Transport and Services Working Group
Internet-Draft
Intended status: Experimental
Expires: 3 September 2026

M. P. Tahiliani
National Institute of Technology Karnataka

2 March 2026

Flow Queue PIE: A Hybrid Packet Scheduler and Active Queue Management Algorithm
draft-ietf-tsvwg-fq-pie-01

Abstract

This document presents Flow Queue Proportional Integral controller Enhanced (FQ-PIE), a hybrid packet scheduler and Active Queue Management (AQM) algorithm to isolate flows and tackle the problem of bufferbloat. FQ-PIE uses hashing to classify incoming packets into different queues and provide flow isolation. Packets are dequeued by using a variant of the round robin scheduler. Each such flow is managed by the PIE algorithm to maintain high link utilization while controlling the queue delay to a target value.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://mohittahiliani.github.io/draft-ietf-tsvwg-fq-pie/draft-ietf-tsvwg-fq-pie.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-tsvwg-fq-pie/.

Discussion of this document takes place on the Transport and Services Working Group Working Group mailing list (mailto:tsvwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tsvwg/. Subscribe at https://www.ietf.org/mailman/listinfo/tsvwg/.

Source for this draft and an issue tracker can be found at https://github.com/mohittahiliani/draft-ietf-tsvwg-fq-pie.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

This Internet-Draft will expire on 3 September 2026.

Table of Contents

1. Introduction

Flow Queue Proportional Integral Controller Enhanced (FQ-PIE) combines flow queuing with the PIE (Proportional Integral controller Enhanced) [RFC8033] Active Queue Management (AQM) algorithm to provide flow isolation and reduce bufferbloat by controlling queue delay. This is similar to how Flow Queue Controlled Delay (FQ-CoDel) [RFC8290] integrates flow queuing with the CoDel (Controlled Delay) AQM algorithm [RFC8289].

When a packet is enqueued, it is classified into different queues to ensure isolation between flows. While the goal of flow queuing is to assign a unique queue to each flow, flows can instead be hashed into a set of buckets using a hash function, where each bucket corresponds to its own queue. The PIE AQM operates independently on each of these queues, enabling each flow to receive appropriate congestion signals either implicitly (via packet drops) or explicitly (via mechanisms such as Explicit Congestion Notification (ECN) [RFC3168]). For dequeuing, FQ-PIE employs the Deficit Round Robin (DRR) based scheduler described in [RFC8290], which ensures fair packet scheduling across the different queues.

An implementation of FQ-PIE has been incorporated into the mainline Linux kernel as a queuing discipline (qdisc) [LINUX-FQ-PIE] and is supported by several Linux distributions. Another implementation has also been incorporated into FreeBSD [FREEBSD-FQ-PIE]. Finally, an implementation of FQ-PIE is also available in the ns-3 network simulator [ns-3-FQ-PIE].

2. Terminology

This document uses the terms defined in Section 1.1 of [RFC8290] and Sections 4 and 5 of [RFC8033].

3. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. The FQ-PIE Algorithm

The FQ-PIE algorithm consists of two main components: (i) flow queuing, which isolates competing flows by treating flows that build queues differently from those that do not, and (ii) the PIE AQM algorithm, which manages each queue and maintains a target queue delay (such as the 15 ms default in [RFC8033]). Flow queuing works by classifying incoming packets into different queues during the enqueue phase and then scheduling outgoing packets from these queues during the dequeue phase. The PIE algorithm, however, only operates during enqueue.

The details of flow queuing and the PIE algorithm are not covered here; for more information, please refer to [RFC8290] and [RFC8033], respectively.

4.1. Enqueue

The packet enqueue process is described as follows: first, the incoming packets are classified into different queues by hashing the 5-tuple, which includes the protocol number, source and destination IP addresses, and source and destination port numbers, similar to the approach used in FQ-CoDel.

Next, the packet is passed to the PIE algorithm, which uses a drop probability to determine whether the packet should be enqueued or dropped, as outlined in [RFC8033]. This drop probability is updated periodically (every 15 ms, as per [RFC8033]) based on the current queue delay's deviation from the target delay and whether the delay is trending up or down.

[RFC8033] presents two methods for calculating the current queue delay: one uses Little's Law, estimating delay based on the queue length and the average dequeue rate; the other takes direct measurements using timestamps, as implemented in CoDel and FQ-CoDel. However, experimental studies on the PIE algorithm [REVISIT-PIE] indicate that while the dequeue rate is intended to estimate the transmission rate of packets over the outgoing link, it may instead reflect the rate at which packets move from the host stack (e.g., Linux qdisc) to the device driver's transmission ring. Additionally, in FQ-PIE, queue delay estimates from Little's Law can be unreliable, as it's challenging to calculate an accurate per-queue dequeue rate. Consequently, the FQ-PIE algorithm SHOULD calculate the current queue delay using direct measurements with timestamps.

It is important to note that the timestamping approach provides a "per-packet queue delay," while the drop probability is calculated periodically (every 15 ms, as specified in [RFC8033]). Therefore, the FQ-PIE algorithm MAY use the queue delay value from the most recently dequeued packet when calculating the drop probability.

At the time of writing this document, the Linux, FreeBSD and ns-3 implementations use timestamps to calculate the current queue delay and consider the measurements from the most recently dequeued packet when calculating the drop probability. Additionally, these implementations offer an option to use the dequeue rate estimation technique based on Little's Law.

Lastly, if an incoming packet arrives when the total number of enqueued packets has already saturated the queue capacity, FQ-PIE drops the packet without further processing. In contrast, FQ-CoDel identifies the queue with the largest current byte count (i.e., a "fat flow") when the queue capacity is saturated and drops half of the packets from this queue (up to a maximum of 64 packets, as specified in Section 4.1 of [RFC8290]). FQ-PIE does not adopt this approach for the reasons explained below.

