编程作业三：基于UDP服务设计可靠传输协议并编程实现

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，完成三组性能对比实验

一.作业要求

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

采用控制变量法，以传输的吞吐率，时延作为性能评价指标，给出图形结果并进行分析

二.实验设计

基于在实验3-1，3-2，3-3中所迭代的UDP传输程序进行调整以用于性能的比较

采用路由器进行转发的方式，其中前两组对比实验采用指定的路由程序。有无拥塞窗口的对比实验采用自己编写的路由转发程序完成

以路由程序中的丢包率和延时分别作为变量进行测试，其中丢包率选择0%，1%，2%，5%，时延选择0ms，50ms，100ms，200ms

程序中超时重传检测时间统一设定为1000ms，数据包大小为4096字节

在每种情况的测试中，重复测定三次以减小实验中的偶然性

三.停等机制与滑动窗口机制

为更好的进行变量控制，对于滑动窗口机制下文件的传输，选择不限制Server端窗口大小，Client端窗口大小设定为10，

1.丢包率影响：

在不同的丢包率设置下，传输时间，传输字节数如下：

停等机制

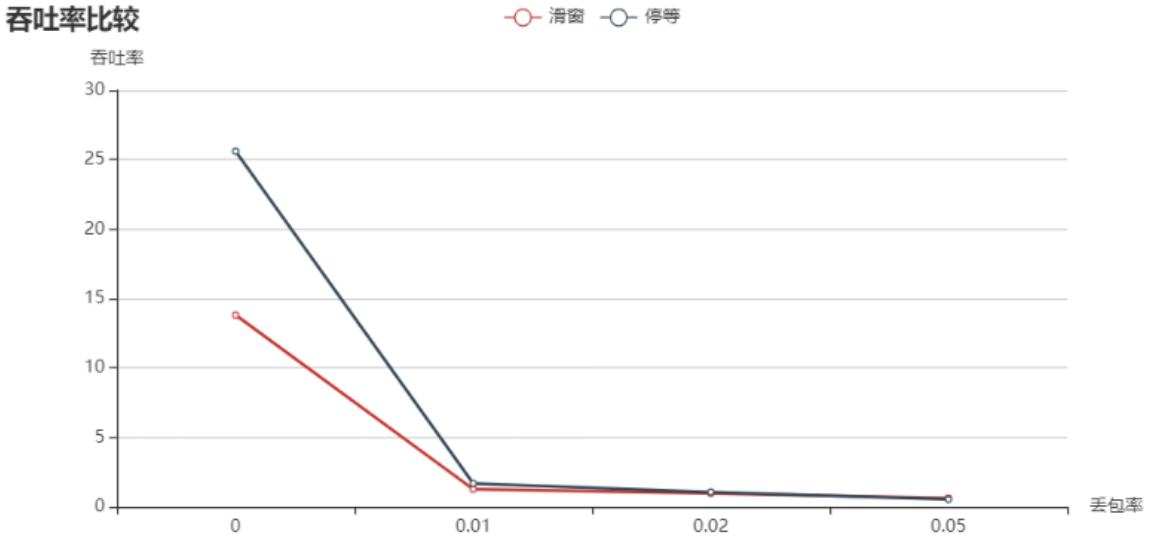
丢包率：	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0%	12075008	3757	25.712
0%	12075008	3798	25.434
0%	12075008	3766	25.651
1%	12193792	55793	1.748
1%	12197888	59007	1.654

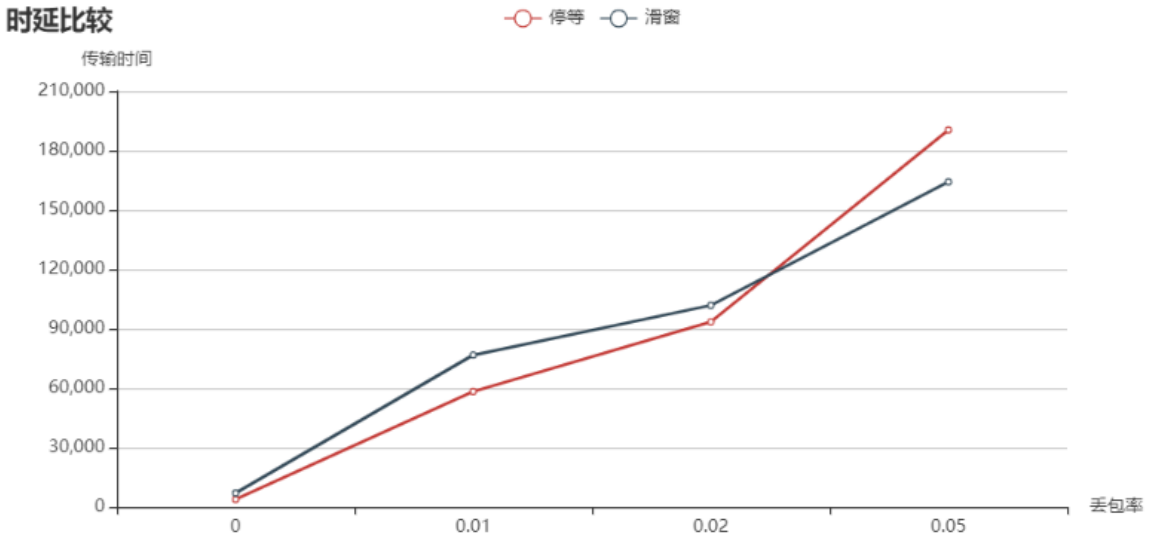
丢包率:	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
1%	12197888	60070	1.624
2%	12320768	92273	1.068
2%	12320768	93070	1.059
2%	12324864	95054	1.037
5%	12709888	189521	0.537
5%	12709888	190476	0.534
5%	12709888	191174	0.532

