Lab10 实验报告

一. 实验任务

1. 重现 BufferBloat 问题。
2. 解决 BufferBloat 问题。

二. 实验设计

1. BufferBloat 问题的复现

a. CWND 随时间和队列大小变化：

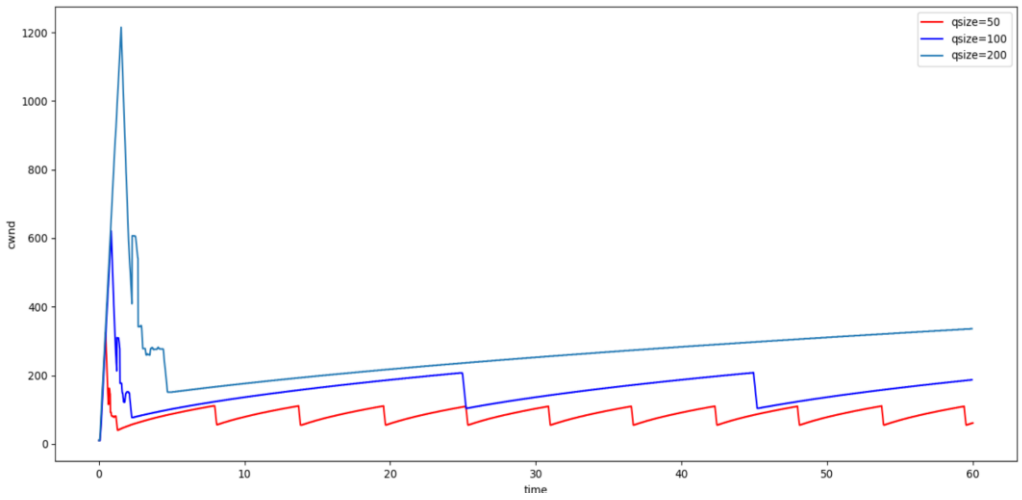

图 1：CWND 随队列最大大小的变化

我们可以看出，CWND 在刚开始达到拥塞峰值，且峰值大小和队列长度线性相关，qlen=50 时候为 300，qlen=100 时为 600，qlen=200 时有 1200。接着拥塞队列很快落回 qlen 大小左右，并随时间上下波动，队列越小，波动周期越小，波动越明显。

b. 队列长度随时间和队列大小变化

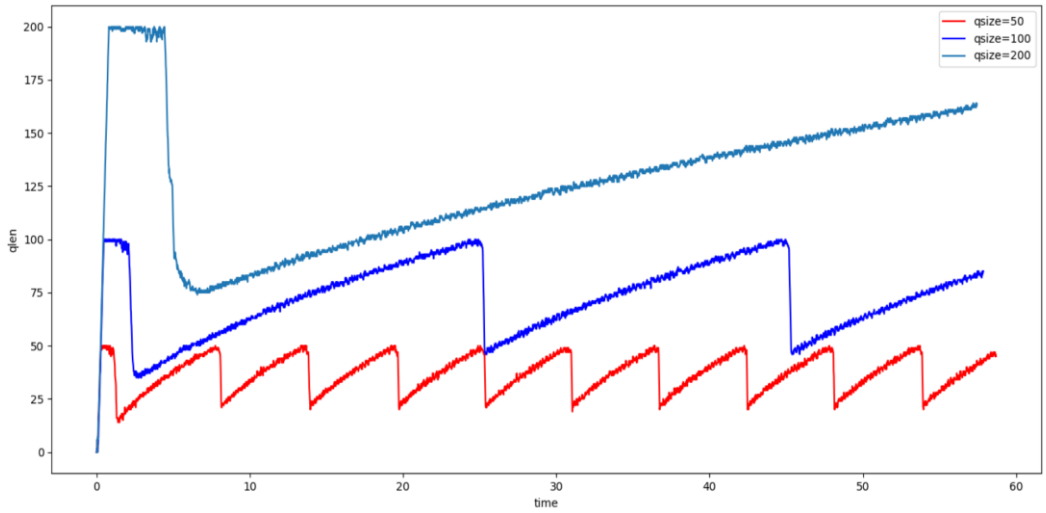

图 2：队列长度随时间和队列最大大小的变化

从图中我们可以看出，队列长度也是从一开始达到峰值，即队列的最大大小。之后迅速下降，到最大长度的 40%左右，接着再慢慢增长，直到接近最大队列长度时又下降到最大长度的 40%左右，循环往复。

