南開大學
Nankai University

计算机网络 实验报告

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： 辛浩然

学号： 2112514

年级： 2021 级

学院：网络空间安全学院

班级： 信息安全、法学

实验要求

基于给定的实验测试环境，通过改变延时和丢包率，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（累积确认和选择确认两种情形）；
3. 滑动窗口机制中相同窗口大小情况下，累积确认和选择确认的性能比较。

停等机制与滑动窗口机制性能对比

停等机制与滑动窗口机制性能对比中传输2.jpg作为测试文件，主要包括两部分：

1. **固定无时延，改变丢包率**，记录传输时间和吞吐率；
2. **固定无丢包率，改变时延**，记录传输时间和吞吐率。

由于测试文件均为2.jpg，MSS不变，文件大小和传输的轮次相同，所以成功传输的数据量是相同的。因此性能对比**只需比较吞吐率**即可。

无时延条件下改变丢包率

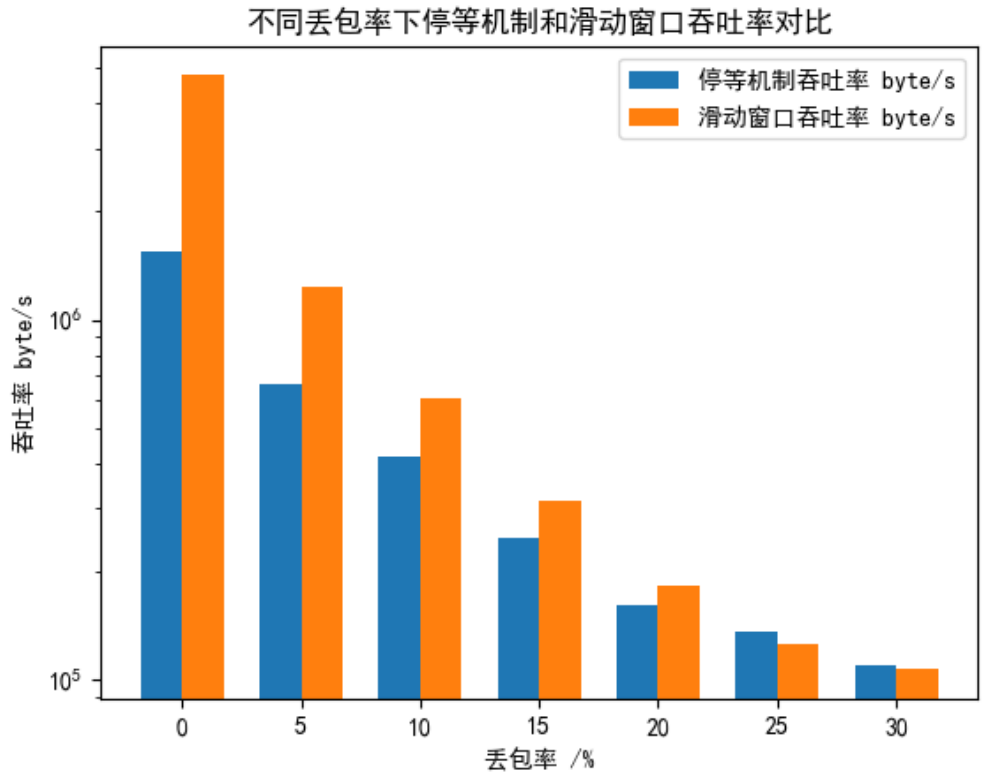
停等：在时延为0的条件下，在0~30%范围内以5%为梯度改变丢包率，使用停等机制传输2.jpg文件，测得传输时间和吞吐率如下表：

