南開大學
Nankai University

计算机网络实验三

基于**UDP**服务设计可靠传输协议并编程实现

实验3-4：性能对比实验

姓名：刘伟

学号：**2013029**

专业：物联网工程

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：

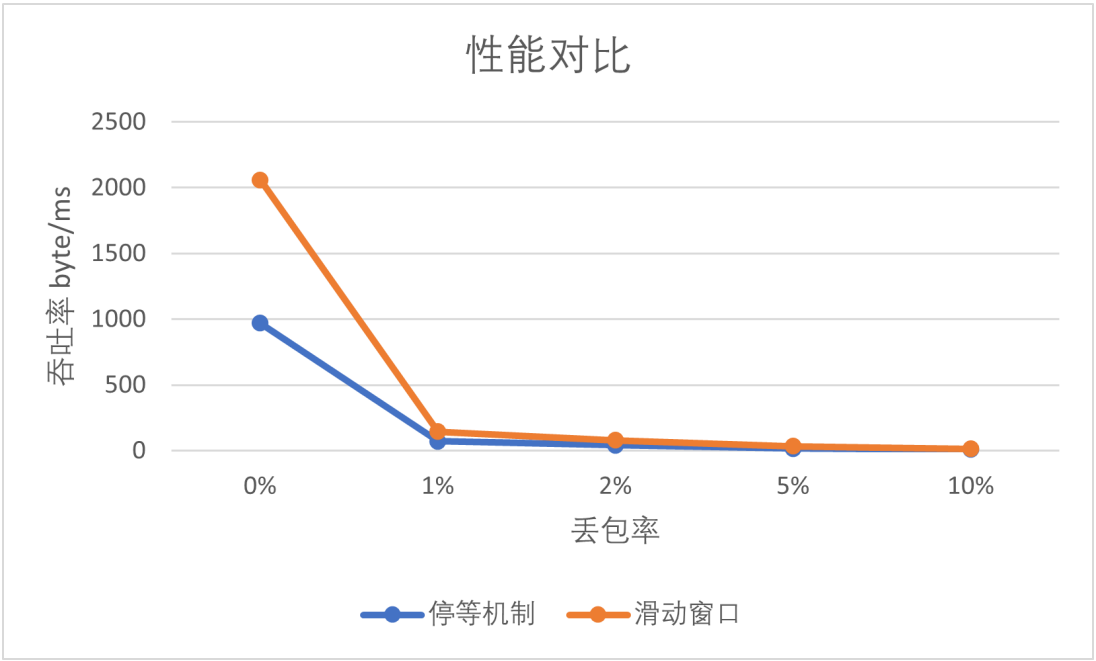
> 注意：性能测试中选用的是1.jpg与helloworld.txt做测试。测试结果均是通过多次测量取均值求得

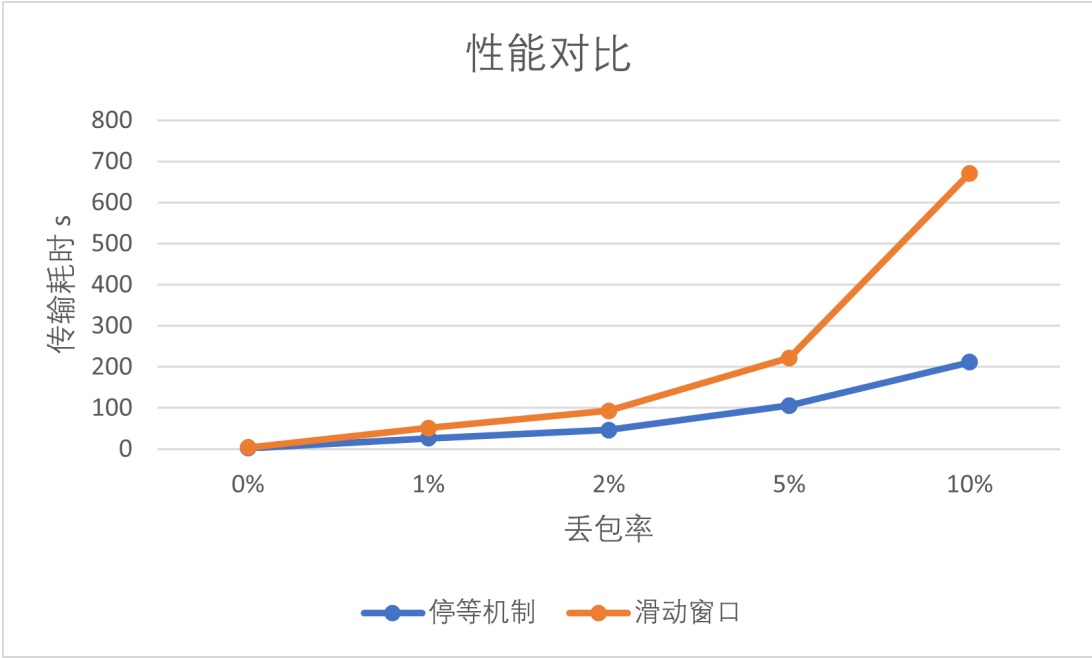
（**1**）停等机制与滑动窗口机制性能对比：（窗口大小选择**window = 6**、超时重传设定为**>0.5s**）

