南開大學
Nankai University

计算机学院

计算机网络实验报告

计算机网络大作业第四部分

姓名：徐文斌

学号：2010234

专业：计算机科学与技术

2022 年 12 月 30 日

目录

1 实验内容

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。

2 实验要求

1. 完成给定测试文件的传输，显示传输时间和平均吞吐率。
2. 控制变量法。
3. 性能测试指标包括吞吐率和时延，给出图形结果并进行分析。

3 实验结果

3.1 停等机制与滑动窗口机制性能对比

这里我们固定最大报文端长度 MSS 为 6666 个字节。滑动窗口我们使用 GBN 算法，并固定窗口大小为 10。然后，我们测试程序的传输性能。首先，固定延时为 0ms，分别取不同的丢包率。测试两种机制在测试文件 1.jpg 上的吞吐率和传输时延，所得数据如表1所示。

丢包率		0	0.1	0.3	0.5	1	1.5	2	3	5
时延	停等机制	3.262	4.393	4.928	5.498	6.017	6.505	7.06	8.518	12.31
	滑动窗口	1.252	3.896	4.272	4.507	5.71	7.126	8.492	11.466	20.948
吞吐率	停等机制	569.39	422.80	376.89	337.82	308.68	285.53	263.08	218.05	150.88
	滑动窗口	1483.51	476.73	434.77	412.10	325.28	260.64	218.72	161.99	88.66

表 1: 停等机制与滑动窗口机制不同丢包率下文件传输时延 (s) 及吞吐率 (KB/s)