c. RTT 随时间和队列大小的变化

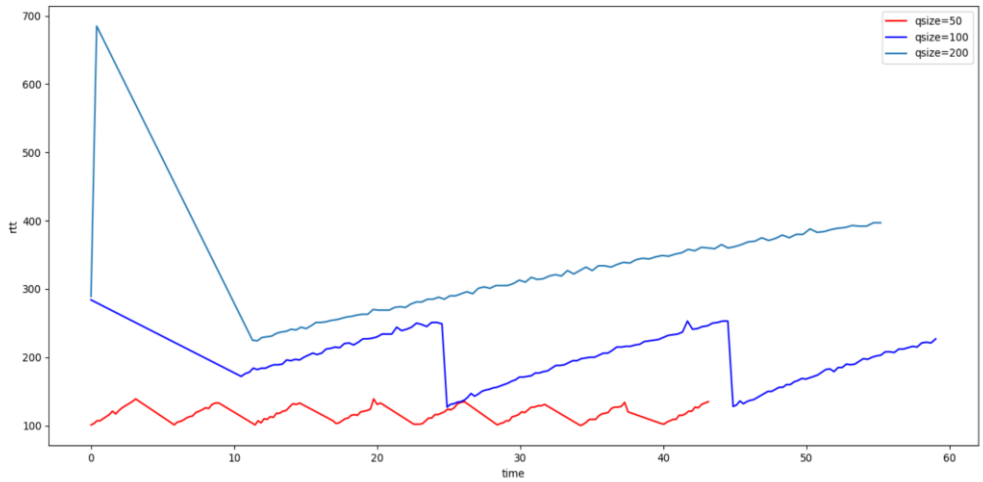

图 3：RTT 随队列最大大小和时间的变化

我们可以看出，结合图二和图三，RTT 随队列缩短而缩短，随队列加长而加长。

d. Iperf 吞吐率的变化

结果如下：

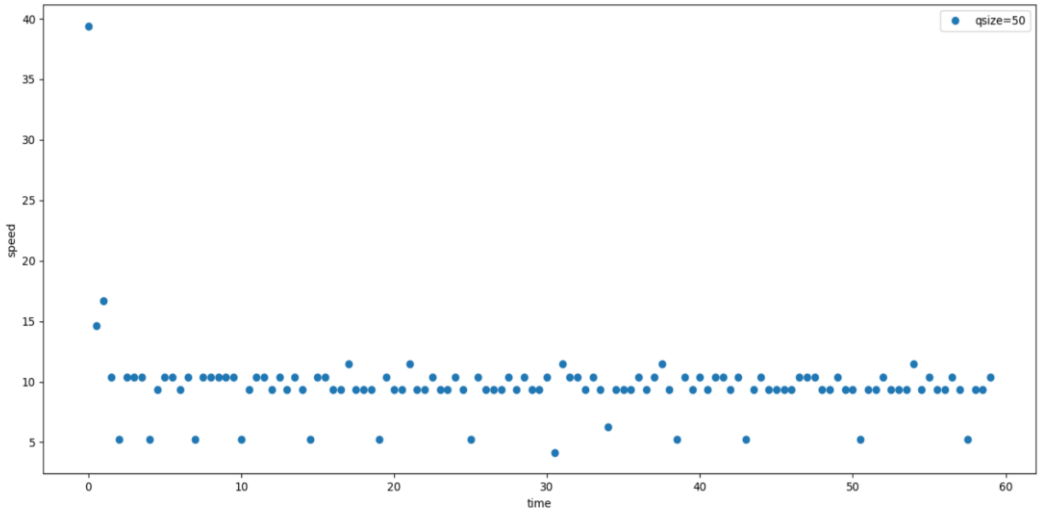

图 4：qsize=50 时候传输速率随时间变化

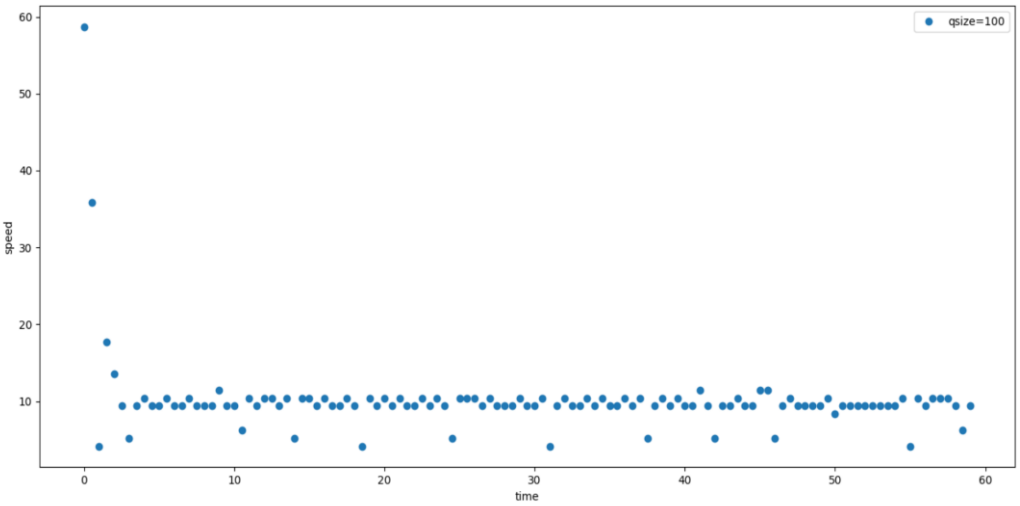

图 5：qsize=100 时候传输速率随时间变化

