NAME

FQ-PIE - Flow Queue Proportional Integral controller Enhanced

SYNOPSIS

tc qdisc ... fq_pie [**limit** PACKETS] [**flows** NUMBER]
[**target** TIME] [**tupdate** TIME]
[**alpha** NUMBER] [**beta** NUMBER]
[**quantum** BYTES] [**memory_limit** BYTES]
[**ecn_prob** PERENTAGE] [[**no**]**ecn**]
[[**no**]**bytemode**] [[**no_**]**dq_rate_estimator**]

DESCRIPTION

FQ-PIE (Flow Queuing with Proportional Integral controller Enhanced) is a queuing discipline that combines Flow Queuing with the PIE AQM scheme. FQ-PIE uses a Jenkins hash function to classify incoming packets into different flows and is used to provide a fair share of the bandwidth to all the flows using the qdisc. Each such flow is managed by the PIE algorithm.

ALGORITHM

The FQ-PIE algorithm consists of two logical parts: the scheduler which selects which queue to dequeue a packet from, and the PIE AQM which works on each of the queues. The major work of FQ-PIE is mostly in the scheduling part. The interaction between the scheduler and the PIE algorithm is straight forward.

During the enqueue stage, a hashing-based scheme is used, where flows are hashed into a number of buckets with each bucket having its own queue. The number of buckets is configurable, and presently defaults to 1024 in the implementation. The flow hashing is performed on the 5-tuple of source and destination IP addresses, port numbers and IP protocol number. Once the packet has been successfully classified into a queue, it is handed over to the PIE algorithm for enqueuing. It is then added to the tail of the selected queue, and the queue's byte count is updated by the packet size. If the queue is not currently active (i.e., if it is not in either the list of new or the list of old queues) , it is added to the end of the list of new queues, and its number of credits is initiated to the configured quantum. Otherwise, the queue is left in its current queue list.

During the dequeue stage, the scheduler first looks at the list of new queues; for the queue at the head of that list, if that queue has a negative number of credits (i.e., it has already dequeued at least a quantum of bytes), it is given an additional quantum of credits, the queue is put onto the end of the list of old queues, and the routine selects the next queue and starts again. Otherwise, that queue is selected for dequeue again. If the list of new queues is empty, the scheduler proceeds down the list of old queues in the same fashion (checking the credits, and either selecting the queue for dequeuing, or adding credits and putting the queue back at the end of the list). After having selected a queue from which to dequeue a packet, the PIE algorithm is invoked on that queue.

Finally, if the PIE algorithm does not return a packet, then the queue must be empty and the scheduler does one of two things:

If the queue selected for dequeue came from the list of new queues, it is moved to the end of the list of old queues. If instead it came from the list of old queues, that queue is removed from the list, to be added back (as a new queue) the next time a packet arrives that hashes to that queue. Then (since no packet was available for dequeue), the whole dequeue process is restarted from the beginning.

If, instead, the scheduler did get a packet back from the PIE algorithm, it subtracts the size of the packet from the byte credits for the selected queue and returns the packet as the result of the dequeue operation.

PARAMETERS

limit

It is the limit on the queue size in packets. Incoming packets are dropped when the limit is reached. The default value is 10240 packets.

flows

It is the number of flows into which the incoming packets are classified. Due to the stochastic nature of hashing, multiple flows may end up being hashed into the same slot. Newer flows have priority over older ones. This parameter can be set only at load time since memory has to be allocated for the hash table. The default value is 1024.

target

It is the queue delay which the PIE algorithm tries to maintain. The default target delay is 15ms.

tupdate

It is the time interval at which the system drop probability is calculated. The default is 15ms.

alpha
beta

alpha and beta are parameters chosen to control the drop probability. These should be in the range between 0 and 32.

quantum

quantum signifies the number of bytes that may be dequeued from a queue before switching to the next queue in the deficit round robin scheme.

memory_limit

It is the maximum total memory allowed for packets of all flows. The default is 32Mb.

ecn_prob

It is the drop probability threshold below which packets will be ECN marked instead of getting dropped. The default is 10%. Setting this parameter requires **ecn** to be enabled.

[**no**]**ecn**

It has the same semantics as **pie** and can be used to mark packets instead of dropping them. If **ecn** has been enabled, **noecn** can be used to turn it off and vice-a-versa.

[**no**]**bytemode**

It is used to scale drop probability proportional to packet size **bytemode** to turn on bytemode, **nobytemode** to turn off bytemode. By default, **bytemode** is turned off.

[**no_**]**dq_rate_estimator**

dq_rate_estimator can be used to calculate queue delay using Little’s Law, **no_dq_rate_estimator** can be used to calculate queue delay using timestamp. By default, **dq_rate_estimator** is turned off.

EXAMPLES

```
# tc qdisc add dev eth0 root fq_pie
# tc -s qdisc show dev eth0
qdisc fq_pie 8001: root refcnt 2 limit 10240p flows 1024 target 15.0ms tupdate 16.0ms alpha 2 beta 20
quantum 1514b memory_limit 32Mb ecn_prob 10
```

```
 Sent 159173586 bytes 105261 pkt (dropped 24, overlimits 0 requeues 0)
 backlog 75700b 50p requeues 0
  pkts_in 105311 overlimit 0 overmemory 0 dropped 24 ecn_mark 0
  new_flow_count 7332 new_flows_len 0 old_flows_len 4 memory_used 108800

# tc qdisc add dev eth0 root fq_pie dq_rate_estimator
# tc -s qdisc show dev eth0
qdisc fq_pie 8001: root refcnt 2 limit 10240p flows 1024 target 15.0ms tupdate 16.0ms alpha 2 beta 20
quantum 1514b memory_limit 32Mb ecn_prob 10 dq_rate_estimator
 Sent 8263620 bytes 5550 pkt (dropped 4, overlimits 0 requeues 0)
 backlog 805448b 532p requeues 0
  pkts_in 6082 overlimit 0 overmemory 0 dropped 4 ecn_mark 0
  new_flow_count 94 new_flows_len 0 old_flows_len 8 memory_used 1157632
```

SEE ALSO

tc(8), **tc-pie**(8), **tc-fq_codel**(8)

SOURCES

RFC 8033: https://tools.ietf.org/html/rfc8033

AUTHORS

FQ-PIE was implemented by Mohit P. Tahiliani. Please report corrections to the Linux Networking mailing list <netdev@vger.kernel.org>.