数据包队列管理实验报告

热伊莱·图尔贡　2018K8009929030

一、 实验题目：

数据包队列管理实验

二、 实验内容：

1. 重现 Bufferbloat 结果
   - ✧ 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)
   - ✧ 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

三、 实验流程：

1. 重现 Bufferbloat 结果
   - ✧ 重现 CWND、Qlen、RTT 时间曲线图
   - ✧ 改变数据包队列大小，观察其对 CWND、Qlen、RTT 和吞吐

率图像的影响

2. 解决 BufferBloat 问题

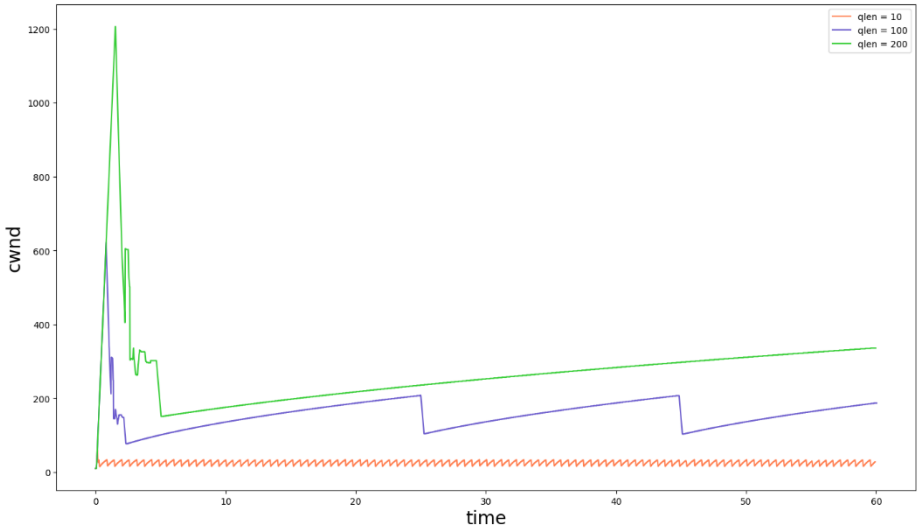
✧ 重现不同队列管理策略在动态带宽下的往返延迟结果。

四、 数据处理及结果：

1. 重现 Bufferbloat 结果

（1） CWND 实验结果

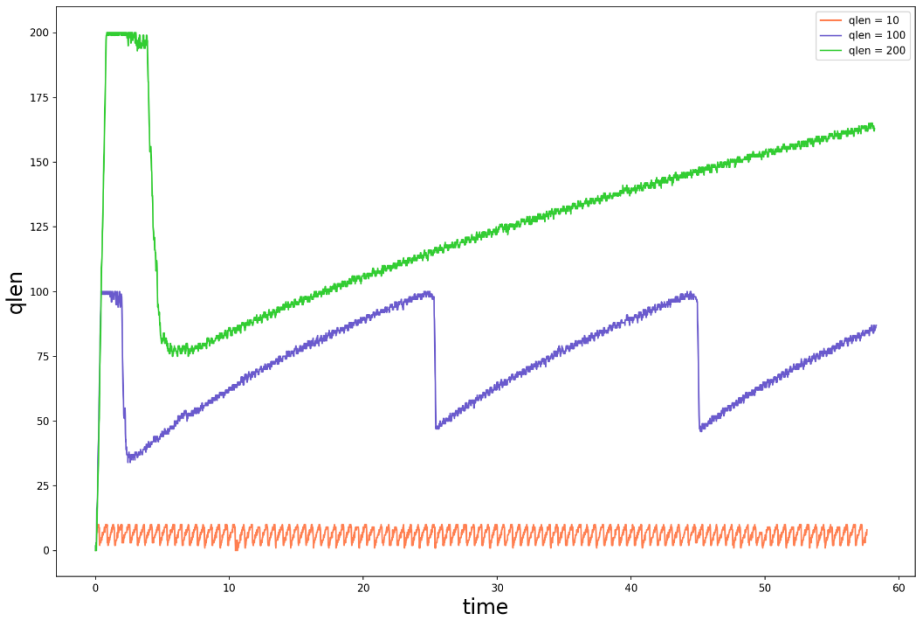
改变数据包队列大小，作出 CWND 与时间的曲线图如下，maxq=100 时与讲义中实验结果一致。



