计算机网络 3-4 实验报告

实验要求

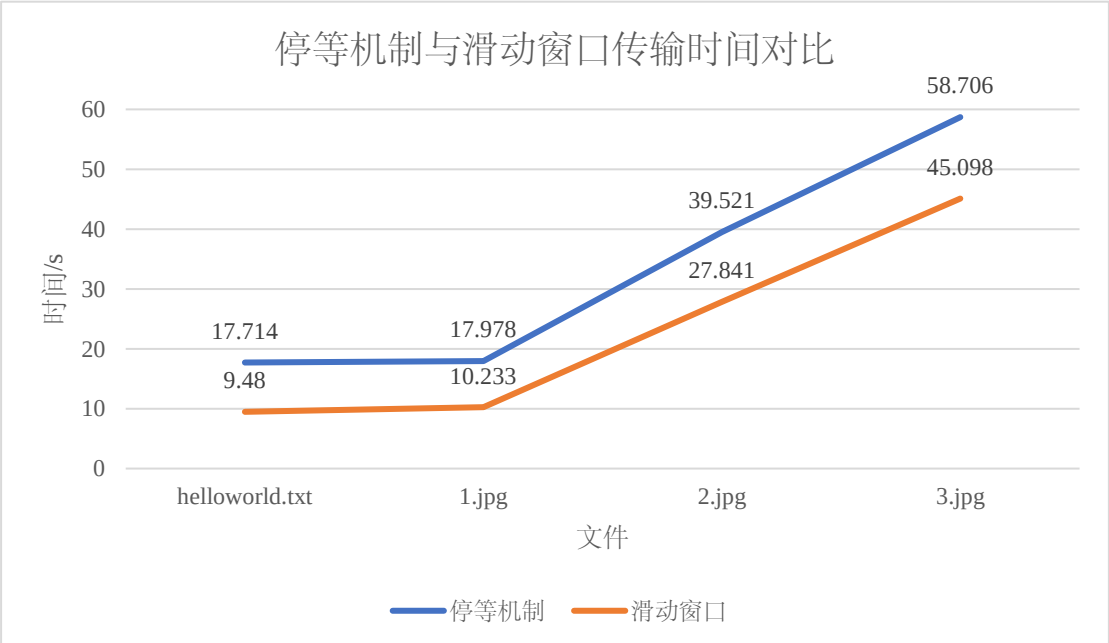
基于给定的实验测试环境，通过改变延迟时间和丢包率，完成下面 3 组性能对比实验：（1）停等机制与滑动窗口机制性能对比；（2）滑动窗口机制中不同窗口大小对性能的影响；（3）有拥塞控制和无拥塞控制的性能比较。