1.jpg 1,857,353字节

延时为**0** 更改丢包率

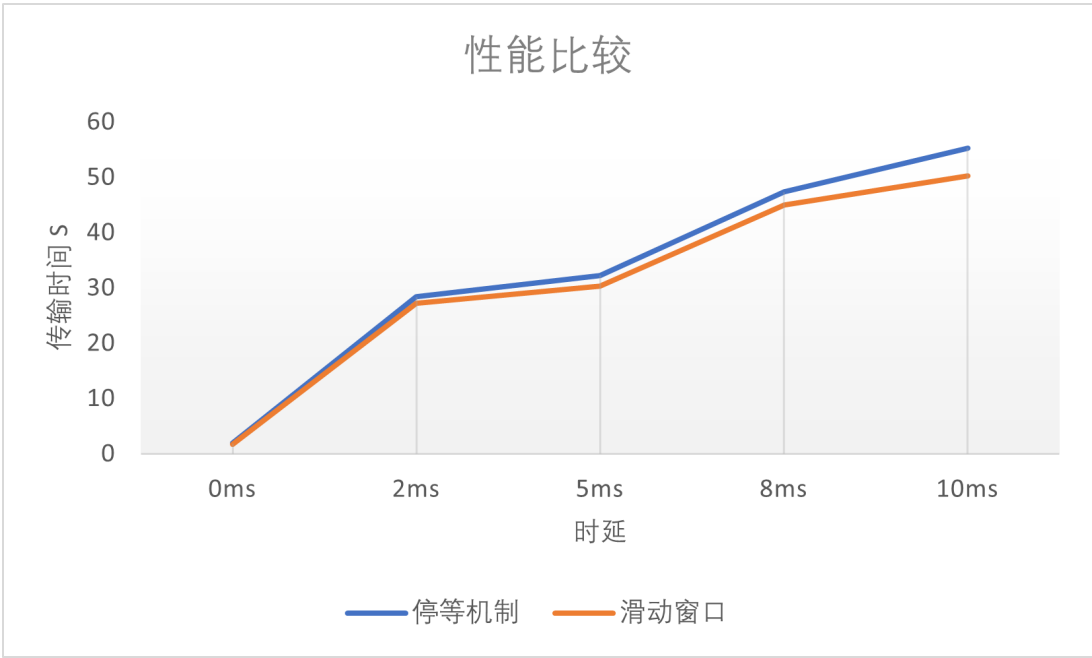
机制\丢包率	**0%**	**1%**	**2%**	**5%**	**10%**
停等机制-传输时间s	1.914	26.386	46.153	105.826	211.202
停等机制-吞吐率byte/ms	970.4038	70.3916	40.2434	17.551	8.7942
滑动窗口-传输时间s	1.704	25.211	47.028	115.73	460.2
滑动窗口-吞吐率byte/ms	1090	73.6723	39.4946	16.049	4.0359

