Lab of Computer Network: BufferBloat **Fall 2024**

Report 6 — October 10

Lecturer: Wu Qinghua *Completed by: Zhang Jiawei*

6.1 实验内容

1. 根据附件材料中提供的脚本，重现 PPT 中 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) 的实验结果；
2. 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；
3. 根据附件材料中提供的脚本，重现 PPT 中 Tail Drop、RED、CoDel 三种队列管理算法的实验结果。

6.2 实验结果与分析

6.2.1 重现 BufferBloat 实验

取 `maxq` 为 100，在终端中输入：

```
sudo python3 reproduce_bufferbloat.py --maxq 100
```

程序会输出三个.txt 文件，我使用了 python 脚本读取这三个文件中的内容，获得相应数据，并输出所绘制曲线图。重现实验结果如下：

1. CWND 结果

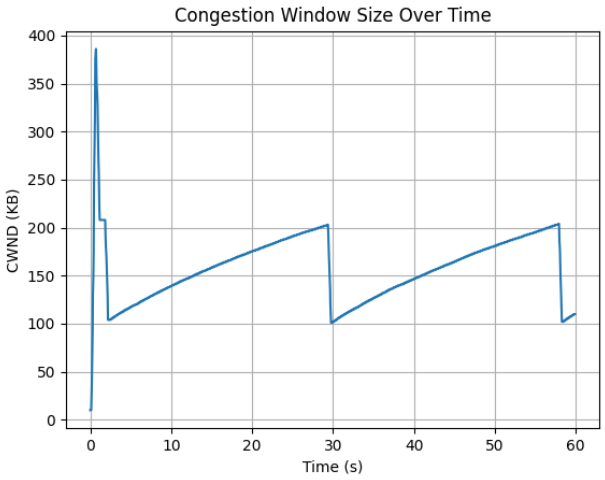

图 **6.1.** CWND 重现结果

与 PPT 中的结果十分一致。

2. QLEN 结果

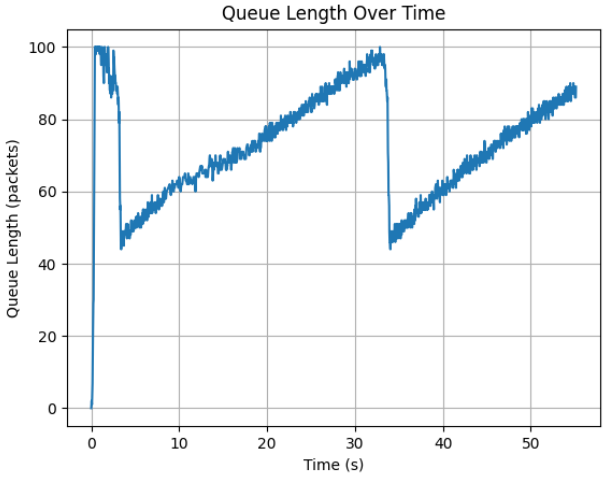

图 **6.2.** QLEN 重现结果

与 PPT 中的结果十分一致。

3. RTT 结果