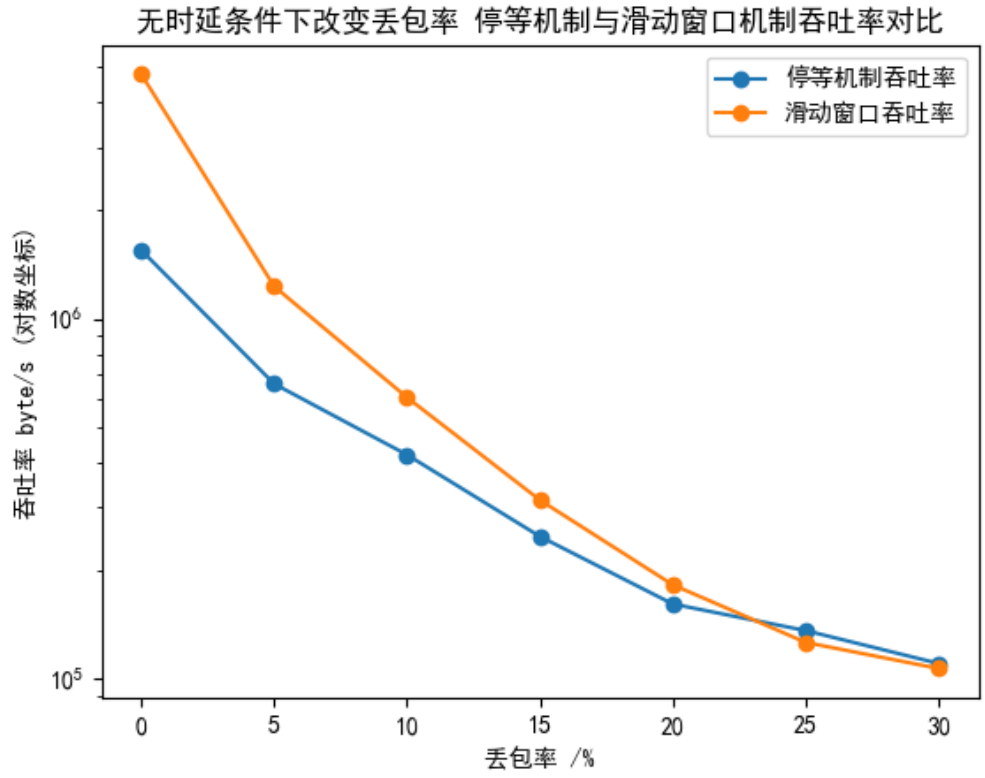
丢包率 /%	0	5	10	15	20	25	30
传输时间/s	3.805	8.94	14.072	23.72	36.522	43.29	53.412
吞吐率 byte/s	1552624.70	660820.69	419822.13	249061.43	161758.31	136468.86	110606.92

滑动窗口：在时延为0的条件下，在0~30%范围内以5%为梯度改变丢包率，使用滑动窗口机制传输2.jpg文件，累积确认，回退N步进行差错恢复，窗口大小为6，测得传输时间和吞吐率如下表：

丢包率 /%	0	5	10	15	20	25	30
传输时间/s	1.236	4.794	9.753	18.794	32.309	46.687	55.161
吞吐率 byte/s	4779722.49	1232318.94	605735.36	314341.65	182851.13	126539.23	107099.89

针对停等机制和滑动窗口机制的吞吐率，绘制柱状图和折线图（为了更好地展示数据的变化趋势，纵轴为对数坐标轴）：





在图中可以看出：

1. 随着丢包率的提高，停等机制和滑动窗口机制的传输吞吐率都有所下降，这是因为丢包率增加导致重传的频率提高，降低了有效的数据传输速率。
2. 滑动窗口协议在一定范围内对丢包率的适应性更好，吞吐率能够相对保持较高水平，但是随着丢包率的增加，滑动窗口协议的性能逐渐下降，甚至在较大的丢包率情况下不如停等机制。
    - 低丢包率情况下，滑动窗口机制的吞吐率更高，这是因为：
        - 停等机制每个分组都需要**等待确认**之后才能发送下一个分组，在等待确认的时间内，发送方无法继续发送新的数据分组，导致空闲时间较长；
        - 滑动窗口机制引入**流水线机制**，允许发送方在**等待确认之前发送多个数据包**，而不像停等机制那样每次只能发送一个数据包。这样，流水线机制能够更好地利用网络带宽，提高数据传输效率。
        - 此外，通过**累积确认**的方式允许多个数据包一起被确认，在接收端回复的某个ACK丢失的情况下，发送端收到之后的ACK可以进行累积确认，减少了重传次数。
    - 高丢包率情况下，滑动窗口机制的性能反而会不如停等机制：
        - 这是因为实验中滑动窗口采用的差错恢复是GBN，当丢包发生时，GBN需要**重传包括丢失的数据包在内的窗口内的所有数据包**，这导致了**额外的重传**。在低丢包率情况下，对传输吞吐率的影响较小；但随着丢包率的升高，额外重传对吞吐率的影响越来越大，使得性能大幅度下降，甚至不如停等机制。
        - 而停等机制的发送方只需重传当前丢失的数据包，不会导致之后的数据包重传。它一次只发送一个数据包，不会引起额外的重传。