滑动窗口机制（窗口大小为10）

丢包率:	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0%	12066816	6987	13.816
0%	12066816	6817	14.161
0%	12066816	7175	13.454
1%	12185600	73479	1.327
1%	12189696	78369	1.244
1%	12189696	78231	1.247
2%	12316672	101690	0.969
2%	12312576	100927	0.976
2%	12312576	102724	0.959
5%	12701696	163897	0.620
5%	12701696	164205	0.619
5%	12701696	164277	0.619

根据实验结果绘制分析图表





根据折线图结果，结合表格数据，可得

（1）随丢包率的增长，停等机制下与滑窗机制下文件传输吞吐率均呈下降趋势。因为丢包率增长，出现了更多的超时重传错误，在超时检测下花费了更多时间

（2）在丢包率为0%，1%，2%时，滑动窗口下的吞吐率明显低于停等机制，可能因为在滑动窗口的多线程设置下，线程之间的调度与互斥锁的调整花费了较多时间，从而降低了整体的吞吐率。根据表格中数据也可得，二者完成的数据传输量相近，主要区别体现在传输完成时间上。

（3）当丢包率达到5%时，滑动窗口与停等机制下吞吐率均收到了限制。此时滑动窗口的优势有了一定的显现，传输吞吐率略优于停等机制,传输时间也较短

2.延时影响

在路由器不同的延时设置下，传输时间，传输字节数如下：

停等机制

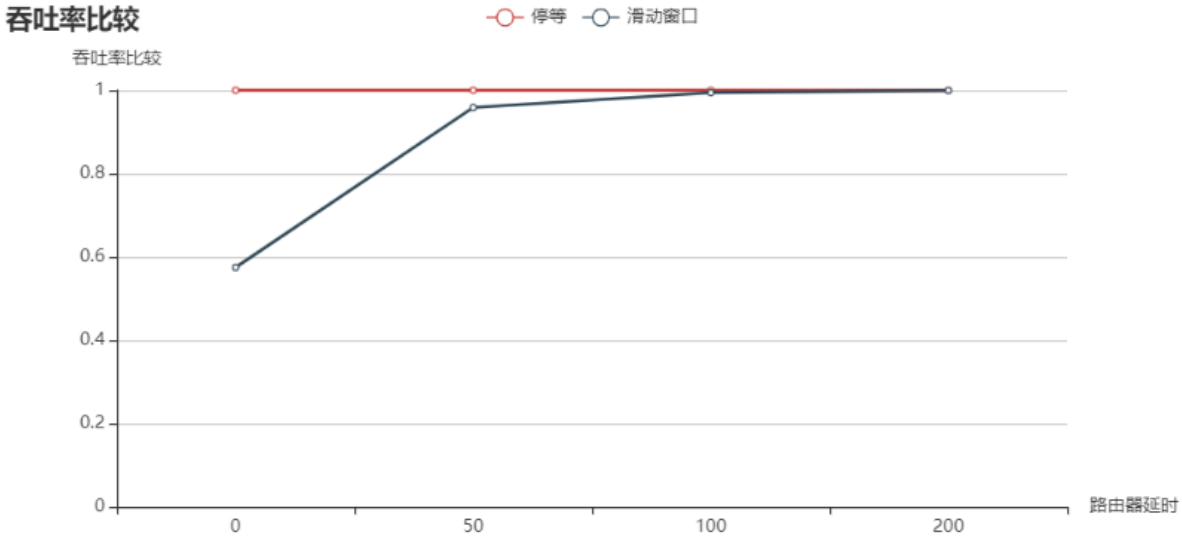
转发延时（ms）	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12075008	3757	25.71202129
0	12075008	3798	25.43445603

