计算机网络第六次实验报告

网络空间安全学院 物联网工程 2111673 岳志鑫

一、实验目的

基于 UDP 服务设计可靠传输协议并编程实现（3-4）

二、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。

- 控制变量法：对比时要控制单一变量（算法、窗口大小、延时、丢包率）
- Router：可能会有较大延时，传输速率不作为评分依据，也可自行设计
- 延时、丢包率对比设置：要有梯度（例如 30ms,50ms, …；5%，10%，…）
- 测试文件：必须使用助教发的测试文件（1.jpg、2.jpg、3.jpg、helloworld.txt）
- 性能测试指标：时延、吞吐率，要给出图、表并进行分析

三、实验内容

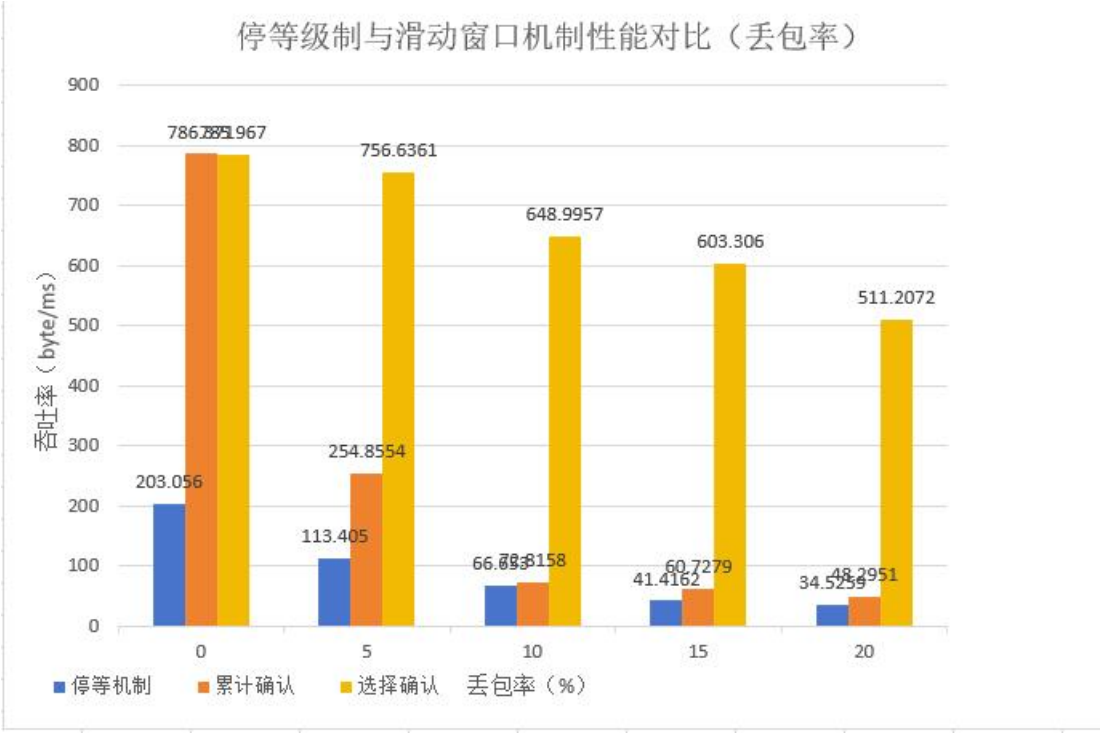
1. 停等机制与滑动窗口机制性能对比

在此实验中固定延时为 0 的条件不变，只改变丢包率的大小，观察停等级制与滑动窗口机制的性能。由表可看出在最初时累计确认和选择确认的吞吐率要远大于停等级制，由于滑动窗口设置为 10，所以滑动窗口的效率理应是停等机制的十倍，但是可能由于存在端到端延迟、文件读写和处理等操作以及一些其他的开销，所以传输效率其实并没有达到理论值。

随着丢包率的不断上升，发现停等级制大概处于一个线性下降的状态，因为重传的包的次数增加，导致整个传输过程的时间增加，吞吐率会呈现逐渐下降的趋势。而累计确认则是骤降，甚至在更高的丢包率状态下可能传输效率会低于停等机制，这是因为 GO BACK N 的机制会导致将之前传输过的数据包再次反复传输，在丢包率高的情况下反而会多次重传，严重影响性能。对于选择确认来说，虽然丢包率上升，但是其 SR 的机制使得不会重传多余的包，所以其效率较为稳定，传输速度也最快。

