南開大學
Nankai University

计算机学院

计算机网络实验报告

实验 3-4: 基于 UDP 服务设计可靠传输协议并编程实现

姓名：谢畅

学号：2113665

专业：计算机科学与技术

2023 年 12 月 23 日

目录

1 实验要求与实验环境概述

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（累计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，累计确认和选择确认的性能比较

其中实验环境，采取 MSS:8016 的固定值进行以下实验，在 x64 下 windows11 的 visual studio 上进行测试。

2 停等机制与滑动窗口机制性能对比

2.1 设计思路

在这里我们分别针对延时、丢包两个变量进行测试，对比停等机制与滑动窗口的性能上的差异，这里性能的指标：传输时延与吞吐率。

2.2 性能测试

2.2.1 针对延时进行比对

这里设置路由器延时为单一变量。滑动窗口的窗口大小为 10，路由器丢包率设置 2%，程序中的 timeout 设定为 500ms。

1. 性能：传输时间

在这里，我们针对路由器的延时作单一变量的实验，对比停等机制、滑动窗口机制的性能。性能上采用传输延时指标。有如下结果。

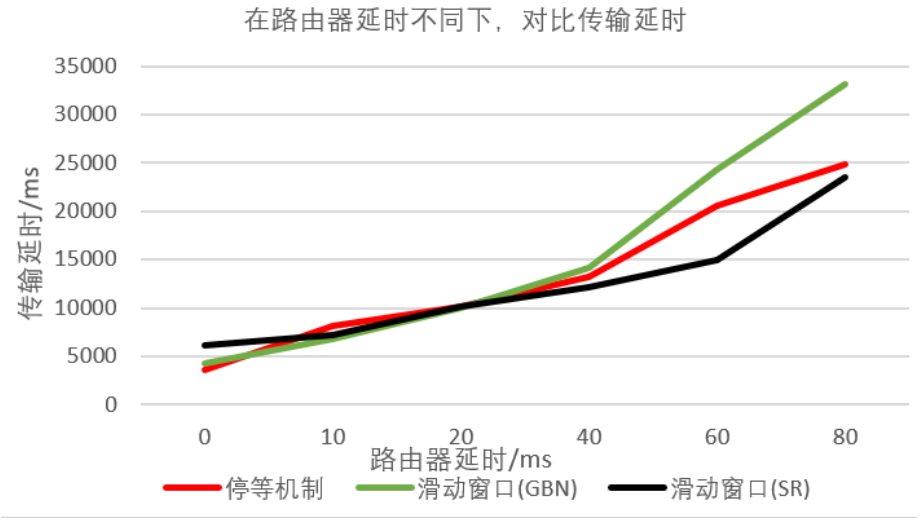

图 2.1

机制 \ 延时时间 (ms)	0	10	20	40	60	80
停等机制	3578	8094	10172	13234	20609	24875
滑动窗口 (GBN)	4219	6844	10078	14109	24328	33157
滑动窗口 (SR)	6204	7156	10141	12125	15000	23562

表 1: 在路由器延时不同下，对比传输延时 (单位:ms)

- **分析**：由表1、图2.1可以发现，滑动窗口 GBN、SR 在一开始 0-20ms 延时下实际上优于停等机制；在 40-80ms 下滑动窗口 GBN 性能最差，传输延时最大，而滑动窗口 SR 最优，传输延时最小，最后停等机制的性能处于两者之间。

- **原因**：滑动窗口 GBN 协议，在路由器延时较大时，每次都要将发送缓存区中的包全部重发，所以传输延时反而比停等协议更长。而滑动窗口 SR 协议都是针对特定的超时包进行重发，因此效率会最高。在延时较小时，停等机制最差；而在延时处于 40-80ms 时，滑动窗口 GBN 最差，滑动窗口 SR 传输延时最小，而停等机制的性能介于两者之间。

2. 性能：传输吞吐率

机制 \ 延时时间 (ms)	0	10	20	40	60	80
停等机制	4152.83	1835.78	1460.76	1122.78	720.99	597.34
滑动窗口 (GBN)	3521.88	2171.07	1474.38	1053.15	610.77	448.135
滑动窗口 (SR)	2395.04	2076.41	1465.22	1225.47	990.59	630.627

表 2: 在路由器延时不同下，对比传输吞吐率 (单位:kbps)

- **分析**：由表2、图2.2可以发现，滑动窗口 GBN、SR 在一开始 10-20ms 延时下的传输吞吐率实际上优于停等机制；在 40-80ms 下滑动窗口 GBN 性能最差，传输吞吐率最大，而滑动窗口 SR 最优，传输吞吐率最小，最后停等机制的性能处于两者之间。

- **原因**：滑动窗口 GBN 协议，在路由器延时较大时，每次都要将发送缓存区中的包全部重发，所以传输吞吐率反而比停等协议更长。而滑动窗口 SR 协议都是针对特定的超时包进行重发，而且同时发送多个包，因此效率会最高。

