实验报告

数据包队列管理实验

一、实验内容

了解为什么需要数据包队列以及数据包队列的原理、功能，了解如何根据环境设置数据包队列的大小。认识数据包队列过大过小引起的问题，了解 BufferBloat 问题的现象和原因。学习解决 BufferBloat 问题的三种方法：改变队列大小、改进传输控制策略、改进队列管理策略。了解 2 种典型的队列管理策略：RED (Random Early Detection)和 CoDel (Controlled Delay)。

最后根据给定的实验环境，重现 BufferBloat 问题。改变数据包队列大小，观察其对拥塞窗口值(cwnd)、队列长度(qlen)、往返延迟(rtt)、吞吐率的影响。使用不同队列管理策略，对比在动态带宽下的往返延迟(rtt)，重现讲义中实验结果。

二、实验流程

1. 重现 BufferBloat 问题，得到 CWND、Qlen、RTT 这 3 个时间曲线图。
2. 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。
3. 重现不同队列管理策略在动态带宽下的往返延迟结果。

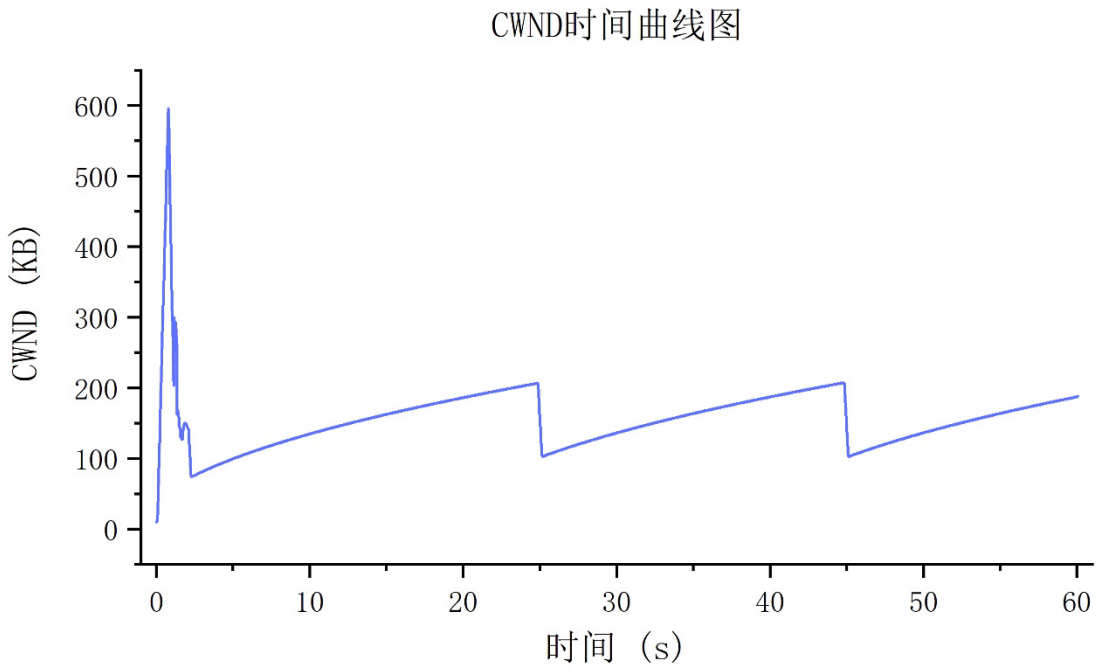
三、实验结果及分析

（一）重现 BufferBloat 问题

1、重现 CWND、Qlen、RTT 实验结果

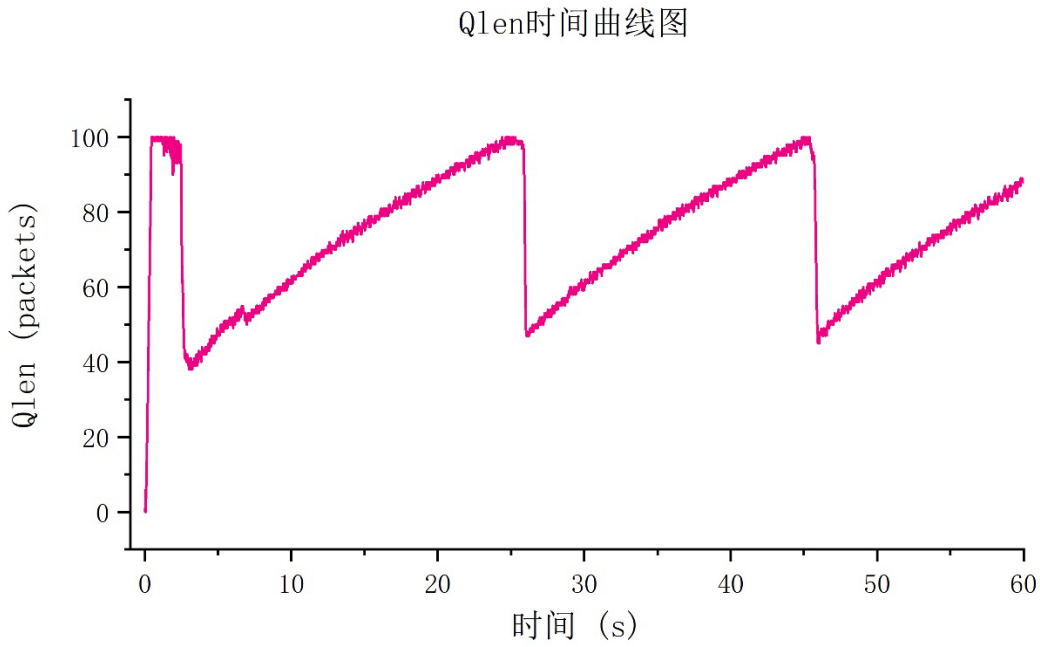
（1）CWND 实验结果

取 qmax=100，作出 cwnd 与时间的曲线图如下，与讲义中实验结果一致。



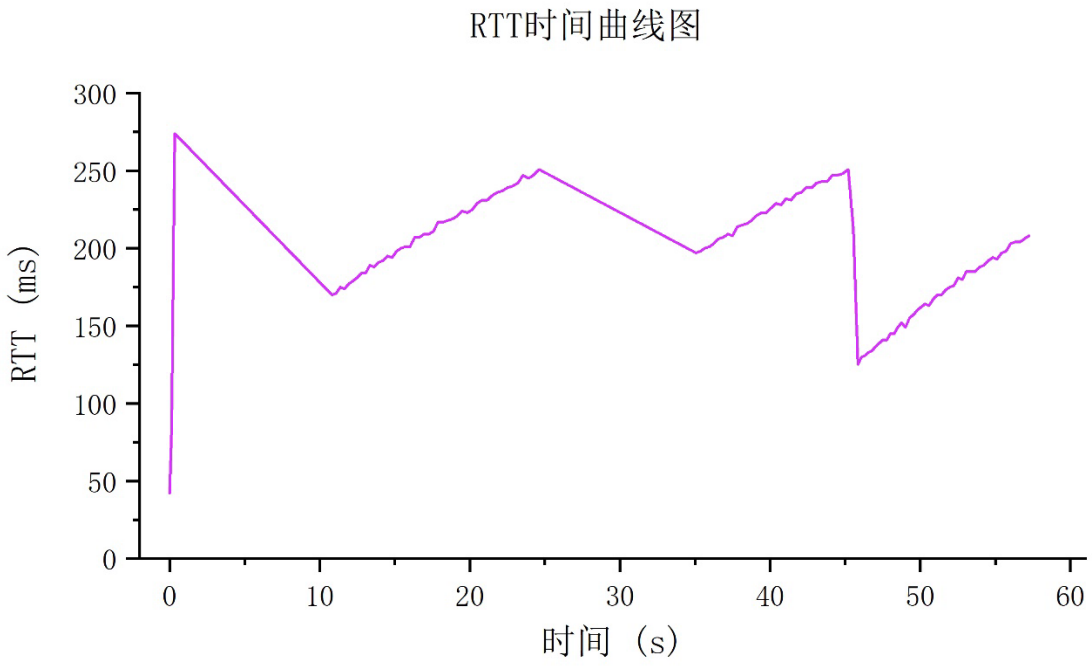
（2）Qlen 实验结果

取 qmax=100，作出 qlen 与时间的曲线图如下，与讲义中实验结果一致。



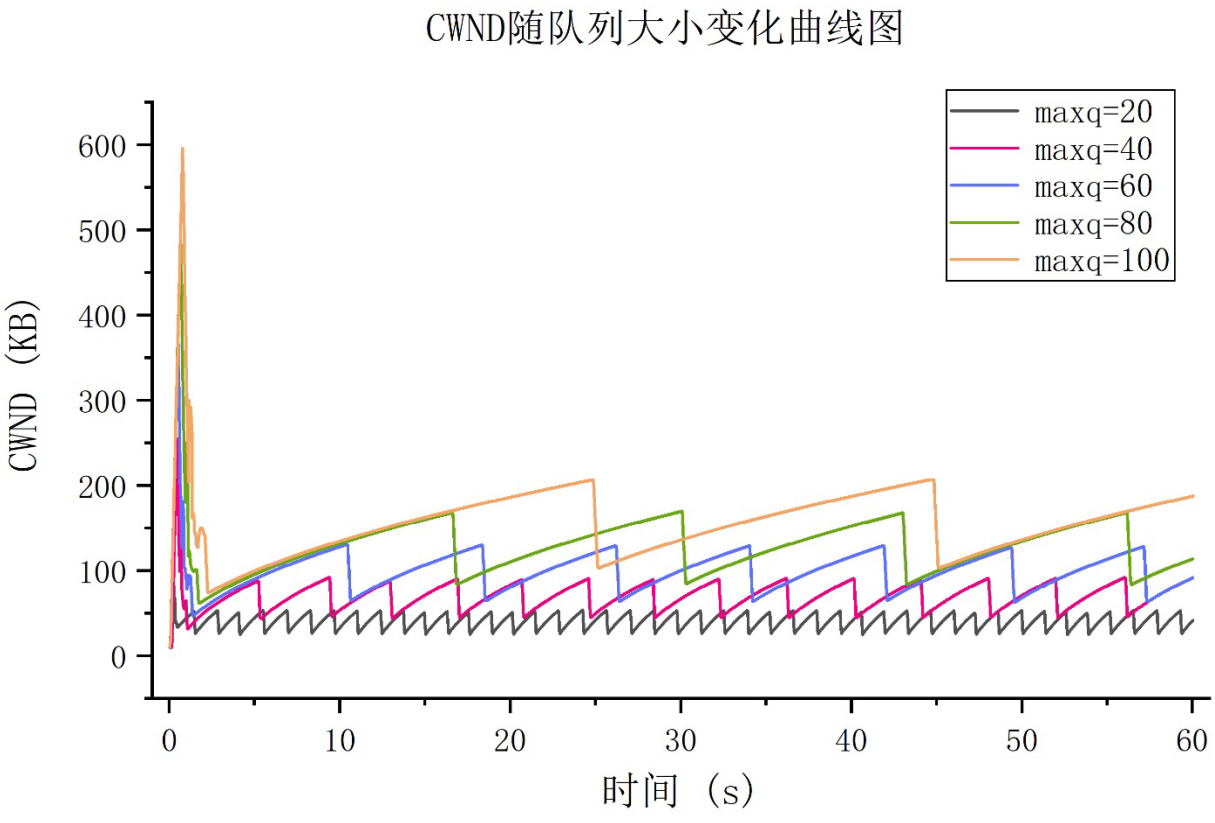