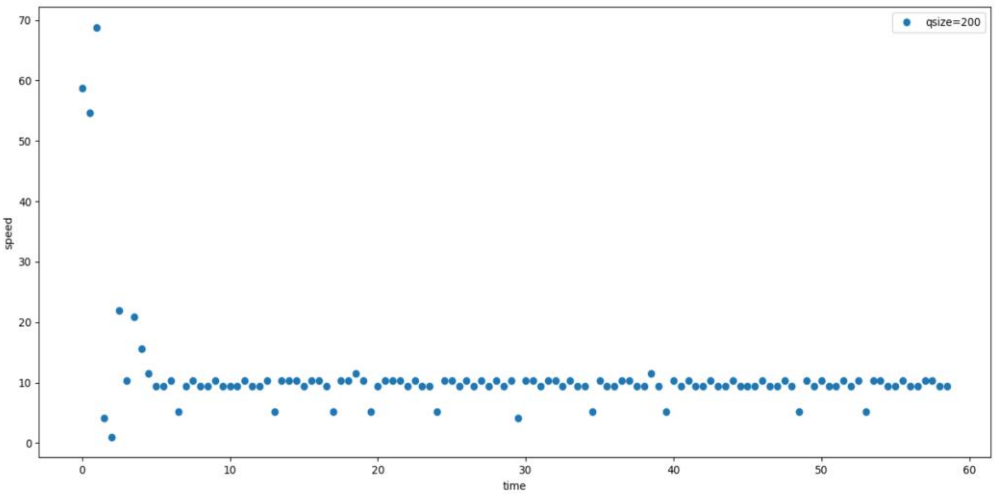

图 6：qsize=200 时候传输速率随时间变化

我们可以看出，刚开始传输速率达到顶峰，后来随着数据包的到来很快下来。之后三种队列长度的带宽基本就一样了。

e. 总结分析

接收队列的长度受到拥塞窗口的影响，因其长度取决于发送方的发包速率，而发包速率又同时受到滑动窗口和拥塞窗口的制约。因此，当拥塞窗口较大时，可以实现更高的发送速率，接收队列的增长也会随之加快。但是从实验结果中可以看到随着拥塞窗口增大，接收队列增速反而减缓，这可能是因为时延增大引起的。

