Lab3-4——基于UDP服务设计可靠传输协议并编程实现

学号：2013921
姓名：周延霖
专业：信息安全

一、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

性能测试指标：吞吐率、时延，给出图形结果并进行分析

二、性能测试

本次实验的数据均为五次测试后取平均值，`3-2`和`3-3`快速重传中均采用`go-back-n`机制，发送方和接收方采用`rdt3.0`，拥塞控制中的拥控算法采用`reno`算法，由于路由器有对应的丢包和延时功能，所以分别设置不同的丢包率和延时时间来测试不同网络传输算法的性能

(一)停等机制与滑动窗口机制性能对比

在第一个对比中滑动窗口的大小为10

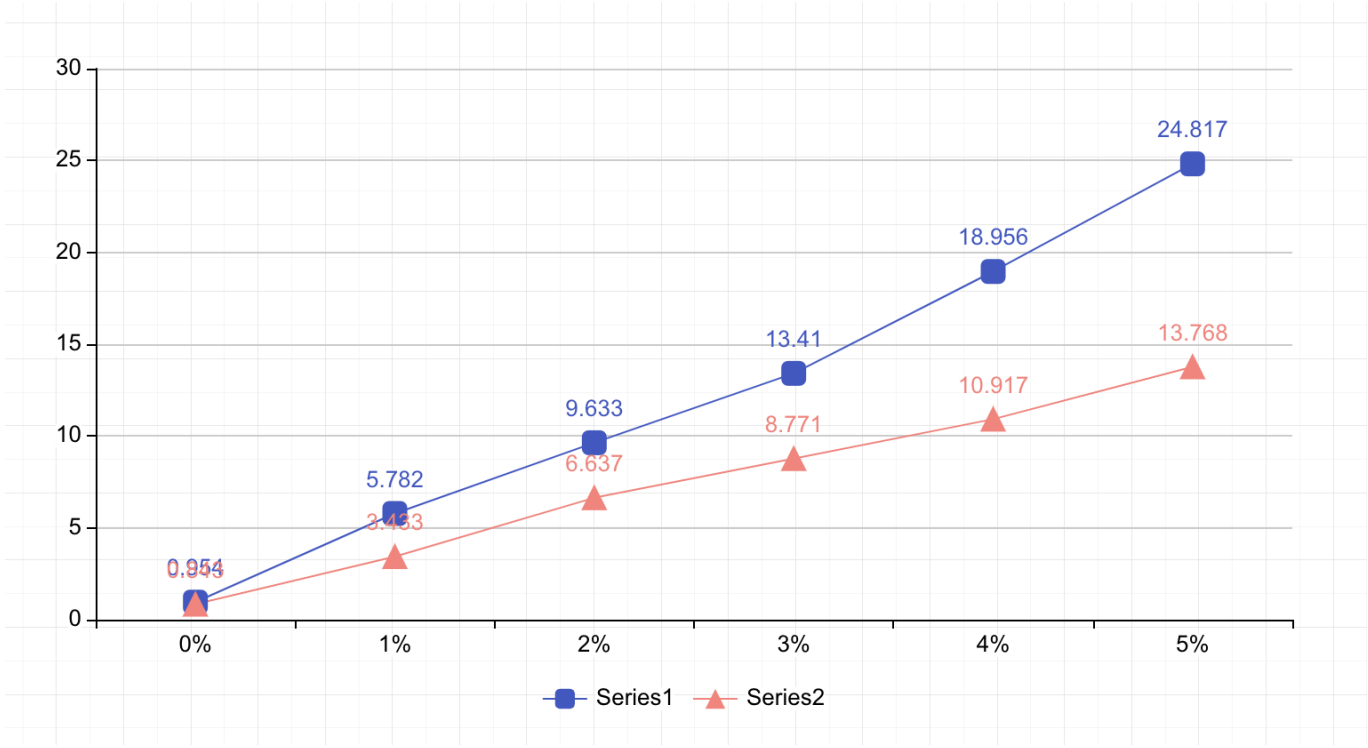
更改丢包率：

> 传输时间(单位：s)

实验数据：

丢包率	0 %	1 %	2 %	3 %	4 %	5 %
停等机制	0.954	5.782	9.633	13.410	18.956	24.817
滑动窗口	0.843	3.433	6.637	8.771	10.917	13.768