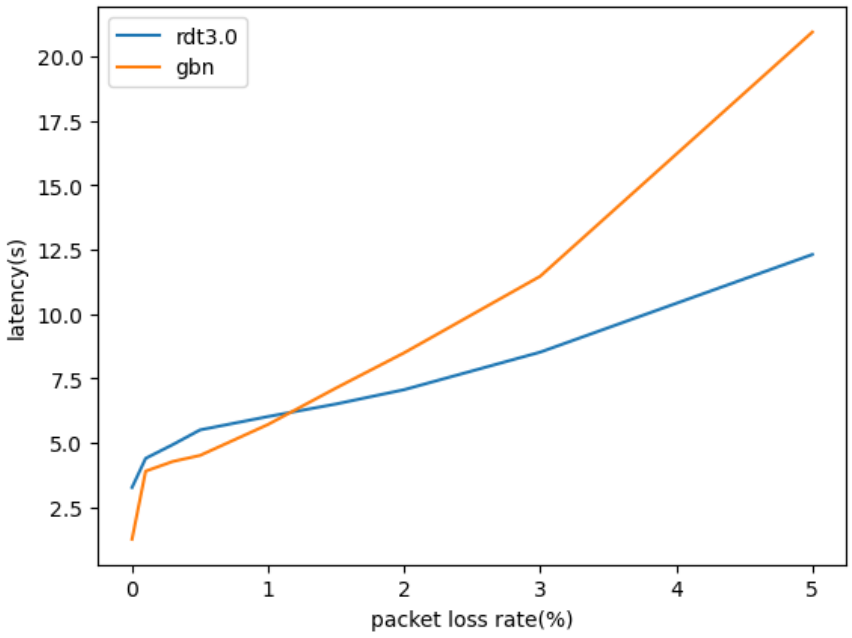

图 3.1: 停等机制与滑动窗口机制不同丢包率下文件传输时延

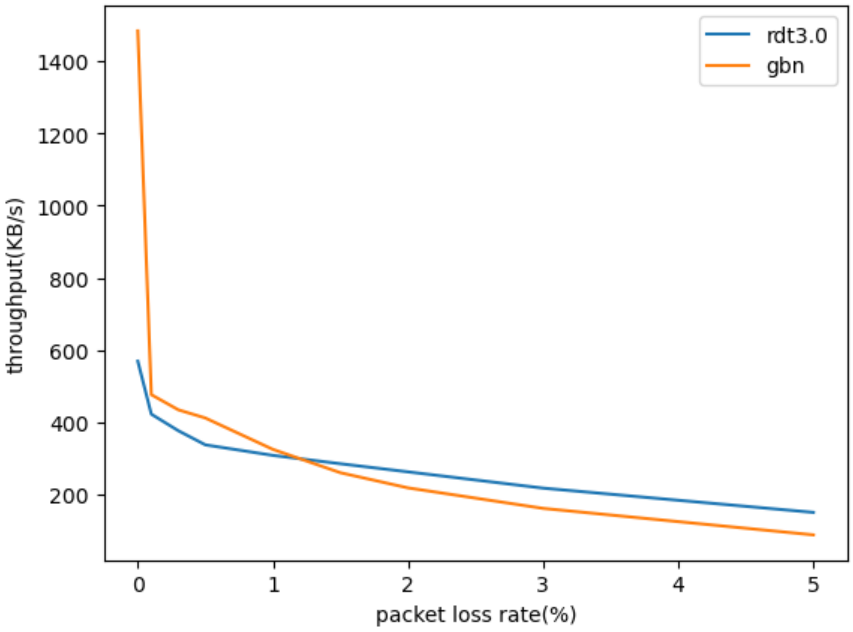

图 3.2: 停等机制与滑动窗口机制不同丢包率下文件传输吞吐率

分析图3.1和图3.2，我们可以发现，当丢包率较小时，滑动窗口机制性能优于停等机制，而随着丢包率的增大，滑动窗口机制的时延开始大于停等机制。显然，当丢包率较小时，与停等机制相比，滑动窗口一次能发送多个数据包，并且采用累积确认，可以有效利用网络带宽，性能较好；而随着丢包率的增大，二者发生超时重传的次数也在逐渐增多，而 GBN 滑动窗口机制发生超时时会重传当前窗口内的所有已发送但未确认的报文，开销较大，与之相比停等机制只会重传一个报文，重传开销较小，因此，不难推断出，当丢包率较大时，GBN 滑动窗口机制的超时重传开销过大，从而降低了其文件传输性能。

下面，我们固定丢包率为 0%，分别取不同的延时。测试两种机制在文件 1.jpg 上的吞吐率和传输时延，所得数据如表2所示。不难发现，在测试的延时下，滑动窗口机制的性能均好于停等机制，可见滑动窗口能更加充分地利用网络带宽，从而提升传输性能。

延时 (ms)		0	10	20	30
时延 (s)	停等机制	3.262	9.572	14.215	17.625
	滑动窗口	1.252	8.858	13.259	15.21
吞吐率 (KB/s)	停等机制	569.39086	194.04022	130.66148	105.38173
	滑动窗口	1483.5088	209.68085	140.08243	122.11394

表 2: 停等机制与滑动窗口机制不同延时下文件传输时延 (s) 及吞吐率 (KB/s)

3.2 滑动窗口机制中不同窗口大小对性能的影响

这里我们固定丢包率为 0%，延时为 0ms，最大报文段长度为 2300 字节。设置 GBN 不同的窗口大小，测试程序传输文件 1.jpg 的时延和吞吐率。测试结果如表3及图3.3和图3.4所示。

窗口大小	2	5	10	15	20	25	30	40	50
时延 (s)	9.435	8.963	8.635	8.431	8.343	9.426	25.649	27.746	31.609
吞吐率 (KB/s)	196.86	207.22	215.09	220.3	222.62	197.05	72.41	66.94	58.76