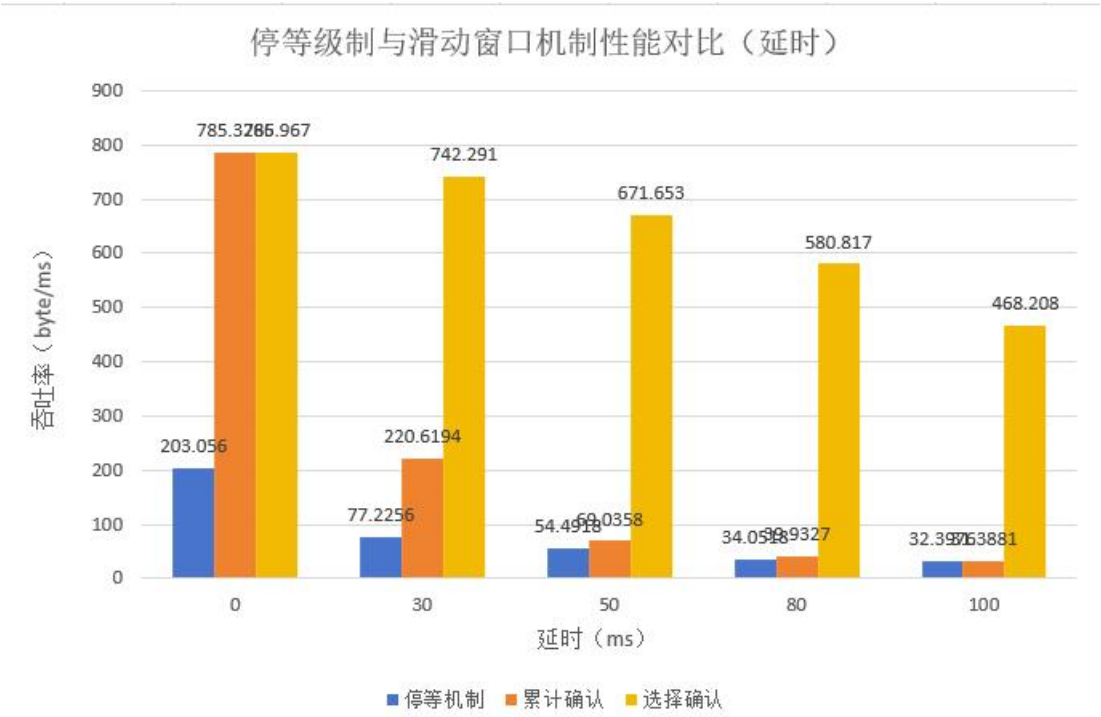
综上所述认为，当丢包率较小的时候，由于流水线和滑动窗口的存在，累计确认和选择确认的性能优于停等机制。但当丢包率较高的时候，GO BACK N 由于存在较大的重传开销，导致性能下降，此时其性能不如停等机制，选择确认的性能较为稳定传输速度也较快。

名称	丢包率(%)	0	5	10	15	20
停等机制	传输时间（ms）	9143	16378	27866	44846	53796
	吞吐率(byte/ms)	203.056	113.405	66.653	41.4162	34.5259
累计确认	传输时间（ms）	2364	7113	25469	30571	38441
	吞吐率(byte/ms)	786.371	254.8554	72.8158	60.7279	48.2951
选择确认	传输时间（ms）	2363	2471	2588	3071	3329
	吞吐率(byte/ms)	785.967	756.6361	648.9957	603.306	511.2072



在此实验中固定丢包率为 0 的条件不变，只改变延时的大小，观察停等级制与滑动窗口机制的性能。发现依旧是最开始的时候累计确认和选择确认要高于停等级制，但是随着延迟逐渐增加，累计确认逐渐递减少直至跟停等机制相差不多，可能是因为延迟时间超过了 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，累计确认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，累计确认的性能迅速下降，甚至不如停等机制。针对于选择重传来说也会对其效率产生影响，但是由于滑动窗口的机制所在，并且不会像 GO BACK N 那样重传窗口内的所有数据包，所以重传的数据包并不是那么多，效率最高。

