计算机网络 3-4

个人信息

- 学号：2013750
- 姓名：管昀玫
- 专业：计算机科学与技术

实验要求

实验3-4：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。

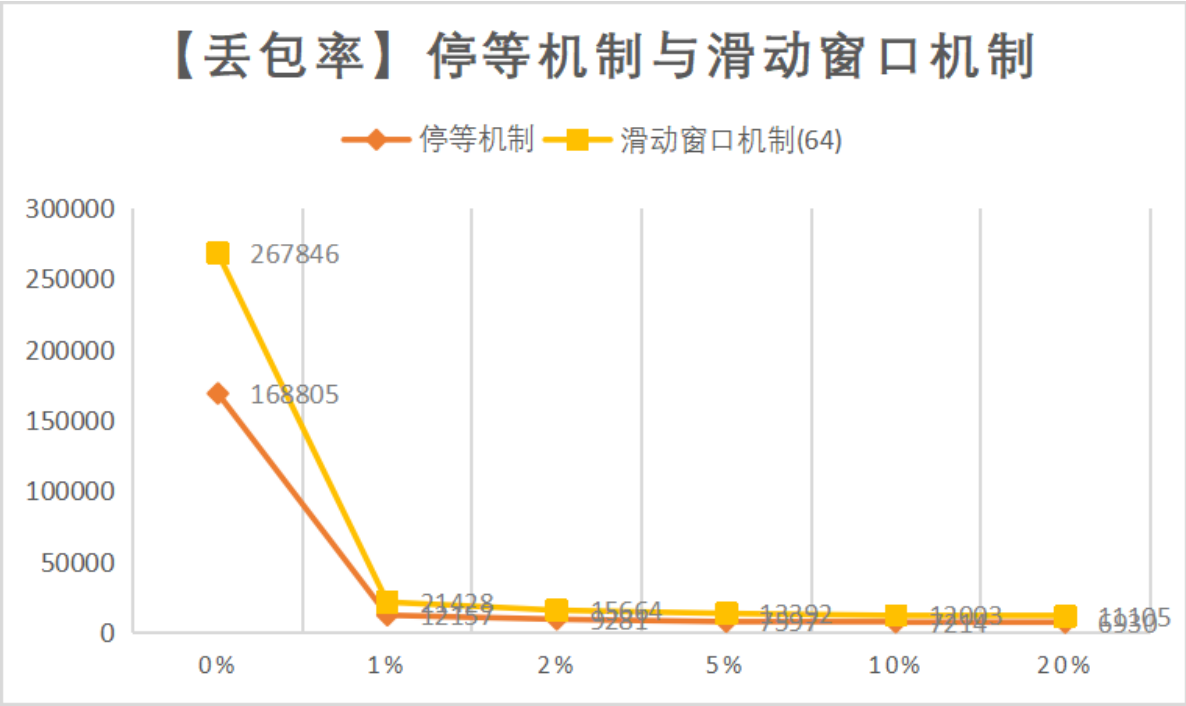
实验环境

- Windows x86
- 使用规定路由器
- 使用规定文件
    - 1.jpg: 1.77 MB
    - 2.jpg: 5.62 MB
    - 3.jpg: 11.4 MB
    - helloworld.txt: 1.57 MB
- 使用回环地址127.0.0.1
    - Client Port: 12880
    - Router Port: 8889
    - Server Port: 12660

1. 停等机制与滑动窗口机制性能对比

不同丢包率的性能对比

	0%	1%	2%	5%	10%	20%
停等机制	168805	12157	9281	7597	7214	6930
滑动窗口机制	267846	21428	15664	13392	12003	11105