表 3: 滑动窗口机制中不同窗口大小文件传输时延及吞吐率

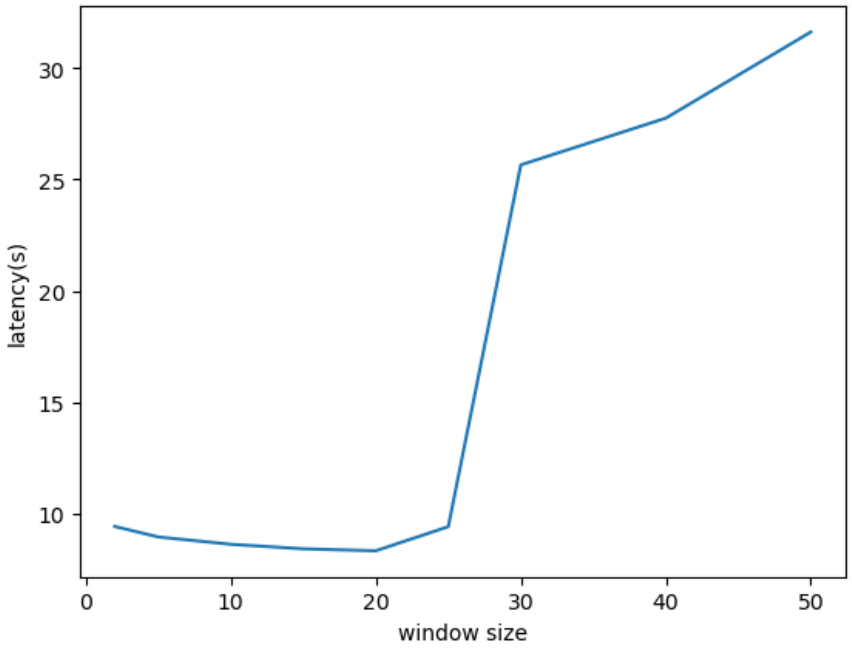

图 3.3: 滑动窗口机制中不同窗口大小文件传输时延

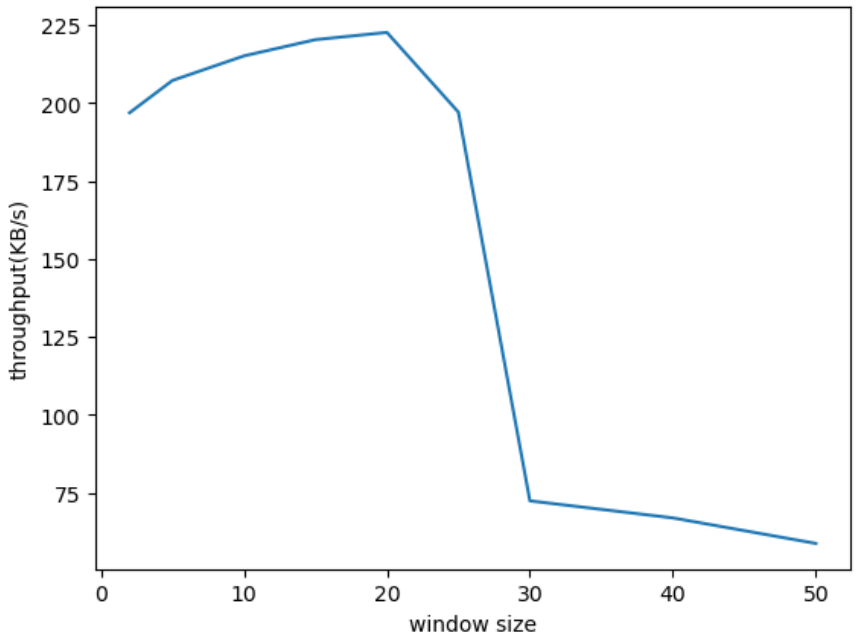

图 3.4: 滑动窗口机制中不同窗口大小文件传输吞吐率

根据图表可以看出，随着窗口逐渐增大，发送端可以发送多个包，发包效率提升，吞吐率也增加。当窗口数达到 20 左右，吞吐率达到顶峰，之后继续增大窗口，容易造成网络堵塞，由于接收端的缓冲区大小有限，一次性发送太多数据包会导致产生丢包现象，从而增大文件传输时延，吞吐率开始下降。可以看到，窗口大小在 25-30 左右，吞吐率发生了大幅度的下降。

3.3 有拥塞控制和无拥塞控制的性能比较

使用 reno 拥塞控制算法。这里我们固定窗口大小为 15，对文件 1.jpg 进行传输测试。分别改变丢包率和延时，比较无拥塞控制和有拥塞控制的算法的性能差异。结果如下所示。

3.3.1 不同丢包率，无延时

丢包率		0%	1%	3%	5%
时延 (s)	无拥塞控制	5.76	15.625	36.688	110.214
	有拥塞控制	4.831	13.149	27.028	40.629
吞吐率 (KB/s)	无拥塞控制	322.457	118.871	50.626	16.852
	有拥塞控制	384.545	141.254	68.72	45.715