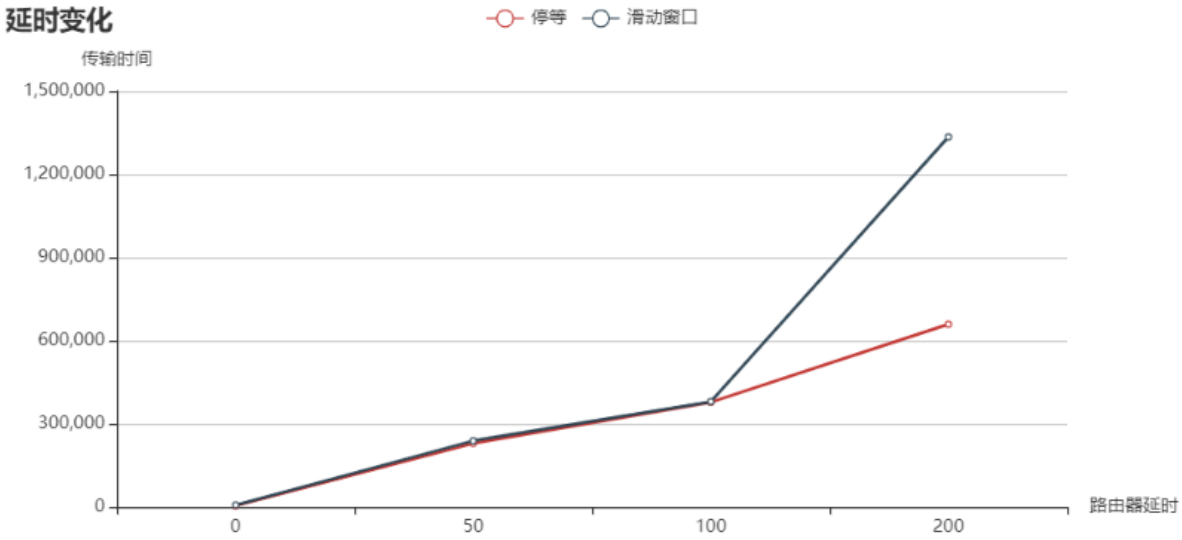
转发延时（ms）	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12075008	3766	25.65057461
50	12075008	222219	0.434706591
50	12075008	230239	0.419564296
50	12075008	234206	0.412457683
100	12075008	377712	0.255750582
100	12075008	377281	0.256042748
100	12075008	377771	0.25571064
200	12075008	658658	0.146661946
200	12075008	658175	0.146769573
200	12075008	659343	0.146509577

滑动窗口机制（窗口大小为10）

转发延时（ms）	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12066816	6774	14.25074225
0	12066816	6645	14.52739323
0	12066816	6289	15.34974209
50	12066816	237608	0.406276422
50	12066816	238963	0.403972699
50	12066816	238938	0.404014966
100	12066816	379823	0.254156615
100	12066816	379687	0.254247651
100	12066816	379238	0.254548669
200	24727552	1349957	0.146538309
200	24154112	1318355	0.146571216
200	24477696	1336014	0.146571494

根据实验结果绘制分析图表





在绘制路由器延时影响下的传输吞吐率比较时，为较为直观的体现停等机制与滑动窗口机制下传输吞吐率的相对情况，以停等机制下的传输吞吐率为基准1进行了比较，

根据折线图结果，结合表格数据，可得

（1）随路由器转发时延的变化，滑动窗口机制的传输性能普遍低于停等机制。原因可能与滑动窗口机制下的多线程调度相关

（2）随着转发延时的进一步增大，停等机制与滑动窗口机制的传输吞吐率逐渐趋近相同。

（3）由传输时间的变化可得，在路由器延时逐渐增大的情况下，滑动窗口机制所用的传输时间相较于停等机制有较为明显的变化。但二者的吞吐率却十分接近。结合表格中数据可得。在传输延时较大的情况下，滑动窗口所完成的数据包传输也远远多于停等机制。较大的传输延时使得滑动窗口中较多的数据包处于等待确认状态，在超时检测的情况下， 有更多的数据包会因为超时而不断重传，因此导致滑动窗口机制的传输时间远远多于停等机制。

