

Pacing in Transport Protocols

draft-irtf-iccrp-pacing-00

Michael Welzl, Wesley Eddy, Vidhi Goel, Michael Tüxen

ICCRG
IETF 124

What's new?

- Updated [Linux](#) kernel version to 6.12 (LTS)
 - Double checked that this matches the description

- Much more [FreeBSD](#) text
(**RACK stack** only!)
 - Socket options with explanation
 - packetdrill example using 1.5 Mbit/s and burst size 4, transmitting IW

Option Name	Data Type	Semantic
TCP_RACK_PACE_RATE_SS	uint64_t	Pace rate in B/s during slow start
TCP_RACK_PACE_RATE_CA	uint64_t	Pace rate in B/s during congestion avoidance
TCP_RACK_PACE_RATE_REC	uint64_t	Pace rate in B/s during recovery
TCP_RACK_PACE_ALWAYS	int	Enable/Disable pacing
TCP_RACK_PACE_MAX_SEG	int	Micro burst size in number of full sized segments

Table 1: Socket Options

What's new, part 2: general considerations

- This old paper is well known; now we discuss it:
Aggarwal, A., Savage, S., and T. Anderson, "Understanding the performance of TCP pacing", IEEE Infocom 2000, March 2000
- Primarily, this paper shows some downsides of pacing
 - ...using a pacing rate calculation of $c * \text{cwnd} / \text{RTT}$, where $c=1$
 - Does anyone really do this? E.g. in Linux, $c > 1$
 - ...using Reno
 - ...with ns-2 simulations only
- Problems are attributed to synchronization of multiple connections
 - still interesting to look at, but I think there's something misleading here

A problem shown in this paper

- Double loss, FR fails to repair...
- Is this really due to synchronization?
 - I don't think so!

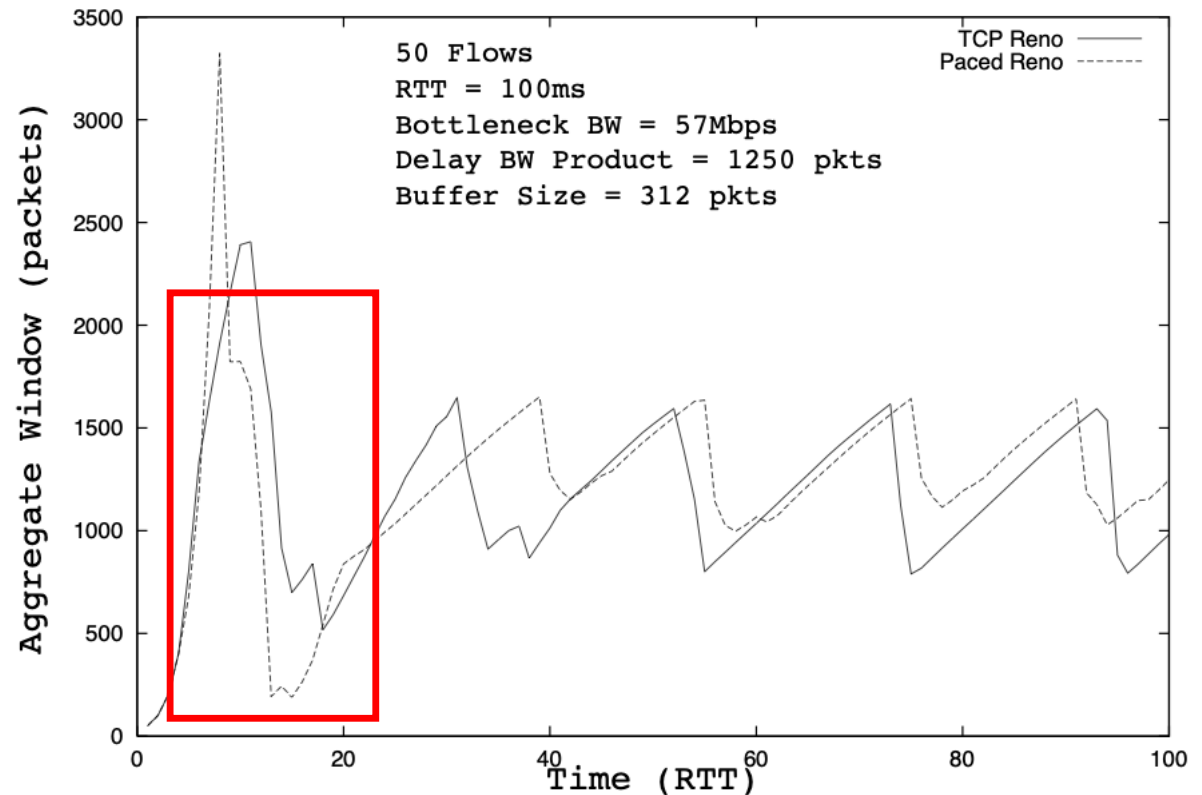
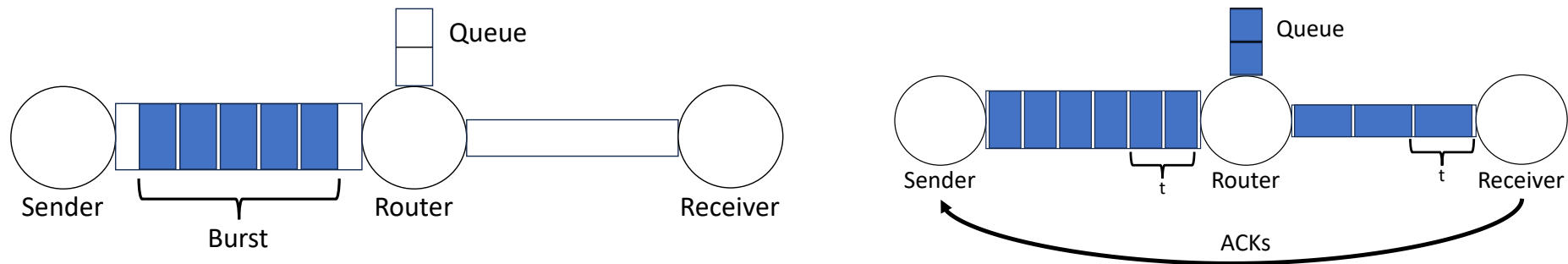