无丢包条件下改变时延

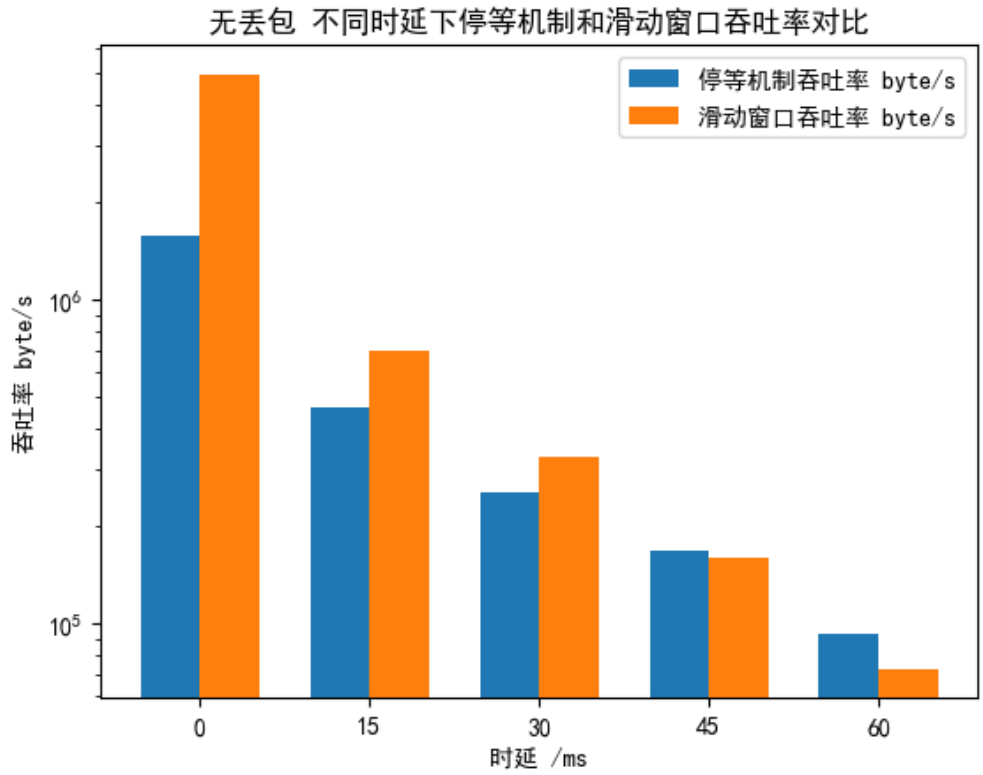
停等：在丢包率为0的条件下，在0~60ms范围内以15ms为梯度改变时延，使用停等机制传输2.jpg文件，测得传输时间和吞吐率如下表：