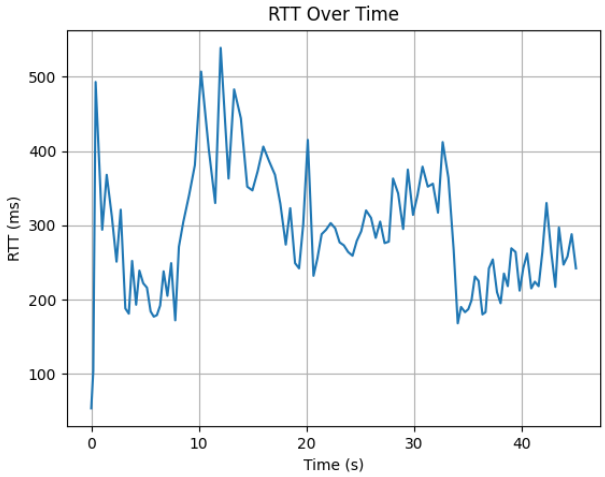

图 **6.3.** RTT 重现结果

与 PPT 中的曲线走势较为一致，均有显著周期性的上升下降趋势。但由于相邻 ping 程序间隔会发生变动，导致实验结果与 PPT 中的结果有一定差异。

6.2.2 变化队列大小对 iperf 吞吐率和指标的影响

分别取 `maxq` 为 100、80、60、40、20，实验结果如下：

1. CWND 结果

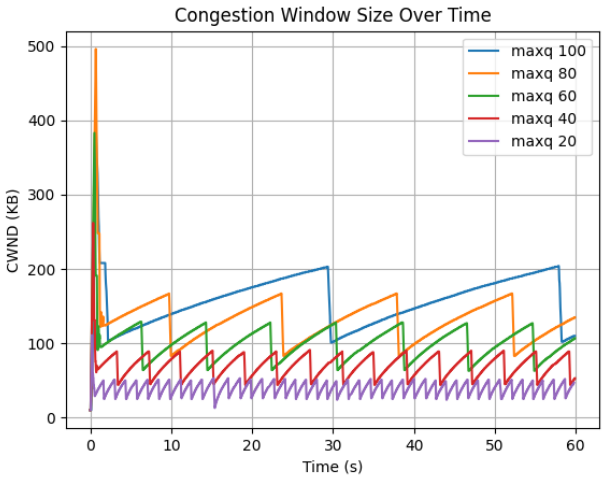

图 **6.4.** CWND 变化 maxq 结果

刚开始测试时，拥塞窗口达到峰值，随后迅速回落至队列大小相近水平，随时间推移，拥塞窗口逐渐增大，接着随着丢包，拥塞窗口迅速回落，循环往复。

2. QLEN 结果

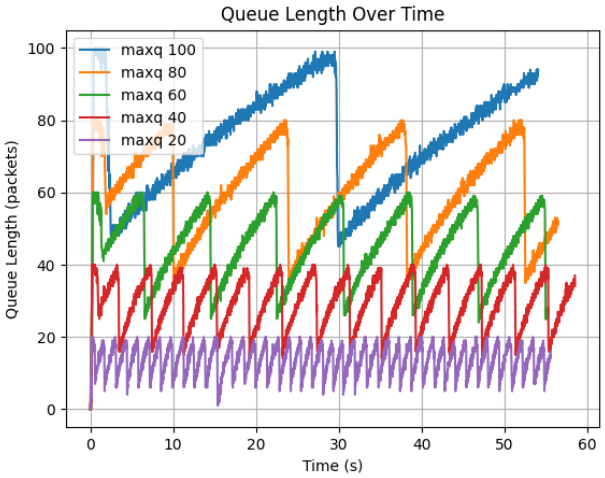

图 **6.5.** QLEN 变化 maxq 结果

同样，刚开始接收到数据包，队列长度迅速增大，当队列长度达到最大值时，开始丢包，队列长度迅速回落，循环往复。随着 maxq 增大，队列长度的峰值逐渐增大，丢包周期和变化幅度也逐渐增大，BufferBloat 现象更加明显。

3. RTT 结果

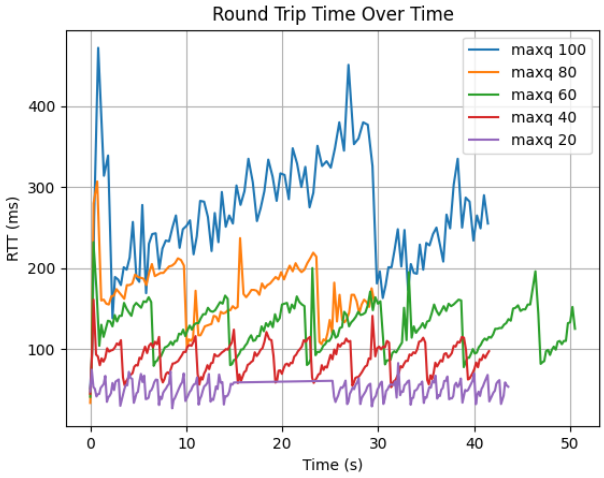

图 **6.6.** RTT 变化 maxq 结果

RTT 随着队列长度的变化，也呈现出周期性的上升下降趋势，且与队列长度的变化趋势一致。随着 maxq 增大，RTT 的峰值逐渐增大，变化幅度也逐渐增大。由于 ping 程序输出不稳定，图中曲线可能出现数据丢失(如图中 maxq 为 20 时的水平线段)。