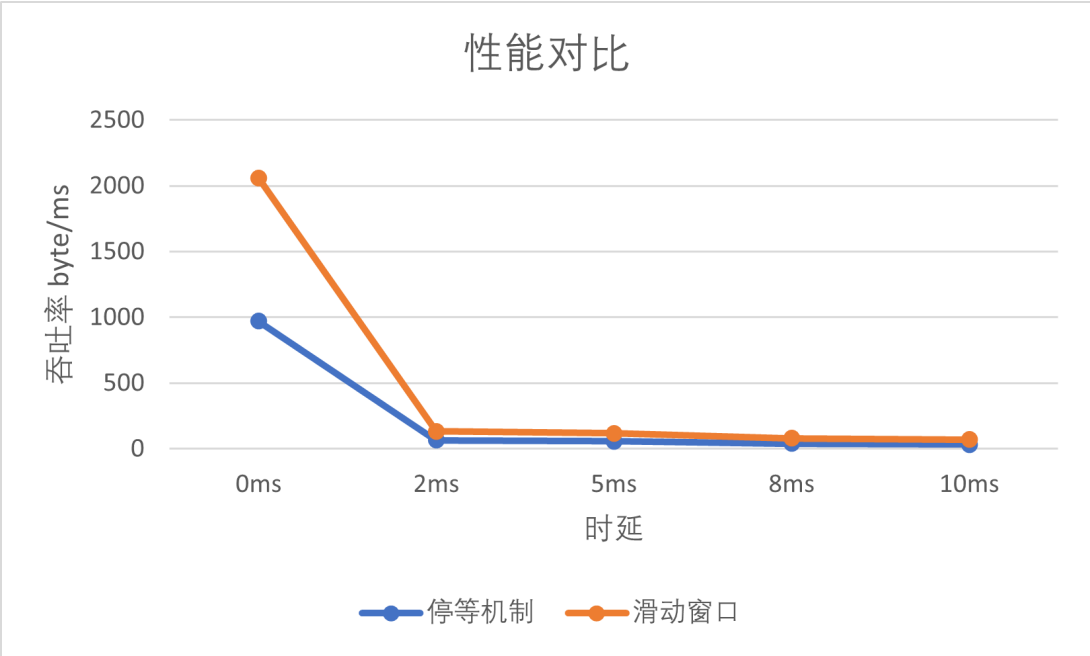




丢包率为**0** 更改延时

机制\延时	**0ms**	**2ms**	**5ms**	**8ms**	**10ms**
停等机制-传输时间s	1.914	28.374	32.245	47.349	55.249
停等机制-吞吐率byte/ms	970.4038	65.4597	57.6013	39.2268	33.6179
滑动窗口-传输时间s	1.704	27.187	30.356	44.930	50.238
滑动窗口-吞吐率byte/ms	1090	68.3176	61.1856	41.3388	36.9711





helloworld.txt 1,655,808字节

延时为**0** 更改丢包率

机制\丢包率	**0%**	**1%**	**2%**	**5%**	**10%**
停等机制-传输时间s	1.709	25.461	44.937	97.078	192.382
停等机制-吞吐率byte/ms	968.8753	65.0331	36.8473	17.0565	8.60688
滑动窗口-传输时间s	1.548	23.897	47.342	125.437	416.35
滑动窗口-吞吐率byte/ms	1069.64	69.2893	34.9754	13.2003	3.9768

3丢包率为**0** 更改延时

机制\延时	**0ms**	**2ms**	**5ms**	**8ms**	**10ms**
停等机制-传输时间s	1.709	25.391	29.886	46.427	49.261
停等机制-吞吐率byte/ms	968.8753	65.6124	55.4041	35.6648	33.613
滑动窗口-传输时间s	1.548	24.831	28.992	45.180	47.884
滑动窗口-吞吐率byte/ms	1069.64	66.683	57.1125	36.6491	34.5795

helloworld.txt的曲线图同1.jpg在曲线走势上是一样的，不作过多展示。从数据上同样可以清晰看出变化与比较。

> 停等机制与滑动窗口的比较总结：
>
> - 二者的传输性能均严重的依赖丢包率。随丢包率提高而显著降低。
> - 在没有丢包或丢包率较低的情况下：滑动窗口的性能优于停等机制。

