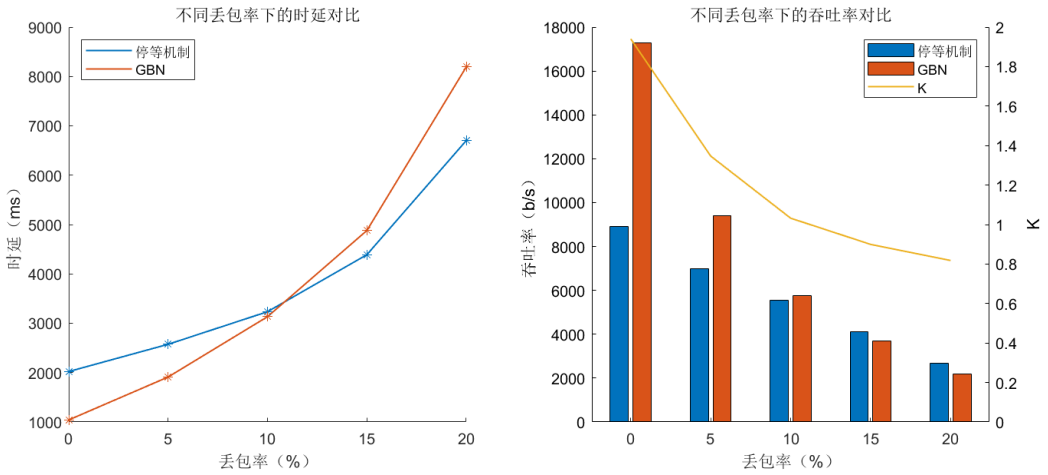
停等机制和滑动窗口性能对比

丢包率影响

在本次实验中，通过固定延迟时间为0，调整丢包率大小，来对比了停等机制和滑动窗口的性能差异。分别测量了时延和丢包率，绘制出下图。



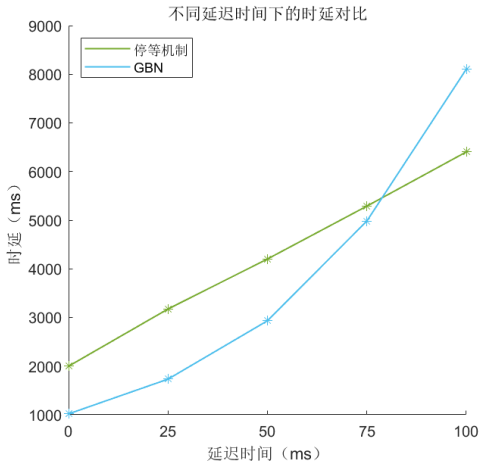
从图像中可以看出，当丢包率较小的时候，GBN的性能要优于滑动窗口，并且通过吞吐率的比值可以看到，在丢包率为0的时候，GBN的吞吐率大约是停等机制的2倍。在实验测量的时候，选取的滑动窗口大小为5，由于流水线机制，如果不考虑上层应用的其余开销的话，理论上GBN的吞吐量能够达到停等机制的5倍，但是在实验中，由于存在端端延迟以及文件读写和处理等操作，所以加速比并没有达到理论值。

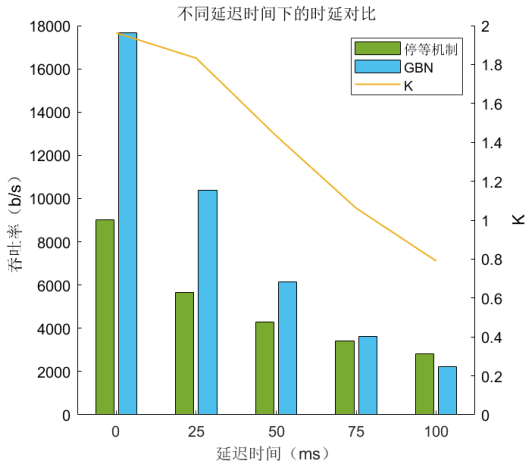
随着丢包率的增加，GBN的性能逐渐下降，并且在较大的丢包率的情况下，GBN的性能甚至不如停等机制。这个原因也很容易分析，由于丢包率较大，因此在GBN中会出现大量的重复ACK，导致超时重传。重传的时候会将窗口中的所有数据包全部重新发送，一方面会占用带宽资源，另一方面很有可能再次出现丢包。此外，由于GBN必须要顺序确认数据包，因此当窗口中的一组数据包中第一个数据包发生丢失的时候，后续到达的其他数据包也将被丢弃，使得原本只需要重传第一个数据包变成了所有数据包都为被确认，因此在一定程度上也严重影响了性能。

因此我们可以得出结论，当丢包率较小的时候，由于流水线和滑动窗口的存在，GBN的性能优于停等机制。而当丢包率较高的时候，GBN由于存在较大的重传开销，导致性能下降，此时GBN的性能不如停等机制。