  这里路由器延时为 0，实际上停等机制性能最好，可能是因为滑动窗口相关协议的多线程所导致，也可能与路由器有关，也可能是因为延时 0 实际上停等的情况也就不存在。

  在路由器延时 10-20 时，停等机制最差，滑动窗口由于发送多个包，性能会较好；而在路由器延时处于 40-80ms 时，滑动窗口 GBN 最差，因为重发的包影响效率；滑动窗口 SR 传输吞吐率最小，由于并行发送而且只会重传特定包，所以传输吞吐率最高，而停等机制的性能介于两者之间。

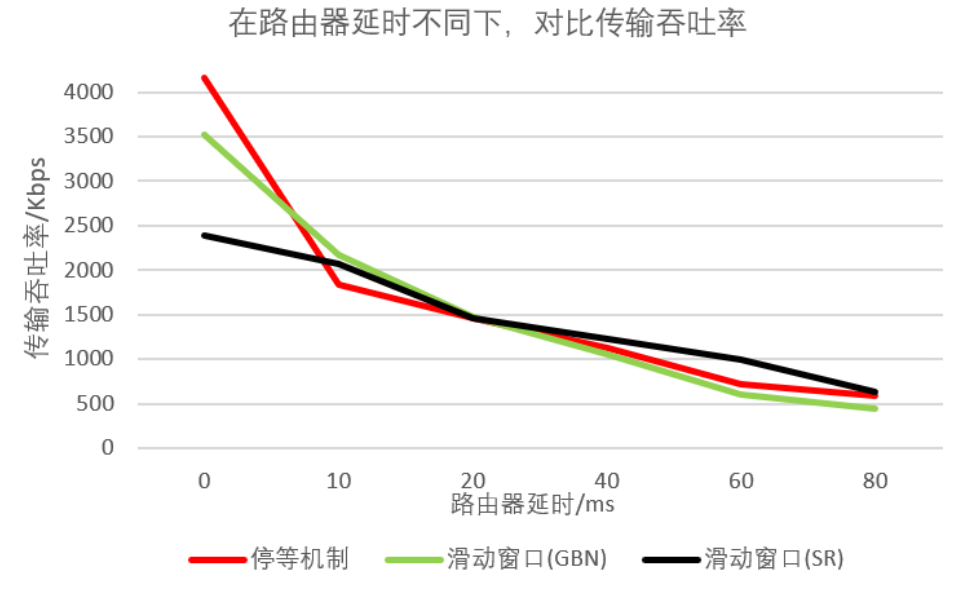

图 2.2

2.2.2 针对丢包进行比对

这里设置路由器丢包率为单一变量。滑动窗口的窗口大小为 10，路由器传输延时设置 20ms，程序中的 timeout 设定为 500ms。

1. 性能：传输时间

机制 \ 丢包率	0%	2%	4%	6%	8%	10%
停等机制	7172	9250	13875	17484	23532	26110
滑动窗口 (GBN)	7312	10735	17547	31563	87453	36063
滑动窗口 (SR)	7375	8766	9625	10391	12735	8735

表 3: 在路由器丢包率不同下，对比传输时延 (单位:ms)

- **分析**：由表3、图2.3可以发现，随着丢包率的变大，滑动窗口 SR 的传输延时最小，滑动窗口 (GBN) 的传输延时最大，而停等机制介于两者之间。所以其实滑动窗口 SR 的性能最好，滑动窗口 GBN 的性能最差。
- **原因**：路由器发生丢包时，滑动窗口 GBN 的协议，会重发整个窗口内的文件，而接收端只能接受 1 个包，所以造成性能很差；而滑动窗口 SR 协议，在发生丢包时，接收端仍然可以接受在接收窗口内的包，同时只会重发超时的特定包，所以性能最好。