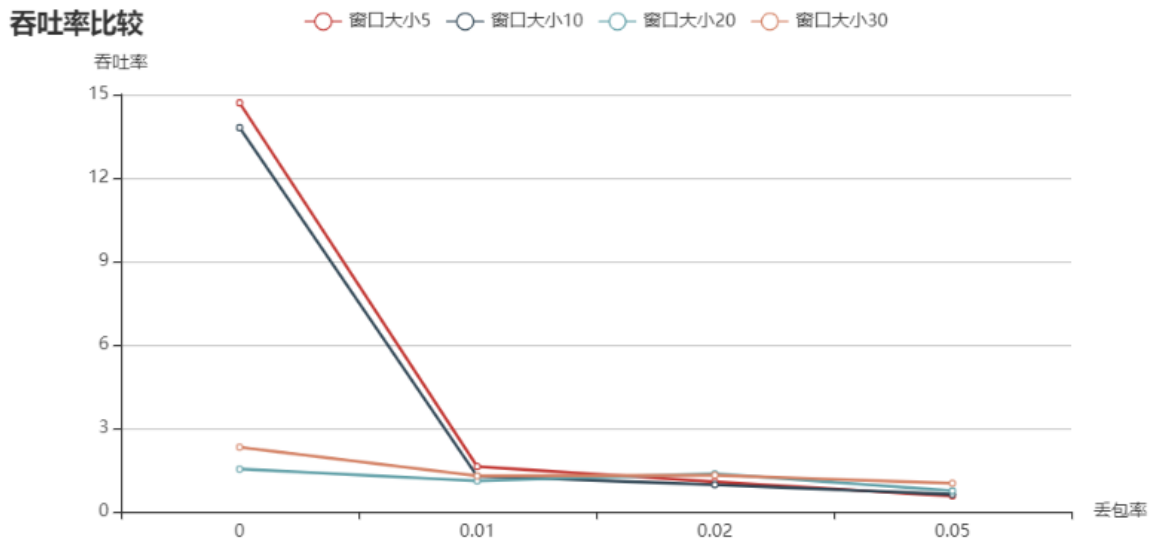
四.不同滑动窗口大小

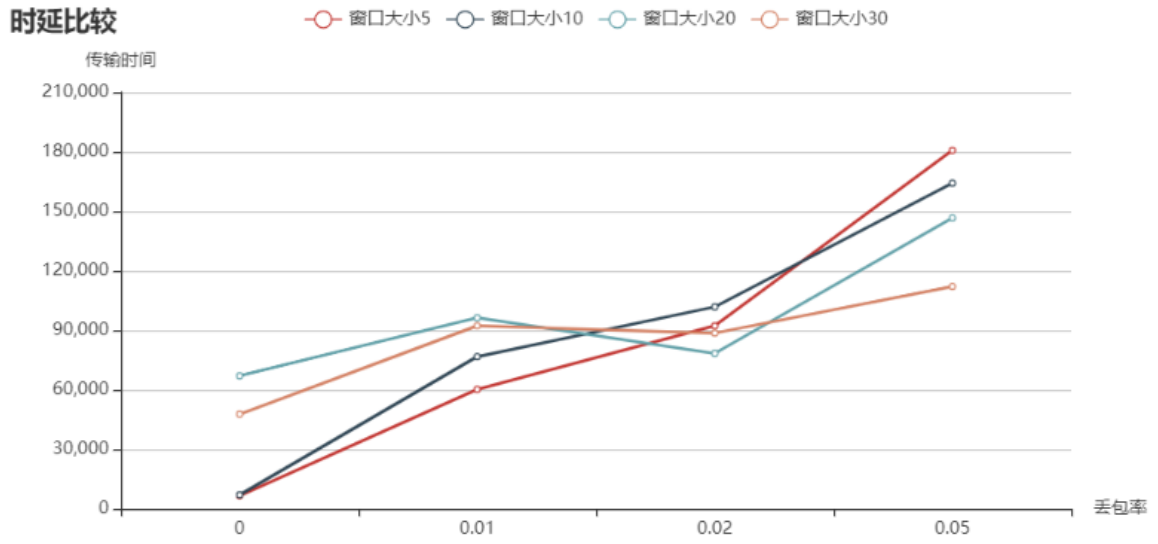
在Server端窗口大小设定为50，即不做相关限制，在Client端，设定窗口大小分别为5，10，20，30，比较不同窗口大小下传输的性能

1.丢包率影响

丢包率	窗口大小	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	5	12066816	6569	14.69472214
0	10	12066816	6993	13.80445131
0	20	12839595	67049	1.531973154
0	30	13784405	47652	2.314178684
1%	5	12188331	60191	0.007999403
1%	10	12188331	76693	1.271389114
1%	20	13332480	96281	1.107793549
1%	30	14798848	92269	1.283100026
2%	5	12313941	92205	1.068392978
2%	10	12313941	101780	0.96788375
2%	20	13294251	78361	1.357237117
2%	30	14473899	88570	1.307336046
5%	5	12701696	180702	0.562325711
5%	10	12701696	164126	0.619118005
5%	20	13656064	146587	0.745279347
5%	30	14262272	112098	1.017840075