表 4: 有拥塞控制和无拥塞控制不同丢包率下文件传输时延及吞吐率

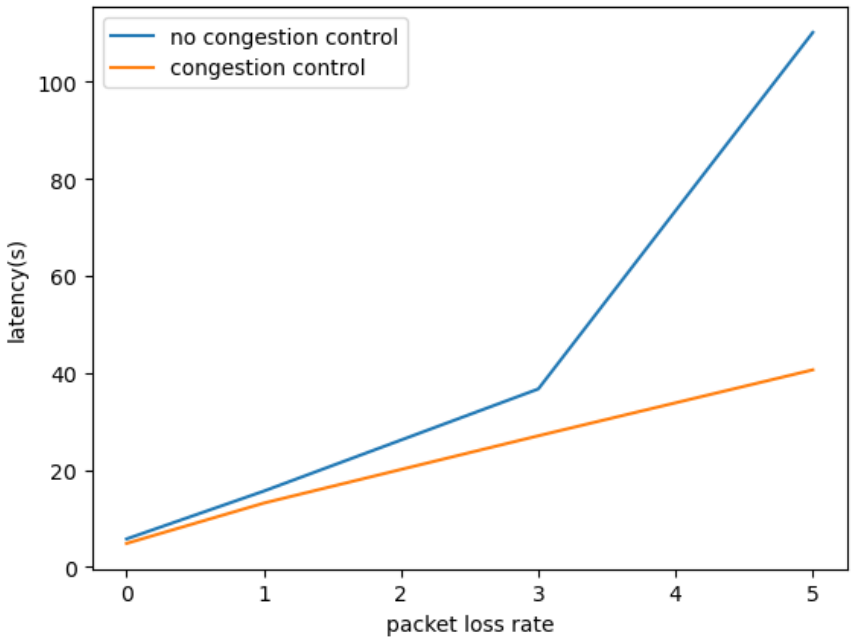

图 3.5: 有拥塞控制和无拥塞控制不同丢包率下文件传输时延

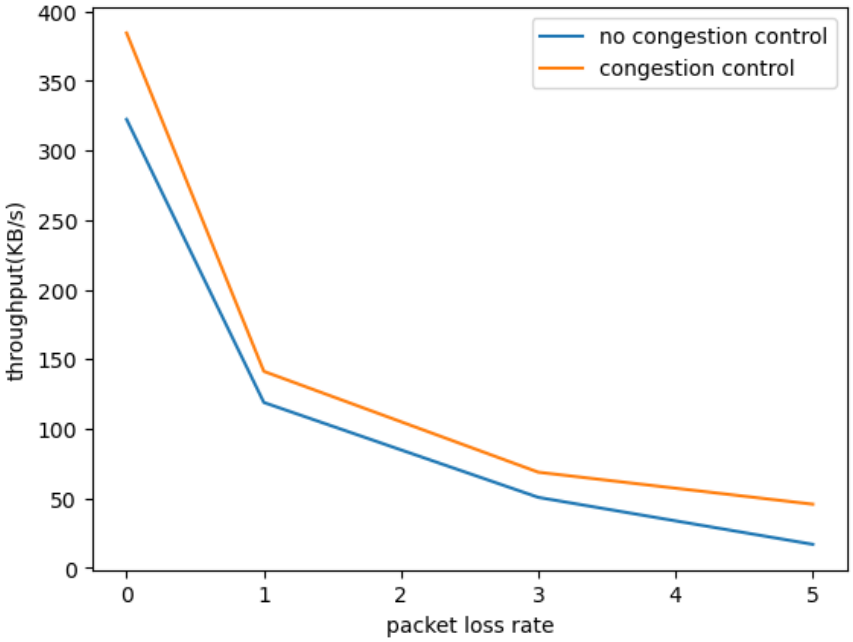

图 3.6: 有拥塞控制和无拥塞控制不同丢包率下文件传输吞吐率

从图3.5中我们可以看到，随着丢包率的增大，有拥塞控制和无拥塞控制的滑动窗口算法的文件传输时延都在增加，但我们可以明显的看到，有拥塞控制的滑动窗口算法的文件传输时延增加速率更慢。显然，这是由于拥塞控制算法在产生丢包时及时限制了窗口的大小，从而避免了更进一步的网络拥塞的产生。

进一步思考，当出现丢包的情况，无拥塞控制的滑动窗口只能等到触发超时才会重传报文，丢包率越高，吞吐率越低；而拥塞窗口机制在收到三次重复 ACK 会触发快速重传，或者超时的情况下重传数据包，因此拥塞控制机制在丢包的情况可以有更好的性能。综上，如果存在丢包的情况，有拥塞

控制机制要明显优于无拥塞控制机制的滑动窗口性能。

3.3.2 不同延时，丢包率 1%

延时 (ms)		0	1	3	5
时延 (s)	无拥塞控制	17.617	40.205	42.041	61.412
	有拥塞控制	13.149	35.689	35.787	36.491
吞吐率 (KB/s)	无拥塞控制	105.42959	46.197065	44.179563	30.244138
	有拥塞控制	141.25432	52.0427303	51.900215	50.898934