Fig. 8. Aggregate window size for 50 flows as a function of time. In slow start, Paced flows synchronize. In steady state, Paced flows have a larger aggregate window owing to de-synchronization.

Backing off after slow start

- See my IETF-120 (Vancouver 2024) ICCRG presentation
 - Two reasons for packet loss: 1) burst too large, 2) capacity limit exceeded

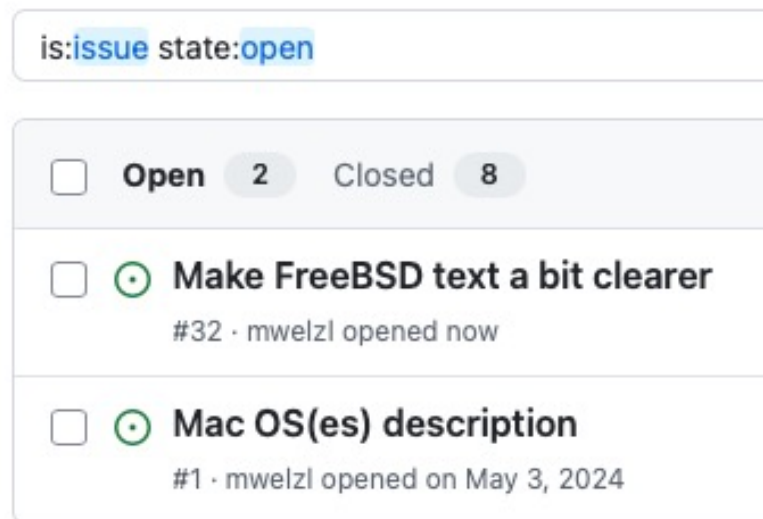


- Reason 2 in slow start: can happen when queue is large; with pacing, more likely to also happen with a shorter queue
- This is better: slow start goes on longer! but then, backing off with $\beta=0.5$ is not enough (0.5 is too large)

=> already explained in the draft; added an example to make it clearer.

Next steps

- Not much in our todo-list anymore



- What do folks think? What's still missing?
 - Tell us! => <https://github.com/mwelzl/draft-iccrg-pacing>