  所以，滑动窗口 SR 的传输延时会表现的最小；而滑动窗口 GBN 的传输延时会表现的最大；而停等机制的性能介于两者之间。

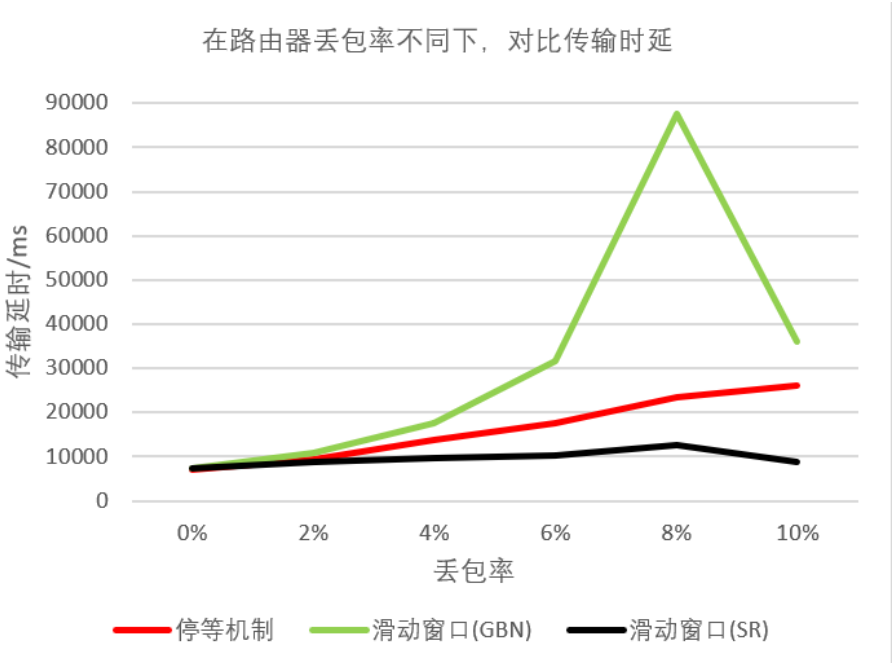

图 2.3

2. 性能：传输吞吐率

机制 \ 丢包率	0%	2%	4%	6%	8%	10%
停等机制	2071.78	1606.36	1070.91	849.853	631.43	569.09
滑动窗口 (GBN)	2032.11	1384.15	846.801	470.767	169.906	412.024
滑动窗口 (SR)	2014.76	1695.05	1543.77	1429.97	1166.77	1701.07

表 4: 在路由器丢包率不同下，对比传输吞吐率 (单位:kbps)

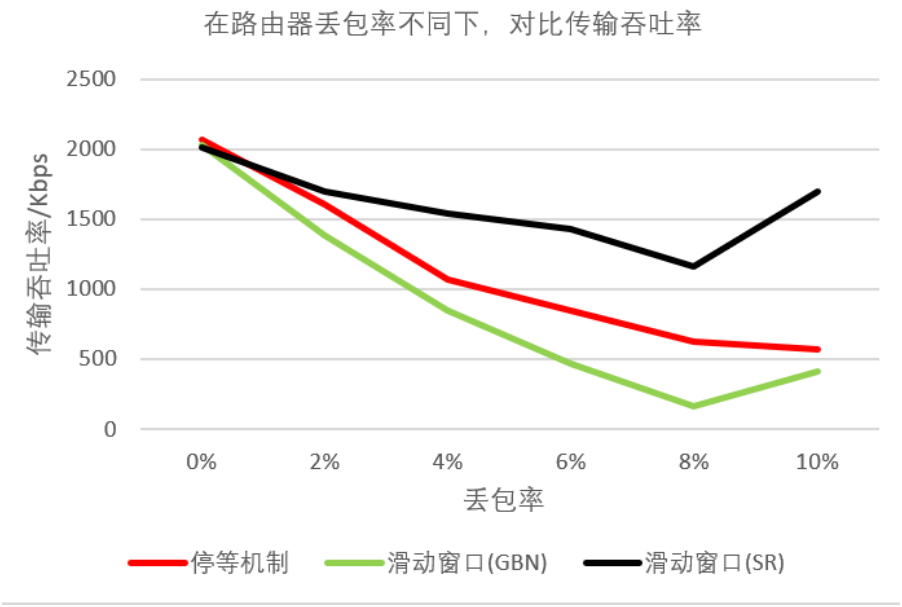

图 2.4

- **分析**：由表4、图2.4可以发现，随着丢包率的变大，滑动窗口 SR 的传输吞吐率最大，滑动窗口 (GBN) 的传输吞吐率最小，而停等机制介于两者之间。所以其实滑动窗口 SR 的性能最好，滑动窗口 GBN 的性能最差。

- **原因**：路由器发生丢包时，滑动窗口 GBN 的协议，会重发整个窗口内的文件，而接收端只能接受 1 个包，所以造成性能很差；而滑动窗口 SR 协议，在发生丢包时，接收端仍然可以接受在接收窗口内的包，同时只会重发超时的特定包，所以性能最好。

  同时这里发现在丢包率为 10% 时，滑动窗口 SR 的性能居然会比之前的还好，这里猜测是因为路由器丢包率与窗口大小的关系。由于丢包率为 10% 而窗口大小是 10，所以每次只会丢掉窗口的第一个包，而在丢包率为其他值时，每次丢失的包不是窗口的特定位置的包。而只丢掉特定位置的包，显然性能会更好，所以造成了这种现象。同时，可以发现滑动窗口 GBN 也有这种现象，无论是在图2.3还是图2.4中。

3 滑动窗口机制中不同窗口大小对性能的影响

3.1 设计思路

针对滑动窗口 GBN 与滑动窗口 SR 进行实验，此节中重点关注不同窗口大小变量，同时在不同网络情况下进行测试，所以会设置相关路由器延时，路由器丢包。由于关注的是不同窗口大小，所以部分地方只会进行 GBN/SR 的测试。

3.2 性能测试

3.2.1 延时与丢包率为定值

设置延时 0，丢包率 0，程序的 timeout 设置为 2ms。有如下测试结果

机制 \ 窗口大小	2	5	10	15	20	30
滑动窗口 (GBN)	1500	1516	1531	1407	1156	1438
滑动窗口 (SR)	1531	1422	1681	1735	1625	1547