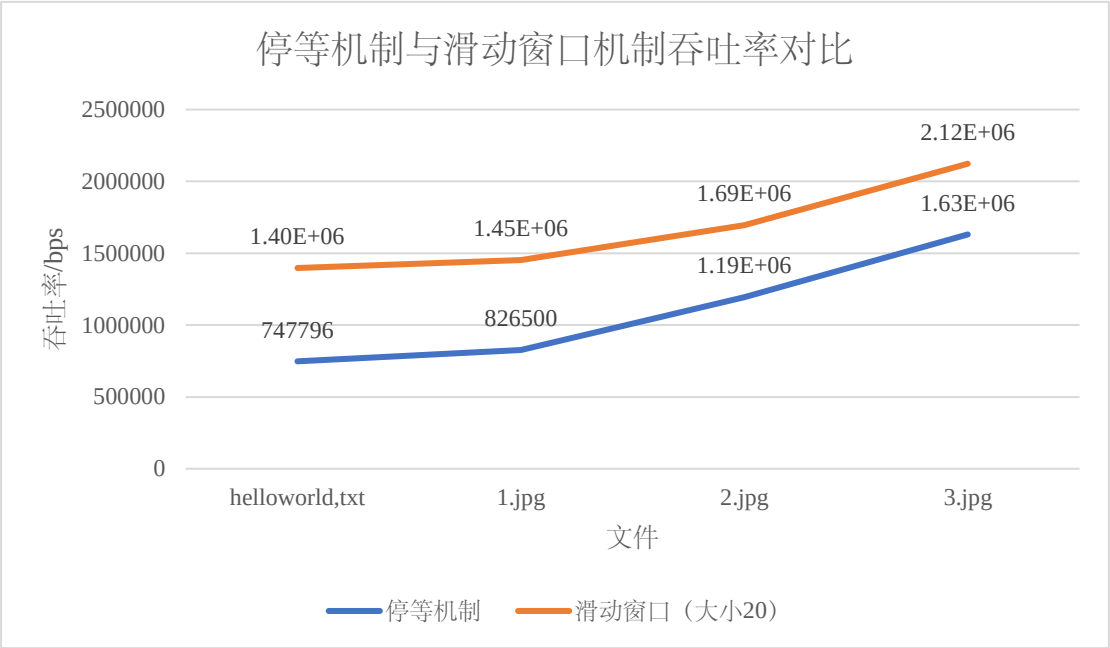
完成给定测试文件的传输，显示传输时间和平均吞吐率。

性能测试指标：吞吐率、时延，给出图形结果并进行分析。

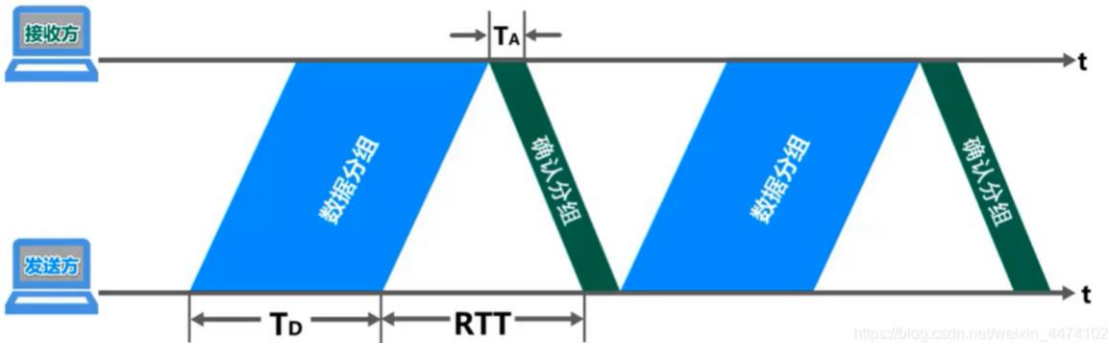
停等机制与滑动窗口机制性能对比

	Helloworld.txt		1.jpg		2.jpg		3.jpg	
	吞吐率/bps	时间/s	吞吐率/bps	时间/s	吞吐率/bps	时间/s	吞吐率/bps	时间/s
停等机制	747796	17.714	826500	17.978	1.194e+06	39.521	1.63104e+06	58.706
滑动窗口	1.39731e+06	9.48	1.45205e+06	10.233	1.69491e+06	27.841	2.1232e+06	45.098