以 maxq=100 为例，在测试开端迅速增长至峰值，然后迅速回落。在回落后发生震荡。拥塞窗口随手包增长，随丢包下降。总体而言，cwnd 随时间呈现周期性变化，周期较小，变化幅度也较小。而随着 maxq 增大，周期变大，每个周期的变化幅度也加剧。平均 CWND 随 maxq 逐渐增大，BufferBloat 问题加剧。

（2） Qlen 实验结果

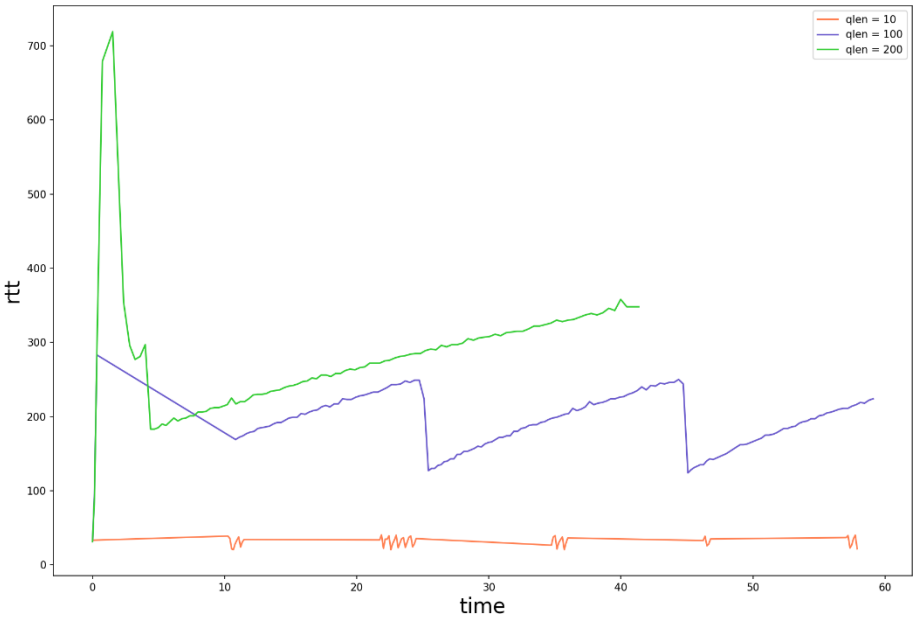
改变数据包队列大小，作出 Qlen 与时间的曲线图如下，maxq=100 时与讲义中实验结果一致。



以 maxq=100 为例，峰值为设定的最大队列大小。当曲线达到峰值后，队列变满，将丢弃新收到的包。随着发送速率的降低，队列大小也将快速下降，并保持在不超过最大队列大小的范围内震荡，做周期性变化。平均 Qlen 随 maxq 逐渐增大，BufferBloat 问题加剧。