表 5: 对比不同滑动窗口大小对应的传输时间 (单位:ms)

机制 \ 窗口大小	2	5	10	15	20	30
滑动窗口 (GBN)	1500	1516	1531	1407	1156	1438
滑动窗口 (SR)	1531	1422	1681	1735	1625	1547

表 6: 对比不同滑动窗口大小对应的传输吞吐率 (单位:kbps)

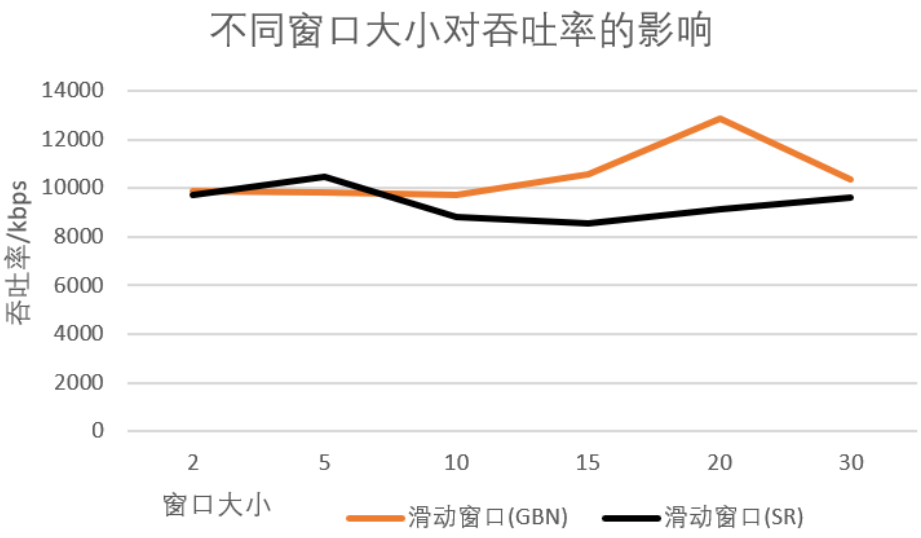

图 3.5

- **分析**：由表5、表6、图3.5，我们可以发现：在无延时、无丢包的情况下，随着窗口大小的增加，GBN 与 SR 的性能首先都得到了一定提升，而在窗口继续增大后，性能开始下降。

  SR 的峰值到达的比较早，窗口大小为 5 就达到了性能的最佳情况；而 GBN 峰值到达的比较晚，在窗口大小为 20 时性能最佳。

- **原因**：当窗口逐渐增大时，发送端可以发送多个包，发包效率提升，吞吐率也增加，当达到一个点时，滑动窗口的性能就会开始下降。而这个点受网络情况 (延时、丢包) 的影响。

  在无延时、无丢包时，滑动窗口 GBN 的峰值比较晚，这是因为 GBN 在没用超时重传时，性能较好，不会进行过多的重传；而 SR 中 ack 确认是针对一个包而言，因此发送的 ack 会比较多，在窗口较大时可能由于 ack 包的增多，导致网络拥挤，出现 ack 丢失等等情况，所以导致在窗口较小时达到了一个峰值。

3.2.2 丢包率变化

1. 针对 GBN

路由器延时 0，程序的 timeout 设置为 100ms

窗口大小 \ 丢包率	0%	2%	5%	10%
2	1344	1344	1438	1406
5	1593	1578	2078	1610
10	2000	1828	1969	1782
15	2250	1781	1813	1973

表 7: 对比不同丢包率、不同窗口大小对 GBN 的传输时间影响 (单位:ms)

窗口大小 \ 丢包率	0%	2%	5%	10%
2	11055.7	11055.7	10333	10568.2
5	9327.57	9416.24	7150.54	9229.08
10	7429.41	8128.46	7546.38	8338.29
15	6603.92	8342.97	8195.71	7671.05

表 8: 对比不同丢包率、不同窗口大小对 GBN 的传输吞吐率影响 (单位:kbps)