根据数据绘制图像：



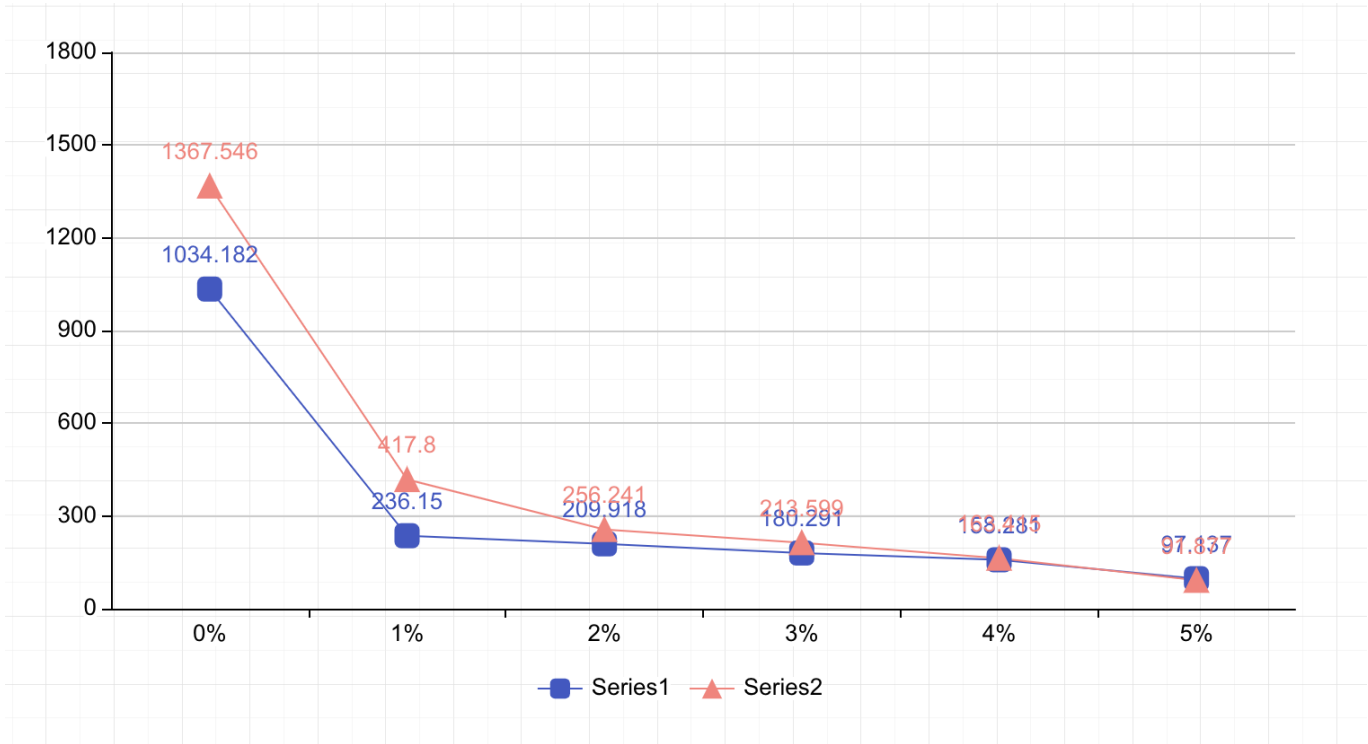
其中Series1为停等机制，Series2为滑动窗口

> 吞吐率(单位：Kbps)

实验数据：

丢包率	0 %	1 %	2 %	3 %	4 %	5 %
停等机制	1034.182	236.150	209.918	180.291	158.281	97.137
滑动窗口	1367.546	417.800	256.241	213.599	163.415	91.877

根据数据绘制图像：



其中Series1为停等机制，Series2为滑动窗口

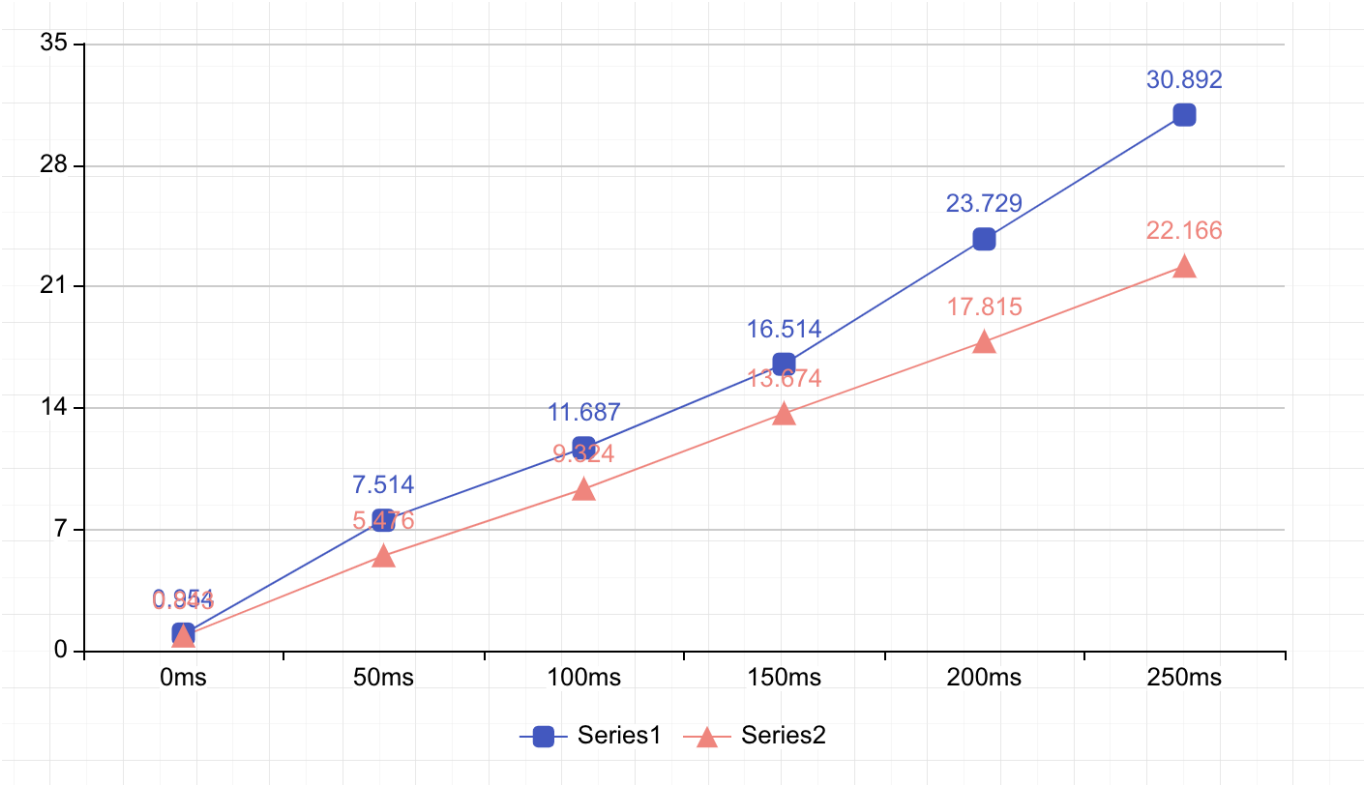
更改延时：

> 传输时间(单位：s)

实验数据：

延时（ms）	0	50	100	150	200	250
停等机制	0.954	7.514	11.687	16.514	23.729	30.892
滑动窗口	0.843	5.476	9.324	13.674	17.815	22.166