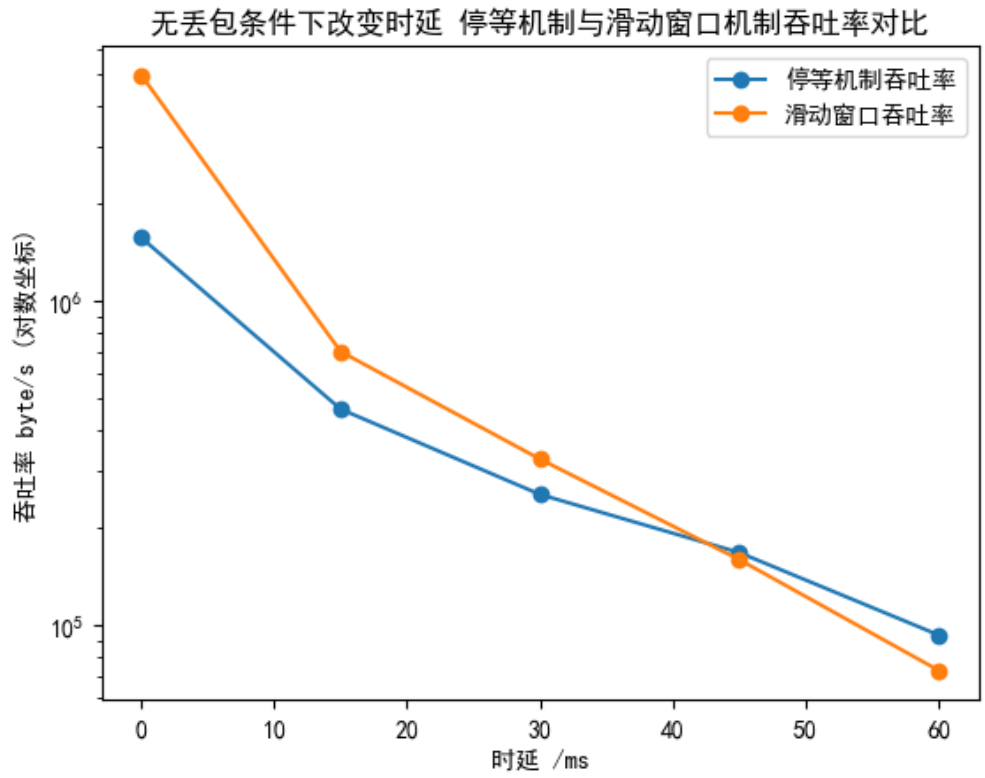
时延 /ms	0	15	30	45	60
传输时间/s	3.757	12.695	23.271	35.173	63.173
吞吐率 byte/s	1572461.27	465359.35	253866.92	167962.27	93516.8

滑动窗口：在丢包率为0的条件下，在0~60ms范围内以15ms为梯度改变时延，使用滑动窗口机制传输2.jpg文件，累积确认，回退N步进行差错恢复，发送端窗口大小为6，测得传输时间和吞吐率如下表：

时延 /ms	0	15	30	45	60
传输时间/s	1.183	8.435	18.101	36.976	81.206
吞吐率 byte/s	4993860.52	700383.76	326376.28	159772.2	72750.01

针对停等机制和滑动窗口机制的吞吐率，绘制柱状图和折线图（纵轴为对数坐标轴）：





随着时延的增加，停等机制和滑动窗口机制传输的吞吐率都在降低：

- 在较低时延时，吞吐率的主要影响原因是传输的延迟时间，数据包在传输中需要等待更长时间，使吞吐率下降；

    - 低时延滑动窗口机制的吞吐率更高，这同样是因为**流水线机制**下，允许发送方在**等待确认之前发送多个数据包**；而停等协议每发送一个数据包后需要等待确认，这会在时延增加时导致吞吐率的急剧下降。
- 但是随着时延的增加，一些数据包的确认报文到达发送端的时间超过了超时时间，会引发超时重传。而GBN会重传**窗口中的全部数据包**，这产生**不必要的超时重传**，这会使性能迅速下降，甚至不如停等机制。

滑动窗口机制中不同窗口大小对性能的影响

探究滑动窗口机制中不同窗口大小对性能的影响中，传输3.jpg作为测试文件。由于在不同程度的丢包率下，不同窗口大小对于性能的影响程度可能不同。因此，进行多组实验，每组实验固定0时延和特定丢包率，改变窗口大小，探究窗口大小对性能的影响。