（3）RTT 实验结果

取 qmax=100，作出 rtt 与时间的曲线图如下，因为 RTT 结果输出不稳定，讲义中图像有更多锯齿，与讲义中实验结果在曲线走势上基本一致。



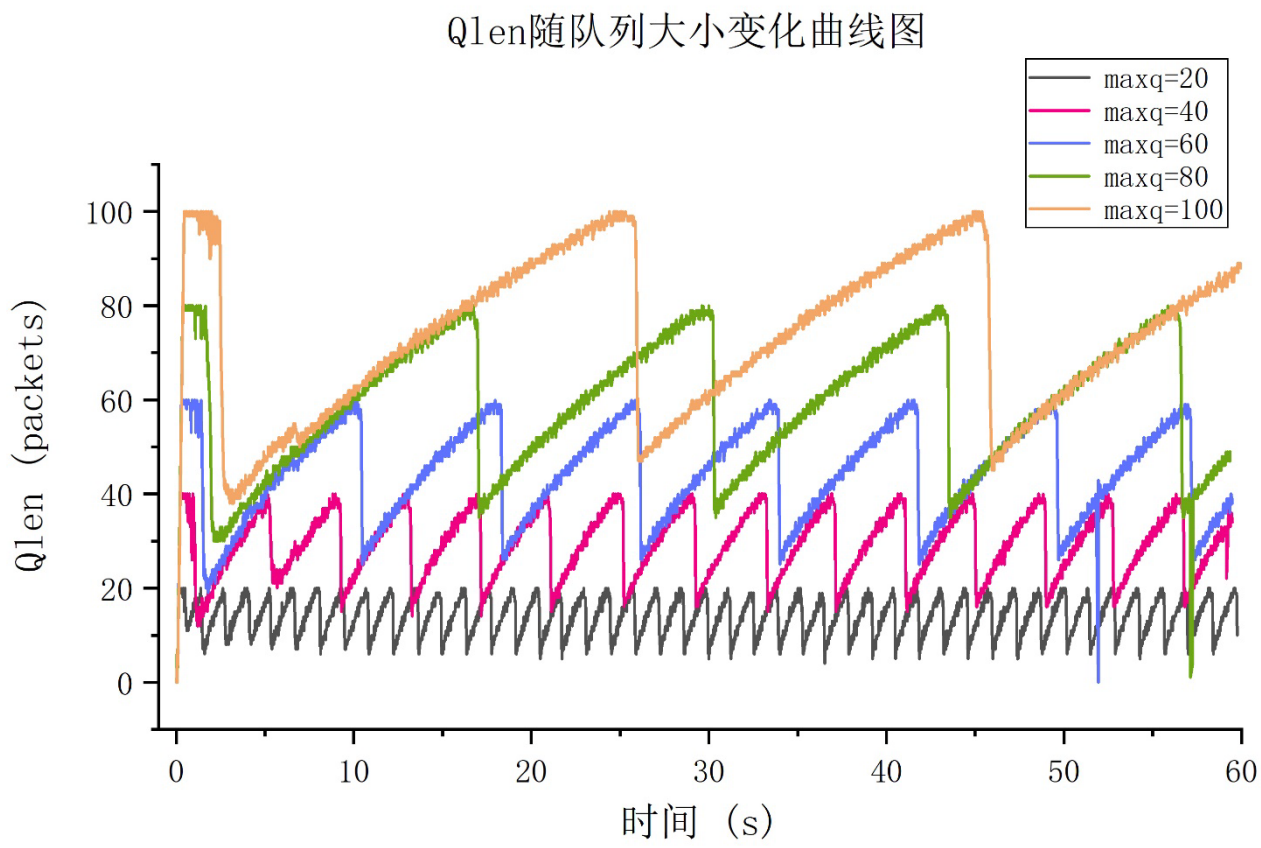
2、观察队列大小对 CWND、Qlen、RTT 和吞吐率图像的影响

（1）CWND 随队列大小变化实验结果



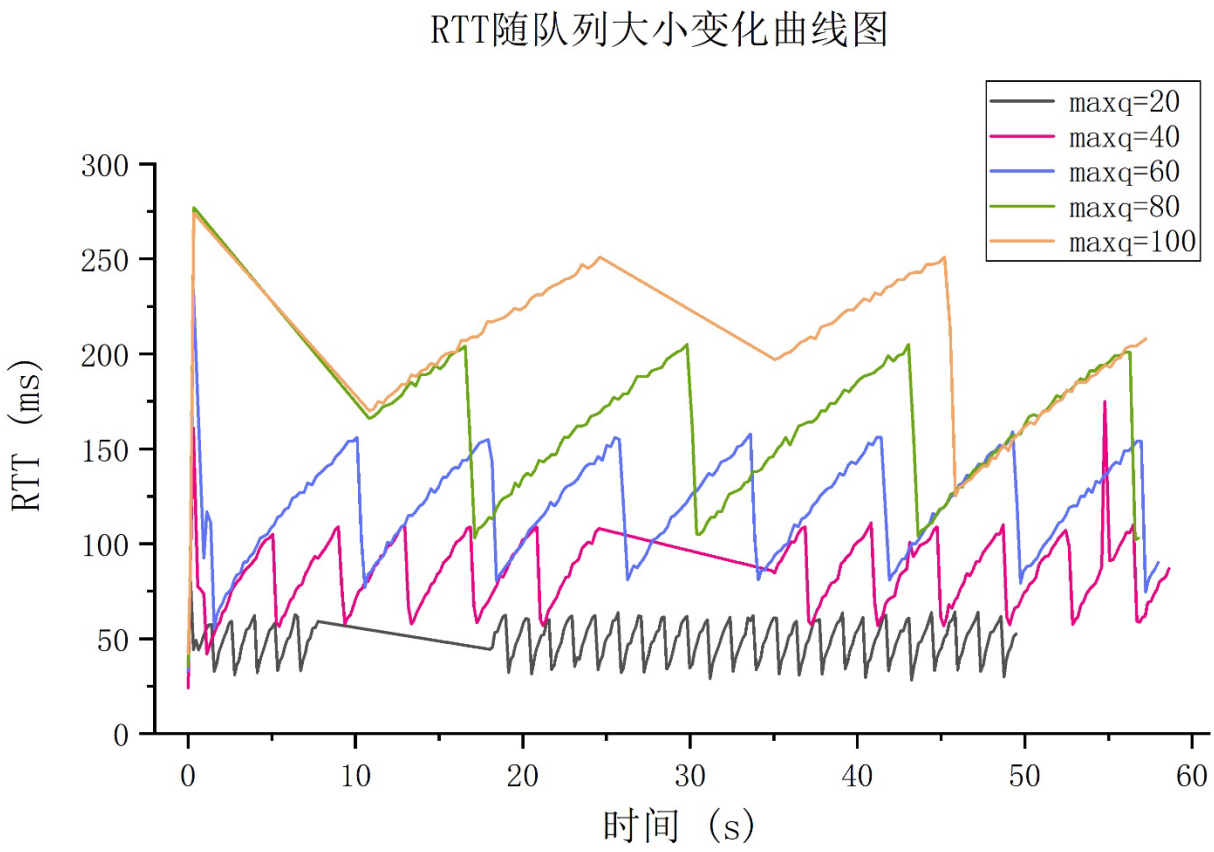
取 qmax=20、40、60、80、100，作出 cwnd 与时间的曲线图如上图所示。以 maxq=20 为基准，我们看到 cwnd 随时间呈现周期性变化，周期较小，变化幅度也较小。而随着 maxq 增大，周期变大，每个周期的变化幅度也加剧。平均 CWND 随 maxq 逐渐增大，BufferBloat 问题加剧。

（2）Qlen 随队列大小变化实验结果