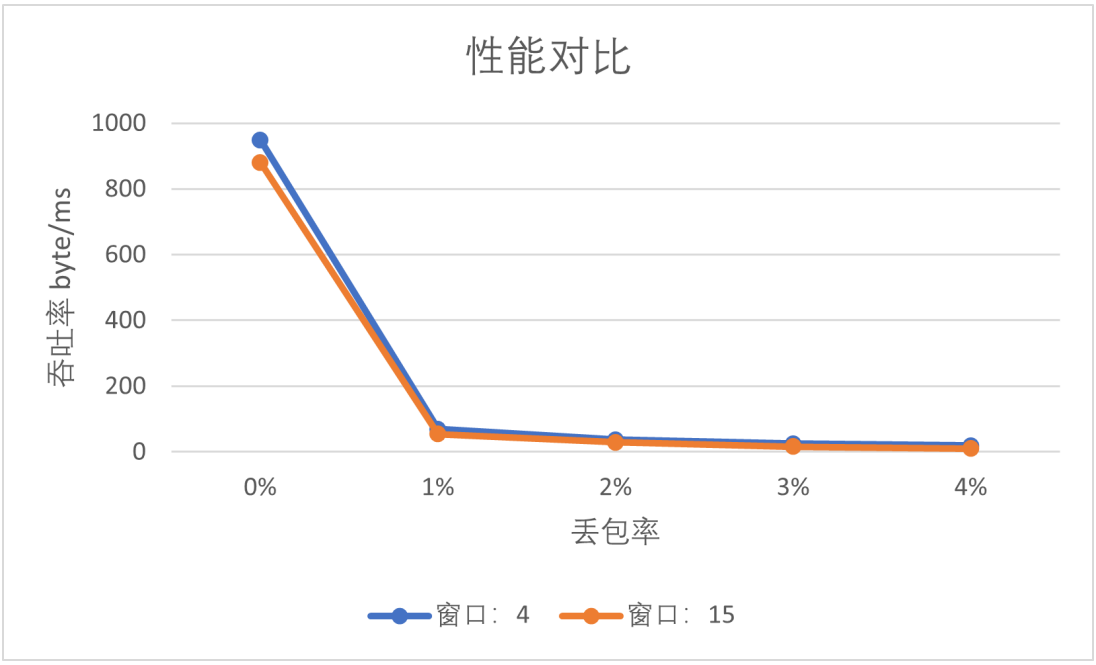
    - 滑动窗口可以允许发送多条，同时等待ACK，减少了RTT在总时间上的影响。
- 丢包率较大的情况下：滑动窗口性能低于停等机制。
    - 滑动窗口GBN在丢包情况下的数据重传严重、重传耗时代价过高、网络大量数据报拥塞。
- 延时对于二者的影响改变的没有丢包率提高那么严重，但是延迟确实所有数据的延迟降低。
- 延时状况下：滑动窗口的性能更优一些。
    - 停等机制需要每条消息单独等待时延和RTT，而窗口可以同时等待多条。

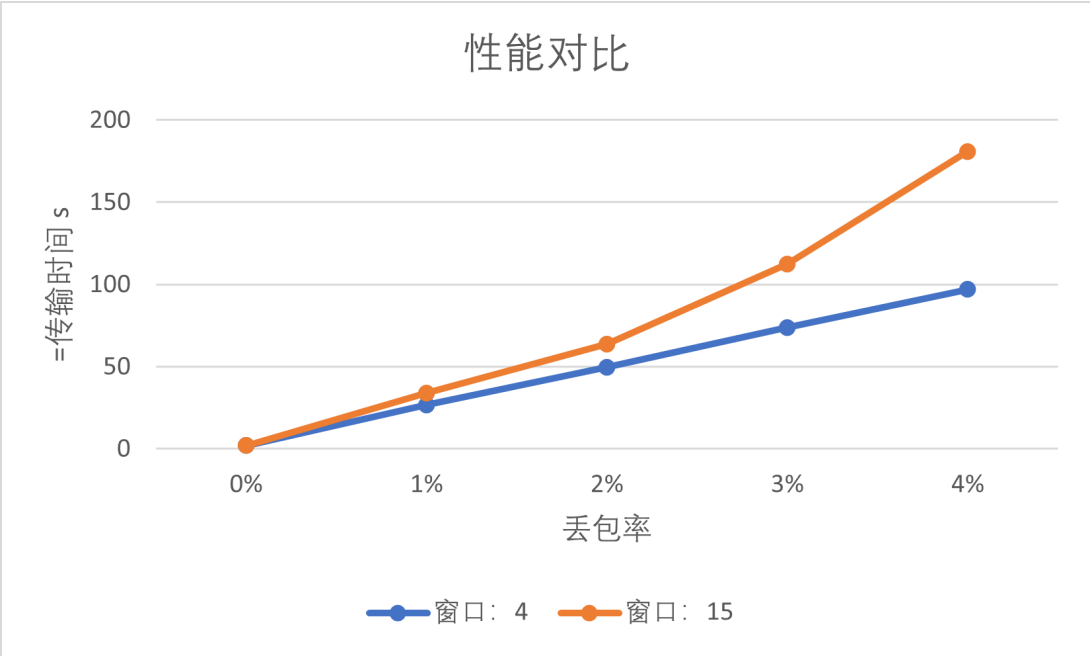
（2）滑动窗口机制中不同窗口大小对性能的影响：（多线程、窗口选择**4、15、35**）（超时重传设定为**>1s**）