累积确认

进行四组实验，每组实验**固定0时延和特定丢包率，改变窗口大小**，探究**累积确认**情形下窗口大小对性能的影响。

丢包率为0%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	3.081	2.123	1.976
吞吐率 byte/s	3886837.39	5640765.9	6060397.77

丢包率为6%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	10.946	10.108	9.675
吞吐率 byte/s	1094038.55	1184739.41	1237761.86

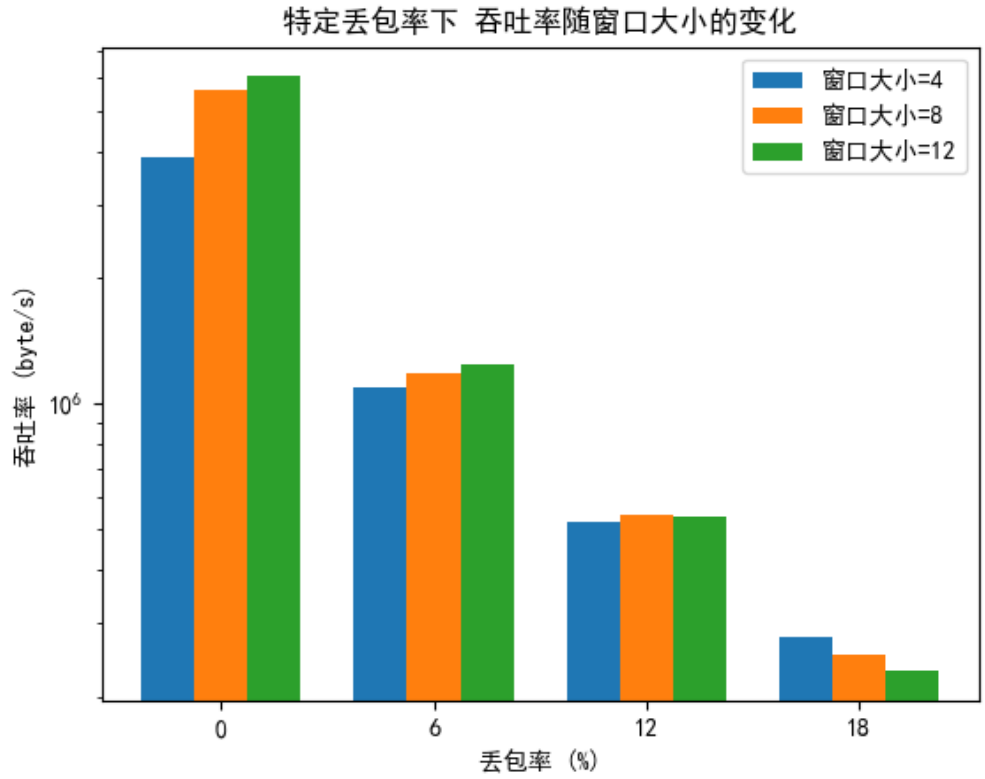
丢包率为12%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	22.823	22.073	22.175
吞吐率 byte/s	524705.17	542533.68	540038.15

丢包率为18%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	43.14	47.763	52.198
吞吐率 byte/s	277592.63	250724.33	229421.55

将上面四组结果绘制柱形图（纵轴为对数坐标轴）：



总得来说，窗口大小对性能的影响在不同丢包率下表现不同。在无丢包或低丢包率的情况下，**适度增大窗口大小可以提高性能**；而在**高丢包率的情况下，窗口大小的增加可能会导致性能下降**。

- 在无丢包或低丢包率的情况下，随着窗口大小的增加，更大的窗口允许不等待确认发送更多的数据包，从而更有效地利用网络带宽，而此时丢失的数据包数量相对较少，重传的代价影响不大，因此吞吐率逐渐增加。
- 但是可以看到，随着丢包率的增加，窗口增大对性能的增幅逐渐降低，甚至会导致性能的下降。在高丢包率的环境中，更多的数据包可能会在传输过程中丢失。由于每次丢包需要**重传当前窗口内的所有数据包**，因此更大的窗口意味着**重传不必要数据包的数量更多**，超时重传的开销更大，对性能产生负作用，使性能降低。

