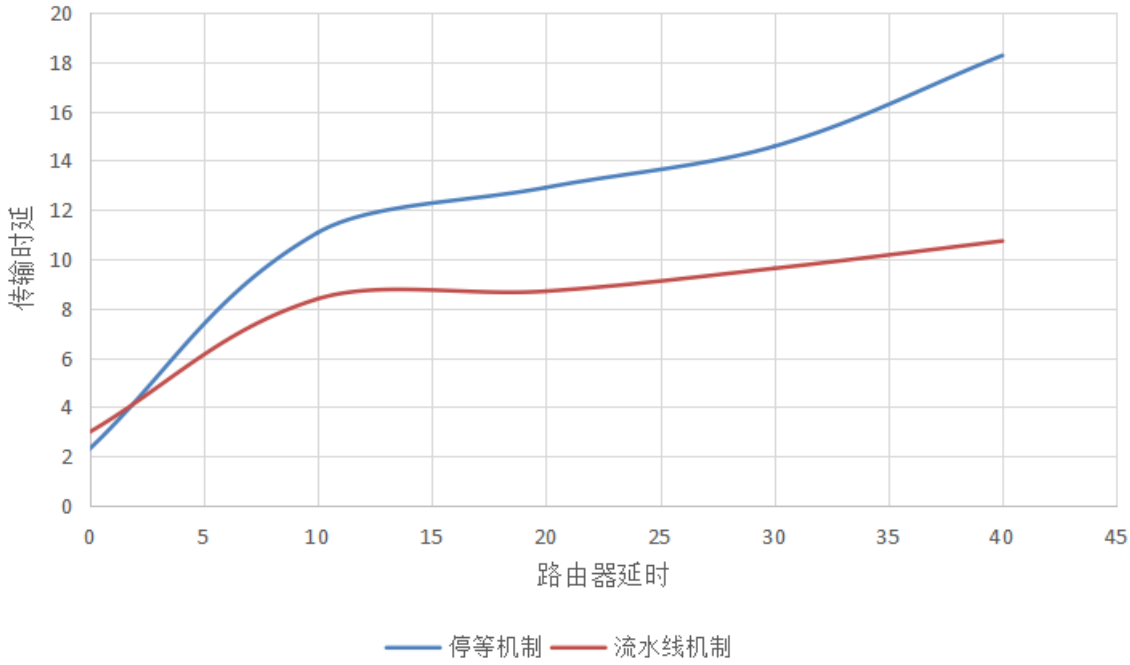
Miss Rate	1%	2%	3%	4%	10%
Latency/ms	0	0	0	0	0
停等机制	810010.0305	509422.1064	366197.358	287426.9576	191835.6744
滑动窗口机制	623691.4036	470811.9138	358009.4449	305837.8067	210084.0403

作出传输时延曲线图：



估计丢包率，改变路由器延迟时间再做一组对比实验：

Miss Rate	1%	1%	1%	1%	1%
Latency/ms	0	10	20	30	40
停等机制	2.293	11.072	12.904	14.575	18.262
滑动窗口机制	2.978	8.386	8.698	9.624	10.734



最后对比不同文件大小两种传输机制的传输时延：

Data Len/Byte	1857353	5898505	11968994
停等机制	2.293	10.861	24.783
滑动窗口机制	2.978	11.962	21.366