取 qmax=20、40、60、80、100，作出 qlen 与时间的曲线图如上图所示。以 maxq=20 为基准，我们看到 qlen 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 Qlen 随 maxq 逐渐增大，BufferBloat 问题加剧。

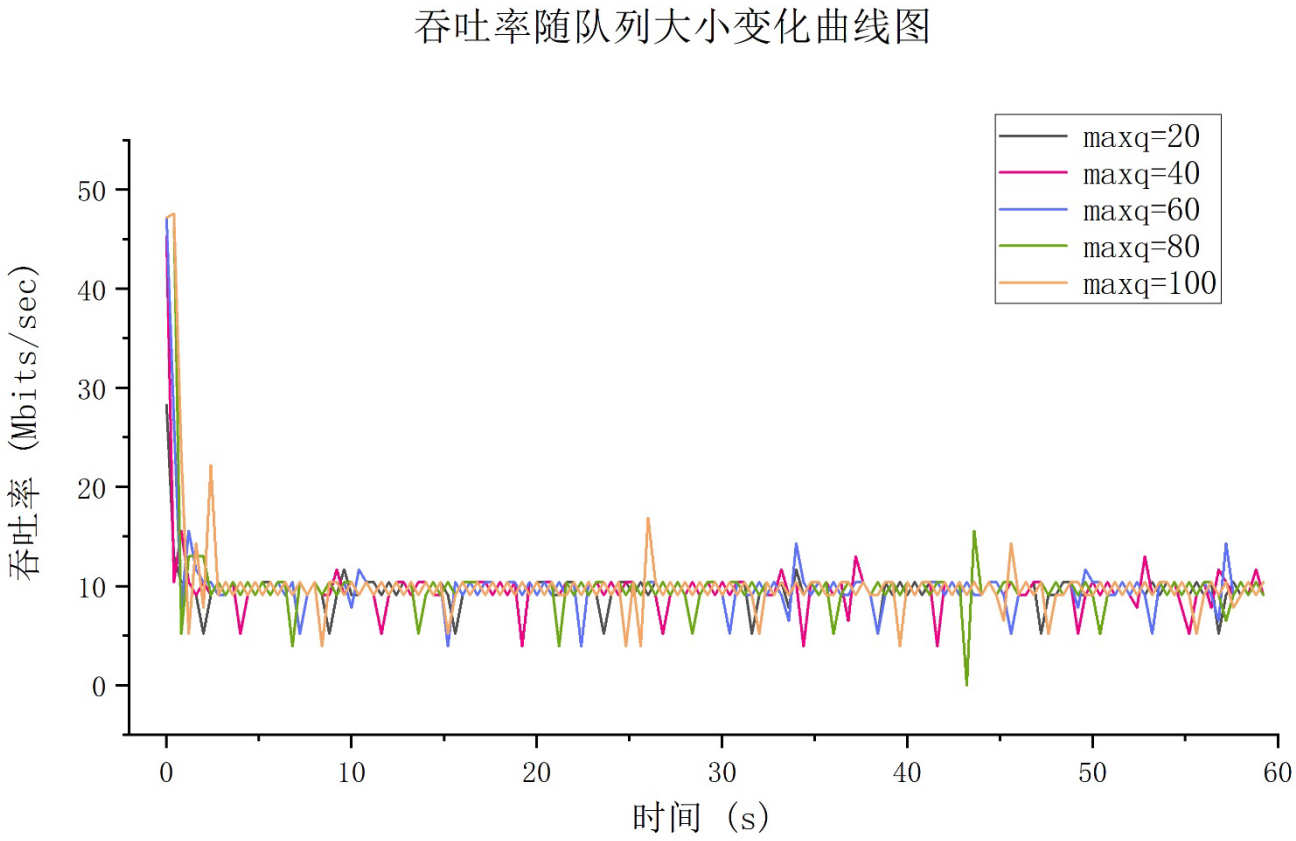
（3）RTT 随队列大小变化实验结果



取 qmax=20、40、60、80、100，作出 rtt 与时间的曲线图如上图所示。以 maxq=20 为基准，我们看到 rtt 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 RTT 随 maxq 逐渐增大，BufferBloat 问题加剧。注意到曲线中有些突变的部分，例如 maxq=20 的 10s 到 20s 段，这是由于 RTT 输出结果不稳定所致。