根据数据绘制图像：



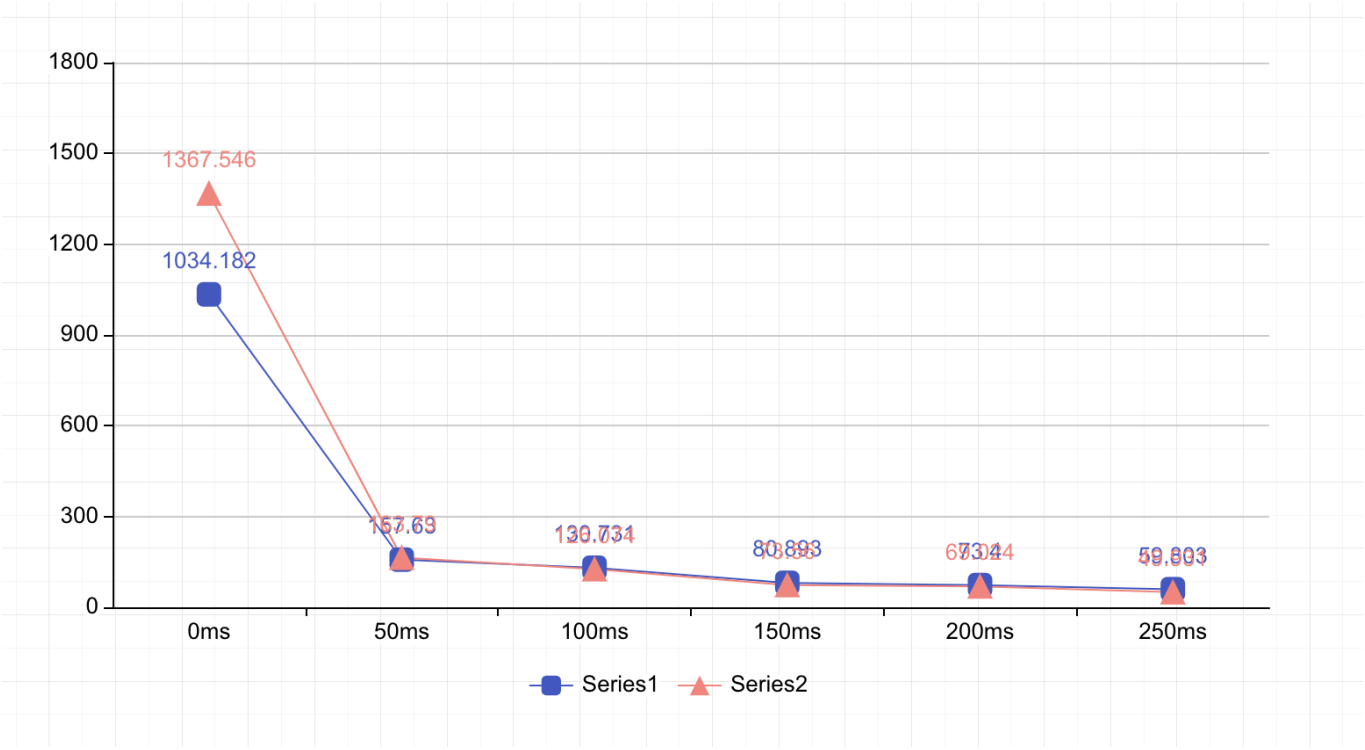
其中Series1为停等机制，Series2为滑动窗口

> 吞吐率(单位：Kbps)

实验数据：

延时（ms）	0	50	100	150	200	250
停等机制	1034.182	157.630	130.731	80.893	73.400	59.803
滑动窗口	1367.546	163.790	126.074	73.560	69.024	49.931

根据数据绘制图像：



其中Series1为停等机制，Series2为滑动窗口

结果分析：

1. 首先通过以上结果可以看出，从总体的大趋势上来看，滑动窗口机制要优于停等机制
2. 但从图中也可以看出当丢包率增大到一定程度时，停等机制慢慢有优于晃动窗口的趋势
3. 在设置延时的情况下滑动窗口更优

- 原因：

1. 首先滑动窗口允许一次发送多条数据报，并在此时等待对方的确认报文ACK，减少RTT对传输的影响
2. 其次停等机制每条消息需要单独等待时延和RTT，而滑动窗口可以同时等待多条数据报
3. 但当丢包率高的时候，更大的窗口意味着更高的重传代价，所以性能大大降低