另外，随着接收队列增长，数据包的接收延时会线性增长。当队列长度很长时，会产生 BufferBloat 问题。就像实验结果图三可以看到，RTT 随队列增大而增大。

2. BufferBloat 问题的解决

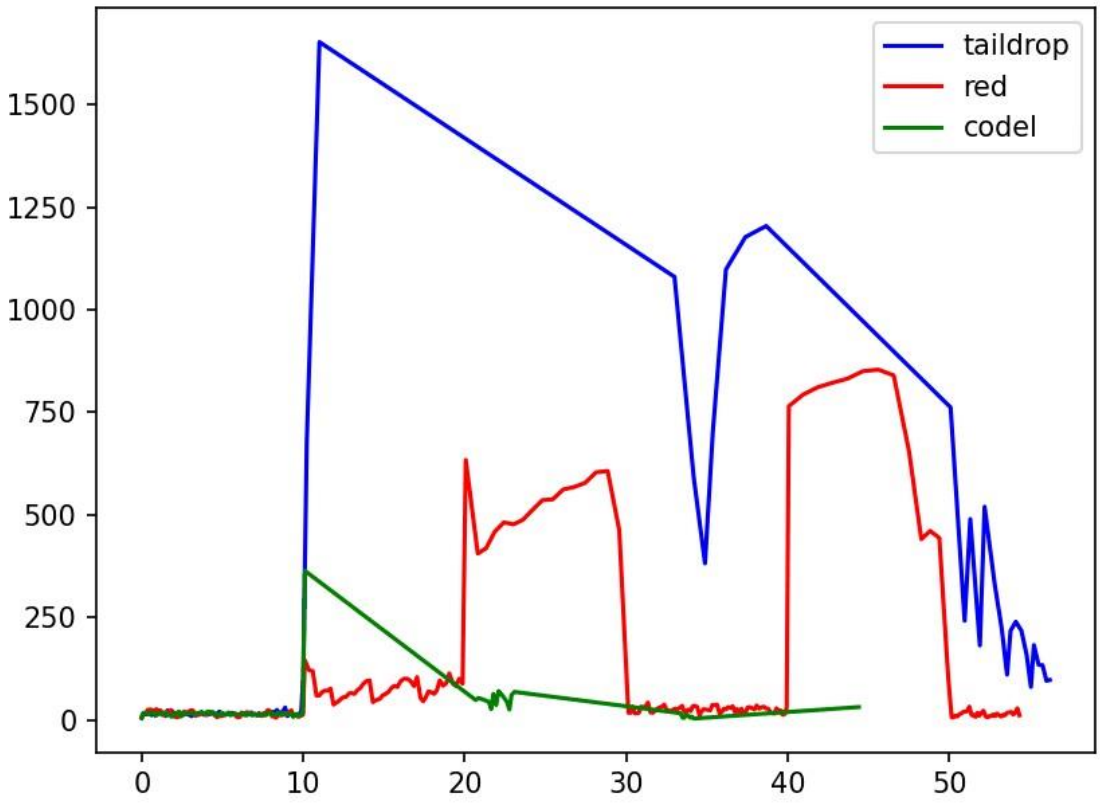

图 7：不同算法的比较

我们可以看出 TailDrop 的延时会高于 Red 和 Codel 算法，Red 和 Codel 算法相比，Codel 算法效率较高。

三． 实验思考

1. BBR 算法

TCP BBR(Bottleneck Bandwidth and Round-trip propagation time)是由 Google 设计，并于 2016 年发布的拥塞算法，以往大部分拥塞算法是基于丢包来作为降低传输速率的信号，而 BBR 基于模型主动探测。

该算法使用网络最近出站数据分组当时的最大带宽和往返时间来创建网络的显式模型。数据包传输的每个累积或选择性确认用于生成记录在数据包传输过程和确认返回期间的时间内所传送数据量的采样率。

TCP WestWood 算法： 其通过对确认包测量来确定一个合适的发送速度，并以此调整拥塞窗口和慢启动阈值。TCP WestWood 算法也是基于带宽和延时乘积进行设计的，但是带宽和延时两个指标无法同时测量，因为这两个值是有些矛盾的极值，要测量最大带宽就要发送最大的数据量但是此时的 RTT 可能会很大，如果要测量最小的 RTT 那么久意味着数据量非常少最大带宽就无法获得。

TCP BBR 算法采用交替采样测量两个指标，取一段时间内的带宽极大值和延时极小值作为估计值。

2. HPCC 算法

论文认为如今高速网络传输普遍存在收敛慢，始终有 standing queue，CC 的参数调优困难等问题。导致这三个问题的根本的原因是现在的拥塞控制算法只有粗粒度的反馈。

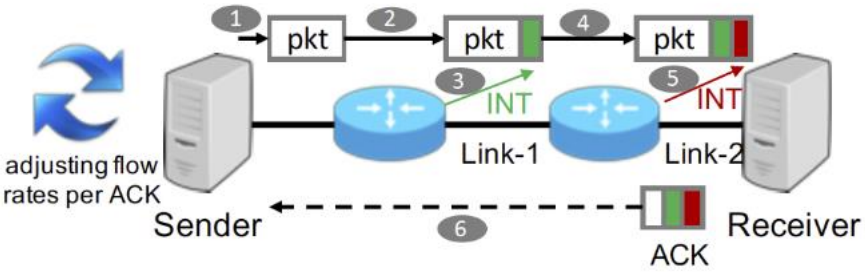

图 8：HPCC 核心思想

HPCC 的核心思想是借助于 INT 提供的细粒度信息，如 timestamp (ts), queue length (qLen), transmitted bytes (txBytes), the link bandwidth capacity (B) 等，去精确计算流速率。

HPCC 的两个核心设计点：基于 inflight bytes 的发送端速率控制、RTT-based 与 ACK-based 反馈结合避免 overreaction.