根据实验结果绘制分析图表





根据折线图结果，结合表格数据，可得

（1）在不同窗口大小下，随丢包率的增加，传输所需时间均不断增大，同时收超时检测的影响，其传输吞吐率也在不断下降

（2）随丢包率的增大，在丢包率为5%时，可发现窗口大小为30时其吞吐率最大，同时所需的传输时间也最短，说明提高窗口大小在一定程度上有助于传输性能的提升

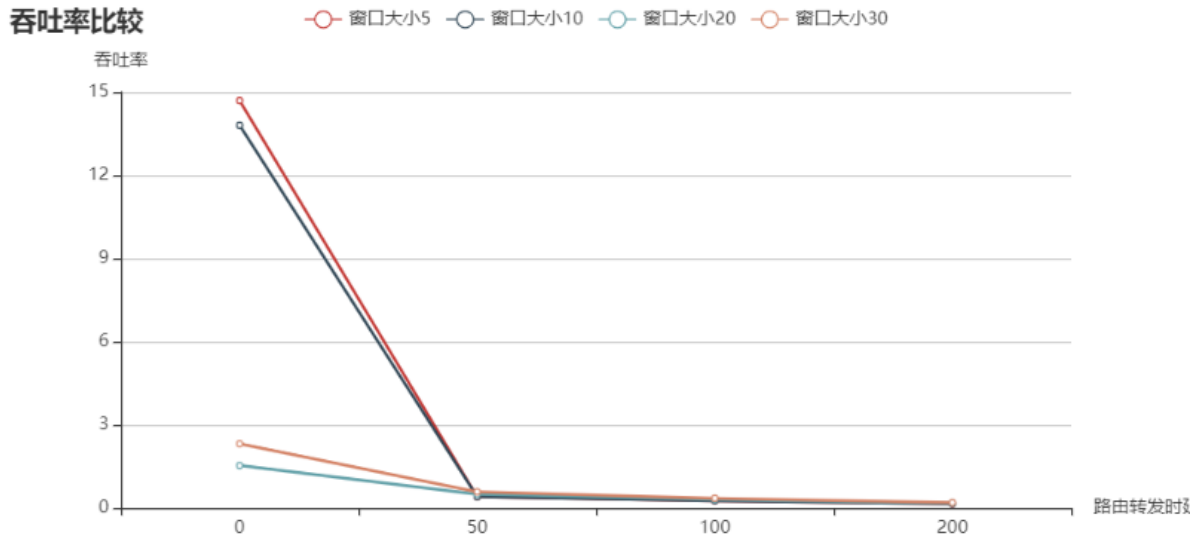
（3）由吞吐率变化折线图中可以发现，当丢包率较小时，窗口大小为20与30的情况下其吞吐率相较于窗口大小为5和10时有较为明显的差距，其原因可能为，同一时间有较多的数据包在缓存窗口中，由超时检测导致较多的数据包被重复发送

