Review: PIE AQM
draft-ietf-aqm-pie-00

Bob Briscoe, BT*

RITE – Reducing Internet Transport Latency

IETF-93, Praha, July 2015

* now independent

Summary of full 20pp review
<http://www.bobbriscoe.net/projects/latency/piervw_tr.pdf>

R/TE
Reducing Internet Transport Latency

The authors' contributions are part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed here are solely those of the authors.

SEVENTH FRAMEWORK PROGRAMME

Main concerns

1. Proposed Standard, but no normative language
    1. work needed to distinguish between design intent and specific implementation
    2. unclear how strongly the enhancements are recommended
2. Has PIE been separately tested with and without each enhancement, to justify each?
3. Needs to enumerate whether it satises each AQM Design Guideline
    1. If not, say why or fix.
    2. Particular concerns:
        1. No spec of ECN behaviour
        2. No autotuning of the two main parameters
        3. Transport specific (Reno-based?) autotuning of α & β
4. Rationale for a PI controller not properly articulated
5. Technical flaws/concerns
    1. Turning PIE off
    2. `Autotuning' α & β parameters
    3. Averaging problems
    4. Burst allowance unnecessary?
    5. Needs a Large Delay to Make the Delay Small
    6. Derandomization: a waste of cycles
    7. Bound drop probability at 100% →DoS vulnerability?
    8. Avoiding Large Packet Lock-Out under Extreme Load.
6. Numerous magic numbers
    1. 20 constants, 13 of which are not in the header block.
    2. About half ought to be made to depend on other constants
    3. Need to state how to set the remaining constants for different environments
7. Implementation suggestion for Autotuning α & β
8. Nits (6pp)

Technical concern #2
`Autotuning' α & β parameters

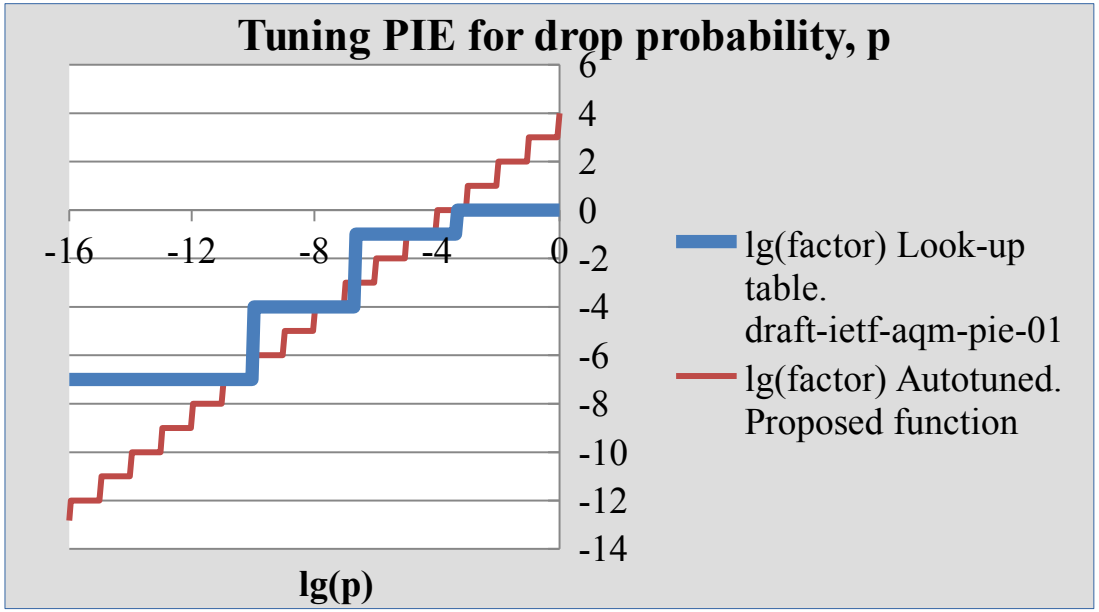
```
if (p < 0.1%) {
   p += [alpha*(qdelay - QDELAY_REF) + beta*(qdelay-PIE->qdelay_old_)]/128;
} else if (p < 1%) {
   p += [alpha*(qdelay - QDELAY_REF) + beta*(qdelay-PIE->qdelay_old_)]/16;
} else if (p < 10%) {
   p += [alpha*(qdelay - QDELAY_REF) + beta*(qdelay-PIE->qdelay_old_)]/2;
} else {
   p +=  alpha*(qdelay - QDELAY_REF) + beta*(qdelay-PIE->qdelay_old_);
}
```