综上可以总结停等机制和滑动窗口机制的性能差异：

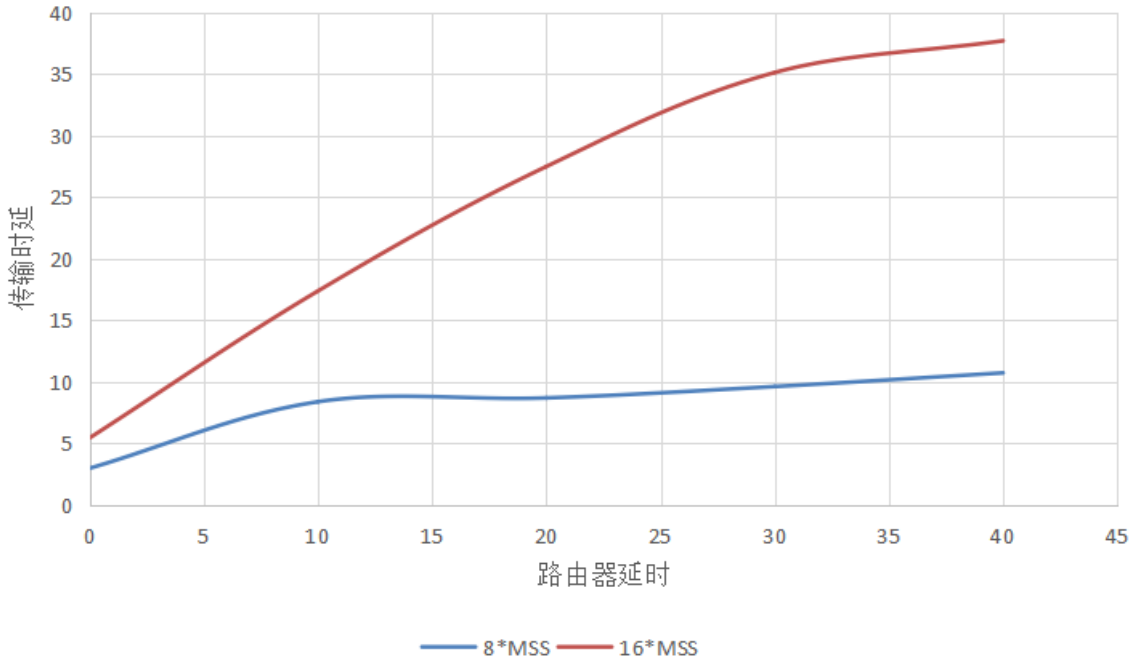
总体而言滑动窗口的传输性能比停等机制更好，因为其可以一次性发送多条信息而减少了等待接收端回复ACK的时间，尤其随着路由器延时的增加RTT增加，滑动窗口的性能会明显优于停等机制。而丢包率的增加都会造成停等机制与滑动窗口性能的下降，但是两者始终性能差距不大，甚至滑动窗口有时性能不如停等机制。而随着传输文件大小增大，滑动窗口的性能优势更加明显。