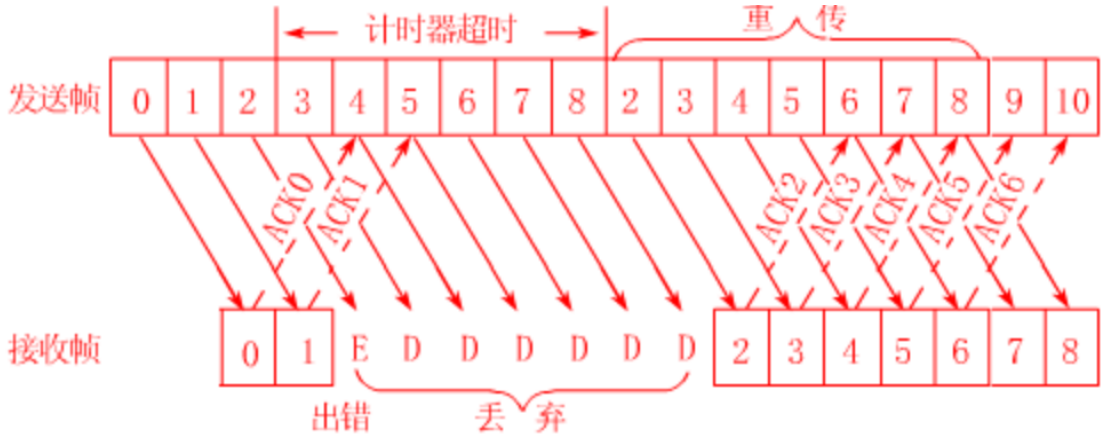




由于停等协议数据包发送时一次只能发送一个，之后必须等待下一个数据包确认后才能继续发送下一个，信道利用率（发送数据所需要的时间/整个发送周期时间）低，因此性能较差。