(二)滑动窗口机制中不同窗口大小对性能的影响

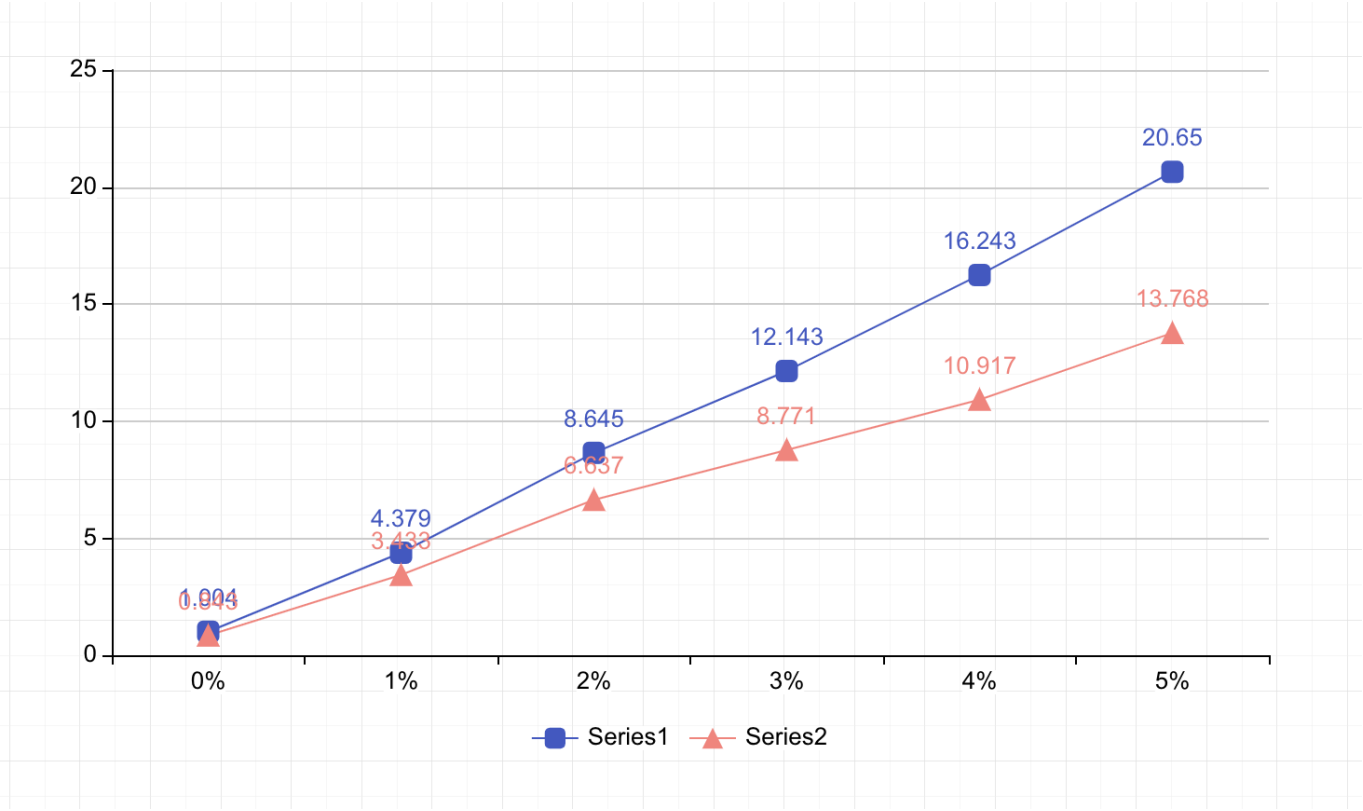
更改丢包率：

> 传输时间(单位：s)

实验数据：

丢包率	0 %	1 %	2 %	3 %	4 %	5 %
窗口大小：7	1.004	4.379	8.645	12.143	16.243	20.650
窗口大小：10	0.843	3.433	6.637	8.771	10.917	13.768

根据数据绘制图像：



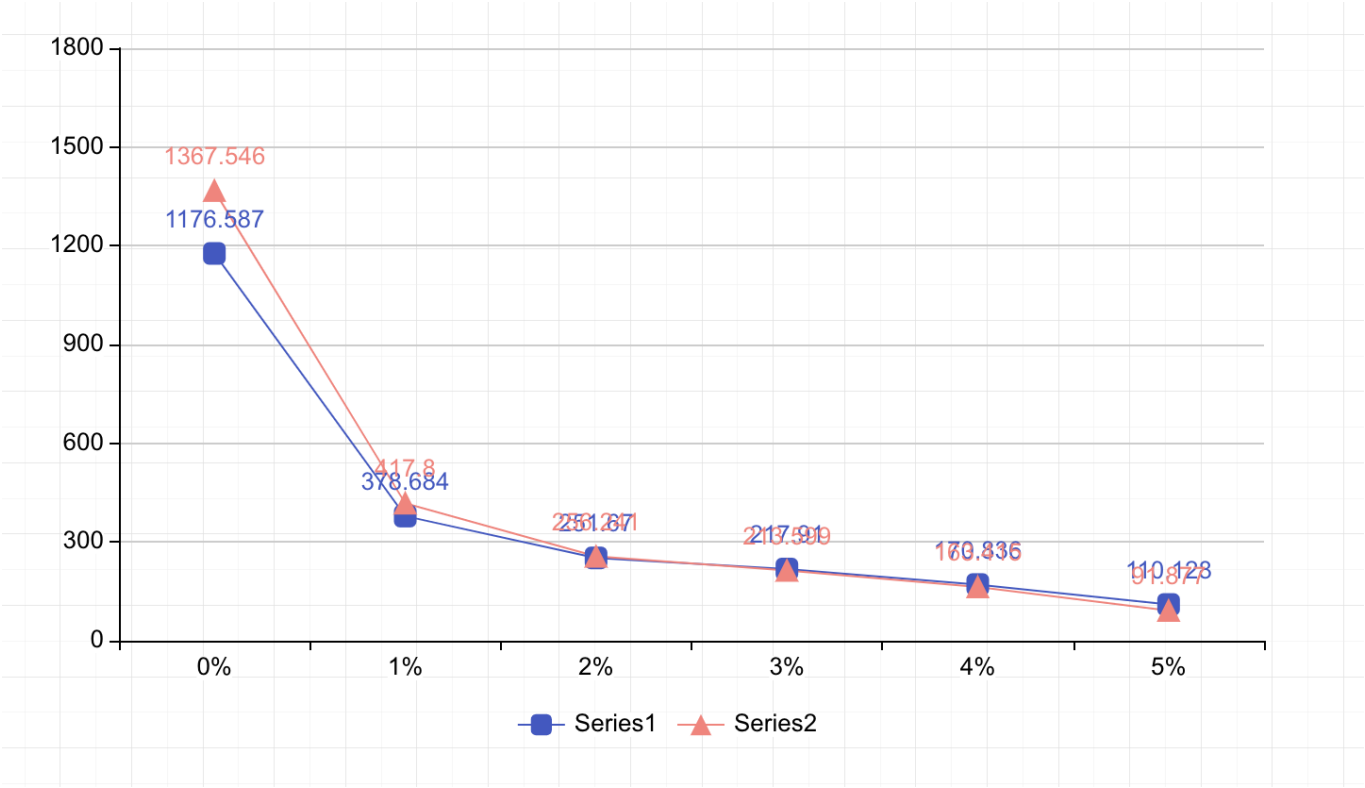
其中Series1窗口大小为7，Series2窗口大小为10

> 吞吐率(单位：Kbps)

实验数据：

丢包率	0 %	1 %	2 %	3 %	4 %	5 %
窗口大小：7	1176.587	378.684	251.670	217.910	170.836	110.128
窗口大小：10	1367.546	417.800	256.241	213.599	163.415	91.877