（3） RTT 实验结果

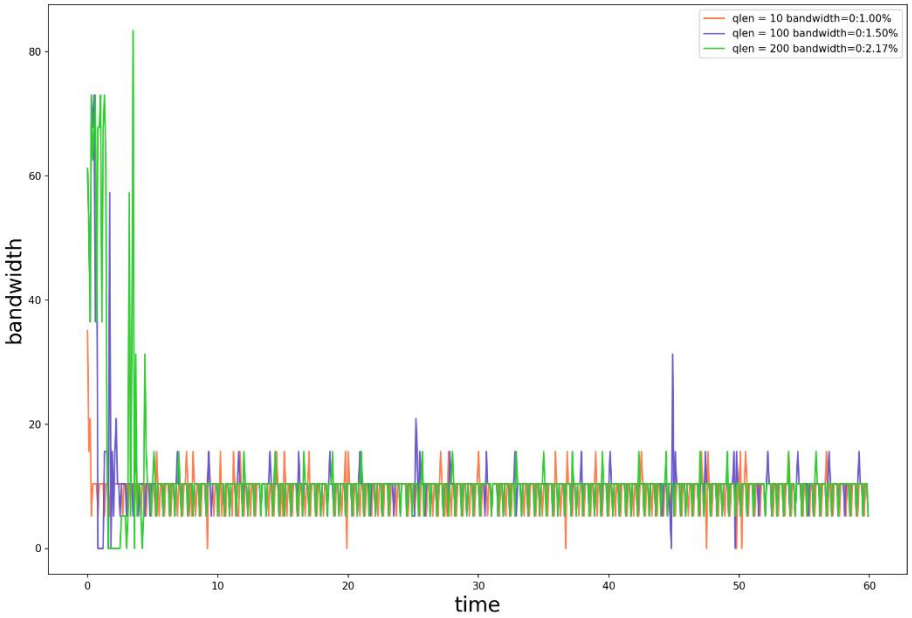
改变数据包队列大小，作出 RTT 与时间的曲线图如下，maxq=100 时与讲义中实验结果一致。



以 maxq=100 为例，测试刚开始时，由于接收队列的快速增长，网络延时急剧增加，随后也随着队列缩短而减少。之后的时间内，延时随着队列长度的变化而变化，波动明显。大队列后续的波动较小队列不明显。平均 RTT 随 maxq 逐渐增大，BufferBloat 问题加剧。

（4） iperf 吞吐率

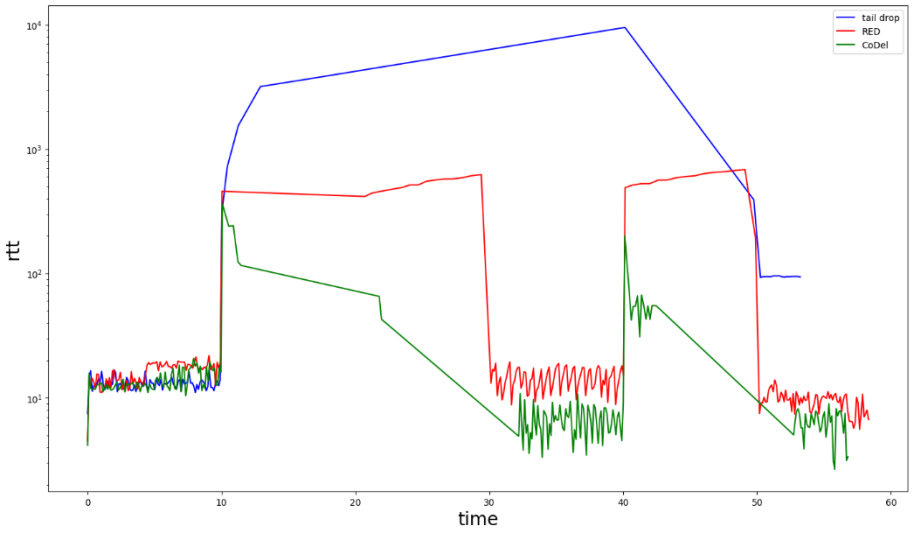
变更队列大小，作出 吞吐率 与时间的曲线图如下



除了开端波动较大，之后的三条线的实验数据成相同的周期性变化，在 10 附近震荡。总体上队列大小对吞吐率影响不大，仅在延迟浮动较大处吞吐率有短暂波动。

2. 解决 BufferBloat 问题

不同队列管理机制延迟时间曲线图如下



tail drop 的队列延时远高于 RED 和 CoDel。其中 CoDel 的性能表现最佳。

五、 调研思考题：

拥塞控制机制对 Bufferbloat 的影响

前文中提到，导致 Bufferbloat 问题的三个关键因素：队列长度，队列管理机制，和拥塞控制机制。同时，分别从上述三个角度都可以解决 Bufferbloat 问题。调研分析两种新型拥塞控制机制（BBR [Cardwell2016]，HPCC [Li2019]），阐述其是如何解决 Bufferbloat 问题的。