-

Mb/s	p
6	0.1%
12	0.04%
24	0.016%
48	0.0065%
96	0.0025%
192	0.0010%
384	0.00040%
768	0.00016%

above table in draft stops here

Instead of a look-up table for the factors, propose to use p*16 *as* the factor
Using bit-shifting to avoid multiplication (see review)



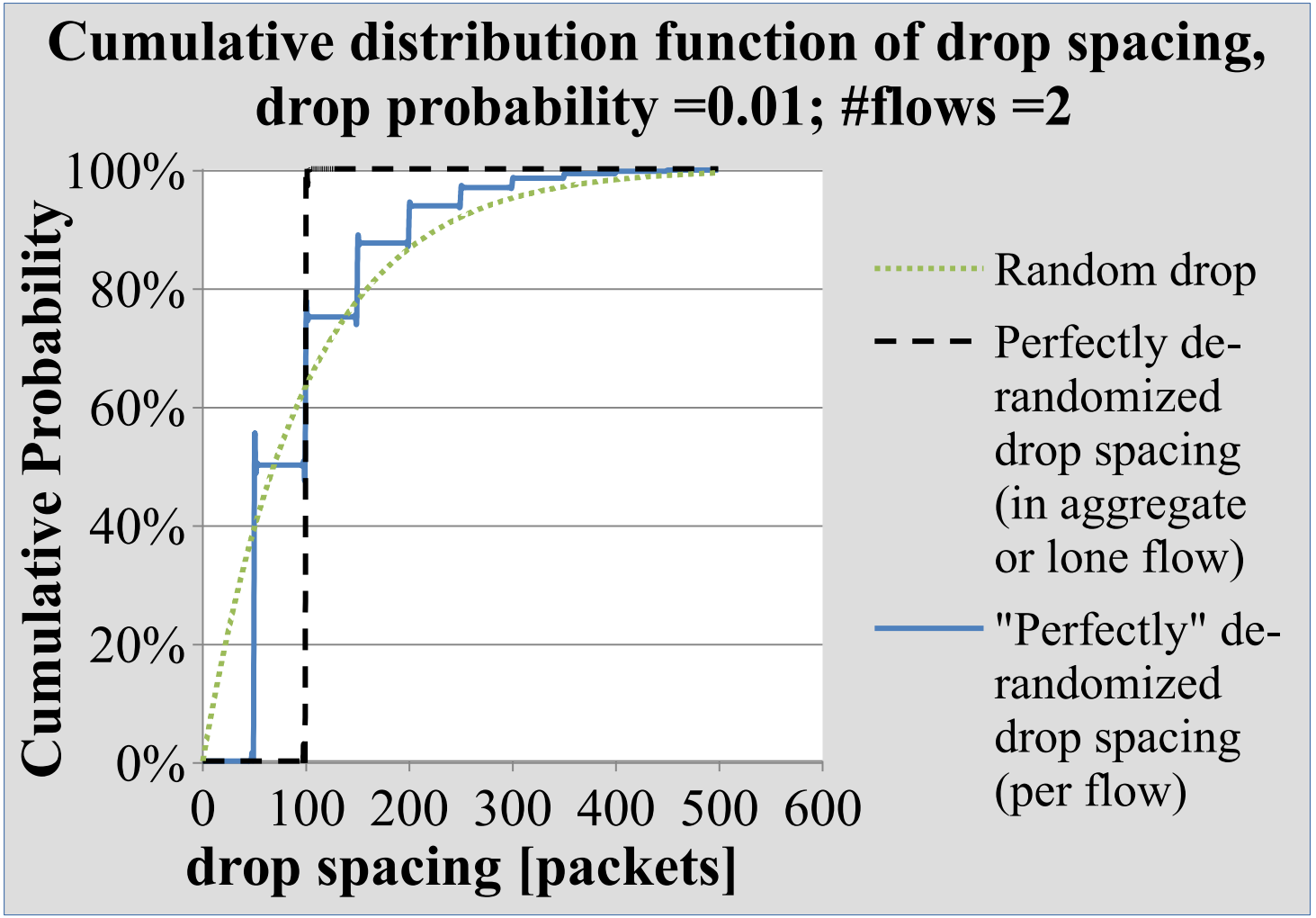
Single Cubic flow, 100ms RTT

Technical concerns #4 & #5

4. Burst allowance too high or even unnecessary?
   - 150ms is 5-15 round trips for most traffic (CDNs)
     - burst allowance is meant to filter out transients
     - If AQM takes >1 RTT to start dropping → delay & tail drop
   - PIE already filters bursts (queue sampling and evolution of drop_prob)
     - Have you tried removing this additional burst allowance?
5. Needs a Large Delay to Make the Delay Small
   - PIE code needs at least 2^{14}B of queue before it starts estimating depart_rate, which is is equivalent to:
     - 16ms of queue at 8Mb/s
     - 66ms of queue at 2Mb/s
   - i.e. for a 2Mb/s uplink, the mechanism PIE uses to hold the queue at 20ms needs 66ms of queue before it kicks in