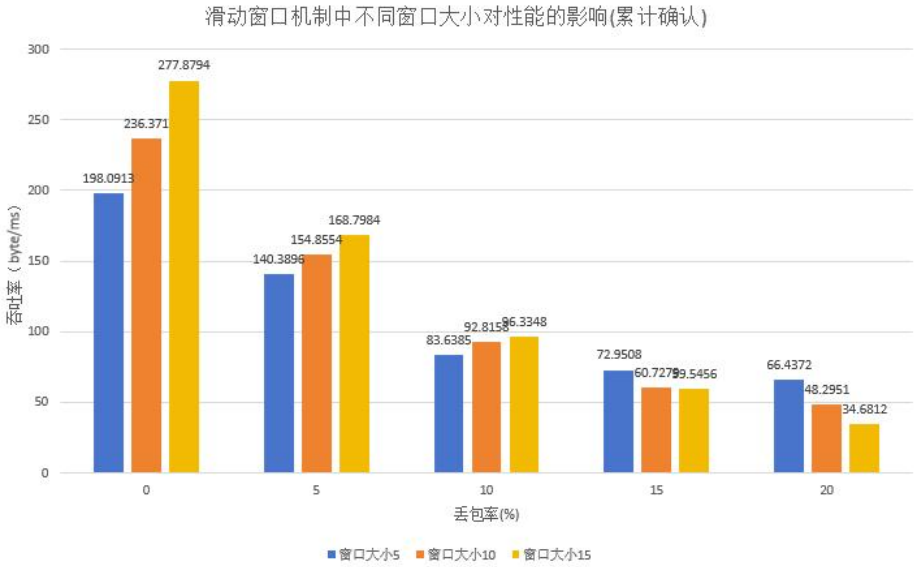
名称	延时(ms)	0	30	50	80	100
停等机制	传输时间（ms）	9143	24051	34085	54545	57330
	吞吐率(byte/ms)	203.056	77.2256	54.4918	34.0518	32.3976
累计确认	传输时间（ms）	2364	8415	26892	46491	59147
	吞吐率(byte/ms)	785.3266	220.6194	69.0358	39.9327	31.3881
选择确认	传输时间（ms）	2364	2451	3325	3811	4311
	吞吐率(byte/ms)	785.967	742.291	671.653	580.817	468.208



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）

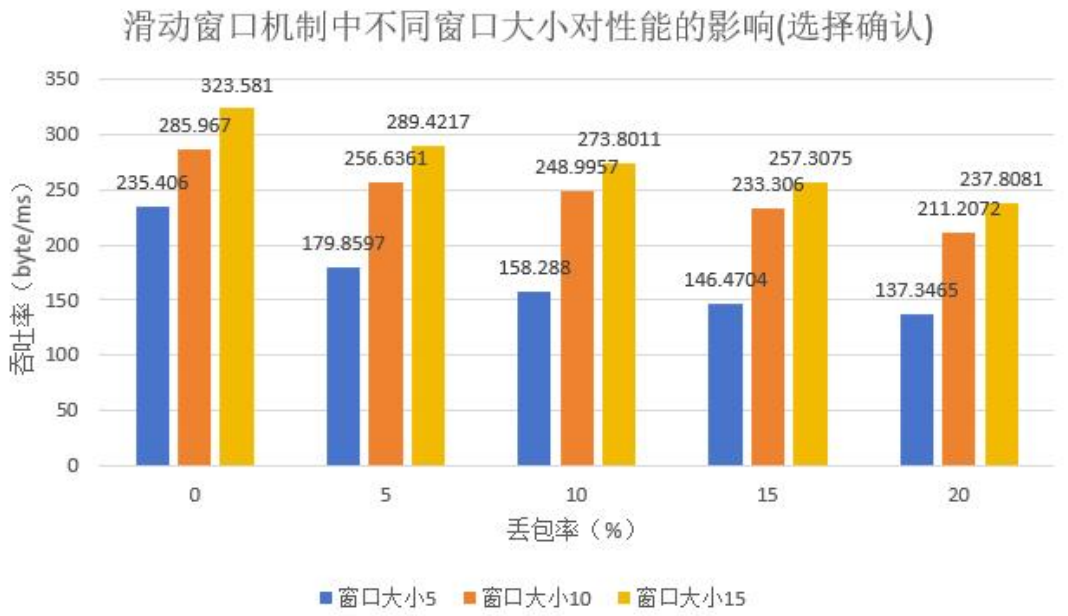
累计确认实验中，我们将延时设置为 0，只观察不同滑动窗口大小和丢包率的影响。发现丢包率小于 15%时，随着滑动窗口变大，吞吐率也在增大，但是丢包率超过 15%时，滑动窗口增大吞吐率反而变小，可能是因为丢包率过高，导致 GO BACK N 机制的影响下滑动窗口内的数据包都需要超时重传，因此滑动窗口越大，超时重传的数据包越多，吞吐率就会越低。