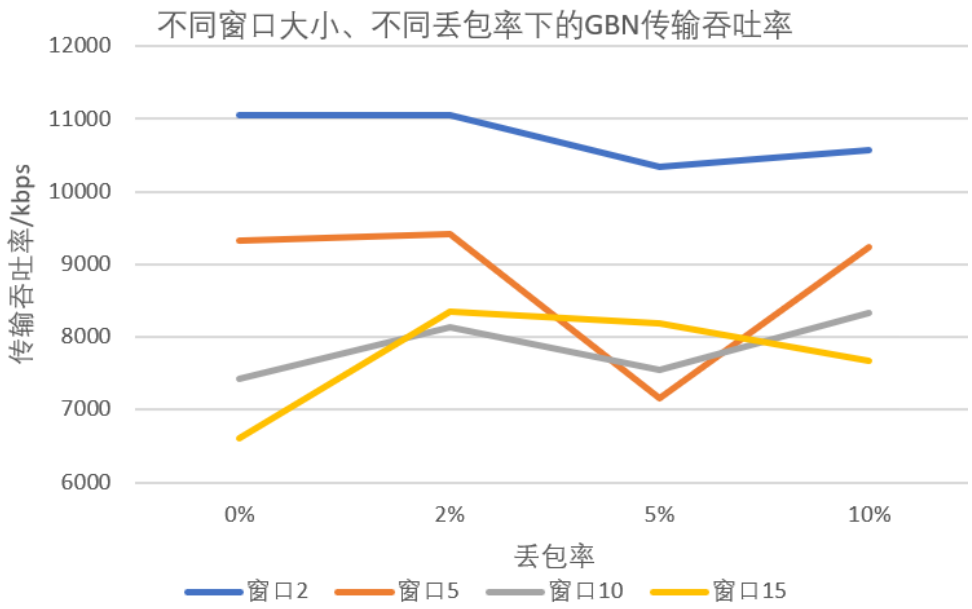

图 3.6

- **分析**：由表7、表8、图3.6，我们可以发现：GBN 在有丢包率的情况下，窗口较小时性能反而更优异，而随着窗口大小的增加以及丢包率的增加，往往性能会比窗口更小的更差。

- **原因**：如果一个包丢失，那么在 GBN 发送缓冲区中后面发送的包都是无意义的，所以窗口越大，一旦数据报丢失，要重发的数据报就越多，这就导致了窗口变大，性能反而下降。所以在有丢包率的情况下，往往是窗口较小的性能更佳，比如在图3.6，我们可以直观发现窗口 2 的性能最佳。

2. 针对 SR

路由器延时 0。设置程序的 timeout 为 100ms，放大性能的差异

窗口大小 \ 丢包率	0%	2%	5%	10%
2	1547	9657	12891	18610
5	1641	2813	3718	4562
10	1625	1906	2094	2672
15	1515	1375	1765	2141

表 9: 对比不同丢包率、不同窗口大小对 SR 的传输时间影响 (单位:ms)

窗口大小 \ 丢包率	0%	2%	5%	10%
2	9604.93	1538.66	1152.65	798.432
5	9054.74	5282.2	3996.46	3257.09
10	9143.89	7795.82	7095.9	5560.94
15	9807.8	10806.4	8418.6	6940.13

表 10: 对比不同丢包率、不同窗口大小对 SR 的传输吞吐率影响 (单位:kbps)

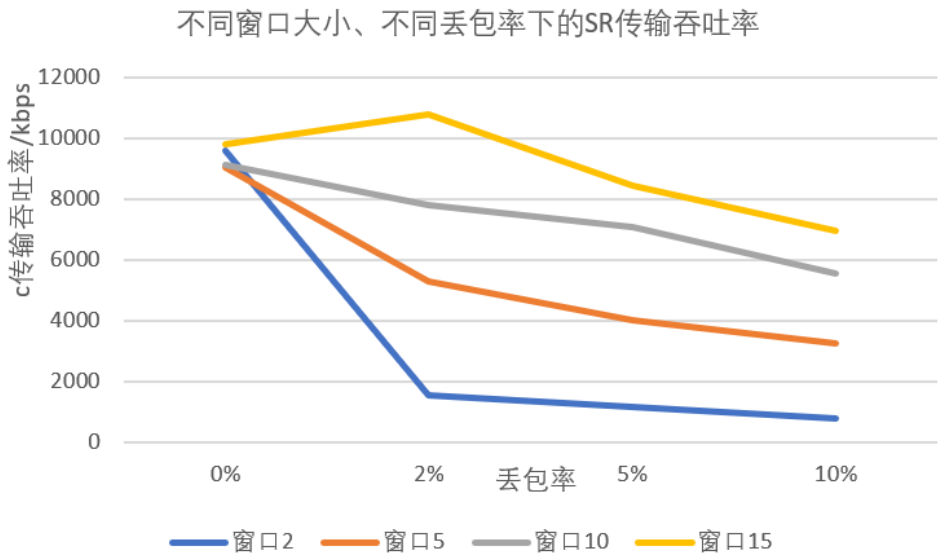

图 3.7

- **分析**：由表9、表10、图3.7，我们可以发现：SR 在有丢包率的情况下，窗口适当的更大，性能会更好。在图3.7可以看出，窗口 15 的性能远远强于其他窗口大小。
- **原因**：在 SR 选择重传中，我们是对每个包进行计时，只会重传那些被丢弃的包，因此如果窗口越大，实际上可以乱序缓存的包也就越多，这会导致阻塞的时间越短。而如果窗口较小，实际上马上就会阻塞，等待着窗口中的包重传，显然性能会较差。

3.2.3 延时变化

1. 针对 GBN

设置丢包率为 0，程序的 timeout 设定为 100ms

窗口大小 \ 延时 (ms)	2	5	10	20
2	3625	3625	3641	7266
5	3641	3656	3625	10265
10	3657	3656	3656	18324
15	7766	7594	7687	20125

