UiO : Faculty of Mathematics and Natural Sciences
University of Oslo

Department of Informatics
Networks and Distributed Systems (ND) group

Pacing in Transport Protocols

draft-welzl-iccrg-pacing

Michael Welzl, Wesley Eddy

ICCRG
IETF-120
26.7.2024

Why this draft?

- Give guidance to implementers via:
  - Discussion of general considerations and consequences
  - An overview of how implementations do it

- Implementations: so far, Linux TCP and QUIC BBR
  - Linux TCP based on: https://tinyurl.com/26698df9
  - QUIC BBR based on open source implementations: Google's *quiche* and Meta's *mvfst*

General considerations & consequences

- More likely to saturate a bottleneck

  Because, during the arrival of a flight of packets, the rate of the flight is lower than with bursty transmission

  – Hence: **loss from saturation** **(type 2 from my previous talk!)** more likely than **burst loss** **(type 1 from my previous talk!)**
  – Hence, beta < 0.5 may be good after slow start overshoot
  – Can work with smaller queues (because **burst loss** less likely)

- Good RTT estimates important
  – Problematic in the beginning
  – Suggestion: RFC 9040 based initialization when possible (temporal or ensemble sharing)

Implementation example: Linux TCP

- Pacing rate updated upon ACK arrival
  - rate = factor * MSS * cwnd / SRTT
  - factor: configurable value. by default, 2 in slow start and 1.2 in congestion avoidance
- Packets sent at that rate, limited to 1 ms granularity
  - I.e., every ms, send 1 ms worth of data at this rate
- Exceptions
  - **The first 10** (not IW!) packets are never paced
  - More than 1 ms worth of data for **very close peers** (configurable, default: min RTT < 3 ms)
  - **Min. burst size: 2 packets**; lower rates achieved via longer pauses
  - **Max. burst size: 64 Kbyte;** higher rates achieved via shorter pauses

Implementation example: QUIC BBR

- Typically in user space
  - More challenges: timing, coupling with the underlying stack and hardware
  - E.g., may "wake up" too late in a highly loaded system
- Tokenless approach (mvfst): compute a regular interval time and batch size (number of packets) to be released every interval and achieve the pacing rate
- Token-based approach: accumulates tokens to permit transmission based on the pacing rate, using a "leaky bucket" to control bursts
  - "Burst tokens" allow back-to-back packets after periods of quiescence (Google's implementation)
  - Heuristics based "lumpy tokens" allow more after burst tokens consumed

Thank you!

Questions?