停等机制与滑动窗口机制在各个丢包率下的传输吞吐率如图表所示(单位：kB/s)。

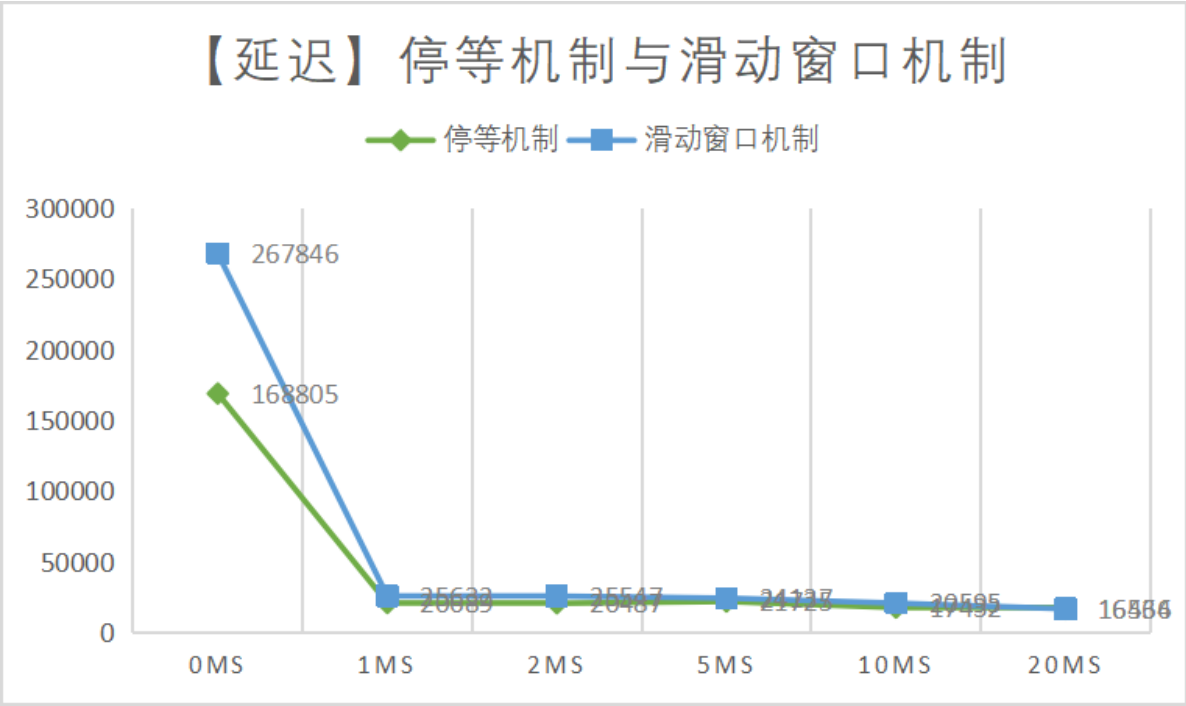
注：滑动窗口大小为$64 \times 1024 \times 1 = 65536B = 64KB$。

从图中可以看到，随着丢包率增大，传输吞吐率如指数般下降，之后趋于平缓下降。滑动窗口为$64KB$时，其传输吞吐率都略高于停等机制，但丢包率$> 1\%$之后与停等机制相差不大。这是因为编写滑动窗口机制时基于SR协议，但没有使用双线程实现发送数据包的同时接收数据包，导致其本质上还是停等机制。

从丢包率0%到丢包率1%之间巨大的吞吐率差距是由超时重传时间RTO造成的。本次实验中的最大等待时间`MAX_WAIT_TIME`设置为1000ms，而由于停等机制是严格按照单次收发判定，若前一个包未能成功送达将会造成超时重传，期间会阻塞下一个包的发送；多次重传将会造成巨大的性能损失，因此停等机制一旦开始丢包性能损失最为严重。

不同延迟的性能对比

	0ms	1ms	2ms	5ms	10ms	20ms
停等机制	168805	20689	20487	21725	17432	16436
滑动窗口机制	267846	25632	25547	24137	20595	16564