表 11: 对比不同延时、不同窗口大小对 GBN 的传输时间影响 (单位:ms)

窗口大小 \ 延时 (ms)	2	5	10	20
2	4098.99	4089.99	4080.97	2044.98
5	4080.97	4064.23	4089.99	1447.52
10	4063.12	4064.23	4064.23	814.90
15	1913.32	1956.65	1932.98	738.33

表 12: 对比不同延时、不同窗口大小对 GBN 的传输吞吐率影响 (单位:kbps)

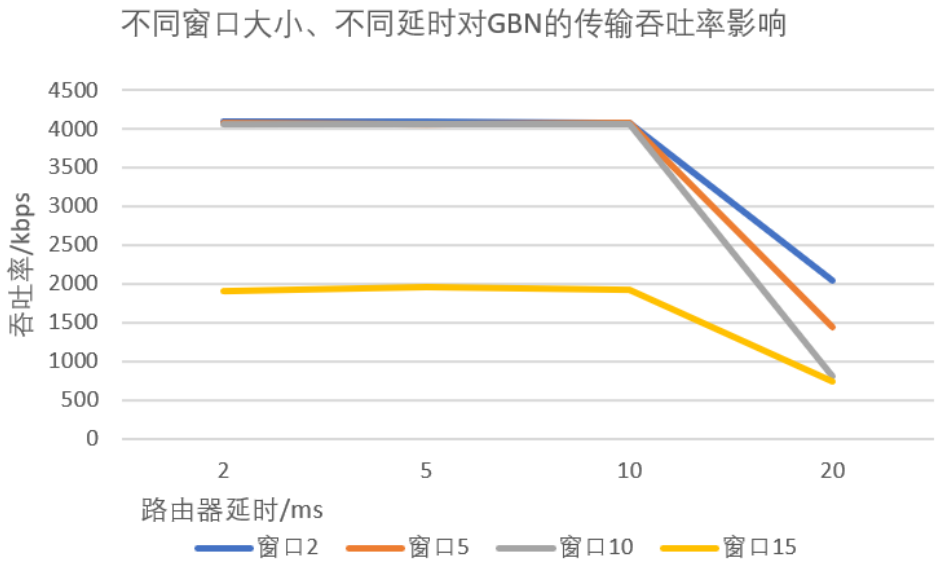

图 3.8

- **分析**：由表11、表12、图3.8，我们可以发现：如果只有延时，而无丢包率，那么窗口小的性能好，在这里延时较小时（0-10ms），窗口 2、5、10 的性能差不多一致，而当延时为 20ms，可以看到窗口 2 的性能最好，而其他更大的窗口性能非常差。

- **原因**：我们知道路由器处理包的时候是一个一个处理，那么尽管 GBN 是连续的发送数据包给路由器，但是一个包要等待前面的包延时完毕，才会被处理。

  那么可以发现一个包要等待：窗口大小 * 一条数据包延时的时间，所以尽管延时只有 20ms，但是我们的窗口越大，比如为 10，那么一个包要等待 200ms 才能发送到接收端，可想而知，由于程序设定的 timeout 为 100ms，每个包一定会重传，所以

性能非常差。

而窗口较小时，一个包被发送成功期间的时间不会超过 timeout，所以不会超时重传，所以性能较好。

所以，如果只发生超时重传，在无丢包的情况下，其实更适合窗口小的 GBN，性能更好一些。

2. 针对 SR

设置丢包率为 0，程序超时的 timeout 设置为 100ms。

窗口大小 \ 延时 (ms)	2	5	10	20
2	2797	2782	4094	6360
5	2531	2813	5219	12734
10	3375	2797	4875	7375
15	3937	3375	5735	7859

表 13: 对比不同延时、不同窗口大小对 SR 的传输时间影响 (单位:ms)

窗口大小 \ 延时 (ms)	2	5	10	20
2	5312.41	5341.06	3629.41	2336.29
5	5870.73	5282.2	2847.06	1166.86
10	4402.61	5312.41	3047.96	2014.76
15	3774.15	4402.61	2590.9	1890.68

表 14: 对比不同延时、不同窗口大小对 SR 的传输吞吐率影响 (单位:kbps)