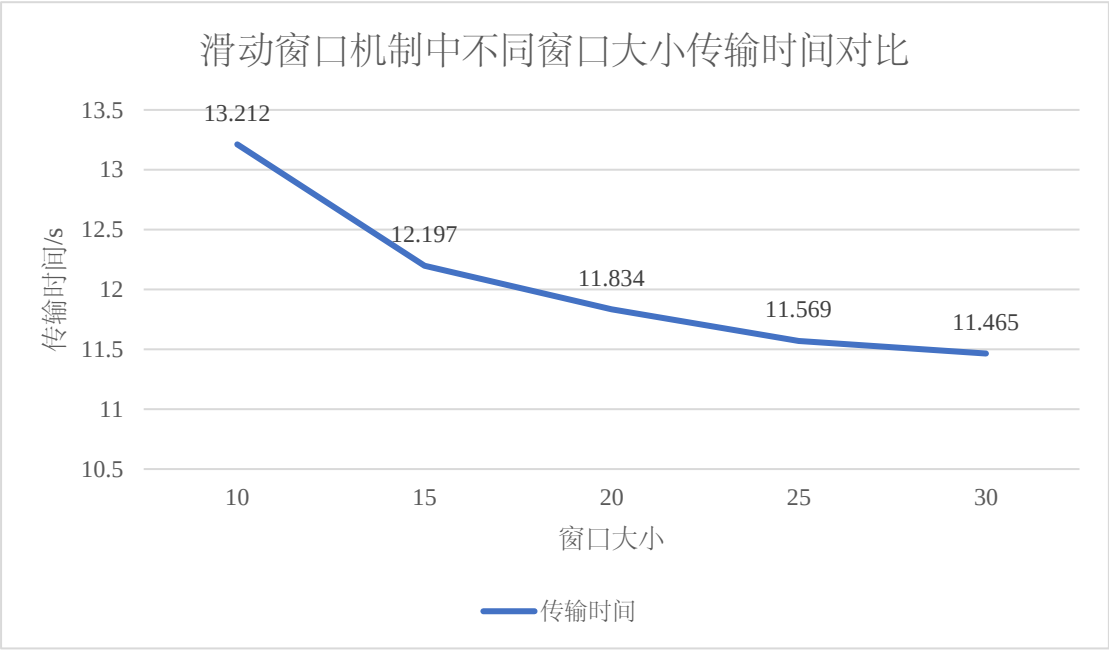


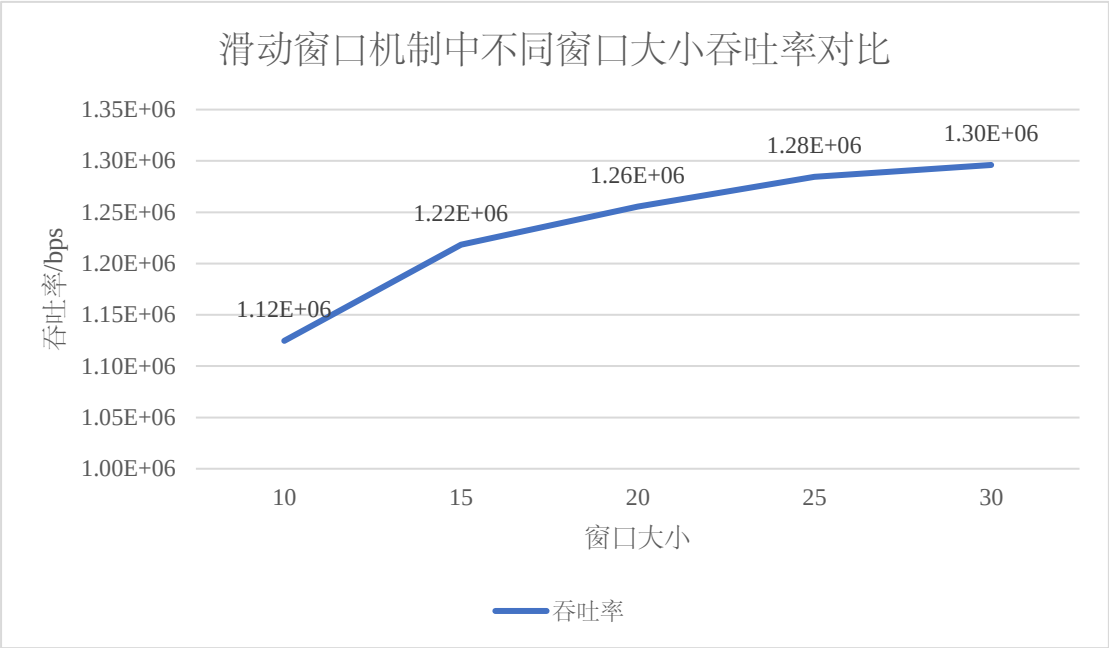
GBN 协议一方面因连续发送数据帧而提高了效率，但另一方面，在重传时又必须把原来已正确传送过的数据帧进行重传（仅因这些数据帧之前有一个数据帧出了错），这种做法又使传送效率降低。由此可见，若传输信道的传输质量很差因而误码率较大时，连续测协议不一定优于停止等待协议。



滑动窗口机制中不同窗口大小对性能的影响

窗口大小	吞吐率/bps	时间/s
10	1.12465e+06	13.212
15	1.21824e+06	12.197
20	1.2556e+06	11.834
25	1.28437e+06	11.569
30	1.29602e+06	11.465



