TSO, fair queuing, pacing: three’s a charm

Eric Dumazet <edumazet@google.com>
Yuchung Cheng <ycheng@google.com>

A world of TCP bursts

- TCP is window-based and ack-clocked
  - Sudden cwnd increase
    - cwnd: stretch ACKs (e.g., LRO)
    - rwnd: receiver buffering
  - Idling between ACK and data
- TSO deferral bugs
- Switches local aggregation
- Modern structured traffic

Burst losses are bad signals to CC as network is often not 100% utilized

Oversize bursts are bad

- Bad throughput
  - Losses and ECN flags
  - Network is often not 100% utilized
  - False congestion alarms to TCP

- Vicious cycle
  - Bursts - losses - recovery - rwin surge - bursts ...

Many solutions exist

- Limit the bursts
    - moderate cwnd = inflight + 3 (linux)
    - send <=max_burst pkts per ACK (bsd)
- Tweak cwnd when idle
    - Reduce cwnd by Y% after X time
        - e.g., cwnd = min(cwnd, IW) after RTO
- Disable TSO/LRO
- Throttle large flows to improve fairness in qdisc

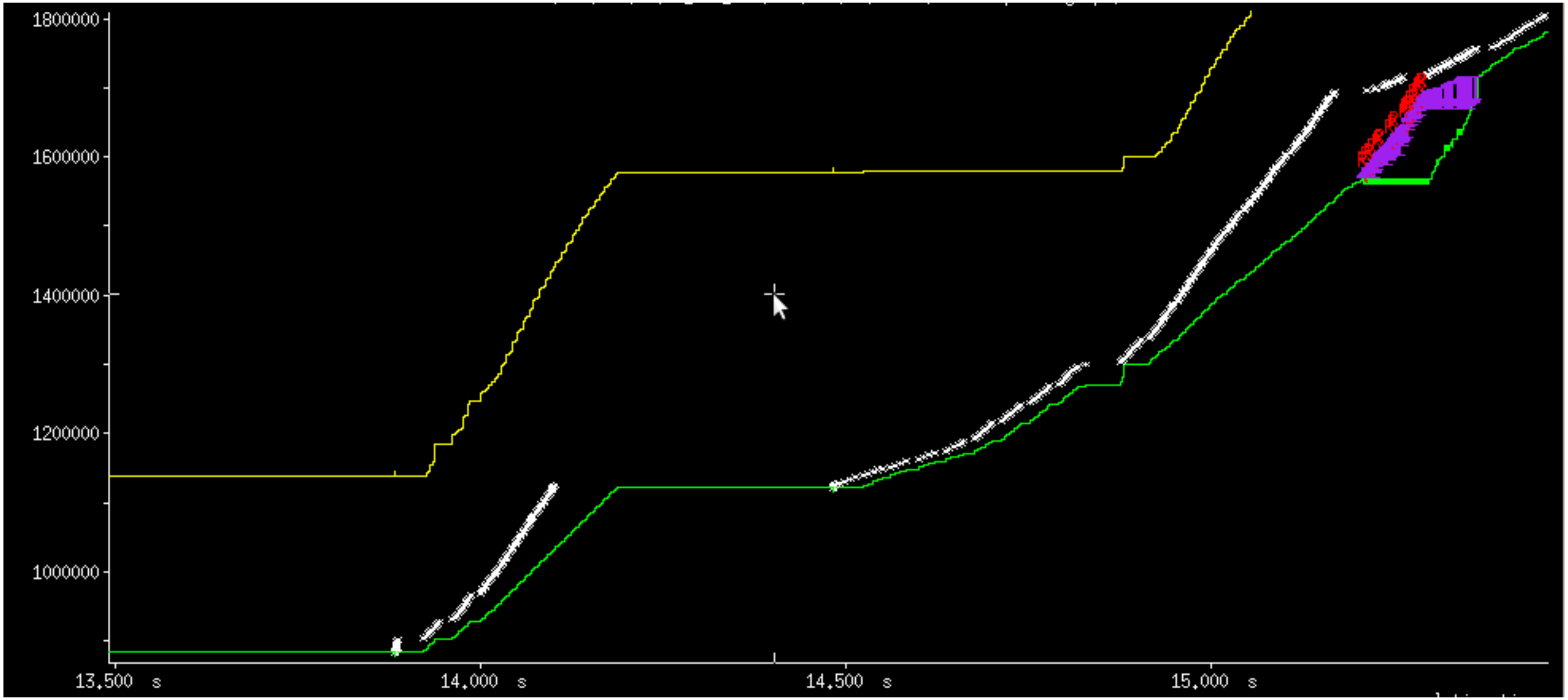
fq/pacing

- TCP cwnd controls the amount to send per RTT
  - Clocked by ACKs
  - No more TCP tweaks for burst

- qdisc layer handles sub-RTT scheduling
  - Pace at cwnd/RTT after idling
  - Flow fair queuing to improve mixing and fairness

- Break large burst into microburst
  - Dynamic TSO sizing base on the pacing rate
  - High performance host I/O

fq/pacing to reduce video burst

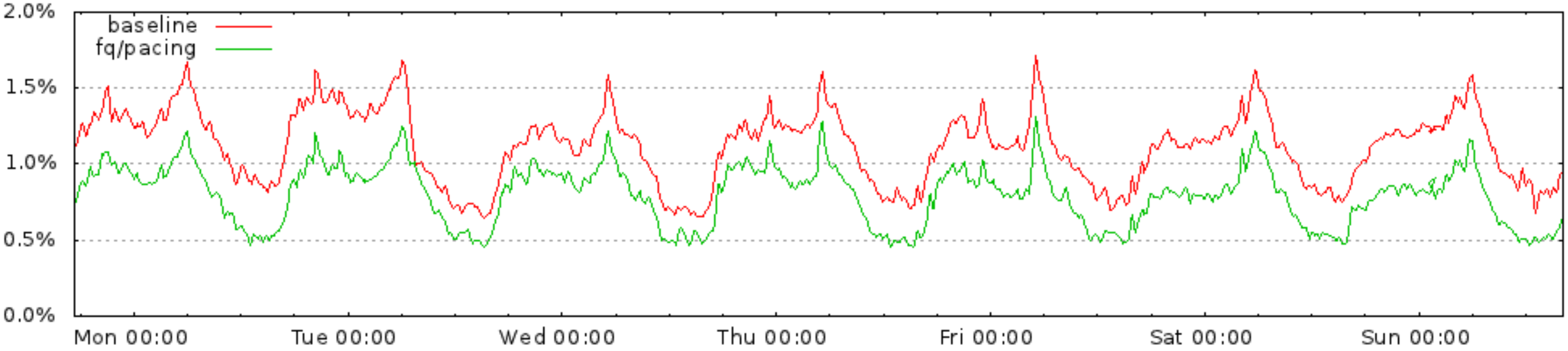


fq/pacing in Linux 3.11-rc7

- High performance: allows millions of concurrent flows per Qdisc
    - Small memory footprint : 8K per Qdisc, and 104 bytes per flow
    - *Single* high resolution timer to pace flows
    - One RB tree to link throttled flows.
    - fast flow match (not stochastic hash like SFQ/FQ_codel)
- Uses the new_flow/old_flow separation from FQ_codel
- Special FIFO queue for high prio packets (no need for PRIO + FQ)

Example usage:

```
tc qdisc add dev $ETH root fq
```


baseline
fq/pacing
Mon 00:00
Tue 00:00
Wed 00:00
Thu 00:00
Fri 00:00
Sat 00:00
Sun 00:00