根据数据绘制图像：



其中Series1窗口大小为7，Series2窗口大小为10

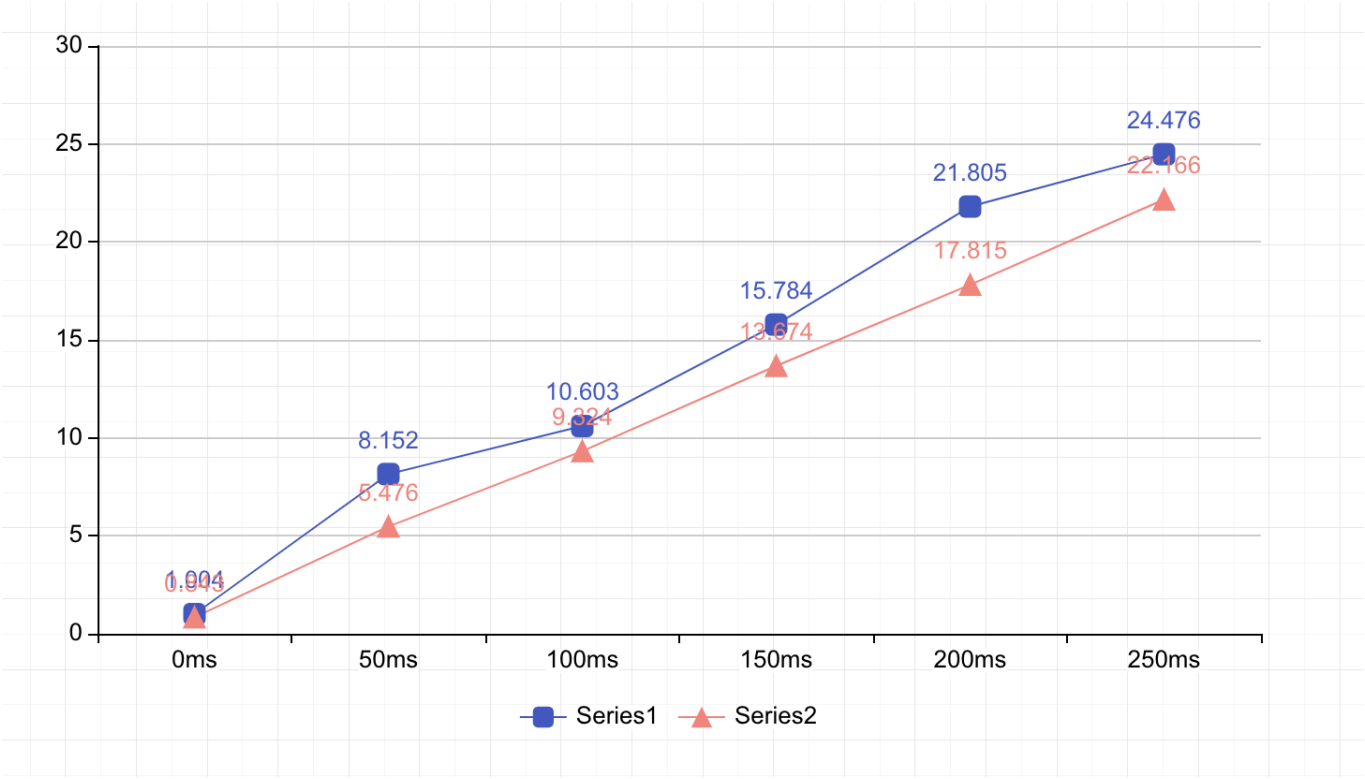
更改延时：

> 传输时间(单位：s)

实验数据：

延时（ms）	0	50	100	150	200	250
窗口大小：7	1.004	8.152	10.603	15.784	21.805	24.476
窗口大小：10	0.843	5.476	9.324	13.674	17.815	22.166

根据数据绘制图像：



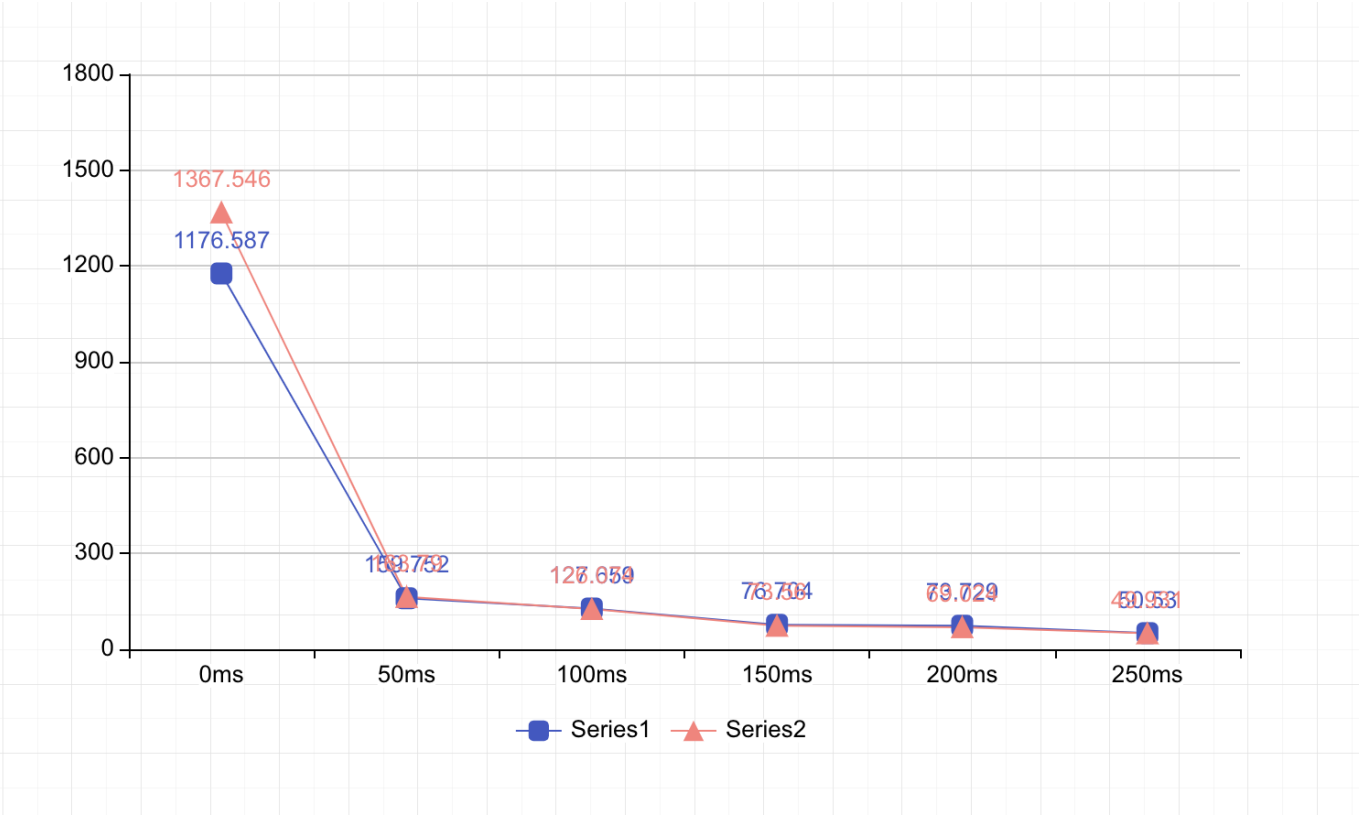
其中Series1窗口大小为7，Series2窗口大小为10

> 吞吐率(单位：Kbps)