Technical concern #6
Derandomization: a waste of cycles

- regular spacing between drops in the aggregate still leaves irregular spacing within each flow



Technical concerns #7 & #8

7. Bound drop probability at 100% → DoS vulnerability?

- In the code, drop_prob is bounded at 100%
  - Even 2x link rate arrivals only need 50% drop
  - 100% drop implies zero throughput!
- RED initially made the same mistake
  - later changed (see link to code in review)

8. Avoiding Large Packet Lock-Out under Extreme Load

```
CURRENT
  if (queue_.is_full()) {
  drop(packet);
  }
SUGGESTED:
  if (queue_.byte_length() > (buffer_size - MTU) ) {
  drop(packet);
}
```

6. Numerous Magic Numbers

- 20 magic numbers
  - 2 called out as parameters
  - 13 embedded in code, not in header (see list in review)
- not all are scenario-independent – should identify them, e.g.
  - `DQ_THRESHOLD` (2^{14}B inappropriate if queues are shorter)
  - `T_UPDATE` (16ms inappropriate if RTT < 16ms)
    better: define `B_UPDATE` in bytes, and derive `T_UPDATE` using the link rate
  - `ALPHA` should be defined in terms of `T_UPDATE`
  - `BETA` should be defined in terms of `ALPHA` and a `RATIO`
  - `QUEUE_SMALL = BUFFER / 3`. Why relate anything to `BUFFER`?
  - 3 parameters for uncongested queue tests seem specific to small #flows
  - EWMA const `avg_ dq_time_` should equal `DQ_THRESHOLD/`2^{16}
  - ...

Main concerns

1. Proposed Standard, but no normative language
   1. work needed to distinguish between design intent and specific implementation
   2. unclear how strongly the enhancements are recommended
2. Has PIE been separately tested with and without each enhancement, to justify each?
3. Needs to enumerate whether it satises each AQM Design Guideline
   1. If not, say why or fix.
   2. Particular concerns:
      1. No spec of ECN behaviour
      2. No autotuning of the two main parameters
      3. Transport specific (Reno-based?) autotuning of α & β
4. Rationale for a PI controller not properly articulated
5. Technical flaws/concerns
   1. Turning PIE off
   2. `Autotuning' α & β parameters
   3. Averaging problems
   4. Burst allowance unnecessary?
   5. Needs a Large Delay to Make the Delay Small
   6. Derandomization: a waste of cycles
   7. Bound drop probability at 100% →DoS vulnerability?
   8. Avoiding Large Packet Lock-Out under Extreme Load.
6. Numerous magic numbers
   1. 20 constants, 13 of which are not in the header block.
   2. About half ought to be made to depend on other constants
   3. Need to state how to set the remaining constants for different environments
7. Implementation suggestion for Autotuning α & β
8. Nits (6pp)

Summary

- The PIE draft ends with two assertions in the Discussion section:

  “PIE is simple to implement”

  “PIE does not require any user conguration”

  - I do not believe either statement is warranted any more
- The implementation has not retained the elegance of the theory
  - The performance benefit from so-called `enhancements' is questionable or non-existent
  - whereas the added complexity is very apparent.
- PIE now contains a large number of hard-coded constants
  - I counted 20
  - Most ought to be scenario-dependent conguration variables.

Review: PIE AQM

draft-ietf-aqm-pie-00

Q&A

spare slides

Technical concern #3
Averaging problems

3.1 Queue Sample Rate

- aim to sample every 100 or even 1000 of packets unnecessarily sluggish?

3.2 Incorrect EWMA

ave(depart_rate) = k * ave(1/t1, 1/t2, …)

!= k / ave(t1, t2, …)

- faster than EWMA at first, then slower
- may be good enough approximation – would be complex to implement correctly – can anyone find scenarios that would lead to pathological errors?

Technical concerns #5

1. Turning PIE Off
   - Background: equipment at the head-end of broadband access, cable or radio network can handle thousands of users, but typically only 1 or 2% are active at a time
   - Since the review: authors confirmed that a regular timer calls status_update() not packet arrivals (even tho the code checks how long since it last ran).
     - Addresses my concern about drop_prob not reducing during inactive periods.
     - But raises another concern about “death by a thousands ticks”.
   - Suggested resolution: Draft should clarify it is for customer premises equipment. For a multi-user scenario, it could be packet triggered, and include code to 'catch-up' drop_prob proportionate to the idle period.