窗口大小 5	传输时间（ms）	9372	13224	22197	25431	28129
	吞吐率(byte/ms)	198.0913	140.3896	83.6385	72.9508	66.4372
窗口大小 10	传输时间（ms）	7866	12055	20179	30571	38441
	吞吐率(byte/ms)	236.371	154.8554	92.8158	60.7279	48.2951
窗口大小 15	传输时间（ms）	6681	11691	19271	31178	54561
	吞吐率(byte/ms)	277.8794	168.7984	96.3348	59.5456	34.6812



对于选择确认我们也将延时设置为 0，只观察不同滑动窗口大小和丢包率的影响。发现选择确认机制下滑动窗口越大，吞吐率越大，这并不受丢包率大小的影响，较为稳定。

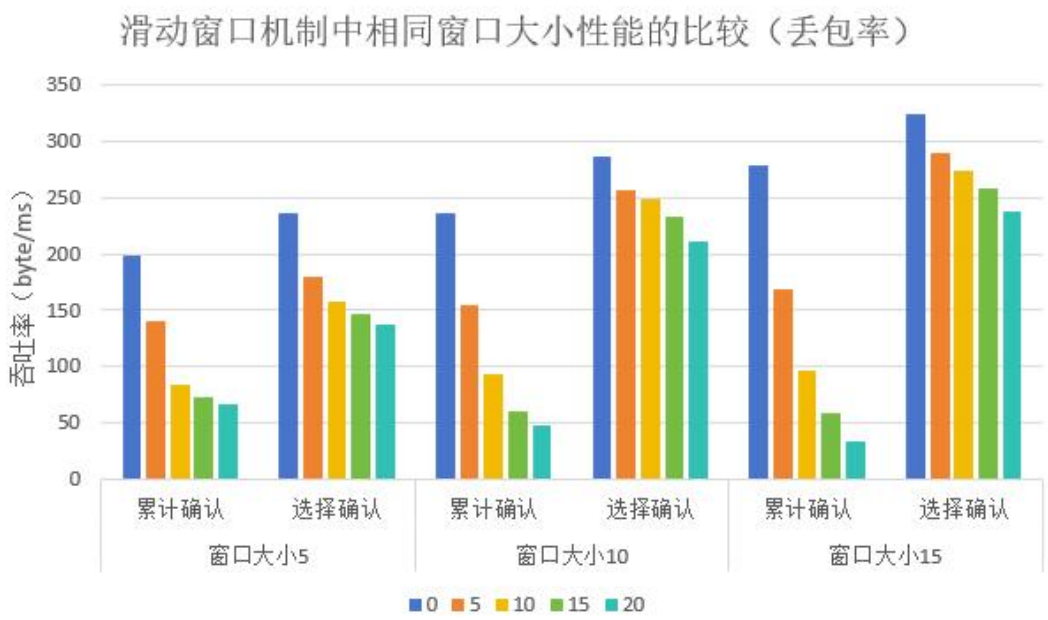
窗口大小 5	传输时间（ms）	7890	10322	11734	12675	13517
	吞吐率(byte/ms)	235.406	179.8597	158.288	146.4704	137.3465
窗口大小 10	传输时间（ms）	6495	7245	7456	7961	8790
	吞吐率(byte/ms)	285.967	256.6361	248.9957	233.306	211.2072
窗口大小 15	传输时间（ms）	5740	6538	6911	7354	7957
	吞吐率(byte/ms)	323.581	289.4217	273.8011	257.3075	237.8081