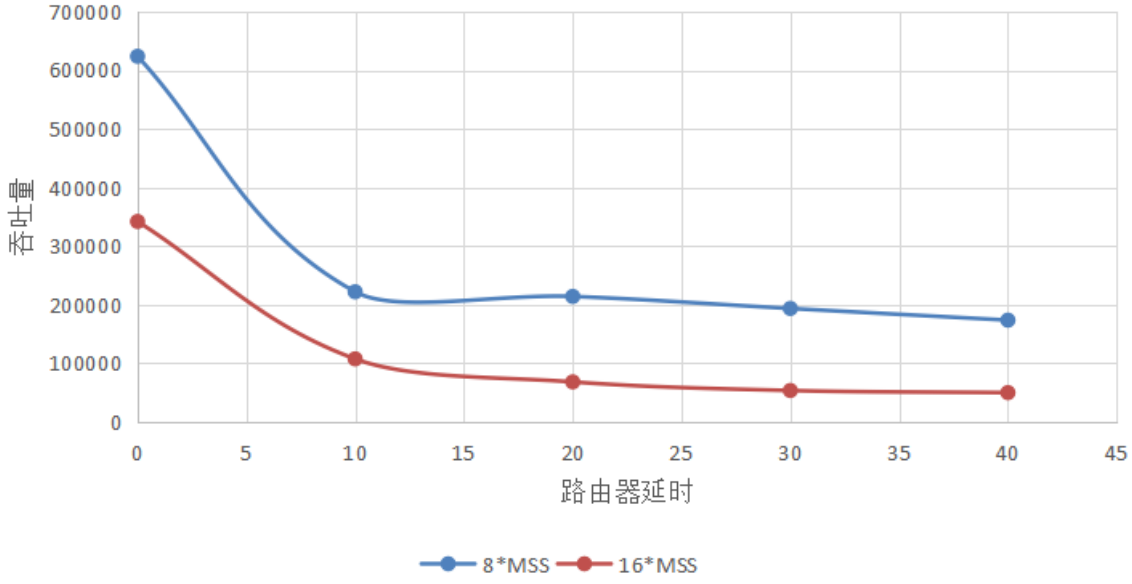
（2）滑动窗口机制不同窗口大小对性能的影响

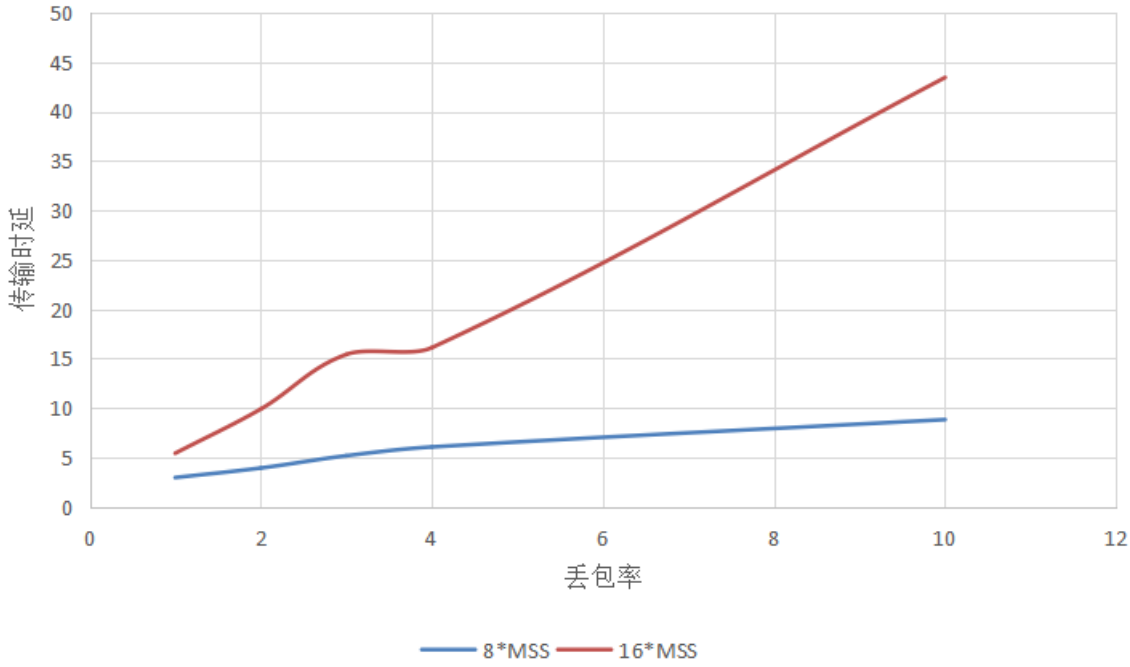
对比窗口大小分别为8×MSS和16×MSS时的文件传输性能：

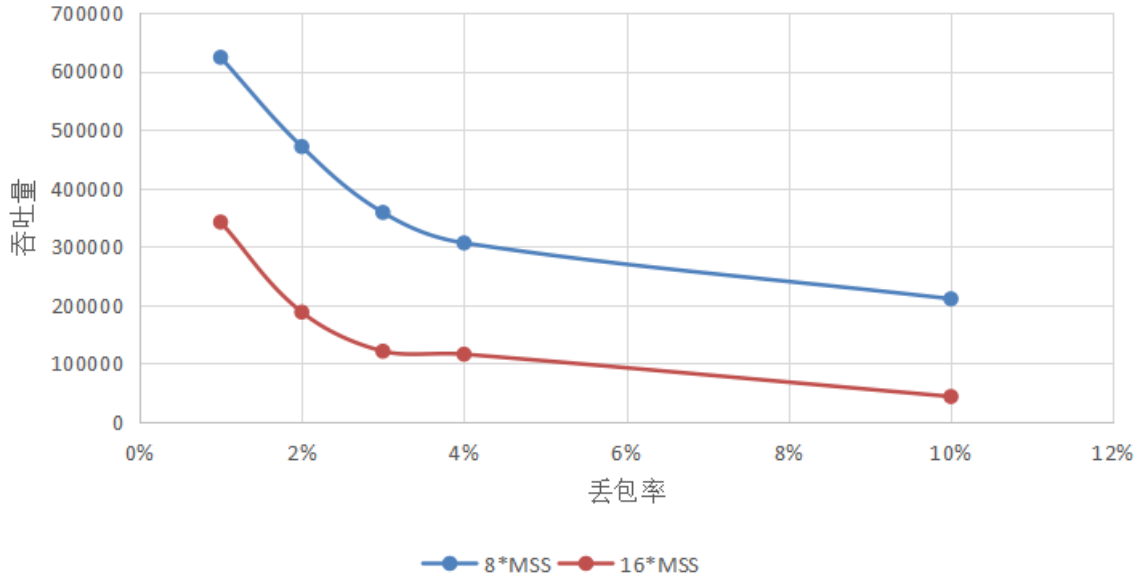
Miss Rate	1%	1%	1%	1%	1%
Latency/ms	0	10	20	30	40
8*MSS	2.978	8.386	8.698	9.624	10.734
16*MSS	5.438	17.386	27.504	35.145	37.724