选择确认

进行四组实验，每组实验**固定0时延和特定丢包率，改变窗口大小**，探究**选择确认**情形下窗口大小对性能的影响。

丢包率为0%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	3.125	2.056	1.986
吞吐率 byte/s	3832110.72	5824584.63	6029882.18

丢包率为6%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	9.876	7.108	6.675
吞吐率 byte/s	1212570.47	1684770.12	1794059.33

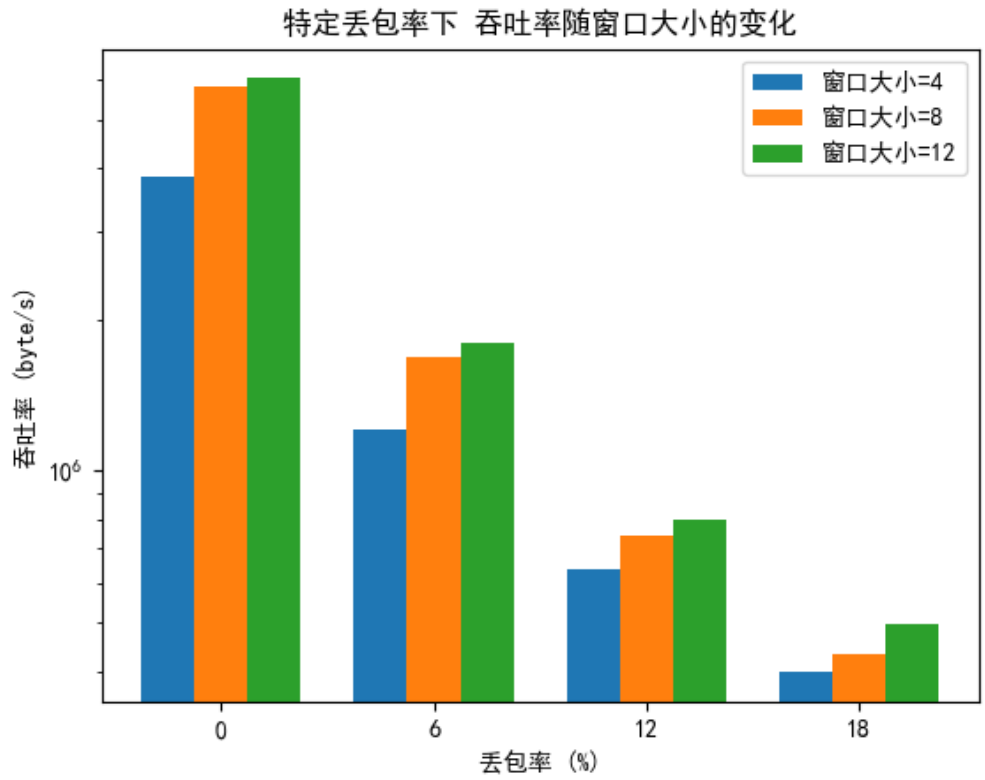
丢包率为12%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	18.823	16.073	14.975
吞吐率 byte/s	636208.15	745059.79	799689.22

丢包率为18%时，改变窗口大小：

窗口大小	4	8	12
传输时间/s	30.14	27.763	24.198
吞吐率 byte/s	397324.02	431341.93	494889.91

将上面四组结果绘制柱形图（纵轴为对数坐标轴）：



从图中可以看到，在不同丢包率下，**适当增加窗口大小都能有效提高传输的吞吐率**。

- 增大窗口大小可以提高吞吐率，同样是得益于流水线机制，在更大窗口下，可以不等待确认而发送更多的数据。
- 即使在较高丢包率下，窗口大小的增加仍然有助于提高性能，因为它**不会产生不必要的重传**。

