数据包队列实验

范子墨
2021K8009929006

一、 实验内容

1、 重现 BufferBloat 结果

a) h1(发送方)在对 h2 进行 iperf 的同时，测量 h1 的拥塞窗口值(cwnd)、r1-eth1 的队列长度(qlen)、h1 与 h2 间的往返延迟(rtt)

b) 变化 r1-eth1 的队列大小，考察其对 iperf 吞吐率和上述三个指标的影响

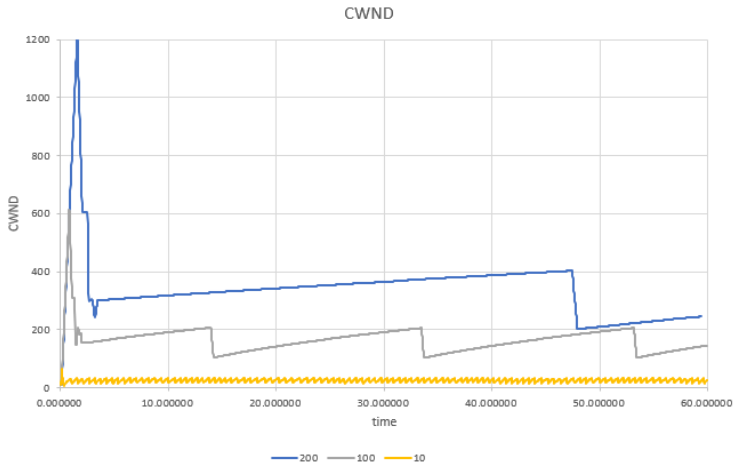
2、 解决 BufferBloat 问题

RED, CoDel, tail drop

二、 重现 BufferBloat 问题

我选取了 10、100、200 三个队列长度进行实验。在 excel 文件中插入三种队列长度的时间与时间对应数据后绘表得到结果。

1、 CWND



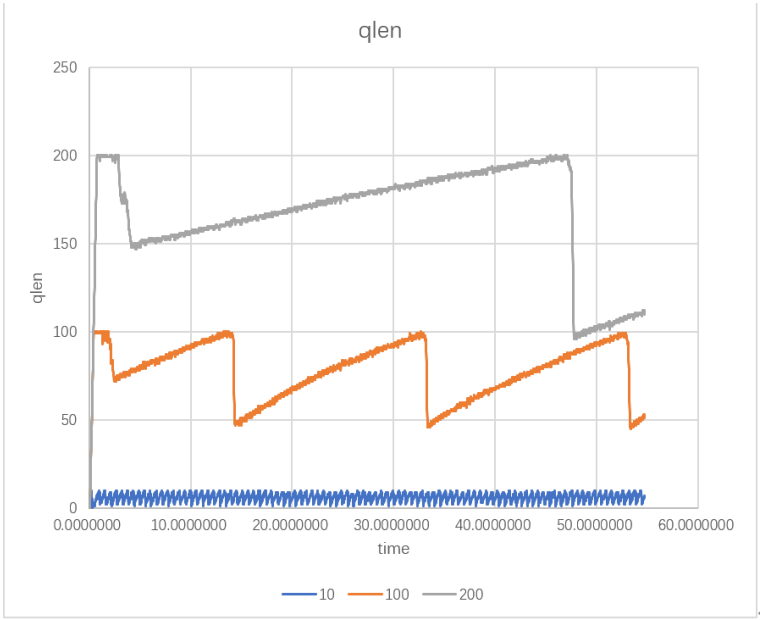
图表 1：CWND

上图为拥塞窗口大小随时间变化曲线，拥塞窗口大小表示了发送方可以在不接收确认的情况下发送到网络的数据量。当测试开始时，由于 TCP 慢启动机制，拥塞窗口大小以指数级别增长，队列大小为 10 的拥塞窗口迅速增长至 60，队列大小为 100 的拥塞窗口迅速增长到 600 的峰值，队列大小为 200 的拥塞窗口迅速增长至 1200 的峰值。拥塞窗口峰值均为队列大小的六倍。由于拥塞程度迅速增加，丢包率增加，拥塞窗口迅速减少，减低发送率，减少拥塞可能性。而后发送方从接收方收到确认后，确定拥塞程度减少，拥塞窗口增加，以提高发送速率。拥塞窗口的大小会不断地根据网络情况进行调整。当网络拥塞减小时，拥塞窗口会增大，从而提高数据传输速率。相反，当网络拥塞程度增加时，拥塞窗口会减小，以降低数据传输速率，以适应网络。

同时，在拥塞窗口的机制中，队列大小与拥塞窗口之间存在关联。当拥塞窗口达到队列大小的两倍时，这是一个峰值点。此时，TCP 发送方会减小拥塞窗口，以缓解网络拥塞。拥塞窗口大小迅速减小至等于队列大小，然后再次开始增长。