Miss Rate	1%	2%	3%	4%	10%
Latency/ms	0	0	0	0	0
8*MSS	2.978	3.945	5.188	6.073	8.841
16*MSS	5.438	9.911	15.43	16.119	43.441




700000
600000
500000
400000
300000
200000
100000
0
吞吐量
0
5
10
15
20
25
30
35
40
45
路由器延时
8*MSS
16*MSS

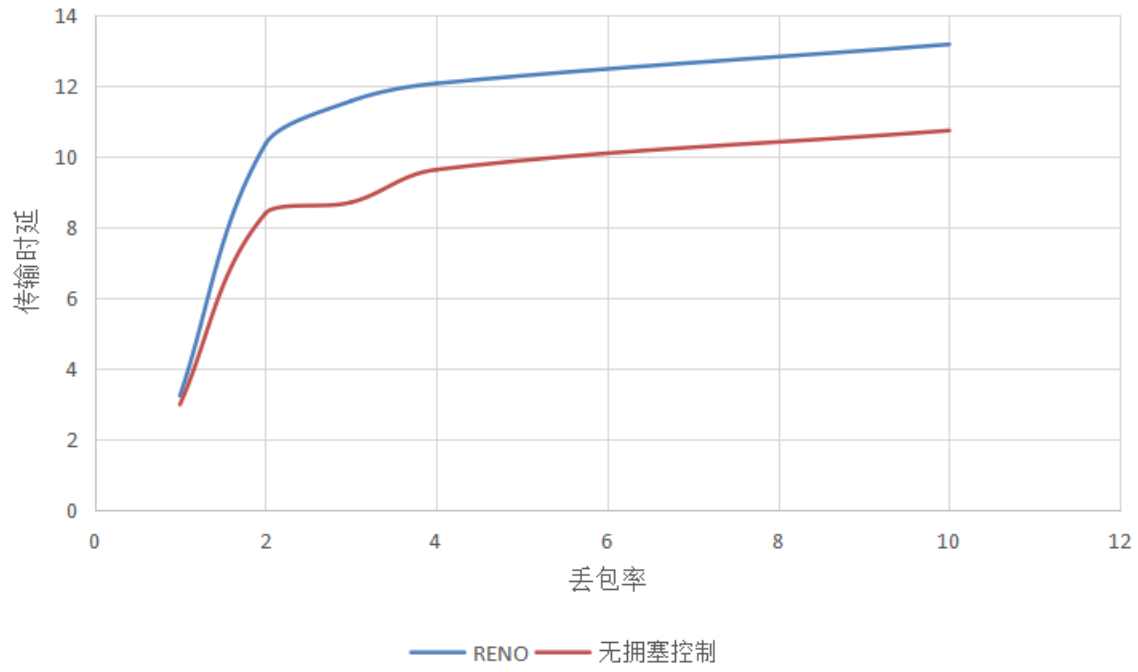

50
45
40
35
30
25
20
15
10
5
0
传输时延
0
2
4
6
8
10
12
丢包率
8*MSS
16*MSS

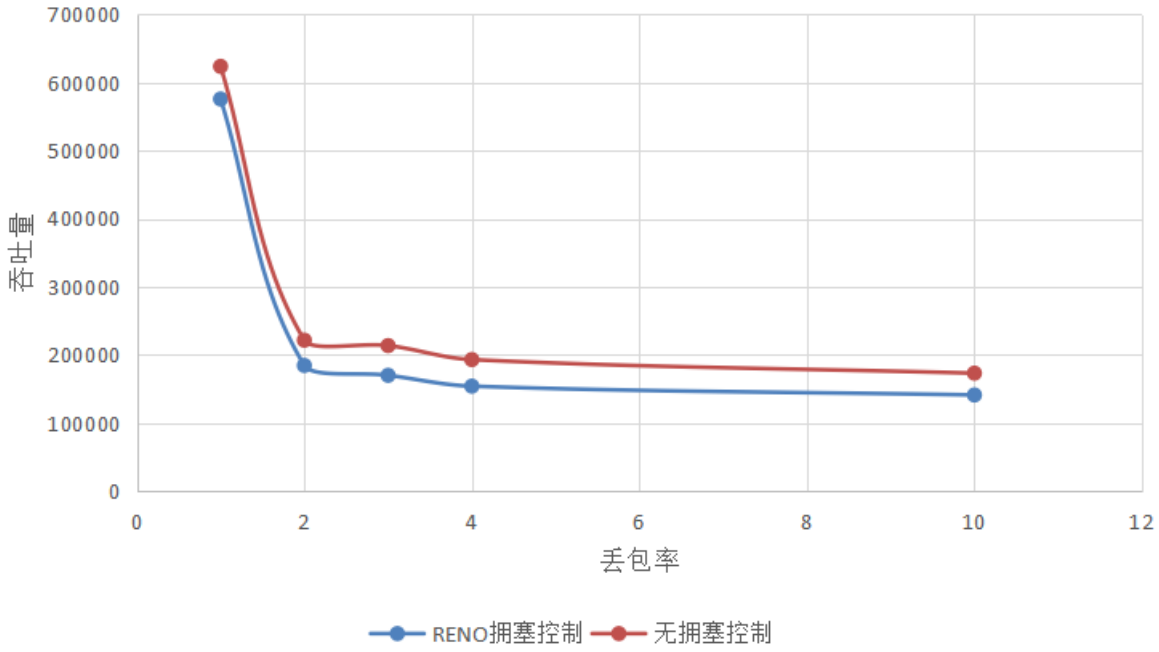

700000
600000
500000
400000
300000
200000