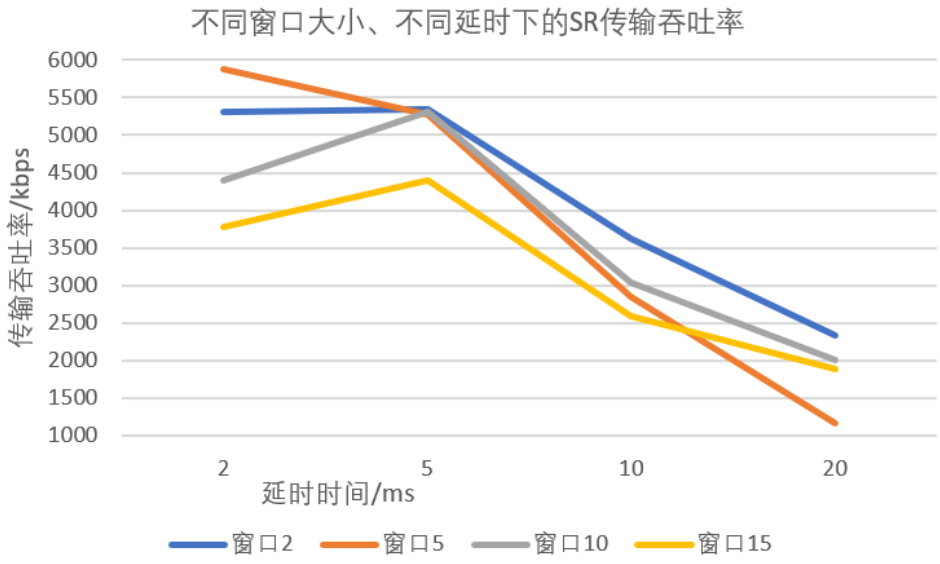

图 3.9

- **分析**：由表13、表14、图3.9，我们可以发现：在延时较小时，窗口 5 表现的最好；而在延时超过 5ms，窗口 2 的性能最佳。

- **原因**：如果网络只存在丢包，那么 SR 的性能十分好。但是如果网络存在较大的延时，比如这里超过 5ms，SR 的性能就不太好了。这可能是因为：

  在窗口过大时，每个窗口内的包迅速一起地发给路由器，而路由器会一个一个处理，那么显然窗口越大，一个包延时的时间就越长，因为路由器不是并行的处理，与路由器的设计有关。这就会导致窗口较大时，一个包实际上的延时远远超过我们设定的 timeout，导致不断重传，性能当然会下降。

  因此也就不难去解释为什么延时较大时，窗口为 2 的性能更好了。因为窗口越大，一个包要延时的时间就越长，最终导致了窗口越大，性能越差，增加了重传的频率，这会降低整体性能。

  此外，我们与 GBN 的测试进行联想，可以发现在只有延时，无丢包的情况，其实窗口小的性能会好一些；但是在有丢包的情况，明显窗口更大的 SR 机制的性能更好。

  所以其实这个现象与路由器的缺陷有关，因为路由器一次只能处理一条数据包，造成了每条数据包等待的延时非常大。如果路由器能够并行的处理到来的数据，而不是一次只处理一条数据包，就可以实现真正的性能测试了。

4 累计确认和选择确认的性能比较

4.1 设计思路

这里我们针对 SR、GBN 两者确认方式，进行性能的比较。窗口大小设定为 10，在不同的延时、丢包下，进行测试

4.2 性能测试

4.2.1 延时变化

丢包率设置 1%，程序的 timeout 设置为 100ms, 窗口大小 10。在丢包率很小情况下，我们测试延时对累计确认和选择确认的影响进行对比。

机制 \ 延时 (ms)	2	5	10	15	20
GBN	3687	3453	5422	5063	12782
SR	4828	4000	4718	5859	7625

表 15: 对比 SR 与 GBN 在不同延时下的传输时间 (单位:ms)

机制 \ 延时 (ms)	2	5	10	15	20
GBN	4030.06	4303.16	2740.47	2934.79	1162.48
SR	3077.64	3714.71	3149.39	2536.07	1948.7

表 16: 对比 SR 与 GBN 在不同延时下的传输吞吐率 (单位:kbps)

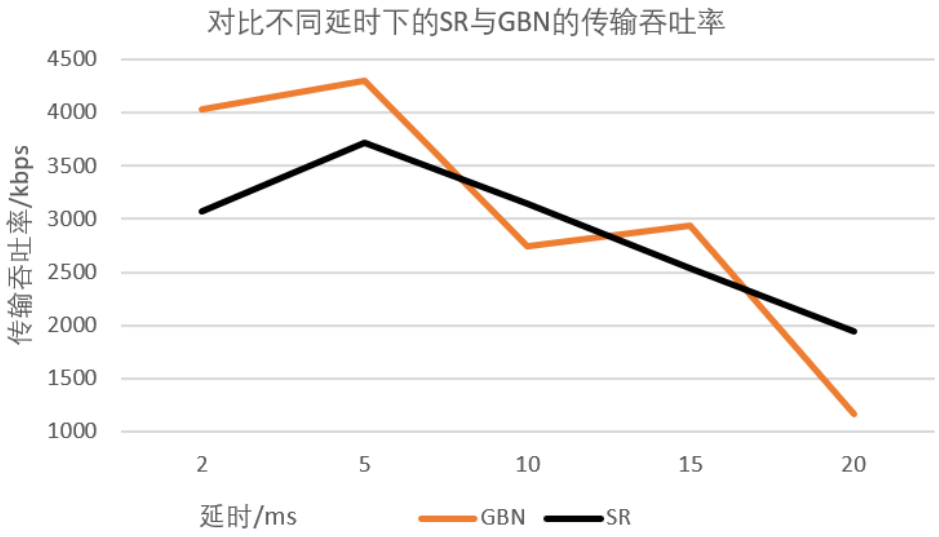

图 4.10

- **分析**：由表15、表16、图4.10，我们可以发现：在延时较小 (2-10ms) 时，GBN 的传输性能较好；而在延时更大 (10-20ms) 时 SR 的性能更好。