Since CoDel performs its queue control operations during the dequeue phase and does not drop incoming packets until the queue is full, it tends to fill its queues more quickly than PIE, which drops packets randomly during the enqueue phase. This is especially true when CoDel has just entered the dropping phase, as it takes time to ramp up its packet dropping frequency. Therefore, the strategy of dropping half of the packets from a fat flow’s queue suits FQ-CoDel but is not appropriate for FQ-PIE. Dropping packets in bulk might lead to underutilization of link capacity, as FQ-PIE already enforces queue control during the enqueue phase.

4.2. Dequeue

The packet dequeue process in FQ-PIE is similar to that in FQ-CoDel, where a DRR-based scheduler is used to dequeue packets from each queue. The key difference is that in FQ-CoDel, CoDel operates during this phase, whereas in FQ-PIE, PIE operates during the enqueue phase. The method for obtaining direct measurements of per-packet queue delay is the same in both FQ-PIE and FQ-CoDel, and is performed during the dequeue phase.

4.3. ECN Support

FQ-PIE MAY support ECN by marking ECN-Capable Transport (ECT) packets [RFC3168] instead of dropping them, in accordance with the recommendations in Section 5.1 of [RFC8033]. The Linux, FreeBSD and ns-3 implementations of FQ-PIE comply with these recommendations at the time of writing this document.

5. Scope of Experimentation

The design of the FQ-PIE algorithm as described in this document has been a part of the Linux kernel since version 5.6 (released on March 29, 2020), FreeBSD since version 11.0-RELEASE (released on October 10, 2016), and the ns-3 network simulator since version 3.34 (released on July 14, 2021). The following aspects can be explored for further study and experimentation:

* The scenarios similar to those summarized in Figure 4 of [RFC7928] MAY be considered for an in-depth experimentation of FQ-PIE.

* Interactions between flow queuing and new congestion control algorithms, such as Bottleneck Bandwidth and Round-trip propagation time (BBR) [I-D.draft-ietf-ccwg-bbr].

* Different packet drop probability thresholds to switch from marking packets to dropping packets.

* Evaluation of the enhancements to the PIE algorithm described in Section 5 in [RFC8033] to decide which enhancements are suitable for deployment with FQ-PIE.

* Effectiveness of FQ-PIE in terms of providing isolation and minimal latency for low volume traffic (short flows) such as web applications, instant messaging applications, interactive applications and IoT applications.

* Different hashing mechanisms to improve the overall working of flow queuing.

6. Security Considerations

The FQ-PIE algorithm introduces no specific security exposures. The flow queuing aspect of the FQ-PIE algorithm is the same as FQ-CoDel, and hence has similar advantages from the security perspective as outlined in Section 8 of [RFC8290]. The PIE aspect of the FQ-PIE algorithm is the same as described in [RFC8033] that does not have any security exposures.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/rfc/rfc3168>.

[RFC7928] Kuhn, N., Ed., Natarajan, P., Ed., Khademi, N., Ed., and D. Ros, "Characterization Guidelines for Active Queue Management (AQM)", RFC 7928, DOI 10.17487/RFC7928, July 2016, <https://www.rfc-editor.org/rfc/rfc7928>.

[RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <https://www.rfc-editor.org/rfc/rfc8033>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8289] Nichols, K., Jacobson, V., McGregor, A., Ed., and J. Iyengar, Ed., "Controlled Delay Active Queue Management", RFC 8289, DOI 10.17487/RFC8289, January 2018, <https://www.rfc-editor.org/rfc/rfc8289>.

[RFC8290] Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm", RFC 8290, DOI 10.17487/RFC8290, January 2018, <https://www.rfc-editor.org/rfc/rfc8290>.

8.2. Informative References

[FREEBSD-FQ-PIE]
Al-Saadi, R. and G. Armitage, "Dummynet AQM v0. 2窶鼎oDel, FQ-CoDel, PIE and FQ-PIE for FreeBSDs ipfw/dummynet Framework", Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. A, 160418 , October 2016, <https://web.archive.org/web/20241018123533/http://caia.swin.edu.au/reports/160418A/CAIA-TR-160418A.pdf>.

[I-D.draft-ietf-ccwg-bbr]
Cardwell, N., Swett, I., and J. Beshay, "BBR Congestion Control", Work in Progress, Internet-Draft, draft-ietf-ccwg-bbr-04, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ccwg-bbr-04>.

[LINUX-FQ-PIE]
Ramakrishnan, G., Bhasi, M., Saicharan, V., Monis, L., Patil, S. D., and M. P. Tahiliani, "FQ-PIE Queue

Discipline in the Linux Kernel: Design, Implementation and Challenges", 2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium) , October 2019, <https://ieeexplore.ieee.org/abstract/document/9000684>.

[ns-3-FQ-PIE]
"FQ-PIE Queue Discipline in ns-3", July 2021, <https://www.nsnam.org/docs/models/html/fq-pie.html>.

[REVISIT-PIE]
Imputato, P., Avallone, S., Tahiliani, M. P., and G. Ramakrishnan, "Revisiting Design Choices in Queue Disciplines: The PIE Case", Computer Networks , April 2020, <https://www.sciencedirect.com/science/article/pii/S1389128619313775>.

Author’s Address

Mohit P. Tahiliani
National Institute of Technology Karnataka
P. O. Srinivasnagar, Surathkal
Mangalore, Karnataka - 575025
India
Email: tahiliani@nitk.edu.in
URI: http://tahiliani.in