停等机制与滑动窗口机制在各个延迟下的传输吞吐率如图表所示(单位：kB/s)。

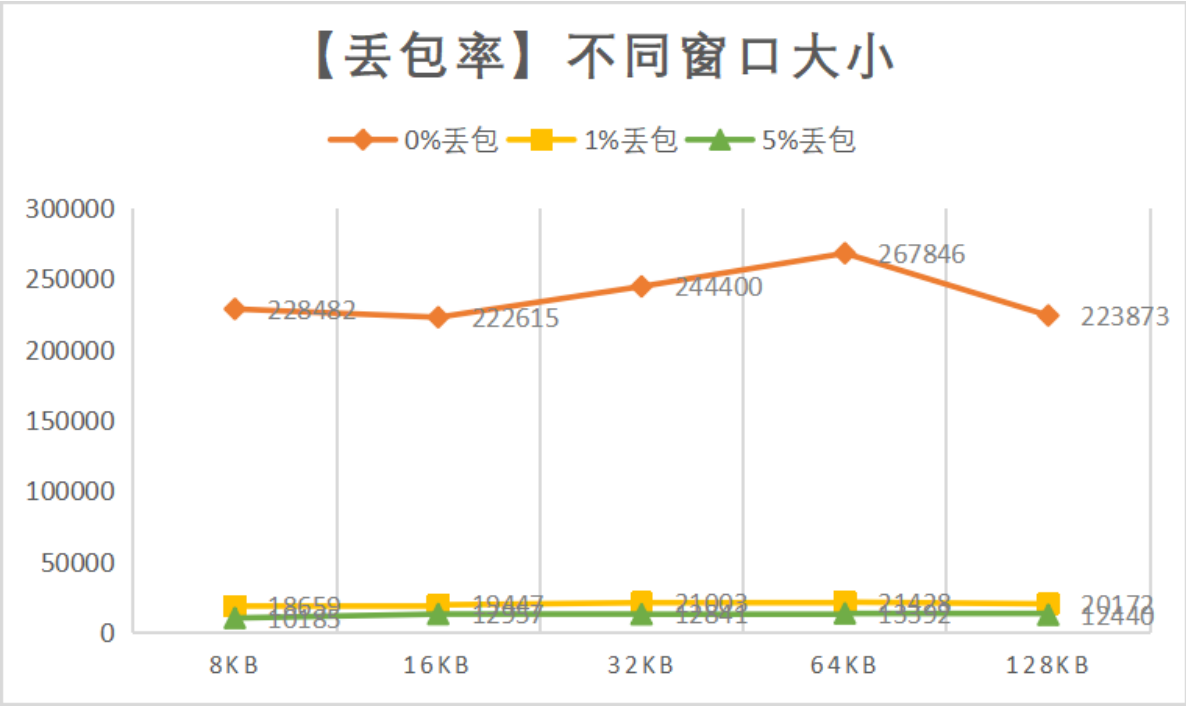
注：滑动窗口大小为$64 \times 1024 \times 1 = 65536B = 64KB$。

从图中可以看出，随着延迟的增大，传输吞吐率随之减小，而且从0ms到1ms传输吞吐率减小得最快，1ms ~ 5ms之间的传输吞吐率基本持平，10ms以上缓慢减小。

2. 滑动窗口机制中不同窗口大小对性能的影响

不同丢包率下不同窗口大小的性能对比

	8KB	16KB	32KB	64KB	128KB
0%丢包	228482	222615	244400	267846	223873
1%丢包	18659	19447	21003	21428	20172
5%丢包	10185	12957	12841	13392	12440



滑动窗口的不同窗口大小在不同丢包率下的传输吞吐率如图表所示(单位：kB/s)。

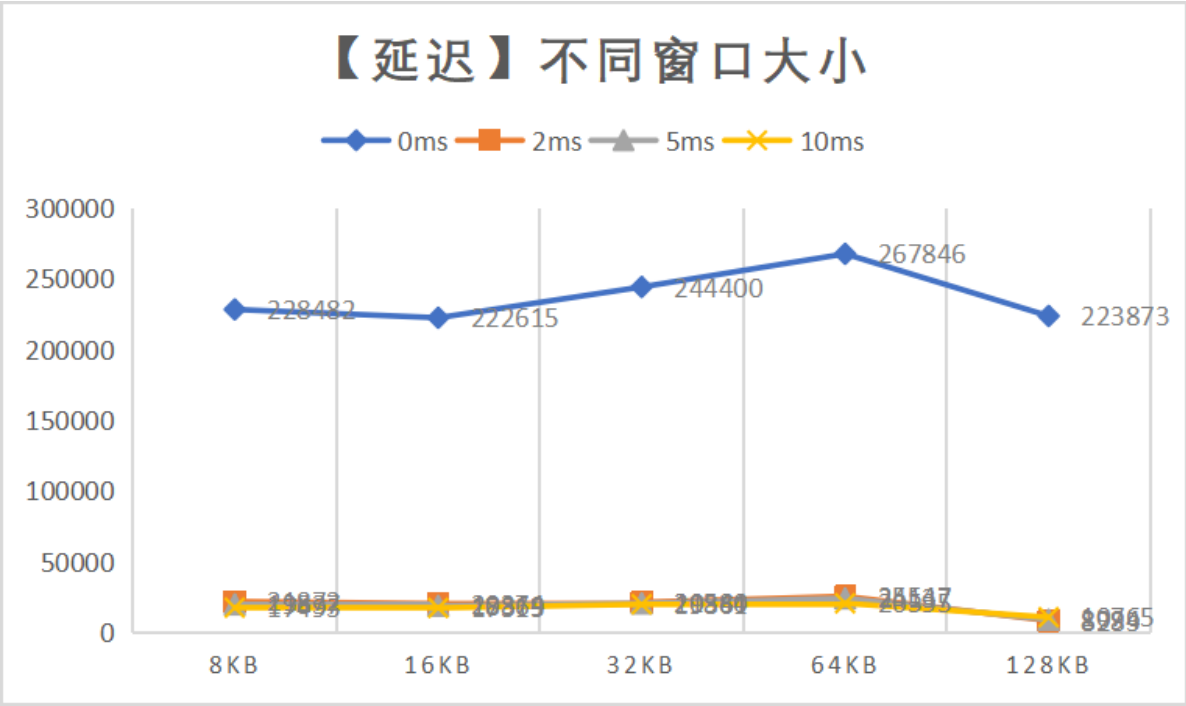
从图中可以看出一个基本趋势：随着滑动窗口增大，程序的吞吐率也将增大，在64KB时达到峰值。但在64KB之后吞吐率又将降落。而且，丢包从无到有会明显降低程序的性能，并随着丢包率的增加吞吐率逐渐降低。