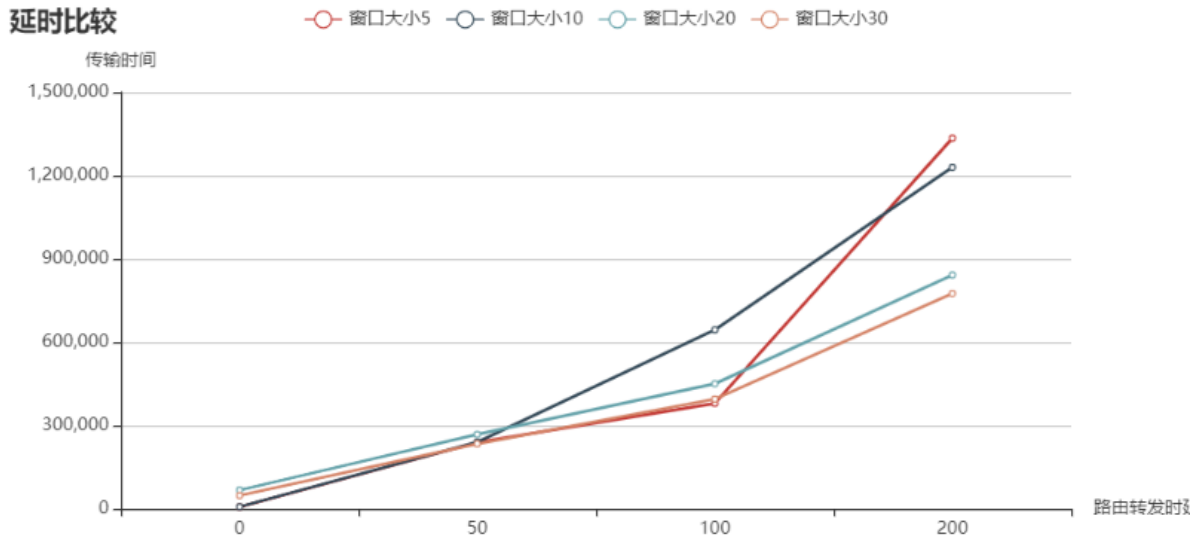
2.延时影响

路由转发延时	窗口大小	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	5	12066816	6569	14.695
0	10	12066816	6993	13.804
0	20	12839595	67049	1.532
0	30	13784405	47652	2.314
50	5	12066816	238503	0.405
50	10	12070912	239695	0.403
50	20	16486400	267751	0.493
50	30	16961566	233874	0.580
100	5	12066816	379583	0.254
100	10	20953771	644091	0.260
100	20	17162240	450039	0.305
100	30	17184085	394882	0.348
200	5	24453120	1334775	0.147

路由转发延时	窗口大小	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
200	10	23090517	1229695	0.150
200	20	18472960	842013	0.176
200	30	18378752	775459	0.190

根据实验结果绘制分析图表：





根据折线图结果，结合表格数据，可得

（1）与丢包率对传输性能的影响类似，在传输时延较小，即网络状况较优时，滑动窗口较小的情况下其传输性能更优

（2）随路由转发时延的逐渐增大，增大滑动窗口所带来的优势开始显现，窗口大小为30时，其最终完成传输的时间约是窗口较小时的一半，在吞吐率上也体现出一定优势。

五.有无拥塞控制

在有无拥塞控制的对照实验中，编写了单独的路由转发程序进行转发，为模拟网络拥塞情况，每转发1000个数据包会出现一次网络拥塞，在接下来的几个数据包转发中将有500ms的延时，网络拥塞会持续5个数据包，后恢复正常的转发延时。

无拥塞控制的滑动窗口大小设定为10

1.丢包率影响

无拥塞控制滑动窗口

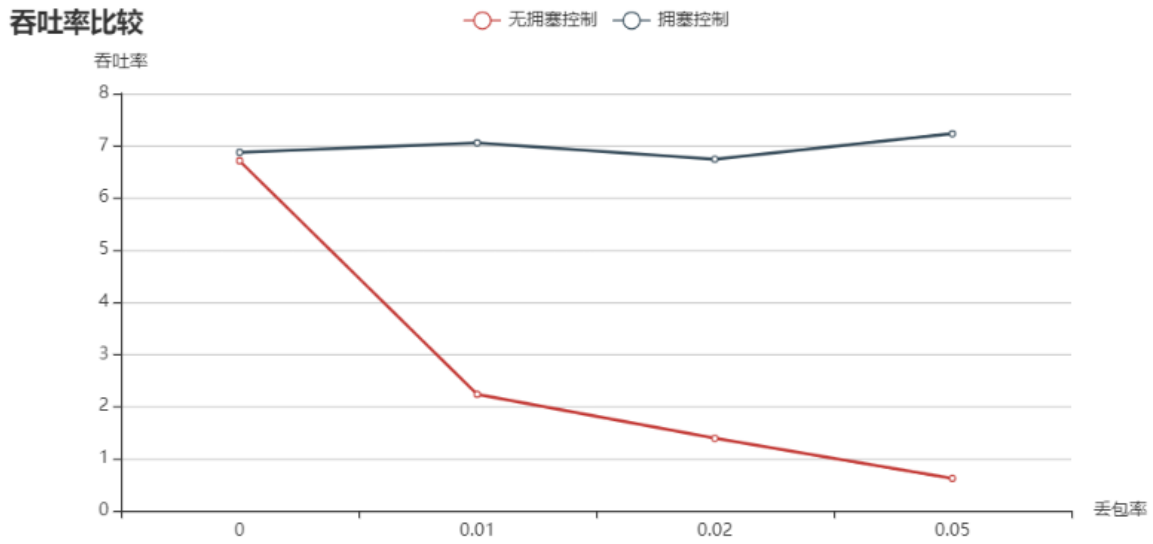
丢包率	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12177408	14524	6.707
0	12177408	14629	6.659
0	12185600	14438	6.752
1%	12304384	44233	2.225
1%	12304384	44077	2.233
1%	12312576	44207	2.228
2%	12369920	71296	1.388
2%	12378112	71331	1.388
2%	12369920	71243	1.389
5%	12750848	165106	0.618
5%	12754944	165821	0.615
5%	12742656	163728	0.623