实验数据：

延时（ms）	0	50	100	150	200	250
窗口大小：7	1176.587	159.752	127.659	76.704	73.729	50.530
窗口大小：10	1367.546	163.790	126.074	73.560	69.024	49.931

根据数据绘制图像：



其中Series1窗口大小为7，Series2窗口大小为10

结果分析：

1. 从得到的折线图中可以看出，不同的窗口大小在不同的条件下的总体趋势大致相同
2. 在网络正常的情况下，滑动窗口大的效率更高

- 原因：

1. 首先更大的窗口可以使得同时发送更多的数据报，减少等待的周期数，更好的解决时延问题
2. 但当丢包率逐渐增加时，越大的窗口会增加重传代价，重新传输更多的数据报，使得效率降低

(三)有拥塞控制和无拥塞控制的性能比较

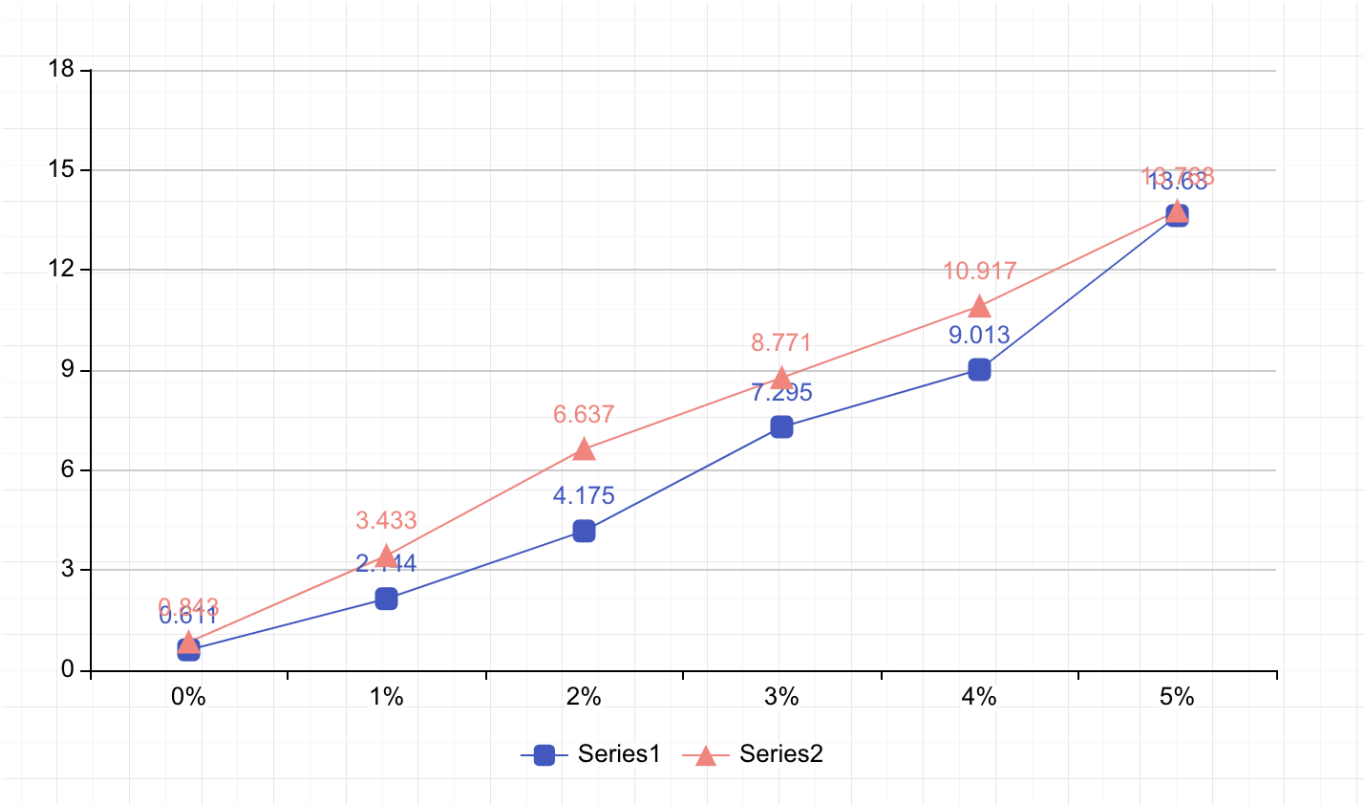
更改丢包率：

> 传输时间(单位：s)

实验数据：

丢包率	0 %	1 %	2 %	3 %	4 %	5 %
有拥塞控制	0.611	2.144	4.175	7.295	9.013	13.630
无拥塞控制	0.843	3.433	6.637	8.771	10.917	13.768