- **原因**：由于窗口大小设置 10，可能会出现一个包要等待 10 个延时的情况，也就是 10 倍的路由器延时。在延时较小，程序设置的 timeout 比一个包最小等待的延时要大，此时显然超时重传的情况发生的较少，所以 GBN 效率很高；而 SR 由于发送的是选择确认，对每个分组计时，实现的过程较为复杂，而且选择确认对单个数据包确认，所以性能差于 GBN。

  而在延时较大，由于窗口大小 10，此时一个包最小等待的延时已经超过了设定的 timeout，导致重传次数显著增多。可以看到在延时为 20 的情况下，GBN 的效率突然巨幅下降，而由于 SR 在重传上有较大的优势，所以性能优于 GBN。因为 SR 重传时采取选择重传，而 GBN 会重传一整个窗口的分组，所以造成延时较大时，SR 的效率更高。

4.2.2 丢包率变化

延时设置为为 0，程序的 timeout 为 100，窗口大小 10

机制 \ 丢包率	0%	2%	4%	8%	10%
GBN	1672	2375	3453	14188	25078
SR	1891	2188	2031	2921	2328

表 17: 对比 SR 与 GBN 在不同丢包率下的传输时间 (单位:ms)

机制 \ 丢包率	0%	2%	4%	8%	10%
GBN	8886.86	6256.35	4303.16	1047.28	592.5
SR	7857.65	6715.05	7316.01	5086.9	6382.66

表 18: 对比 SR 与 GBN 在不同丢包率下的传输吞吐率 (单位:kbps)

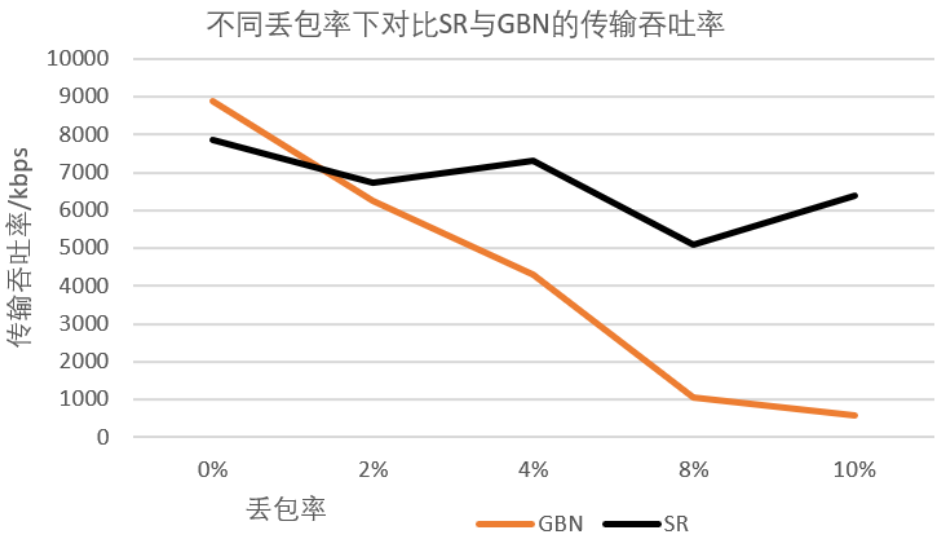

图 4.11

- **分析**：由表17、表18、图4.11，我们可以发现：在丢包率很小时 (0-1%)，GBN 的传输性能更好；而在丢包率变大时 (2%-10%)SR 的传输性能维持稳定，而 GBN 的传输性能陡然下降。

- **原因**：在丢包率近乎没有时，此时超时重传的概率很小，那么 GBN、SR 显然不会用到超时重传机制，此时由于 GBN 是采用累积确认，只针对窗口的首分组计时，程序上更为简单，而 SR 的设计复杂，对每个分组计时，所以会出现 SR 性能差于 GBN 的情况。

  而在丢包率逐渐增长时，SR 的优势便显现出来了。因为 SR 是针对某一个分组进行重传，那么在重传时性能较好；而 GBN 是累积确认，不会接受乱序分组，此时如果需要重传，那么接收方只会阻塞，不会往下接受，而发送方会重传 10 个分组，造成性能显著下降。

综上，在存在丢包率、延时的情况下，选择用 SR 是更优的选择，SR 的性能会维持比较稳定的水平，不会像 GBN 陡然下降。

但是如果只存在延时，而没有丢包，那么在延时较大时，较大窗口的滑动窗口性能会很差。因为路由器是一个一个处理的，导致了每个包都要等待前面的包被延时处理完毕，最后每个包的延时是窗口大小 * 路由器延时，而很容易超过程序设定的 timeout 超时时间。因为每个包都必须要重传才能被收到，所以导致性能陡降。