其原因不难解释：滑动窗口增大后，一次性可以发送更多的数据包，减少了发送次数，因此增大吞吐率。但在达到峰值后不升反降，这是因为一旦需要重传，将会一次性重传更多的包，造成更大的性能损失。

而在丢包率为0%时窗口大小从64KB变为128KB时吞吐率减低是因为：这是由于程序设计引起的。在发包时，所有的包体都是事先打包好的，128个包体可以一起发出；但在接受时却需要按到达顺序接收、拆包、处理，并在128个数据包全部都得到确认后发送端才会发下一轮，这就造成了发送端在等待接收端的一个阻塞的局面，因此性能不升反降。考虑使用双线程来解决收发问题能改善性能。

不同延迟下的不同窗口大小性能对比

	8KB	16KB	32KB	64KB	128KB
0ms	228482	222615	244400	267846	223873
2ms	21872	20374	21340	25547	8283
5ms	19837	18869	20584	24137	8994
10ms	17493	17315	19861	20595	10765



滑动窗口的不同窗口大小在不同延迟条件下的传输吞吐率如图表所示(单位：kB/s)。

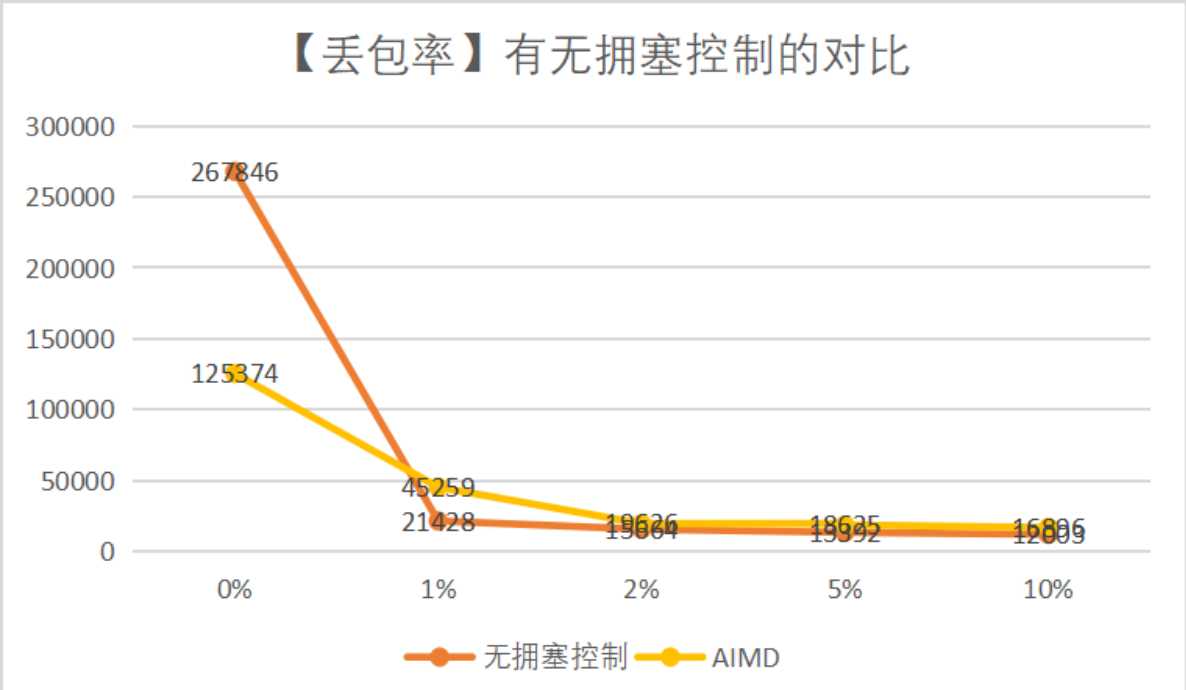
大体趋势与不同丢包率下的性能表现差不多：在64KB之前，随着滑动窗口的增大，吞吐率增大；在64KB之后吞吐率开始降低。且随着延迟的增大，性能降低。