累积确认和选择确认的性能对比

累积确认和选择确认的性能对比中传输3.jpg作为测试文件。

固定窗口大小：GBN协议的发送端窗口大小为8，SR协议的发送端和接收端窗口均为8。

主要包括两部分：

1. **固定无时延，改变丢包率**，记录传输时间和吞吐率；
2. **固定无丢包率，改变时延**，记录传输时间和吞吐率。

由于测试文件均为3.jpg，MSS不变，文件大小和传输的轮次相同，所以成功传输的数据量是相同的。因此性能对比**只需比较吞吐率**即可。

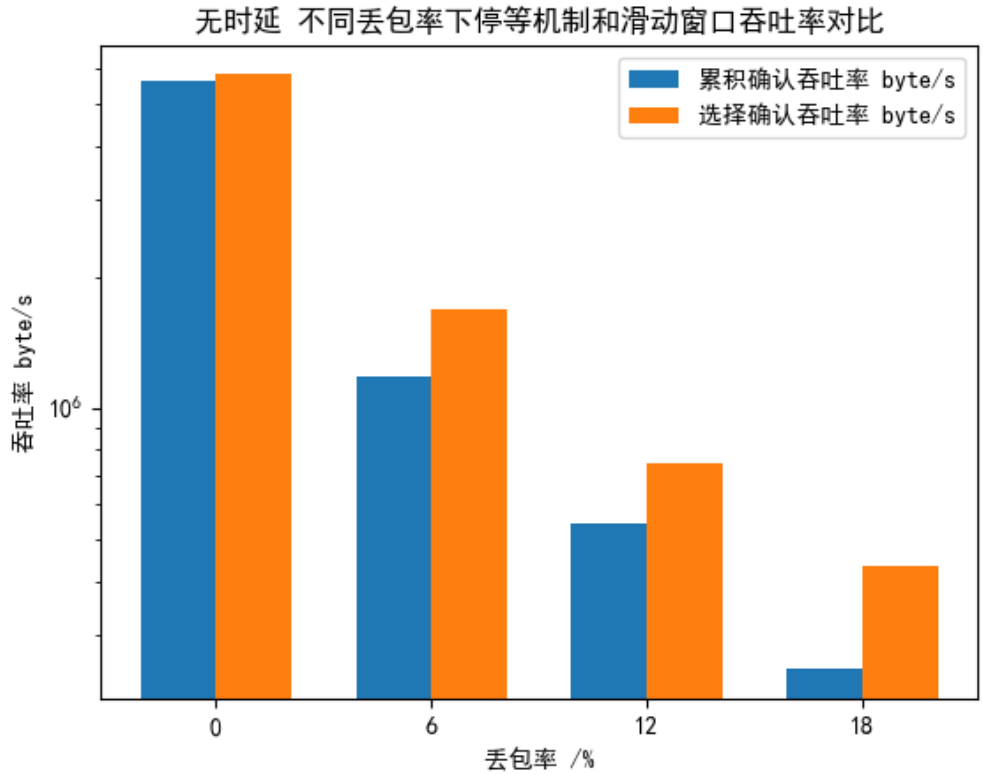
无时延条件下改变丢包率

在时延为0的条件下，在0~18%范围内以6%为梯度改变丢包率，分别使用GBN协议和SR协议传输3.jpg文件，测得吞吐率如下表：

实验结果如下：

丢包率 /%	0	6	12	18
累积确认吞吐率 byte/s	5640765.9	1184739.41	542533.68	250724.33
选择确认吞吐率 byte/s	5824584.63	1684770.12	745059.79	431341.93

绘制柱形图（纵轴为对数坐标轴）：



随着**丢包率的增加，二者吞吐率都呈下降趋势**。这是因为丢包导致了更多的重传，影响了整体的数据传输效率。

无丢包时二者性能差距不大。在丢包情况下，**选择确认的性能明显优于累积确认**，并且随着丢包率的增加，**选择确认的优势更加明显**。这是因为当发生丢包时，累积确认机制下，需要重传当前窗口内的所有数据包，引入了不必要的重传，尤其是在丢包率较高时，产生了大量的重传开销；而选择确认机制下，只需重传丢失的某个数据包，**避免了不必要的重传**。

无丢包情况下改变时延

在丢包率为0的条件下，在0~60ms范围内以15ms为梯度改变时延，传输3.jpg文件，测得传输时间和吞吐率如下表.