表 5: 有拥塞控制和无拥塞控制不同延时下文件传输时延及吞吐率

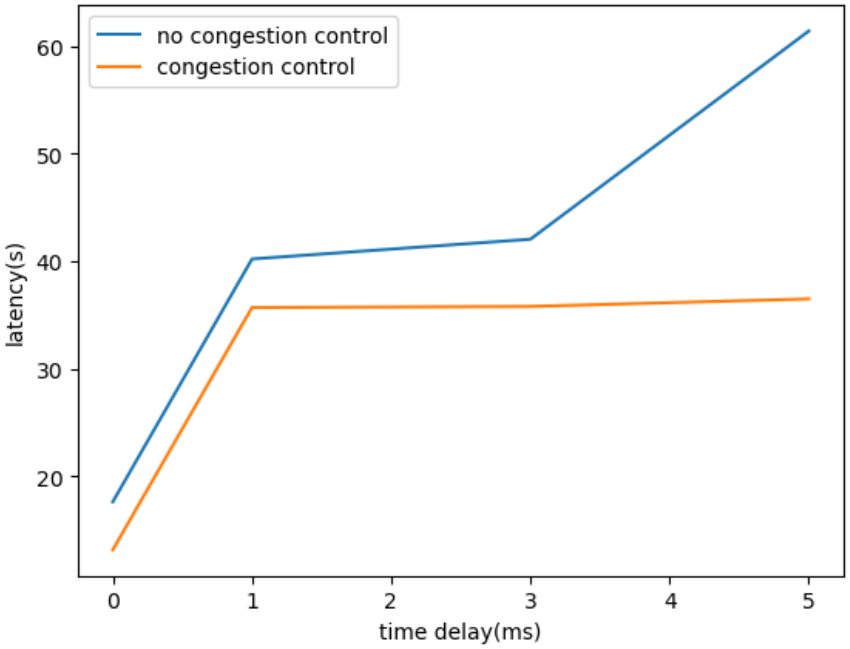

图 3.7: 有拥塞控制和无拥塞控制不同延时下文件传输时延

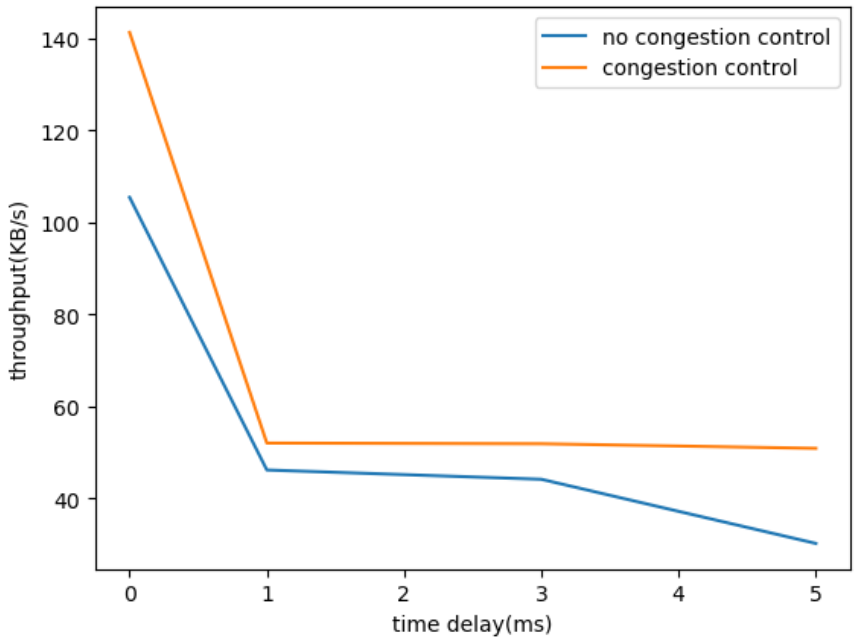

图 3.8: 有拥塞控制和无拥塞控制不同延时下文件传输吞吐率

分析图3.7和图3.8，对比延时为 0ms 和延时为 1ms 时文件的传输效率，可以发现延时对于文件传输的影响还是比较大的。随着延时增大，文件传输效率降低。当出现延时的情况，无拥塞控制机制的滑动窗口的窗口大小不变，每次都发相同数量的包，在延时的情况下，需要等待十分长的时间，容易造成超时发生；而拥塞控制机制在超时的情况下会调整窗口大小，将窗口重新设置为 1，从而限制了发送端的发送。因此延时变长对拥塞窗口机制的吞吐率影响小于对无拥塞控制机制的滑动窗口的影响。所以如果存在延时的情况，拥塞控制机制要优于无拥塞控制的滑动窗口机制。