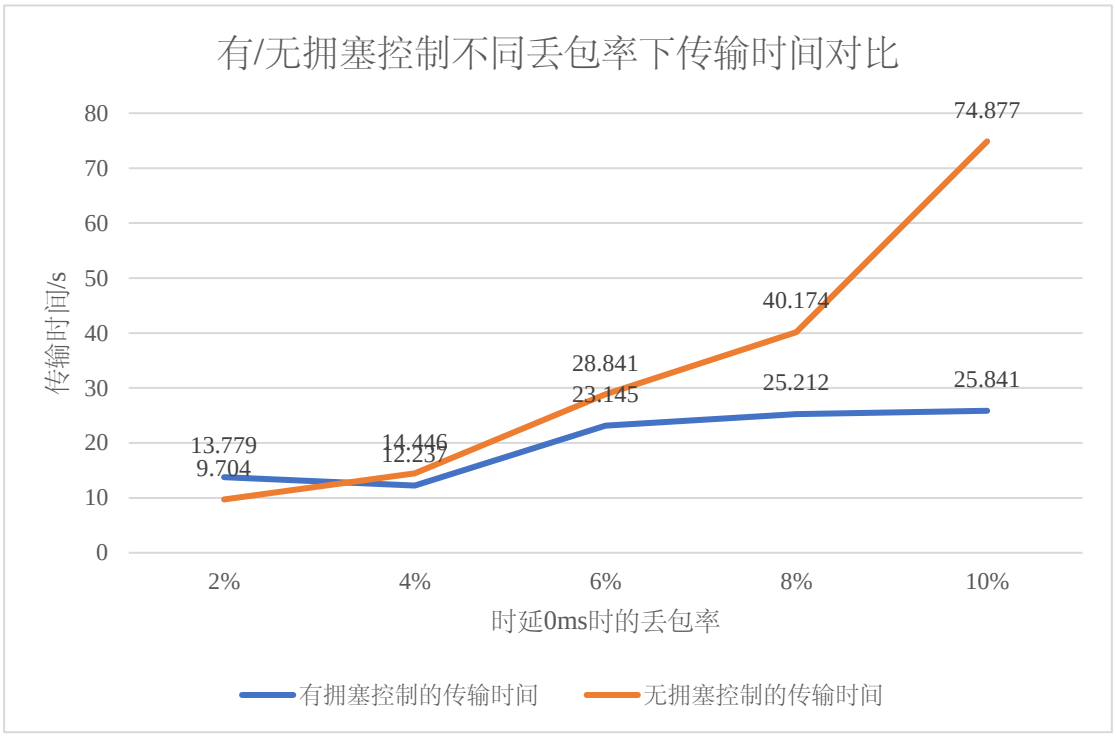


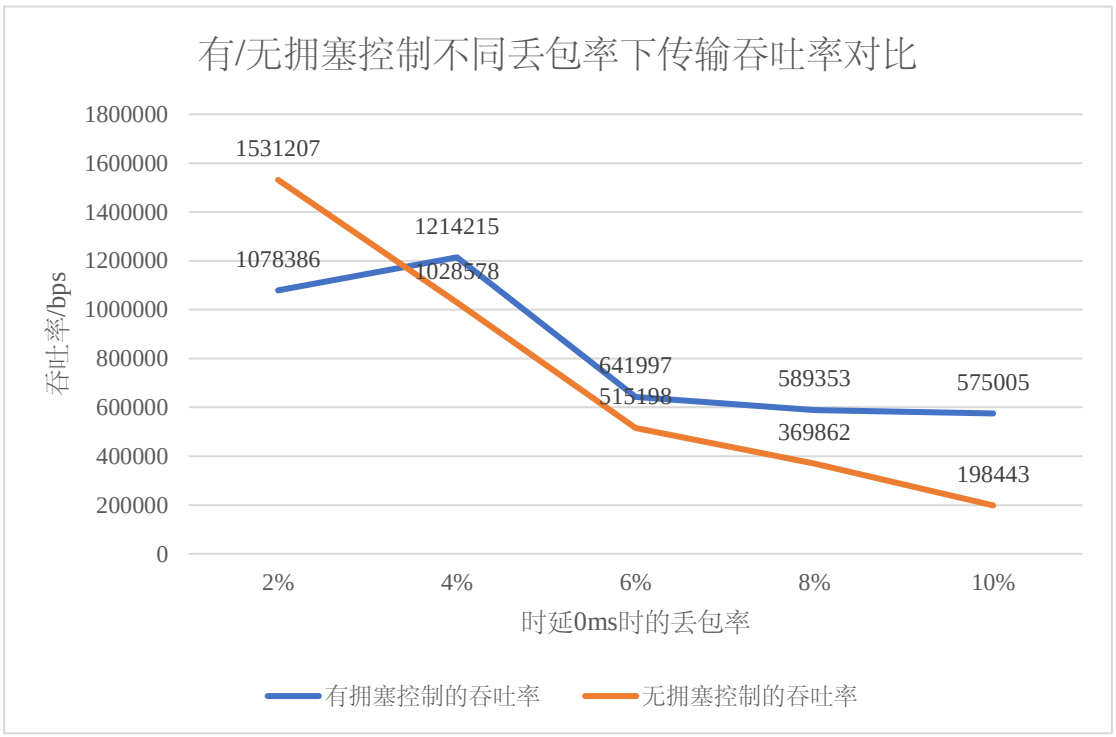
在滑动窗口增大初期，由于窗口加大，一次性可以传送的数据包增多，对信道的利用率增大，因此在一开始吞吐率是在不断增大的；但随着窗口增大到一定程度时，信道利用率趋近饱和上限，吞吐率的增加就不大了。并且由于实现的 GBN 协议在出现丢包时需要重传窗口内所有包，其传输速率相比停止等待协议可能反而有所降低。