1.jpg 1,857,353字节

延时为**0** 更改丢包率

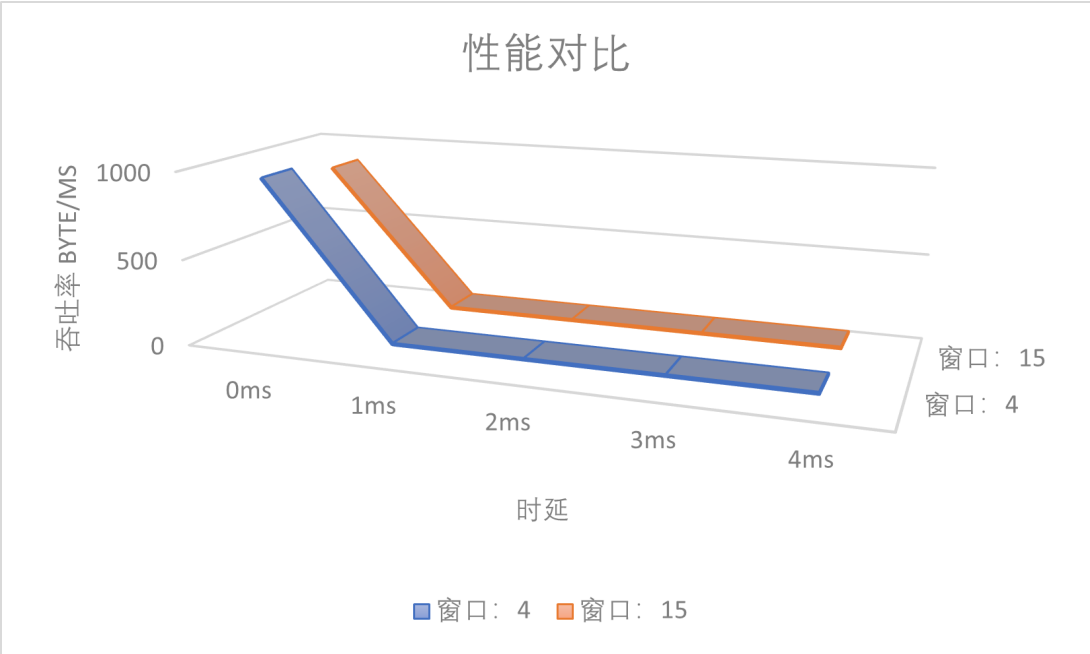
机制\丢包率	**0%**	**1%**	**2%**	**3%**	**4%**
窗口：4-传输时间s	1.957	26.814	49.666	73.968	97.018
窗口：4-吞吐率byte/ms	949.082	69.268	37.3969	25.1102	19.1444
窗口：15-传输时间s	2.109	34.065	63.788	112.557	180.971
窗口：15-吞吐率byte/ms	880.679	54.5238	29.1176	16.5014	10.2633