当队列越大，波动周期越长。

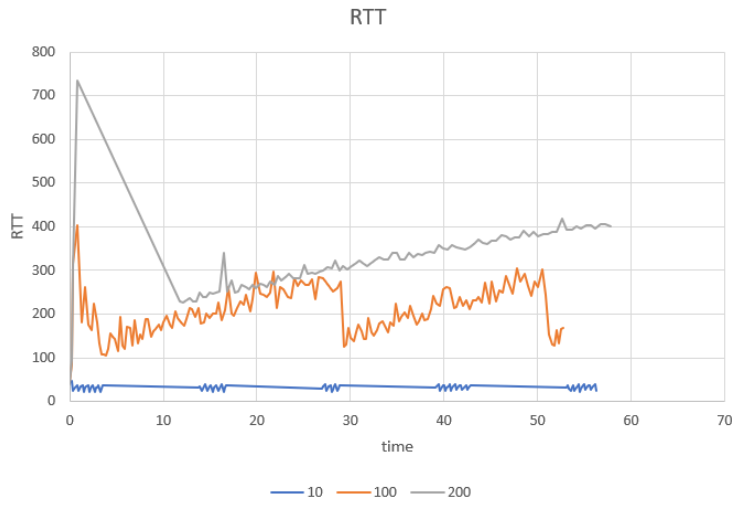
2、 QLEN



图表 2：QLEN

上图为队列长度随时间变化曲线。测试刚开始时队列快速到达上界，队列满时直接丢弃新的包，发送速率骤降至队列长度 50%。与拥塞窗口的图对应可知，当队列满时丢包率迅速上升，拥塞窗口迅速减小减低发送速率，因此队列长度对比拥塞窗口下降时间稍有延迟。由于拥塞窗口大小较周期性峰值为一半，因此队列长度也会下降至周期性峰值的一半。

3、 RTT



图表 3:RTT

测试起始时由于队列的迅速增长，网络延时急剧增加，而后随队列缩短而减少，与队列的变化趋势大体相同。同时可以发现，当队列长度为 200 时，时延大部分时间高于剩余两种队列长度，产生 BufferBloat 问题。

猜测 RTT 图像中毛刺较多可能是由于受到网络中的路由器、链路和网络拓扑的影响。即使在没有拥塞的情况下，RTT 仍可能因网络拓扑中的不同路径和链路条件而发生变化。同时由于 RTT 可以迅速反映网络中的变化，包括数据包的排队和传输，因此图中毛刺会较多。

4、 iperf

队列大小为 10 时，带宽平均大约为 9.45 Mbits/sec ，峰值为 16.8 Mbits/sec，最小值为 6.29 Mbits/sec。

队列大小为 100 时，带宽平均大约为 10.21 Mbits/sec，峰值为 49.3 Mbits/sec，最小值为 1.02 Mbits/sec。

队列大小为 200 时，带宽平均大约为 10.89 Mbits/sec，峰值为 69.9 Mbits/sec，最小值为 0 Mbits/sec。

可以看出随队列长度增加，平均带宽并无显著增加，但是峰值更高，最小值也更小。这可能是由于平均带宽受瓶颈链路影响。同时由于队列长度长，拥塞窗口变化更大，发送速率变化大，进而导致峰值和最小值差距更大。

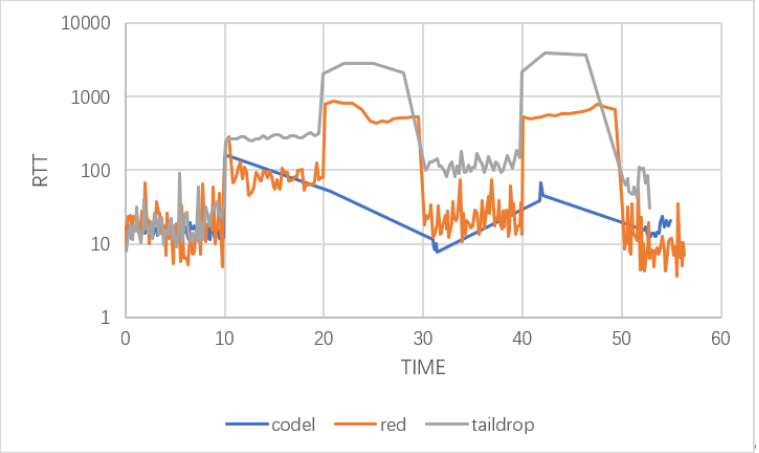
5、 总结

可以看出随队列长度增加，网络延迟明显增加，产生 BufferBloat 问题。同时由于较小的队列难以容忍高并发数据包，产生 TCP incast 问题。

三、 解决 BufferBloat 问题

在 excel 文件中插入三种解决方法的时间与时间对应数据后绘表，纵坐标取 log。

1、 结果



图表 4:解决 BufferBloat 问题 RTT 监测

2、 分析

Taildrop 是一种简单的队列管理策略，当队列已满时，它会直接丢弃后续到达的数据包。Taildrop 队列通常不具备主动拥塞控制，这意味着在拥塞发生时，它无法智能地控制拥塞窗口，因此可能会导致较高的延迟。