100000
0
吞吐量
0%
2%
4%
6%
8%
10%
12%
丢包率
8*MSS
16*MSS

可以明显对比出来，当窗口大小为8×MSS时传输性能明显优于16×MSS，说明并不是窗口越大传输性能效率越快，经过探究，发现在窗口大小为16*MSS时丢包率远大于路由器设定的丢包率，根本原因发送端一次性发送太多内容，远超过接收端的处理能力或者甚至超出UDP的缓冲区，导致频繁丢包，传输性能下降。由此可见，选择合理的窗口大小在滑动窗口机制里具有重要意义。

（3）有拥塞控制和无拥塞控制的性能对比

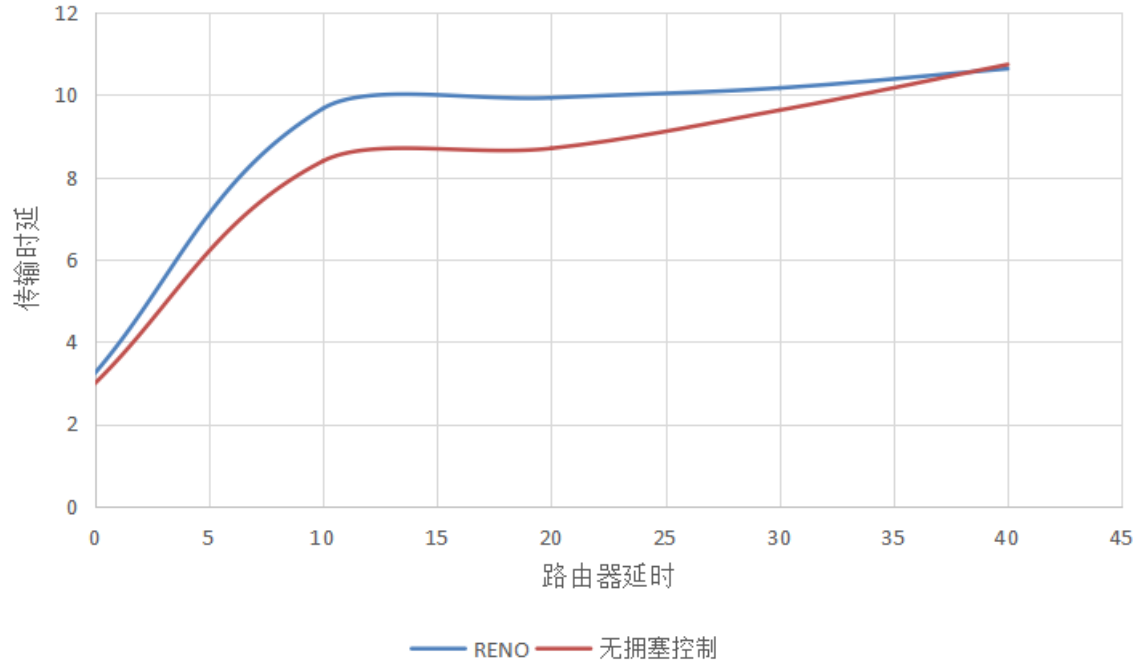
Miss Rate	1%	2%	3%	4%	10%
Latency/ms	0	0	0	0	0
RENO拥塞控制	3.226	10.074	10.933	12.065	13.171
无拥塞控制	2.978	8.386	8.698	9.624	10.734

