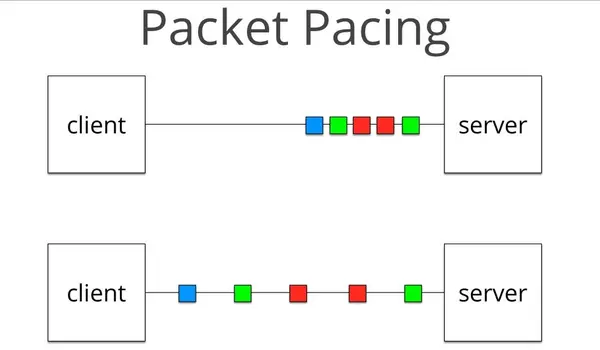
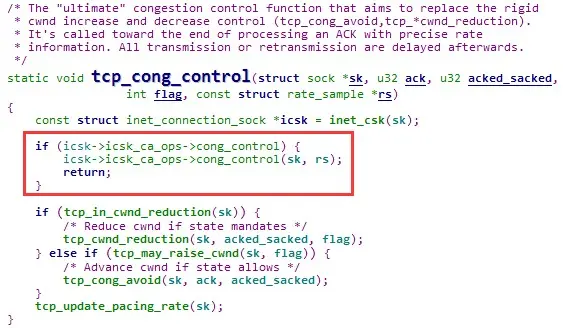
# pacing\_rate

BBR总是在测量最小RTT（10s内），最大Bandwidth（10 Round Trips），并且尽量控制输出到网络的数据包（in-flight）靠近 BDP（without buffer），这样既能保证带宽利用率，又能避免Bufferbloat问题。

现在广泛使用的CUBIC/(new)Reno都是基于丢包的，在算法上重点输出拥塞窗口（cwnd）；  
而BBR输出cwnd和pacing\_rate，且pacing\_rate为主，cwnd为辅，参考



**BBR对TCP的大胆改动：**



红框中是BBR加入时添加的，这里很明显，BBR从TCP接管了充分的控制权。  
从工程实现的角度来看，BBR这个小小的修改把TCP的 可靠传输 / 拥塞控制 解耦了 —— TCP专注于自己的可靠性（当然还有很多其它细节），BBR总是会负责任地告诉你（TCP）现在可以发多少数据，以什么速度发出这些数据。

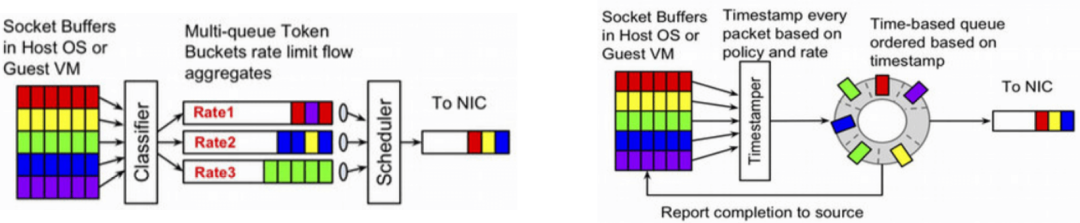
# 影响Pacing rate的性能因素

另外，Pacing就是大家原来说的rate-based congestion control, 给一个Rate发送而不是发送一个窗口，例如TFRC，WebRTC早已这样实现。其性能与控制间隔有关，调整慢了就会导致delay大甚至丢包。

# Shape不再基于排队（queue），而是基于时间戳（EDT）

**两点核心转变：**

* 每个包（skb）打上一个最早离开时间（Earliest Departure Time, EDT），也就是最早可以发送的时间戳；
* 用时间轮调度器（timing-wheel scheduler）替换原来的出向缓冲队列（qdisc queue）

Fig. 传统基于 queue 的流量整形器 vs. 新的基于 EDT 的流量整形器