RED 是一种主动队列管理策略，它在队列即将溢出之前就开始丢弃数据包，以防止队列过载和拥塞。RED 会随机丢弃一些数据包，这有助于减少拥塞窗口的增长速率，并减少网络延迟。RED 的目标是在网络拥塞发生之前就采取措施，因此比 Taildrop 表现稍好

CoDel 是一种新的队列管理策略，旨在更好地控制网络延迟。它根据队列中数据包的排队时间来决定是否丢弃数据包。这使得 CoDel 能够更好地适应网络负载和动态变化，避免了拥塞的快速发展。因此，CoDel 通常在避免 BufferBloat 问题和提供更低延迟方面表现出色。

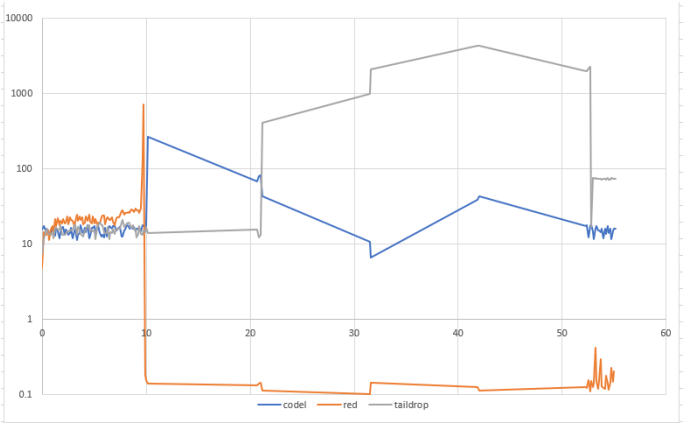
因此推测三种方法效果从好到差是 Codel，RED 最后为 taildrop

而由图可以看出 taildrop 的队列延时远高于 RED 和 CoDel，其中相对来说，CoDel 表现最佳，与猜测相符。在最开始队列为空时，三种方法时延相似，这是由于延迟即为传播时延。

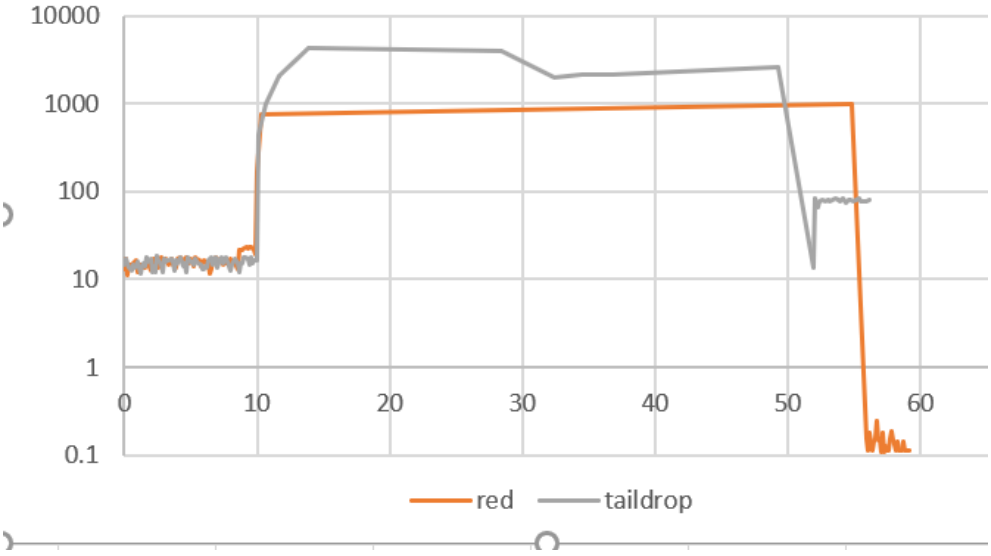
当即将发生拥塞时，队列为满，taildrop 直接丢弃数据包，而后每次能成功接收的数据包都经历了完整的排队过程，因此出现一段稳定的时延。而 RED 也与之类似，只是由于会进行数据包的提前丢弃，因此稳定的时延比 taildrop 低。而 Codel 表现则更好，可以有效的减缓 BufferBloat 问题。

四、 实验中遇到的一些小问题

在解决 BufferBloat 实验中经常会遇到实验内容没有正常输出，以及 RED 的时延只有 0.0 几秒的情况，这远低于传输时延，显然是无效数据。很好奇原因是什么？有延长过检测时间，但是依旧出现问题。因此借用饶嘉怡同学的电脑进行 RED 解决方法时延的测量。



图表 5：无效数据 1



图表 6：无效数据 2

本次时间主要耗费在跑出合理数据上，在重现 BufferBloat 问题中会出现 RTT 监测只有三十多秒的情况，因此花费了很多时间重新跑数据。同时对比同学，似乎我的丢包率格外大，600 个结果常常只能返回 100-200 个结果左右，可能是由于虚拟机配置的问题。