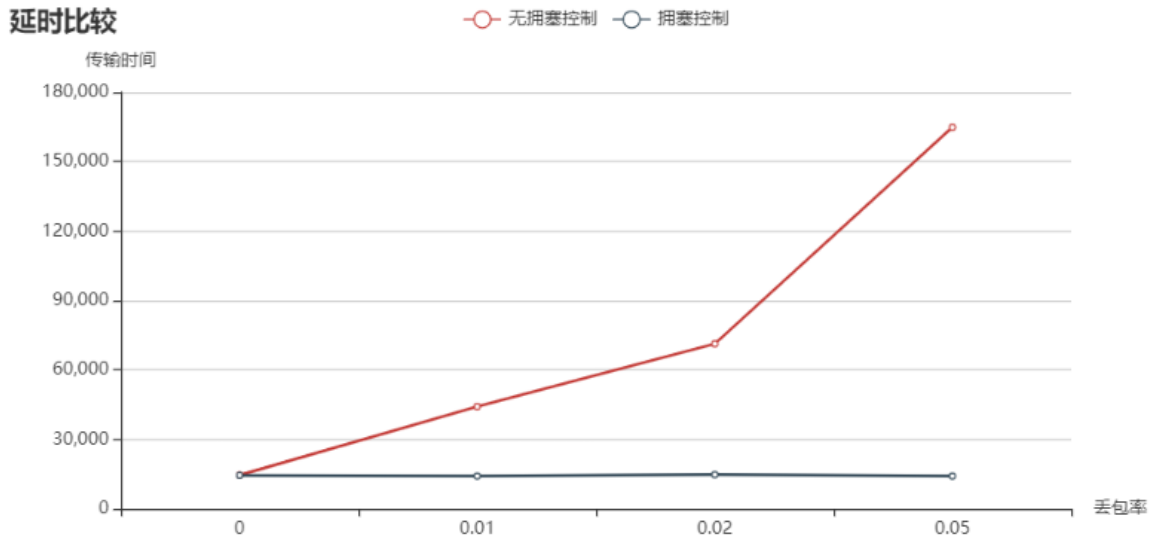
reno算法拥塞控制

丢包率	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12374016	14137	7.002
0	12382208	14797	6.694
0	12390400	14330	6.917
1%	12427204	14039	7.082
1%	12427264	14815	6.711
1%	12414976	13491	7.362
2%	12406784	14429	6.879
2%	12406784	14874	6.673

丢包率	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
2%	12410880	14896	6.665
5%	12742656	13933	7.317
5%	12742656	14328	7.115
5%	12742656	14064	7.248

根据实验结果绘制分析图表：





根据折线图结果，结合表格数据，可得

通过折线图数据可较为明显的看出，在有拥塞控制的条件下，传输效率明显优于无拥塞控制的情况，一方面因为reno算法可以更好的根据网络情况调整窗口大小，避免减少重复数据包的发送，同时也与在reno算法中添加了三次重复ack检测，在检测到三次重复ack时会立即重新发送当前的数据包，从而避免在rdt3.0的情况下，丢失的数据包必须等待超时检测后再重新发送，从而缩短了所需的等待时间，可以发现在不同的丢包率影响下，拥塞控制下的传输时间几乎没有明显变化。