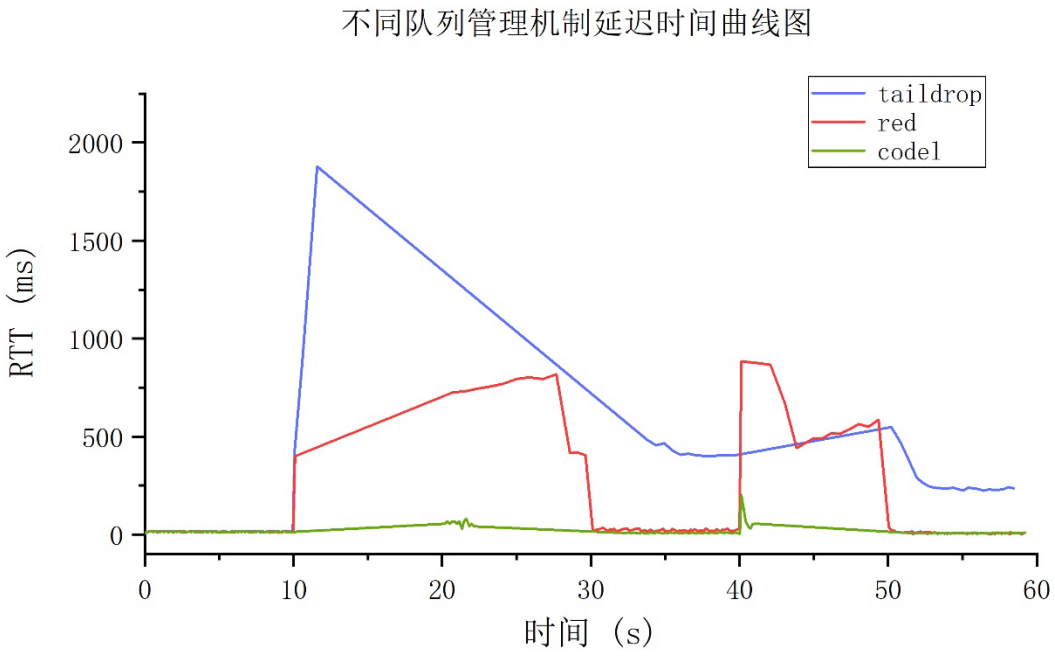
（4）吞吐率随队列大小变化实验结果

取 qmax=20、40、60、80、100，作出吞吐率与时间的曲线图如下图所示。我们可以看到在最初有一个吞吐率峰值，这是刚开始传输产生。之后吞吐率总体平稳，不同队列大小的吞吐率都在 10Mbits/s 附近波动，与链路带宽 10Mbits/s 一致，基本达到满利用率。然后注意到吞吐率有一些周期性的向下波动，这与 cwnd/qlen/rtt 图像中，达到峰值后下降的时间点一致。因此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，其余时刻基本稳定在链路带宽 10Mbits/s 附近。