延迟时间影响

在本次实验中，通过固定丢包率为0，调整延迟时间大小，来对比了停等机制和滑动窗口的性能差异。分别测量了时延和丢包率，绘制出下图。





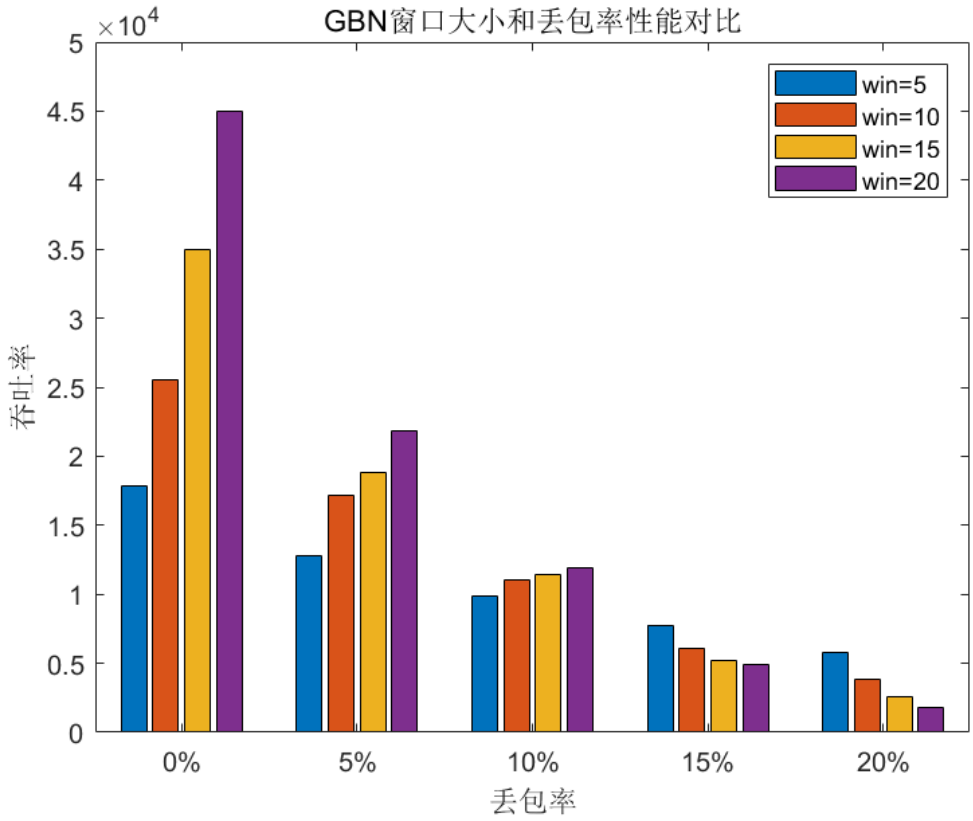
从图像中而已看出，当延迟时间较小的时候，GBN的性能要优于停等机制，并且在延迟时间小于50ms的时候，GBN和停等机制的实验变化大约都是线性的。这是因为在本次实验中，并没有设置丢包，因此所有的数据包都会正常到达，并且RTO设置为50ms，因此时延的差异主要就是由传输的延迟时间导致的，而这个变化是线性的，因此时延变化也大约为线性。并且由于很小概率触发超时重传，因此GBN也几乎不存在超时重传的额外开销。

而当延迟时间超过了50ms之后，两边都会产生超时重传，但是对于超时重传，停等机制的影响不大，GBN则会产生不必要的超时重传，并且会重传窗口中的全部数据包，这会导致较大的额外开销影响性能，因此可以看出当延迟时间超过50ms之后，GBN的性能迅速下降，甚至不如停等机制。

因此可以得出结论，当延迟时间较小的时候，由于流水线机制GBN性能明显优于停等机制。但当延迟时间超过了设定的RTO的时候，GBN就会产生大量的不必要重传，严重影响性能。

滑动窗口机制中不同窗口大小对性能的影响

在实验中，通过控制变量的方式，探究了不同窗口大小对于滑动窗口机制性能的影响。并且额外探究了在不同丢包率下的影响差异，绘制实验结果如图。



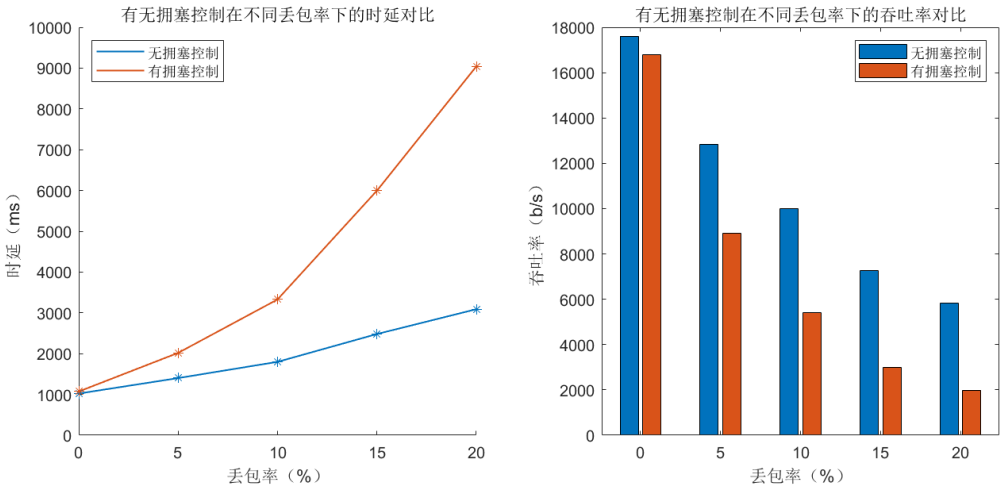
从图像中可以看出，窗口大小对于吞吐量的影响情况会受到丢包率的影响。当丢包率较小的时候，吞吐量随着窗口大小的增加而提升，这是很好理解的。由于丢包率较小，因此影响GBN的非必要重传很少发生，因此GBN的性能由于流水线机制会随着窗口大小的增加而提升。