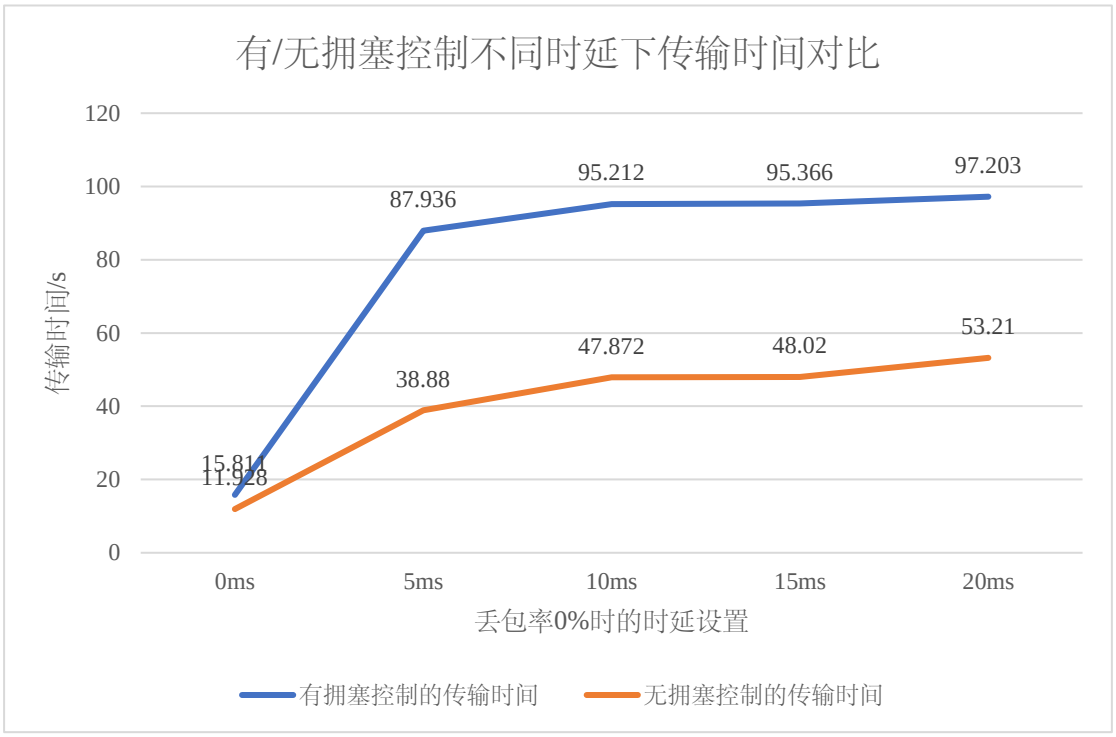
有拥塞控制和无拥塞控制的性能比较

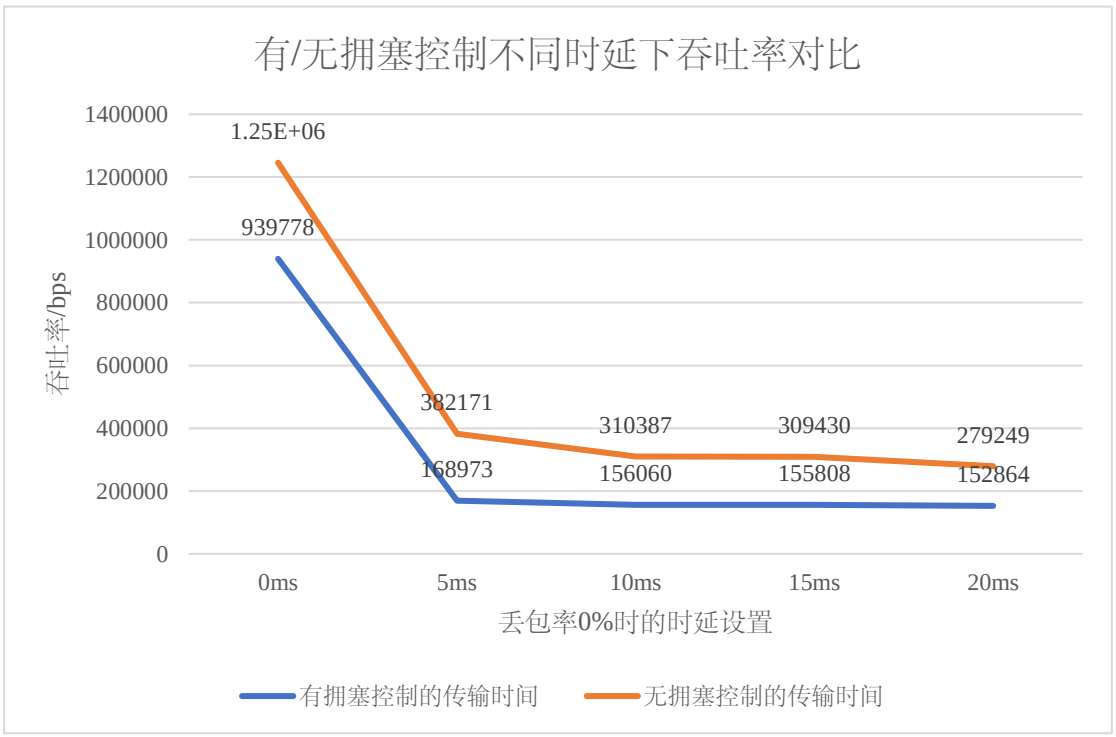
此处对比均使用测试文件 1.jpg；无拥塞控制时设置传输窗口大小恒定为 20，有拥塞控制时设置门限值 ssthresh=14

		有拥塞控制		无拥塞控制	
		吞吐率/bps	时间/s	吞吐率/bps	时间/s
时延 0ms	2%	1078386	13.779	1531207	9.704
	4%	1214215	12.237	1028578	14.446
	6%	641997	23.145	515198	28.841
	8%	589353	25.212	369862	40.174
	10%	575005	25.841	198443	74.877
丢包率 0%	0ms	939778	15.811	1.24571e+06	11.928
	5ms	168973	87.936	382171	38.88
	10ms	156060	95.212	310387	47.872
	15ms	155808	95.366	309430	48.02
	20ms	152864	97.203	279249	53.21


有/无拥塞控制不同丢包率下传输时间对比
传输时间/s
80
70
60
50
40
30
20
10
0