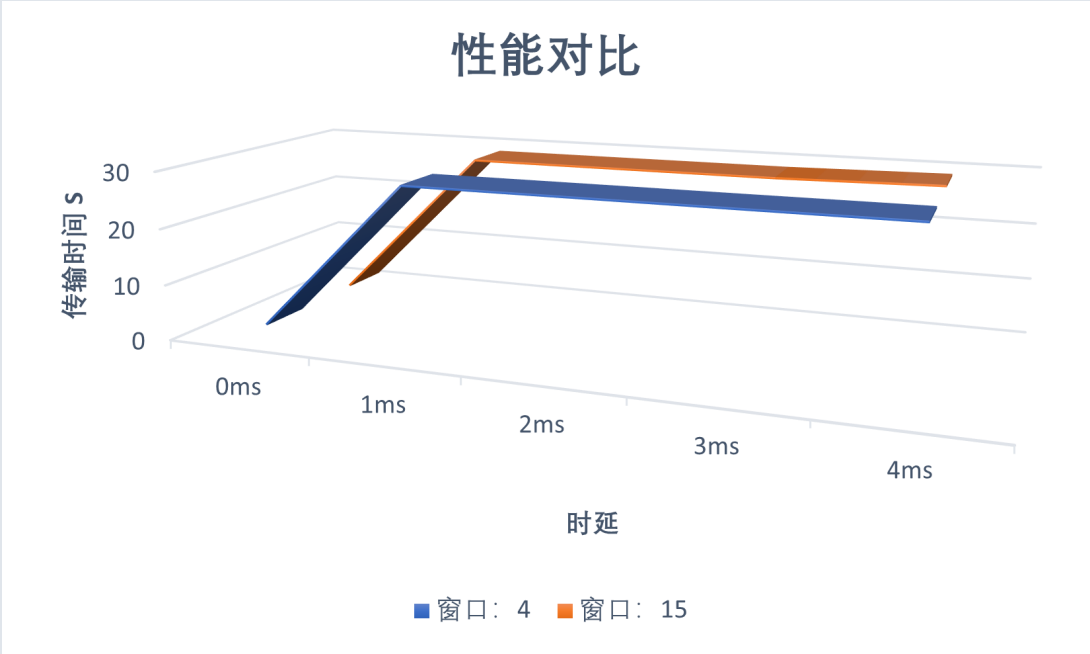




丢包率为**0** 更改延时

机制\延时	**0ms**	**1ms**	**2ms**	**3ms**	**4ms**
窗口：4-传输时间s	1.957	28.767	28.787	28.793	28.826
窗口：4-吞吐率byte/ms	949.082	64.5654	64.5205	64.5071	64.4333
窗口：15-传输时间s	2.109	28.747	28.762	28.787	29.177
窗口：15-吞吐率byte/ms	880.679	64.6103	64.5766	64.5205	63.6581





不同的滑动窗口大小性能对比总结：

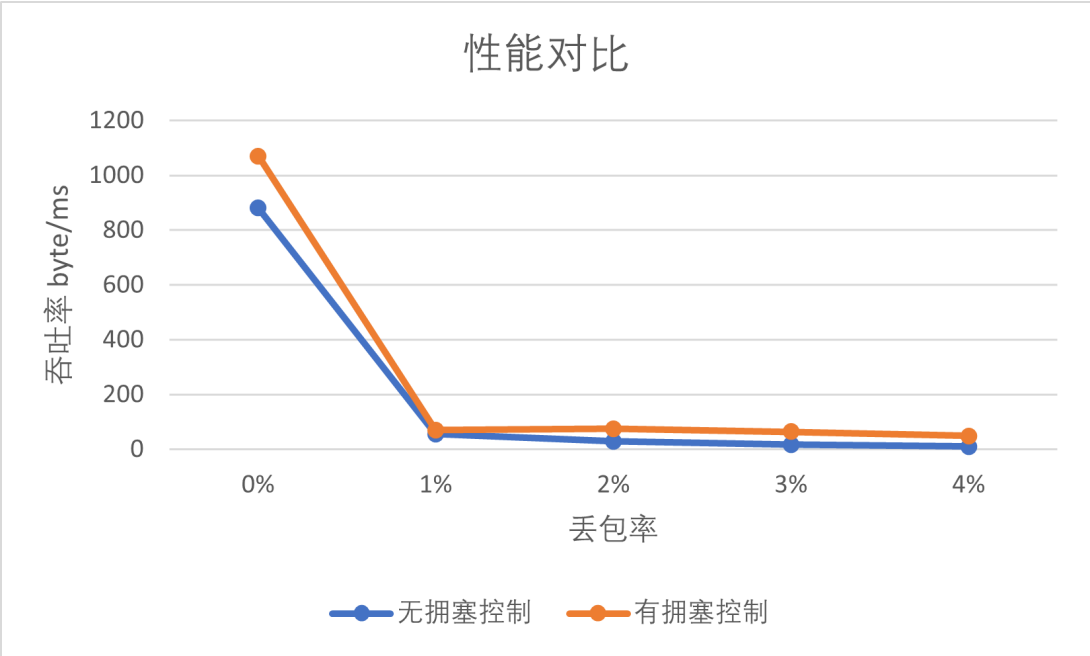
- 不同的滑动窗口大小在丢包率和时延的改变下，性能较低变化整体一致
- 在丢包率改变增大的情况下：
  - 滑动窗口大小越大的性能下降越快、降低更多。
    - 大的窗口带来的重传代价更大
  - 在丢包率为0或者较低的时候，大窗口的性能、效率高一些
- 在时延的改变情况下：滑动窗口的大小区别在这里影响区别不大，保持着同样的速率降低趋势