从延迟和吞吐率两方面可以看出，窗口大小设置为64KB将拥有最优性能。

3. 有拥塞控制和无拥塞控制的性能比较

不同丢包率的性能对比

	0%	1%	2%	5%	10%
无拥塞控制	267846	21428	15664	13392	12003
AIMD	125374	45259	19626	18625	16896



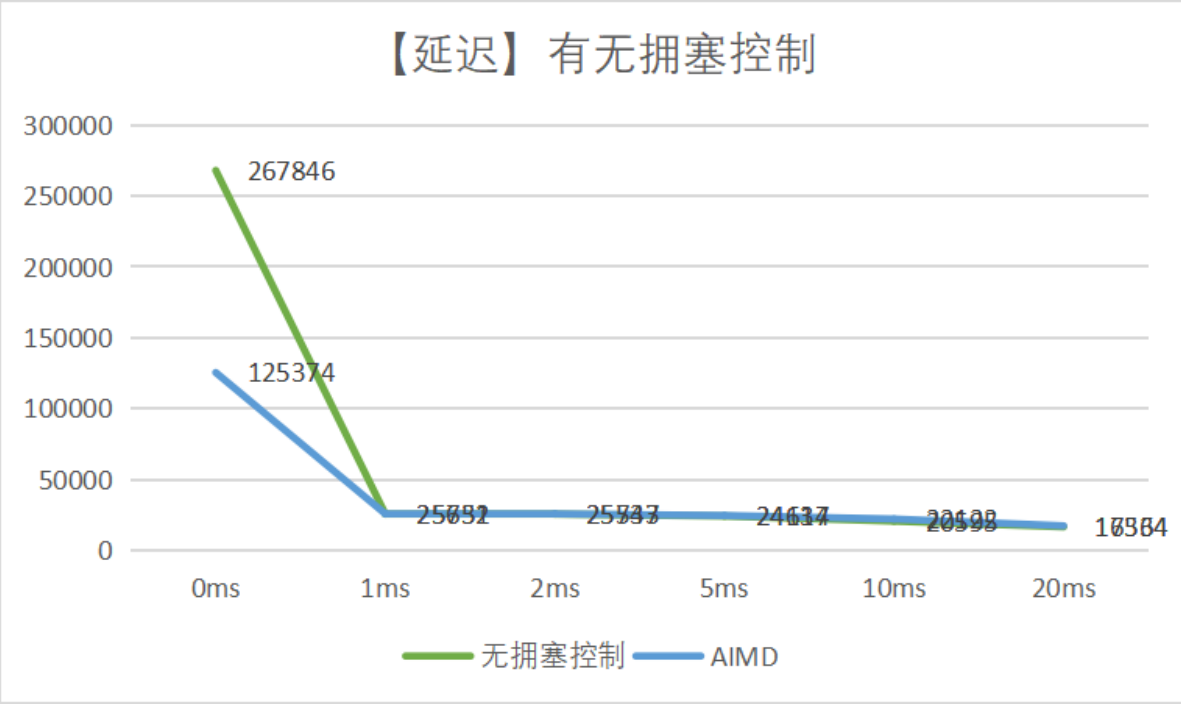
无拥塞控制机制和有拥塞控制机制在不同丢包率下的传输吞吐率如图表所示(单位：kB/s)。

注：拥塞控制机制选择AIMD。

从图中可以看出，无丢包率时，无拥塞控制机制的吞吐率几乎是AIMD的两倍。但是一旦出现丢包，虽然两者的传输吞吐率都会出现不同程度的下降，但AIMD的性能将会超过不使用拥塞控制机制的吞吐率性能。随着丢包率的增加，两者都趋于平稳下降，但AIMD基本优于不使用拥塞控制机制的性能。

不同延迟的性能对比

	0ms	1ms	2ms	5ms	10ms	20ms
无拥塞控制	267846	25632	25547	24137	20595	16564
AIMD	125374	25751	25733	24614	22122	17334



无拥塞控制机制和有拥塞控制机制在不同延迟的传输吞吐率如图表所示(单位：kB/s)。

从图中可以看出，随着延迟的增加，两者的传输吞吐率下降趋势大致与随着丢包率增加的趋势相同。无延迟时，无拥塞控制的性能优于AIMD的性能；一旦出现延迟，二者性能骤降；延迟的增加对二者的性能没有造成太大的差距，二者都呈现出缓慢降低的态势。