Bufferbloat 的原因是数据的发送带宽小于到达带宽，导致数据包在队列中的滞留。

◆ BBR:

BBR 是 2016 年由 Google 发表的优化 tcp 传输的算法，它极大的提高了 tcp 的 throughput。

BBR 的组成

bbr 在实现上由 5 部分组成：即使速率计算，RTT 跟踪，BBR 状态机，pacing rate 和 cwnd，其他外部机制的利用-fq，rack 等。

1.即时速率的计算

计算一个即时的带宽 bw，该带宽是 bbr 一切计算的基准，bbr 将会根据当前的即时带宽以及其所处的 pipe 状态来计算 pacing rate 以及 cwnd(见下文)，这个即时带宽计算方法的突破式改进是 bbr 之所以简单且高效的根源。计算方案按照标量计算，不再关注数据的含义。在

bbr 运行过程中，系统会跟踪当前为止最大的即时带宽。

2.RTT 的跟踪

bbr 之所以可以获取非常高的带宽利用率，是因为它可以非常安全且豪放地探测到带宽的最大值以及 rtt 的最小值，这样计算出来的 BDP 就是目前为止 TCP 管道的最大容量。bbr 的目标就是达到这个最大的容量，这个目标最终驱动了 cwnd 的计算。在 bbr 运行过程中，系统会跟踪当前为止最小 RTT。

3.BBR 状态机的维持

bbr 算法根据互联网的拥塞行为有针对性地定义了 4 中状态，即 STARTUP, DRAIN, PROBE_BW, PROBE_RTT。bbr 通过对上述计算的即时带宽 bw 以及 rtt 的持续观察，在这 4 个状态之间自由切换，相比之前的所有拥塞控制算法，其革命性的改进在于 bbr 拥塞算法不再跟踪系统的 TCP 拥塞状态机，而旨在用统一的方式来应对 pacing rate 和 cwnd 的计算，不管当前 TCP 是处在 Open 状态还是处在 Disorder 状态，抑或已经在 Recovery 状态，换句话说，bbr 算法感觉不到丢包，它能看到的就是 bw 和 rtt。

4.结果输出-pacing rate 和 cwnd

bbr 的输出并不仅仅是一个 cwnd，更重要的是 pacing rate。在传统意义上，cwnd 是 TCP 拥塞控制算法的唯一输出，但是它仅仅规定了当前的 TCP 最多可以发送多少数据，它并没有规定怎么把这么多数据发出去，在 Linux 的实现中，忽略接收端通告窗口的前提下，Linux 会把 cwnd 一窗数据全部突发出去，而这往往会造成路由器的排队，

在深队列的情况下，会测量出 rtt 剧烈地抖动。

bbr 在计算 cwnd 的同时，还计算了一个与之适配的 pacing rate，该 pacing rate 规定 cwnd 指示的一窗数据的数据包之间，以多大的时间间隔发送出去。[i]

◆ HPCC:

HPCC（High Precision Congestion Control）是一种用于大型高速网络的新型流控机制，它主要致力于实现以下三个目标：超低延迟、高带宽、高稳定性。

HPCC 利用网络内遥测技术(INT)来获取精确的链路负载信息，并精确地控制流量。相比于传统的在端上调节流量，以维持网络最佳平衡状态的拥塞控制方法，HPCC 的核心理念是利用精确链路负载信息直接计算合适的发送速率；HPCC 速率更新由数据包的 ACK 驱动。通过处理 INT 信息，HPCC 可以快速利用空闲带宽，同时避免拥塞，并可以保持接近于零的网络内队列，实现超低延迟。

HPCC 是在高性能的云网络环境下，对现有的拥塞控制的一种替代方案。它可让网络中的报文稳定的、以微秒级的延迟传输。当前主流的拥塞控制算法主要依赖于端的信息(如丢包信息，RTT)做拥塞控制，而 HPCC 则运用了网络设备提供的细粒度负载信息而设计了全新的拥塞控制算法来其解决 Bufferbloat 问题。

[i] https://zhuanlan.zhihu.com/p/383910510