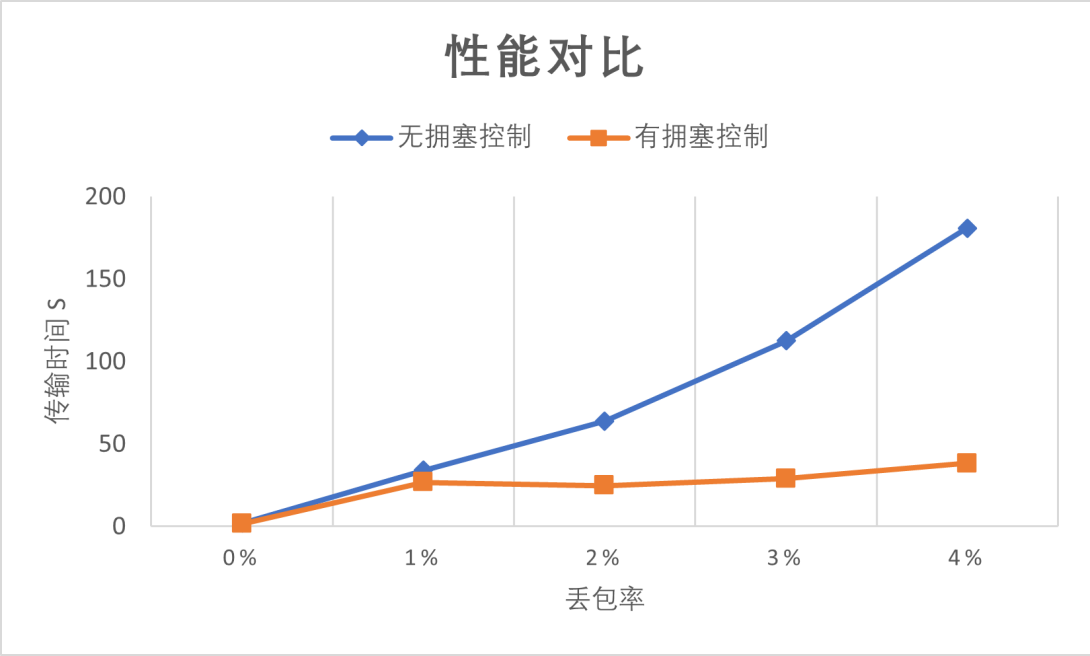
（**3**）有拥塞控制和无拥塞控制的性能比较：（**Reno**拥塞控制：初始阈值**40**、无拥塞**GBN**：窗口**15**）

1.jpg 1,857,353字节

延时为**0** 更改丢包率

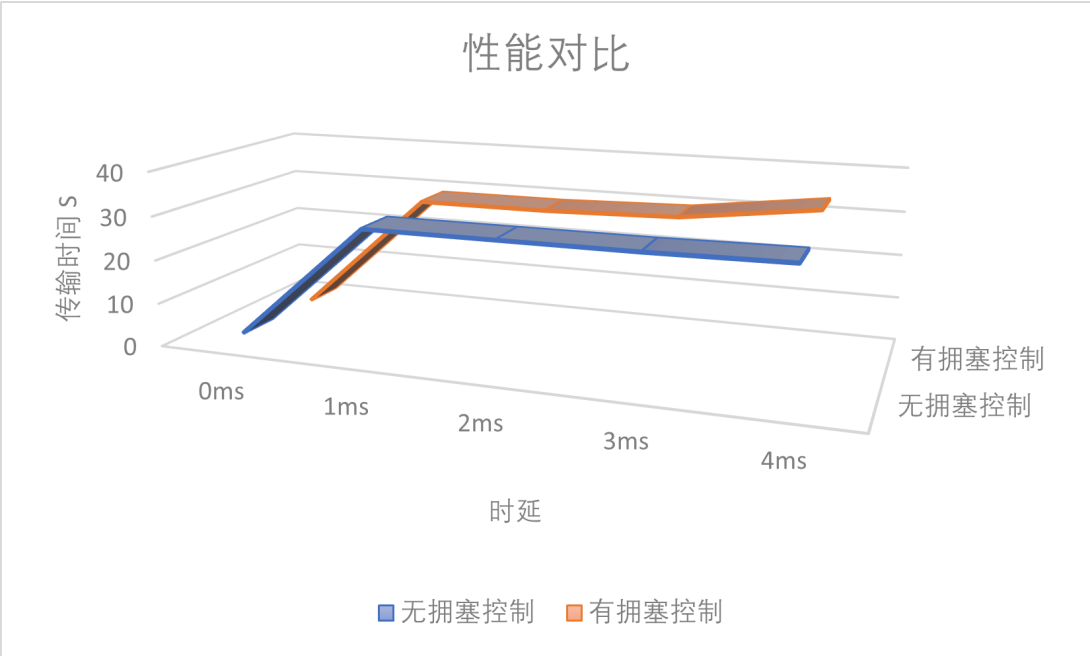
机制\丢包率	**0%**	**1%**	**2%**	**3%**	**4%**
无拥塞控制-传输时间s	2.109	34.065	63.788	112.557	180.971
无拥塞控制-吞吐率byte/ms	880.679	54.5238	29.1176	16.5014	10.2633
有拥塞控制-传输时间s	1.735	26.843	24.844	28.988	38.425
有拥塞控制-吞吐率byte/ms	1070.520	69.1932	74.7606	64.0732	48.3371