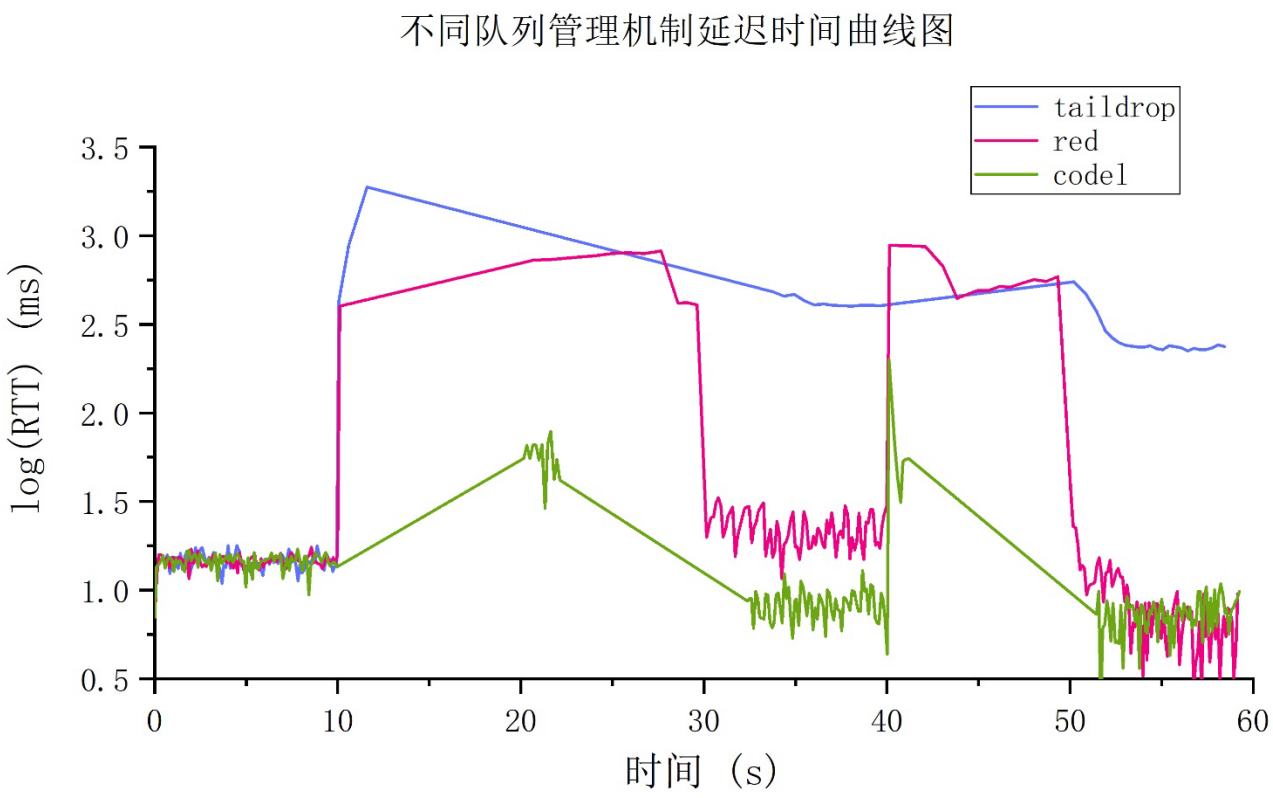
（二）解决 BufferBloat 问题

在给定实验环境下，分别运行 3 种队列管理策略，作出延迟时间随带宽变化的曲线图如下。



可以看出 RED、CoDel 的图线基本与讲义一致，而 taildrop 出现了一个“尖端”的图像，这是由于仿真环境与真实队列有所差异导致的。

将纵坐标取 log 后，再次作图如下。



图像与讲义更为接近。注意到 10-20s 区间，讲义上图像先有一个下降，然后再次上升，这里的图像则上升后保持，之后再下降。造成这一差异的原因可能为：我们的实验中带宽变化速率较快，没来得及反映出下降就再次上升了；仿真环境与真实情况还是有些区别。总体上曲线走势与讲义一致。

四、思考题

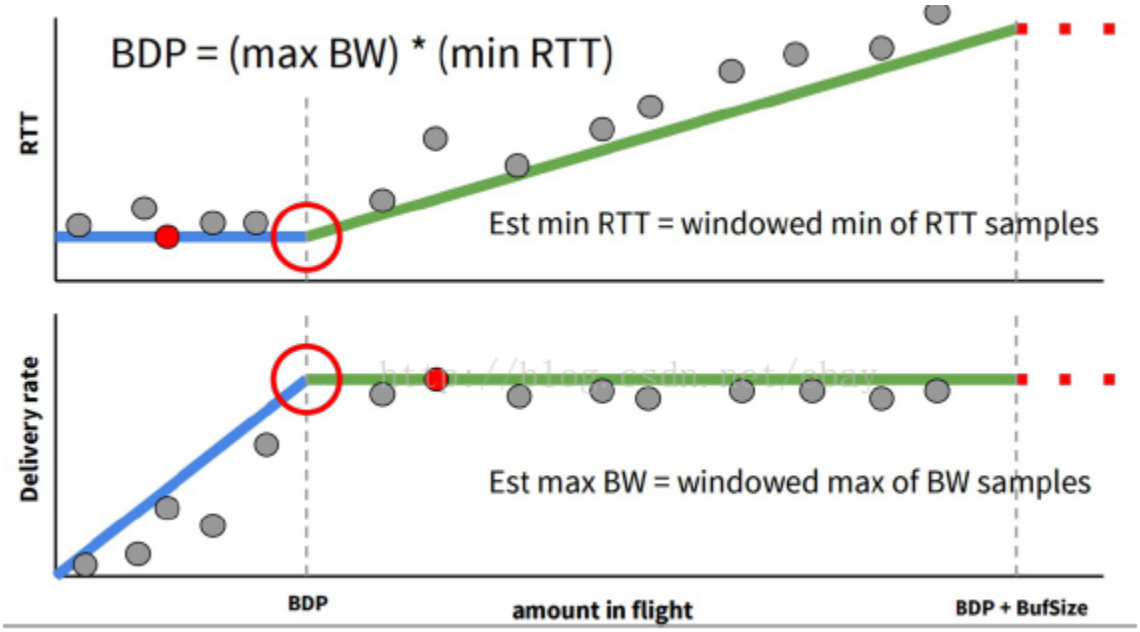
1. 调研分析两种新型拥塞控制机制（BBR [Cardwell2016], HPCC [Li2019]），阐述其是如何解决 Bufferbloat 问题的。

（1）BBR

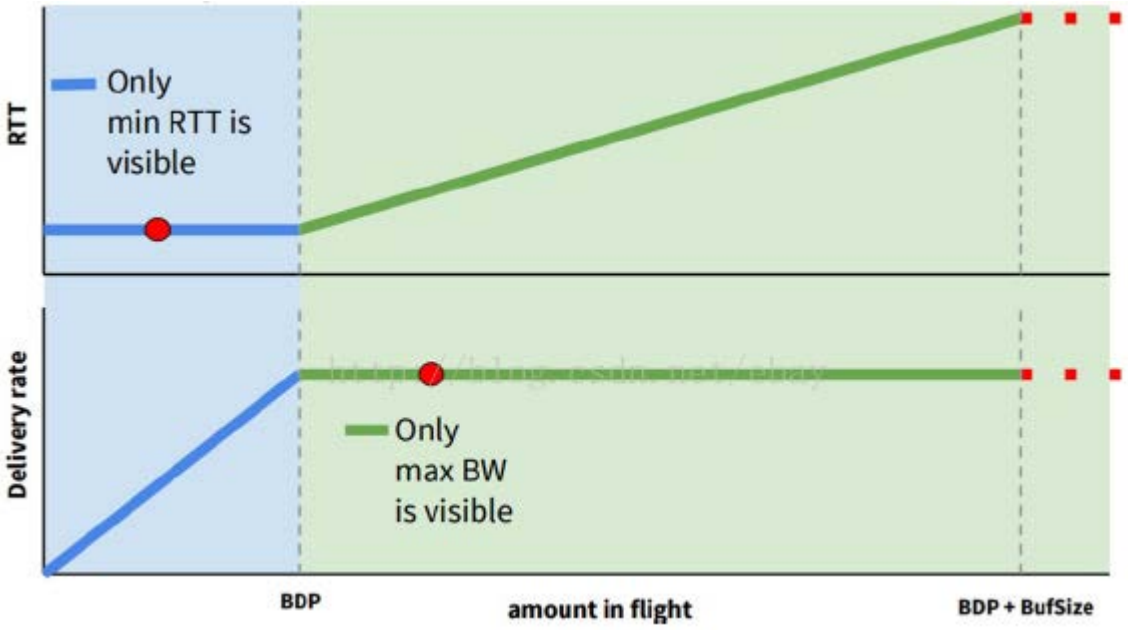
BBR 是 2016 年由 Google 发表的优化 tcp 传输的算法，它极大的提高了 tcp 的 throughput。