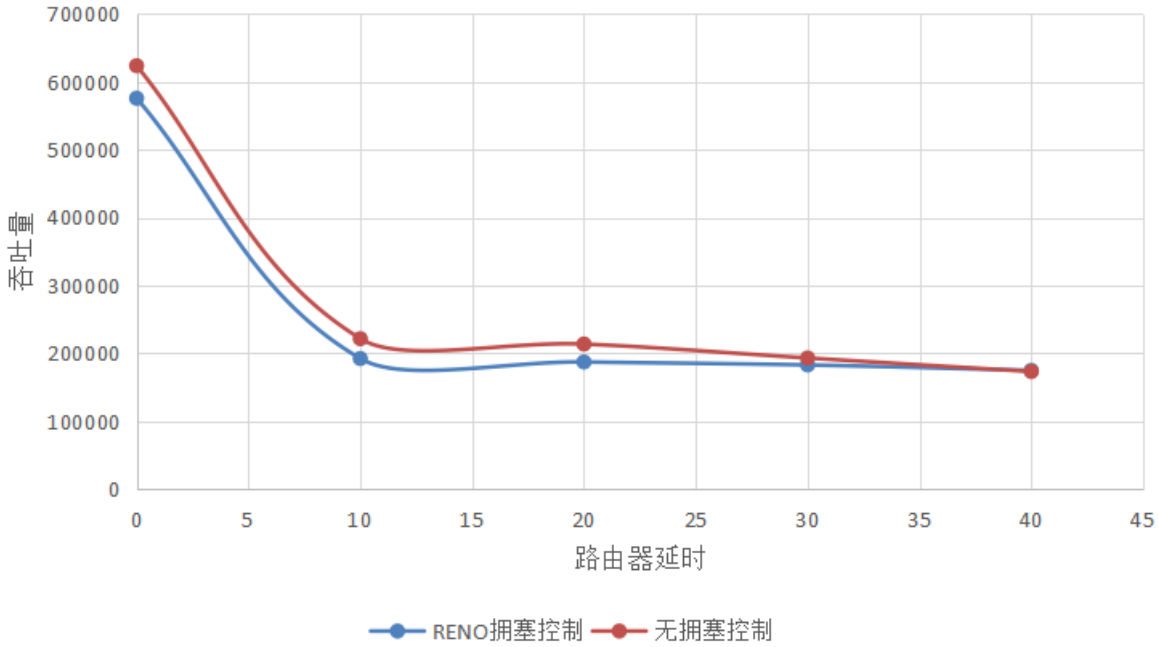




从上图可以发现，在网络状况比较好（路由器延迟为0ms）的条件下，RENO拥塞控制算法下的程序性能不如无拥塞控制，因为RENO算法造成额外的时间消耗且随着丢包率增大窗口变化频率也增大，性能下降。

Miss Rate	1%	1%	1%	1%	1%
Latency/ms	0	10	20	30	40
RENO拥塞控制	3.226	9.664	9.927	10.165	10.633
无拥塞控制	2.978	8.386	8.698	9.624	10.734





但是通过变化路由器延时可以发现，当网络状态变差，RENO拥塞控制算法下的程序运行性能随路由器延时增加下降得并不明显，总体曲线比较平稳。推测主要原因是快速重传机制的引入减少等待超时重传，且RENO算法顺利将滑动窗口设置为适合当前网络状况的大小。