根据数据绘制图像：



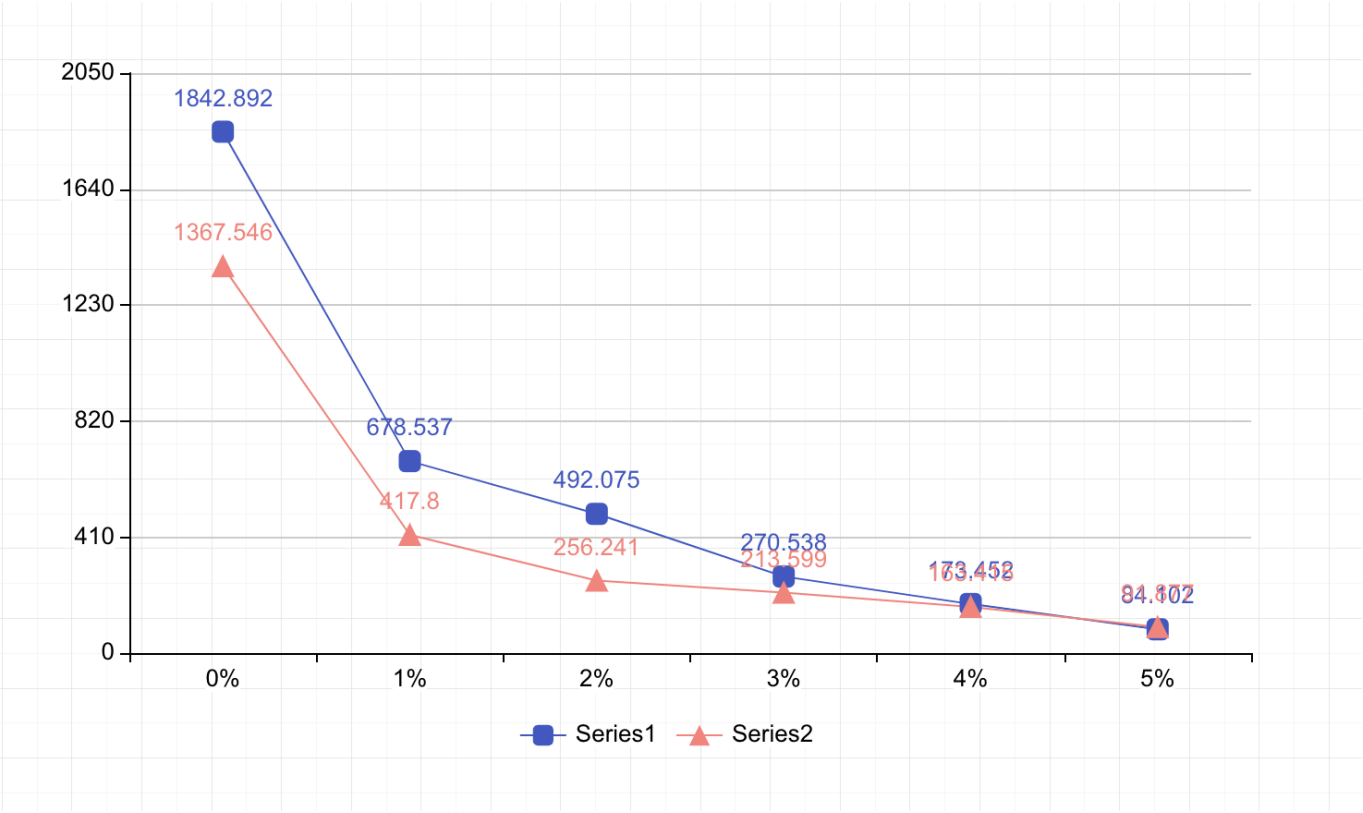
其中Series1有拥塞控制，Series2无拥塞控制

> 吞吐率(单位：Kbps)

实验数据：

丢包率	0 %	1 %	2 %	3 %	4 %	5 %
有拥塞控制	1842.892	678.537	492.075	270.538	173.452	84.102
无拥塞控制	1367.546	417.800	256.241	213.599	163.415	91.877

根据数据绘制图像：



其中Series1有拥塞控制，Series2无拥塞控制

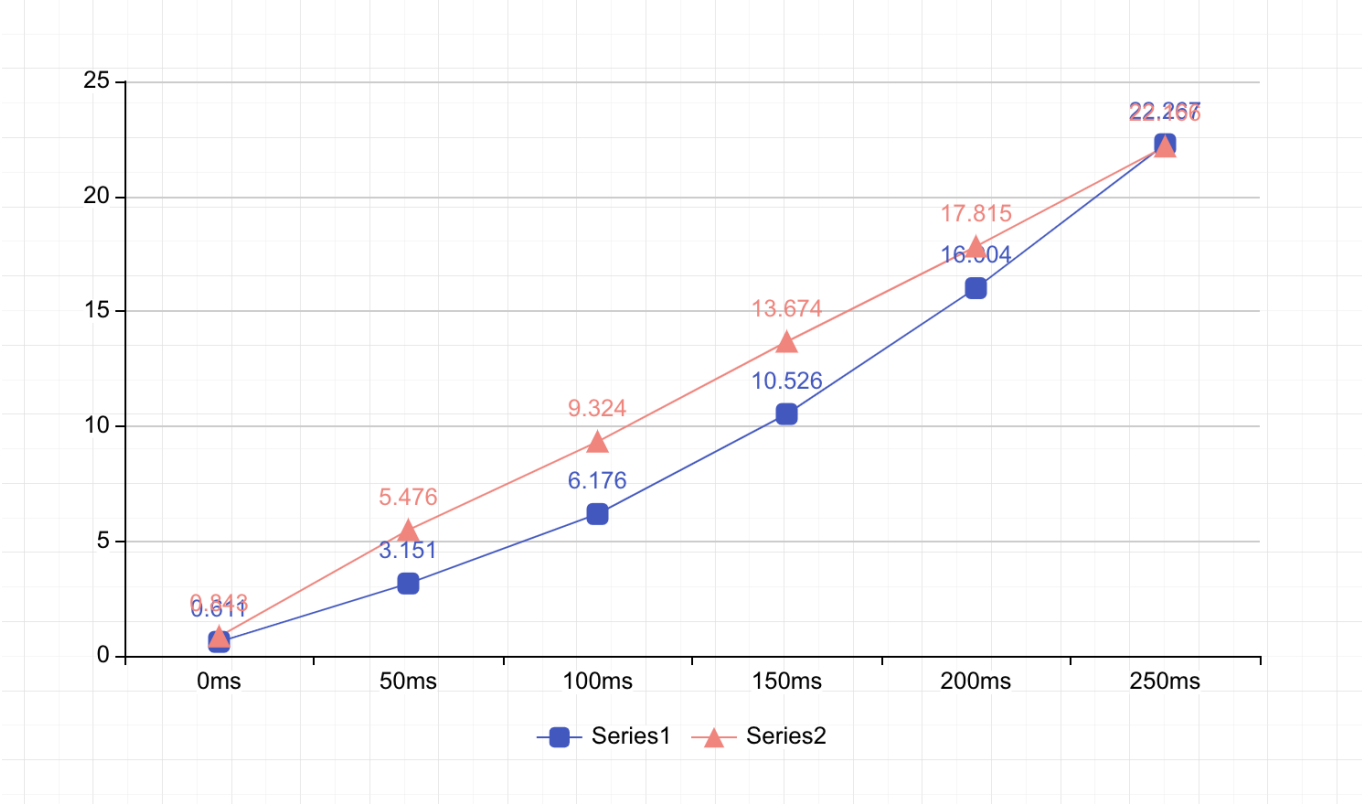
更改延时：

> 传输时间(单位：s)