2.延时影响

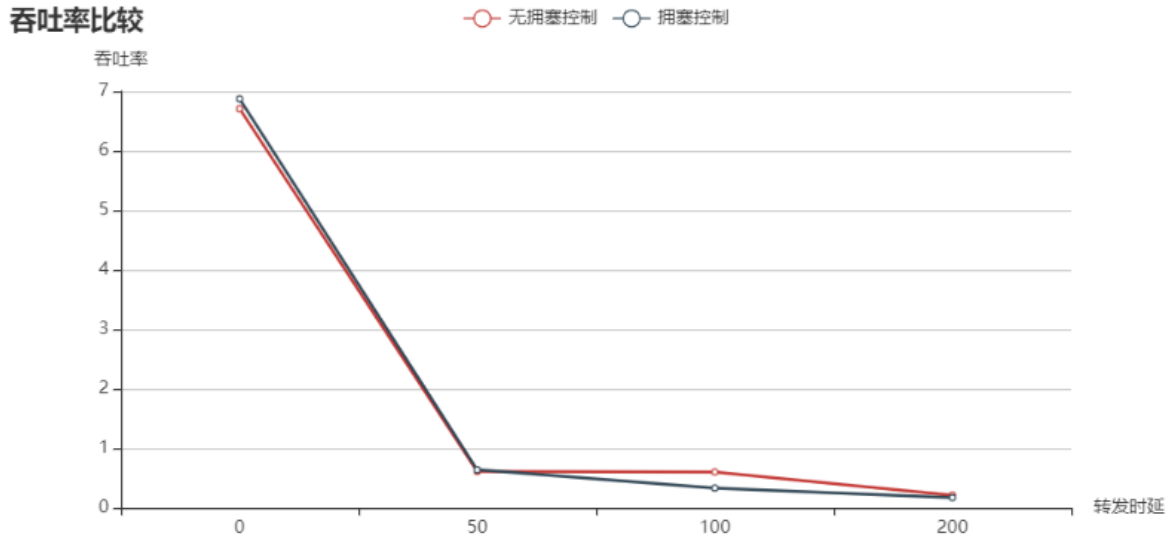
无拥塞控制

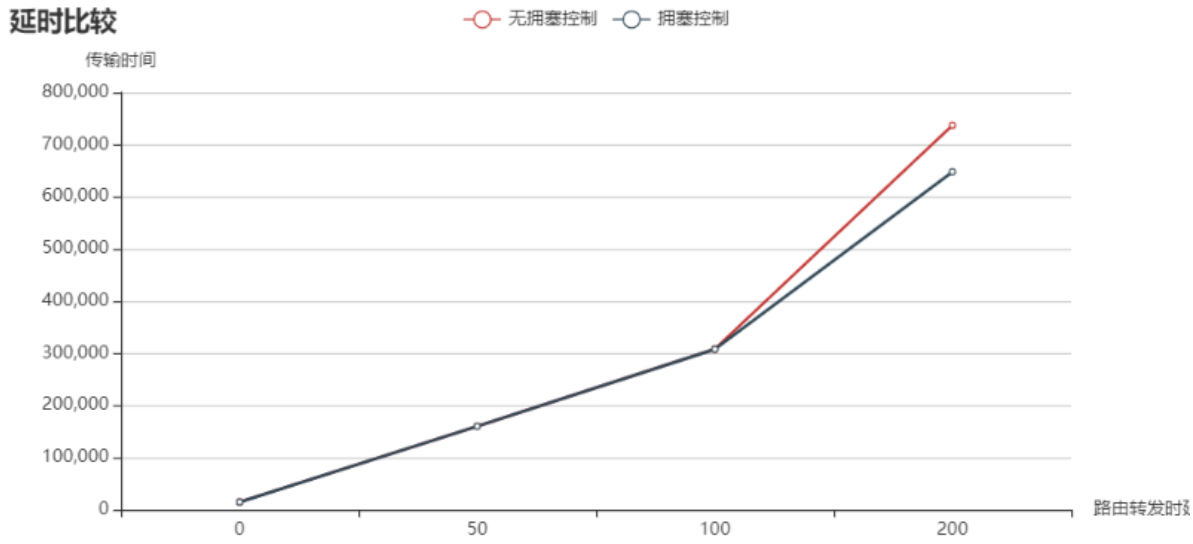
路由转发延时	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12177408	14524	6.707
0	12177408	14629	6.659
0	12185600	14438	6.752
50	12201984	159949	0.610
50	12206080	160119	0.610
50	12197888	159413	0.612
100	22343680	308465	0.579
100	23285760	307608	0.606
100	23670784	307223	0.616
200	19292160	743584	0.208
200	18956288	736268	0.206
200	19185664	730440	0.210

reno算法拥塞控制

路由转发延时	传输大小（Byte）	传输时间（ms）	吞吐率（Mbps）
0	12374016	14137	7.002
0	12382208	14797	6.694
0	12390400	14330	6.917
50	12742656	159638	0.639
50	12742656	159898	0.638
50	12742656	159969	0.637
100	12750848	307735	0.331
100	12750848	307538	0.332
100	12750848	307264	0.332
200	13594624	642303	0.169
200	13692928	647609	0.169
200	13815808	652504	0.169

根据实验结果绘制分析图表：





根据折线图结果，结合表格数据，可得

在网络中转发延时不断加大的情况下，拥塞控制算法在吞吐率的表现上优势并不明显，但在所需的传输时间上，可发现拥塞控制算法所需的传输时间在一定程度上优于无拥塞控制算法的情况，结课表格数据也可得到拥塞控制下其所需传输的数据量也明显低于无拥塞控制情况。说明reno算法对于缓解拥塞网络中数据包的不断重发有一定的作用

五.实验总结

本次实验是对前几次实验的一次综合总结，在对前几次实验的回顾中，也获得了新的理解，在算法的实现，程序的编写上都发现了在过去不曾发现的问题。对计算机网络UDP传输的流程与控制有了更见深刻的理解。

本次实验中还有许多不足之处。对于变量的设计与性能评价指标缺乏可靠的分析，对于传输中各个参数的设置也不尽合理。希望在以后的学习中可以进一步完善这方面能力。