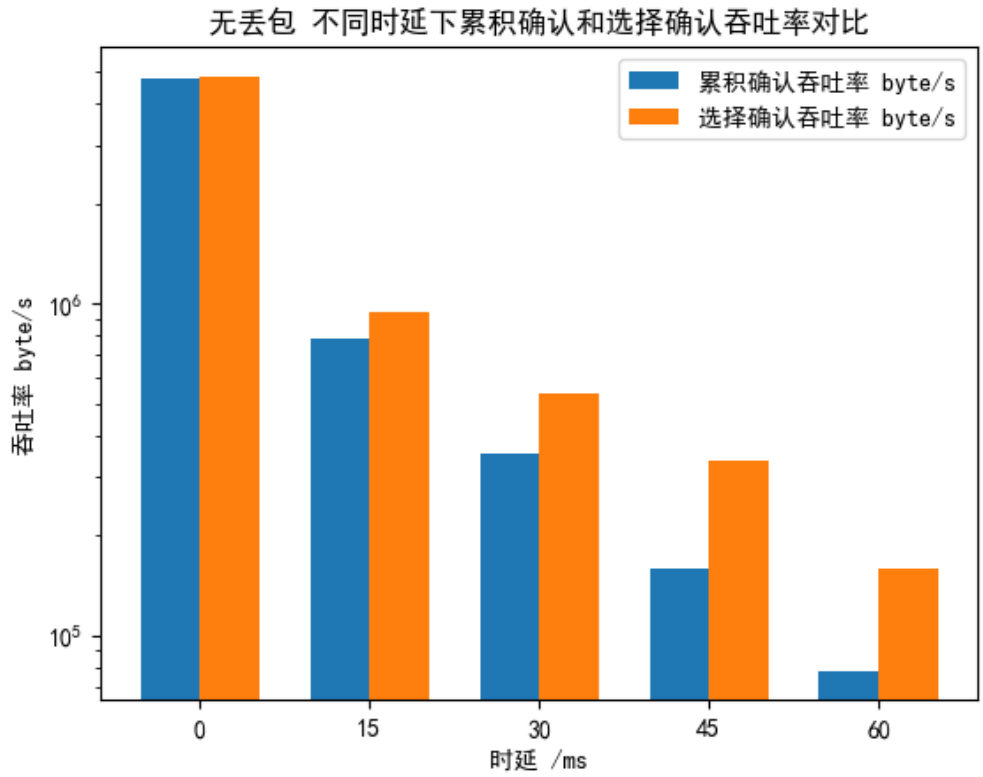
累积确认：

时延 /ms	0	15	30	45	60
传输时间/s	2.5138	15.172	33.804	74.924	152.684
吞吐率 byte/s	4763829.53	789322.13	354261.4	159832.38	78432.25

选择确认：

时延 /ms	0	15	30	45	60
传输时间/s	2.467	12.691	22.407	35.47	75.641
吞吐率 byte/s	4854764.76	943631.39	534451.58	337615.91	158317.43

绘制柱形图（纵轴为对数坐标轴）：



随着时延的增加，**吞吐率都在降低**，在较低时延时，数据包在传输中需要等待更长时间，使吞吐率下降；

- 在低时延时，吞吐率的主要影响原因是**传输的延迟时间**，二者性能差距不大。
- 随着时延的增加，一些数据包的确认报文到达发送端的时间超过了超时时间，会引发超时重传。这时**选择确认性能明显高于累积确认**，并且**优势逐渐拉大**。这同样是二者重传数据包的机制不同。GBN会重传**窗口中的全部数据包**，这产生**不必要的超时重传**，而选择确认机制下，只需重传丢失的某个数据包，**避免了不必要的重传**。