1. BBR 思路

首先我们已经知道所谓的最优工作点。



上图红色圆圈所示即为网络工作的最优点，此时数据包的投递率=瓶颈链路带宽，保证了链路被 100%利用；在途数据包总数=BDP(时延带宽积)，保证未占用 buffer。



然而 max BW 和 min RTT 不能被同时测得。要测量最大带宽，就要把瓶颈链路填满，此时 buffer 中有一定量的数据包，延迟较高。要测量最低延迟，就要保证 buffer 为空，网络中数据包越少越好，但此时带宽较低。

BBR 的解决办法是：

交替测量带宽和延迟，用一段时间内的带宽极大值和延迟极小值作为估计值。

2. BBR 细节设计

BBR 规定四个状态（启动、排空、带宽探测、时延探测）。

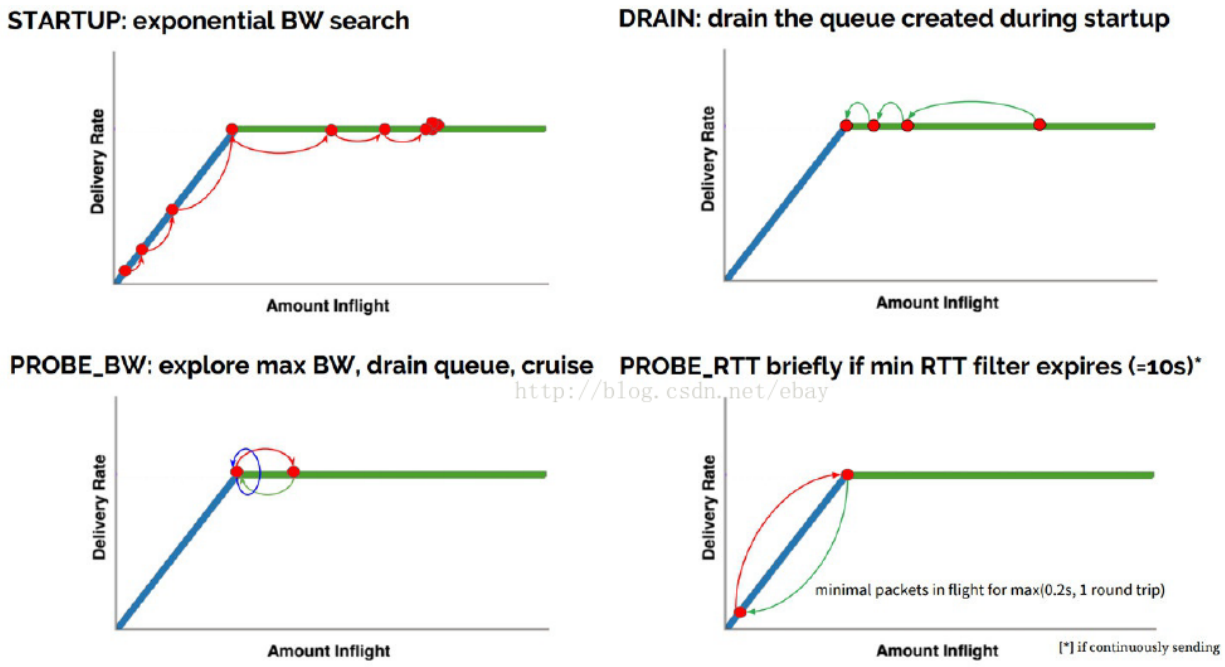
（1）当连接建立时，BBR 采用类似标准 TCP 的 slow start，指数增加发送速率，目的也是尽可能快的占满管

道，经过三次发现投递率不再增长，说明管道被填满，开始占用 buffer 它进入排空阶段（事实上此时占的是三倍带宽*延迟）。

（2）在排空阶段，指数降低发送速率，（相当于是 startup 的逆过程）将多占的 2 倍 buffer 慢慢排空。

（3）完成上面两步，进入稳定状态后，BBR 改变发送速率进行带宽探测：先在一个 RTT 时间内增加发送速率探测最大带宽，如果 RTT 没有变化，减小发送速率排空前一个 RTT 多发出来的包，后面 6 个周期使用更新后的估计带宽发包。

（4）还有一个阶段是延迟探测阶段：BBR 每过 10 秒，如果估计延迟不变，就进入延迟探测阶段。为了探测最小延迟，BBR 在这段时间内发送窗口固定为 4 个包，即几乎不发包，占整个过程 2%的时间。



其中带宽探测占据 BBR 绝大部分时间。在 startup 阶段，BBR 已经得到网络带宽的估计值。在带宽探测阶段，BBR 利用一个叫做 cycle gain 的数组控制发送速率，进行带宽的更新。cycle gain 数组的值为 5/4, 3/4, 1, 1, 1, 1, 1, 1，BBR 将 max BtlBW*cycle gain 的值作为发送速率。一个完整的 cycle 包含 8 个阶段，每个阶段持续时间为一个 RTprop。