而随着丢包率的上升，窗口大小带来的收益逐渐反转。可以看到当丢包率达到20%的时候，窗口越大，性能越差，这也很好理解。原因是丢包率较大，会产生大量超时重传，而窗口越大，超时重传的开销越大，因此性能越差。

有无拥塞控制的性能比较

丢包率影响

在本次实验中，通过固定延迟时间为0，调整丢包率大小，来对比了无拥塞控制和有拥塞控制的性能差异。分别测量了时延和丢包率，绘制出下图。

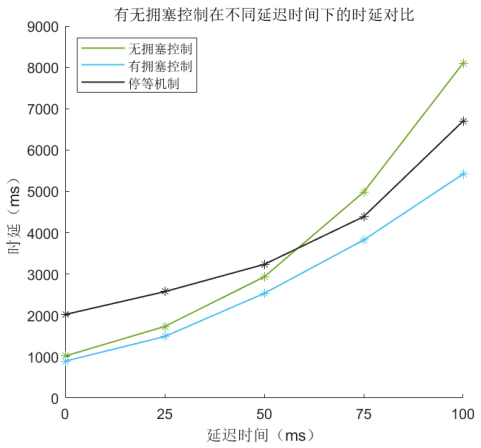


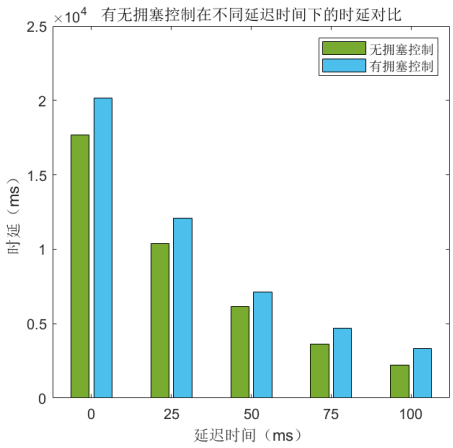
从图像中可以看到，增加拥塞控制之后，性能会收到很大影响，这在直观上并不好理解，我们进行简要分析。在实验测试的时候，固定了延迟时间为0，因此绝大部分的超时重传都是由于丢包导致的。对于无拥塞控制的算法，超时后会重传窗口中所有未被确认的数据包，但是并不会因此更改窗口大小，因此在后续还能够保持一个较大的吞吐量。而对于有拥塞控制的算法，如果遇到丢包，三次重复ACK会提前超时，此时窗口大小近似折半，如果遇到超时重传，窗口大小会被直接置为1，此时会较大影响吞吐量。虽然慢启动阶段和拥塞避免阶段会增加窗口大小，但是通过实验观察可以发现，窗口大小并不能够在较大值保持很长时间，导致平均窗口大小要小于无拥塞控制的算法。

并且随着丢包率不断增加可以看出有拥塞控制的算法的性能会快速下降，这就是由于经常会出现重复ACK导致窗口不断折半，当丢包率较大的时候，窗口大小几乎保持在1左右进行波动，这回严重影响性能。

延迟时间影响

在本次实验中，通过固定丢包率为0，调整延迟时间大小，来对比了无拥塞控制和有拥塞控制的性能差异。分别测量了时延和丢包率，绘制出下图。





为了能够对于拥塞控制机制的性能提升，在实验中额外和停等机制进行了对比。由于在实验测定的时候将丢包率设置为零，因此在实验过程中不会出现重复ACK，因此也不会提前超时，只会在慢启动和拥塞控制之间进行跳转。

可以看到当延迟时间较小的时候，有拥塞控制和无拥塞控制的性能相近，并且由于流水线机制的存在，两者的性能都要优于停等机制。而随着延迟时间的增加，到超过RTO=50ms之后，就会发生超时重传。无拥塞控制的算法超时的时候会重传窗口中所有未被确认的报文，但是其实这完全是没有必要的，而这次重传会导致一定的性能损失。而有拥塞控制的算法则会在超时之后调整窗口的大小，减小由于超时重传产生的额外开销。可以看到，随着延迟时间的增加，无拥塞控制的算法甚至性能已经比停等机制差了，而有拥塞控制的算法则会稍微优于停等机制。