4. iperf 吞吐率结果

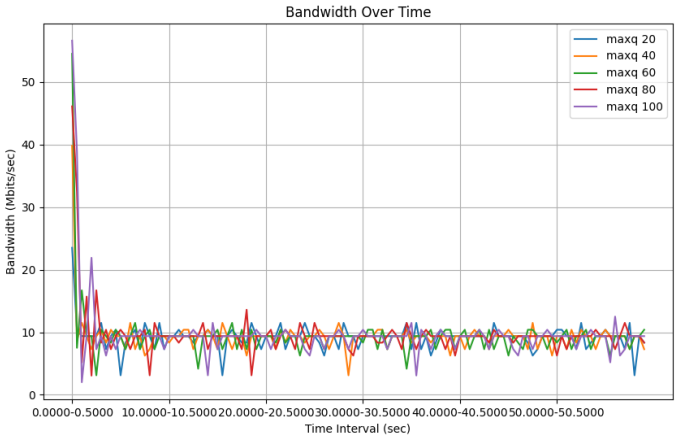

图 **6.7.** iperf 吞吐率变化 maxq 结果

刚开始接收到数据包，iperf 吞吐率也迅速增大，之后迅速回落，稳定在 10Mbps 左右，这正是链路的带宽。此外也观察到吞吐率会有周期性的向下波动，这是由于队列长度的变化导致的丢包，从而导致 iperf 吞吐率的波动。

6.2.3 重现队列管理算法实验

分别使用 Tail Drop、RED、CoDel 三种队列管理算法，在终端中输入：

```
sudo python3 mitigate_bufferbloat.py -a taildrop # or -a red or -a codel
```

实验结果如下：

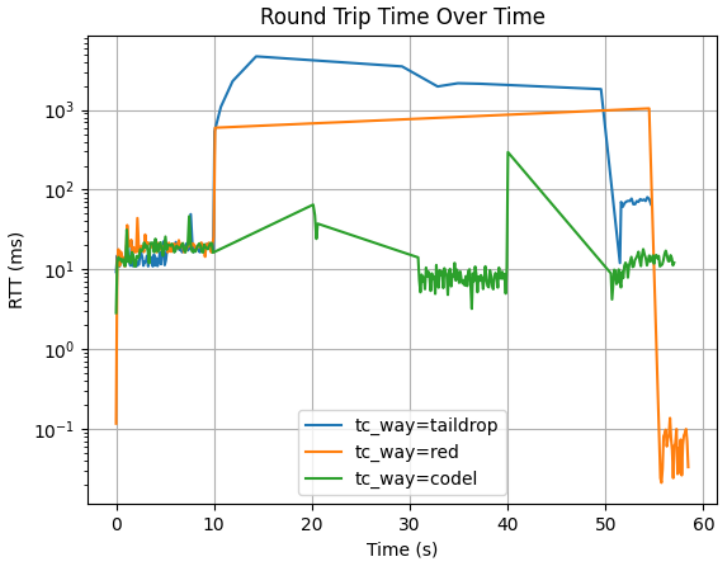

图 **6.8.** RTT 重现队列管理算法实验结果

可以看出，Tail Drop 算法的 RTT 显著高于 RED 和 CoDel 算法，且 CoDel 算法的 RTT 表现更佳。而且 Tail Drop 曲线出现了较稳定峰值，说明实验较为准确。

6.3 实验总结

通过本次实验，我成功重现了 BufferBloat 实验，并且观察到了队列大小对 iperf 吞吐率和指标的影响，这让我更加深入地理解了数据包对了的工作原理和 BufferBloat 现象。此外，我还重现了 Tail Drop、RED、CoDel 三种队列管理算法的实验，发现 CoDel 算法的 RTT 表现更佳，这也让我对队列管理算法有了更深入的了解。