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。

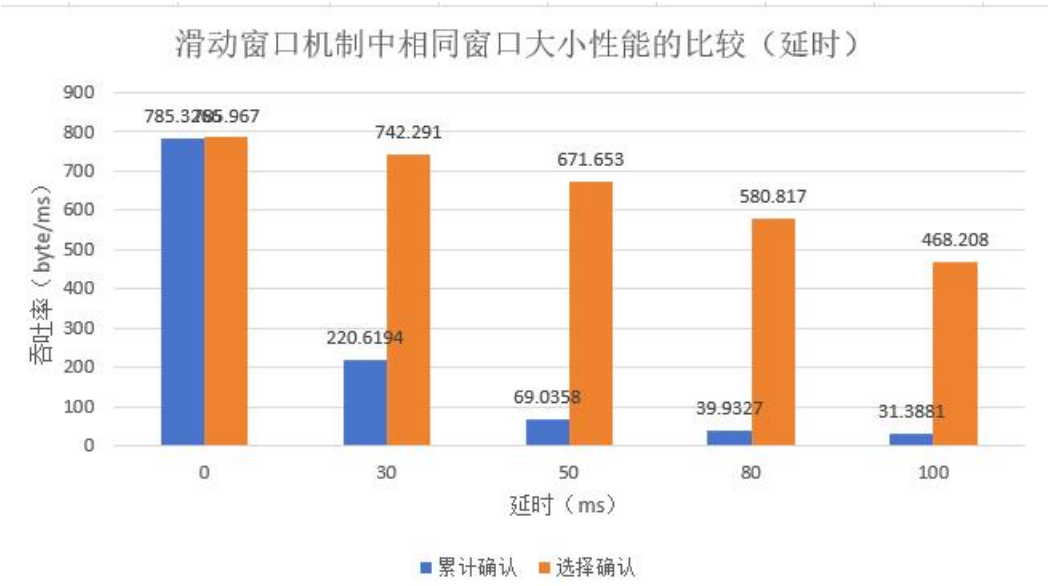
在此实验中我们固定时延为 0 不变，通过改变丢包率和滑动窗口的大小探究相同窗口下累计确认和选择确认的性能差异。首先窗口固定时，发现丢包率为 0 时选择确认和累计确认的吞吐量大小差距不大，这是因为都是基于滑动窗口机制而设置所以丢包率为 0 时差距不大，但随着丢包率逐渐增大，选择确认的吞吐率总是要高于累计确认，也是由于 GO BACK N 在有丢包条件下的数据包重复重传导致的吞吐率降低。滑动窗口增加时发现选择确认在各种丢包率条件下的吞吐率都相应增加，但累计确认在丢包率超过 15 后滑动窗口增加，吞吐率会减少，也符合之前分析的规律。

		0	5	10	15	20
窗口大小 5	累计确认	198.0913	140.3896	83.6385	72.9508	66.4372
	选择确认	235.406	179.8597	158.288	146.4704	137.3465
窗口大小 10	累计确认	236.371	154.8554	92.8158	60.7279	48.2951
	选择确认	285.967	256.6361	248.9957	233.306	211.2072
窗口大小 15	累计确认	277.8794	168.7984	96.3348	59.5456	34.6812
	选择确认	323.581	289.4217	273.8011	257.3075	237.8081



对于延时的实验我们就以窗口大小为 10 的时候进行分析，固定丢包率为 0，查看吞吐率。我们发现初始时选择确认和累计确认吞吐率相差不多，随着延时增大累计确认的吞吐量下降较快，也是因为超时重传的重复数据包过多导致的效率降低，相比之下选择确认的效率也有降低，但是也比累计确认的吞吐率要高。

	延时(ms)	0	30	50	80	100
累计确认	传输时间（ms）	2364	8415	26892	46491	59147
	吞吐率(byte/ms)	785.3266	220.6194	69.0358	39.9327	31.3881
选择确认	传输时间（ms）	2364	2451	3325	3811	4011
	吞吐率(byte/ms)	785.967	742.291	671.653	580.817	468.208



五、附录

相关完整代码参照 GitHub:
https://github.com/Q-qiuqiu/Computer-Networks/tree/main