实验数据：

延时（ms）	0	50	100	150	200	250
有拥塞控制	0.611	3.151	6.176	10.526	16.004	22.267
无拥塞控制	0.843	5.476	9.324	13.674	17.815	22.166

根据数据绘制图像：



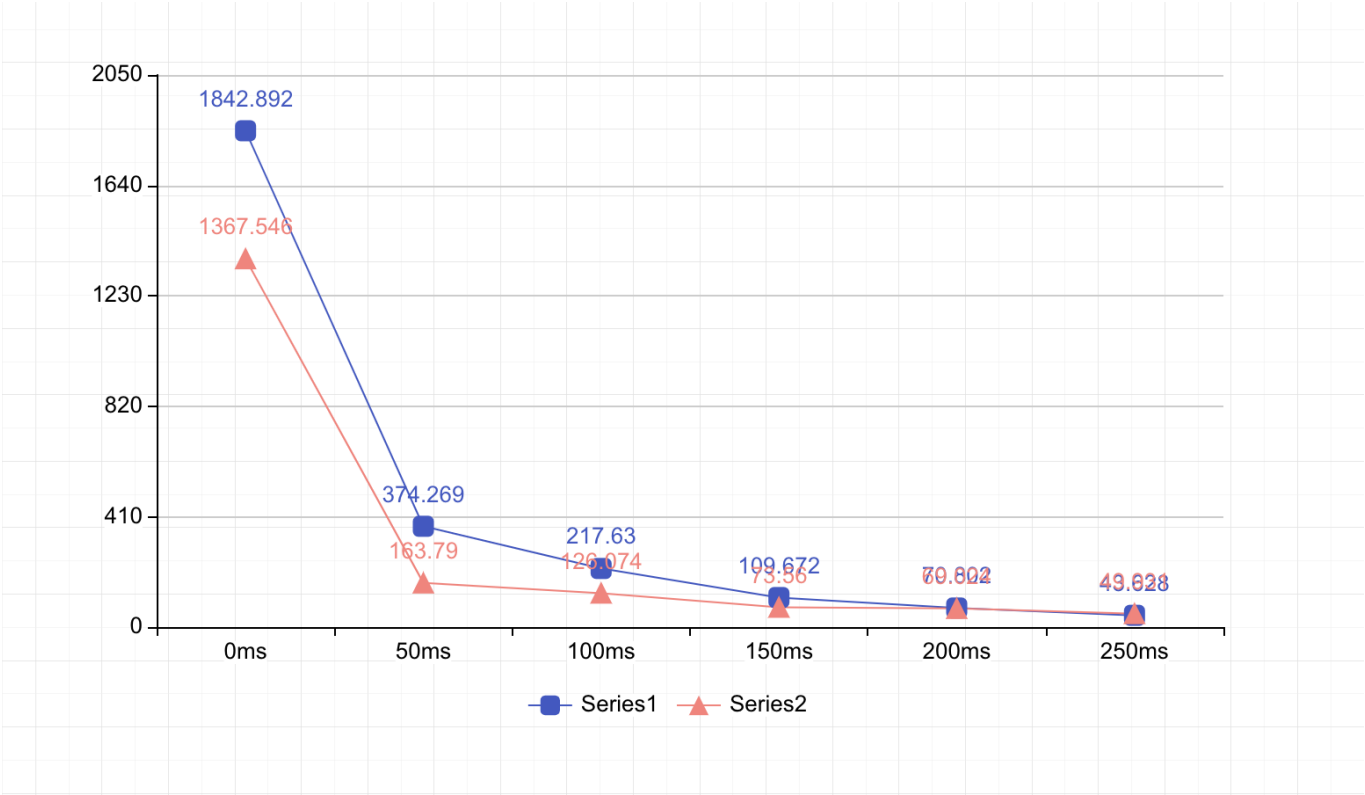
其中Series1有拥塞控制，Series2无拥塞控制

> 吞吐率(单位：Kbps)

实验数据：

延时（ms）	0	50	100	150	200	250
有拥塞控制	1842.892	374.269	217.630	109.672	70.802	43.628
无拥塞控制	1367.546	163.790	126.074	73.560	69.024	49.931

根据数据绘制图像：



其中Series1有拥塞控制，Series2无拥塞控制

结果分析：

1. 在网络一切正常是，有拥塞控制相对于无拥塞控制整体趋势上效率更高
2. 但是当网络逐渐变差，拥控机制可能不必无拥控机制

- 原因：

1. 首先拥塞控制可以使得窗口岁网络的情况而逐渐变化，当网络好的时候会变的更大，传输的数据报更多
2. 并且快速重传机制可以减少等待超时重传的次数
3. 但网络不好时，由于频繁缩小窗口导致窗口较小，甚至小于无拥控的窗口设定值，导致受时延的影响较大
4. 当网络情况更差，拥控机制可能使窗口从几百降至1，并进入重传，使得重传代价迅速增大

三、总结与展望

(1)总结

本次实验是计算机网络的3-4实验，这一次的实验其实也是对之前所有实验的一次汇总，经过本次实验，让我对本学期的实验课业也做以总结，也让我对不同算法实现网络的性能对比更加的清楚。

(2)展望

本次实验是本学期最后的实验，经过了一学期的实验，自己在网络方面编程的技术也进步了许多，期待自己未来更好的发展，万事胜意、心想事成、未来可期。