故如果数组的值是 1，就保持当前的发送速率，如果是 1.25，就增加发送速率至 1.25 倍 BW，如果是 0.75，BBR 减小发送速率至 0.75 倍 BW。

对于多条独立的 BBR 数据流分享瓶颈链路的情况：一条连接独占整个链路时，它的可用带宽近似为链路的物理带宽，n 条连接共享链路时，最理想最公平的情况就是 BW/n。每条连接的 startup 阶段都会尝试增加带宽，所以吞吐量会有一个向上的尖峰，已经在链路中的连接会检测到拥塞而减小自己的发送速率，吞吐量会下降。

最后通过反复的带宽探测，他们都会趋于收敛，带宽得到均分。

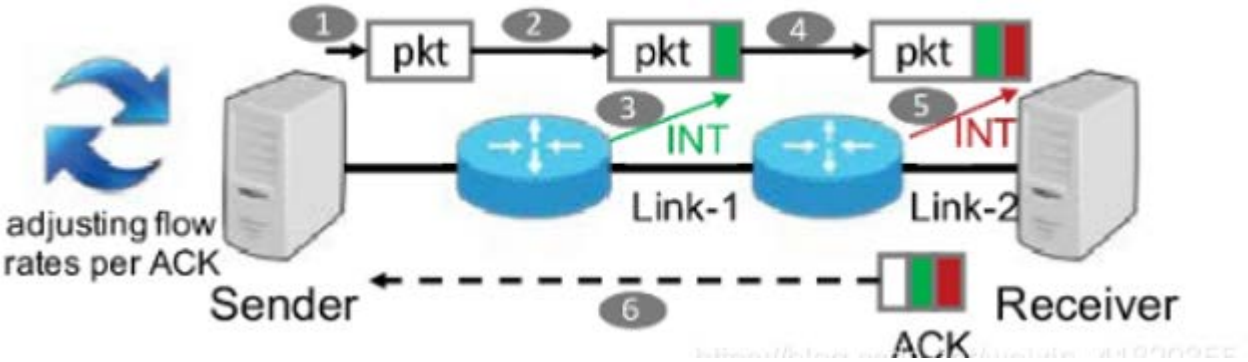
（2）HPCC

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。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。

1. 总体模型

如图所示，发送方发送的每个数据包将由接收方确认。在从发送器到接收器的数据包传播过程中，沿路径的每个交换机利用其 INT 功能来插入一些数据，这些数据包括时间戳（ts），队列长度（qLen），发送字节数（tx 字节）和链路带宽容量（B）的信息。当接收方获取数据包时，它会将记录的所有元数据复制到 ACK 消息中发送回发送方。每次收到带有网络负载信息的 ACK 时，发送方决定如何调整其流速。



2. 细节设计

飞行包的大小直接对应了整个链路利用率。当飞行包的大小小于带宽延时积（BDP）的时候，说明信息量不能填满这样的“管道”，则链路未被充分利用，即还没有发生拥塞。

$$I < B \times T$$

同理当当飞行包的大小大于带宽乘 RTT 的时候，总吞吐量将超过链路带宽，发生拥塞。

$$I >= B \times T$$

所以，HPCC 使用总的飞行包的大小来检测是否发生拥塞，并且目标是控制 I 略小于 B * T。飞行包大小由队列和管道中的数据包组成，有公式:

$$total\ in\ flight\ bytes = qlen + txRate \times T$$

这里设计一个参数 η，它是一个非常接近 1 的数，一般把它取值为 95%。对于每一个链接 j，发送方都会通过参数 K_j 来对发送窗口进行调整：

$$K_j = \frac{I_j}{\eta * B_j * T} = \frac{U_j}{\eta}$$

对 U_j 进一步展开，有

$$U_j = \frac{I_j}{B_j * T} = \frac{qlen + txRate * T}{B_j * T} = \frac{qlen}{B_j * T} + \frac{txRate}{B_j}$$

然后再通过公式：

$$W_i = \frac{W_i}{maxj(K_j)} + W_{AI} = \frac{W_j}{maxj(U_j)/\eta} + W_{AI}$$

最终达到调整发送窗口W，使得 I 略小于 $B * T$ 的最终目的。这里的 W_{AI} 是一个为了确定公平性而存在的一个很小的值。W_{AI}越大，总的吞吐量越大，这个公平性也会越好。但是整体队列长度也会增加。

3. 难点设计

（i）延迟反馈容忍

延迟反馈容忍问题是说由于阻塞的出现导致携带阻塞标记的报文无法及时到达发送端，从而让整个反馈过程没有办法及时进行。

HPCC 中使用窗口 W 解决这个问题，让一次发送的报文限制在一个范围，也就是窗口范围之内，这样的话不管是否发生拥塞反馈都可以及时进行。

（ii）过度反应

如果 HPCC 针对每一个报文都去改变一次窗口大小，那在一个 RTT 之中的窗口会不断的振荡，会让传输的状态变得不够稳定。基于此，论文引入了窗口 W_{ci}，以每一个 RTT 为基础更新窗口状态。只有当接收到 W_{ci} 发送的第一个数组包的 ACK 的时候，发送方才会以 $W_{ci} = W_i$ 来进行更新窗口。具体公式更新如下：

$$W_i = \frac{W_{ci}}{maxj(K_j)} + W_{AI}$$

五、实验总结

通过本次实验，我了解了数据包队列的原理和工作方式，以及如何根据环境设置数据包队列的大小。然后了解到 BufferBloat 问题的现象和原因，知道了解决 BufferBloat 问题的三种方法：改变队列大小、改进传输控制策略、改进队列管理策略。在实验环节，通过改变队列大小、使用不同队列管理策略，直观的认识到这两种方法是如何解决 BufferBloat 问题的。最后在调研环节，对两种新型的传输控制策略进行了调研，这让我对传输控制策略解决 BufferBloat 问题有了更多的了解。