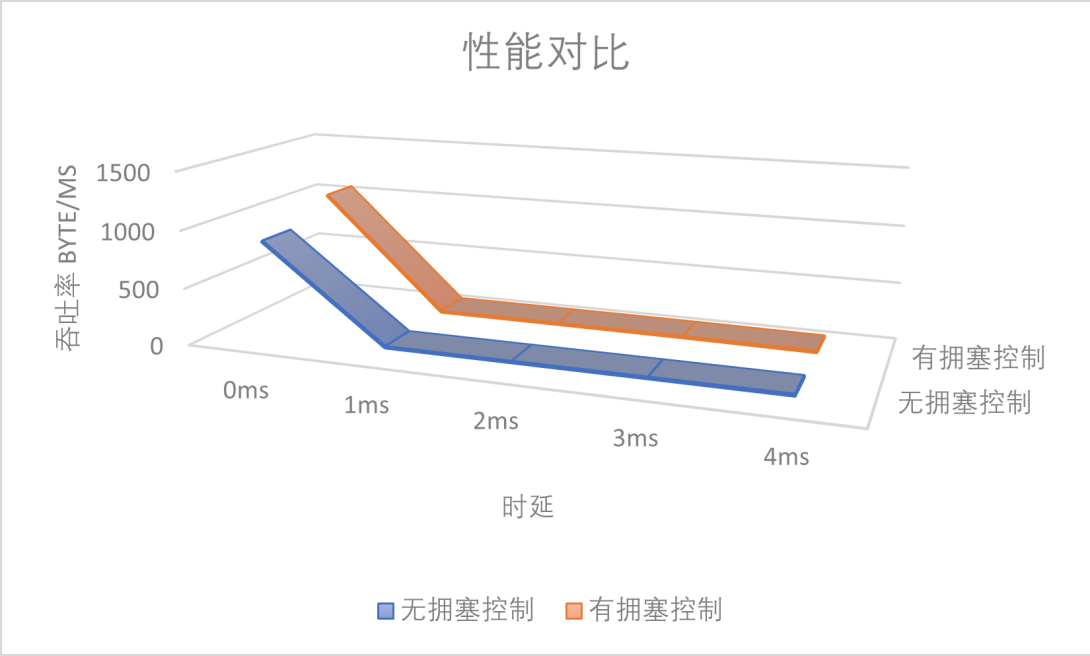




丢包率为**0** 更改延时

机制\延时	**0ms**	**1ms**	**2ms**	**3ms**	**4ms**
无拥塞控制-传输时间s	2.109	28.747	28.762	28.787	29.177
无拥塞控制-吞吐率byte/ms	880.679	64.6103	64.5766	64.5205	63.6581
有拥塞控制-传输时间s	1.735	28.908	28.957	29.719	33.533
有拥塞控制-吞吐率byte/ms	1070.520	64.2505	64.1418	62.4972	55.3888





有拥塞控制和无拥塞控制的性能对比：

- 这里二者最为明显的对比即是二者在不同的丢包率下的性能对比：
  - 在丢包率增大的影响下，无拥塞控制的性能严重下降，但是有拥塞控制其性能下降的速率慢很多
    - 这是由于拥塞控制可以在网络大量丢包时，缩小窗口大小以避免大量的数据重传，减少重传代价。
  - 在延迟的变化影响下：窗口大小变化、有无拥塞控制对于性能的影响力度整体上是一致的。

- 注意：作者测试时延时，设定的时延较低、明显低于超时重传的时间限定。如果增大时延至引发丢包的情况下，Reno拥塞控制的性能会更优于无拥塞控制。因为算法会降低窗口大小，以减少窗口带来的重传代价。