74.877
40.174
28.841
23.145
25.212
25.841
13.779
9.704
14.446
12.237
2%
4%
6%
8%
10%
时延0ms时的丢包率
有拥塞控制的传输时间
无拥塞控制的传输时间


有/无拥塞控制不同丢包率下传输吞吐率对比
吞吐率/bps
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
1531207
1214215
1078386
1028578
641997
589353
575005
515198
369862
198443
2%
4%
6%
8%
10%
时延0ms时的丢包率
有拥塞控制的吞吐率
无拥塞控制的吞吐率





对于 RENO 算法，为了维持一个动态平衡，再加上 AIMD 机制--减少快，增长慢，尤其是在大窗口环境下，由于一个数据报的丢失所带来的窗口缩小要花费很长的时间来恢复，这样，带宽利用率不可能很高且随着网络的链路带宽不断提升，这种弊端将越来越明显。

当丢包率不断上升时，RENO 算法导致其窗口一直被限定在较小范围内，虽然对信道的利用率降低，但是减少了一个窗口反复重传的情况，因此在丢包率较低时 RENO 算法的吞吐率低于无拥塞控制控制情况，而后高于。

当时延不断增高时，RENO 算法中进入快速恢复阶段的可能性不断增高，导致窗口大小受到限制，因此吞吐率低于无拥塞控制时。