CoDel

Caching control law state: Linux bugs

Bob Briscoe

Oct 2021

Summary of `codel_dequeue()` logic in Linux (excluding ECN logic)

more efficient structure:

```
if (_drop) {
        if (dropping) {…}
} else {
        dropping = false;}
}
```

```
bool _drop = should_drop()
if (dropping)
   if (!_drop)
      dropping = false
   else
      if (now > drop_next)
         while (dropping &&
                now > drop_next)
            count++
            Newton_step()
            drop_func()
            if (!should_drop())
               dropping = false
            else
               drop_next = control_law()
else
   if (_drop)
      drop_func()      // while (now>drop_next)?
      _drop = should_drop()      // redundant?
      dropping = true
      _delta = count - lastcount
      if ( (_delta > 1) &&             // >0 ?
           (now-drop_next < 16*interval) )
         count = _delta          // count--; ?
         Newton_step()
      else
         count = 1
         rec_inv_sqrt = FFFF
      lastcount = count
      drop_next = control_law()
```

CoDel: potential problems

count-caching bugs in Linux

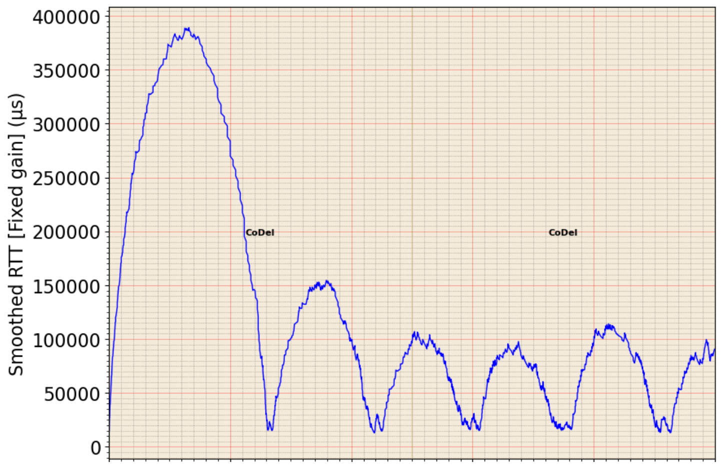
- `count` wrongly set to `_delta` on re-entering dropping mode and `if (_delta > 1)` should be `if (_delta > 0)`
  - AFAICT, when CoDel re-enters dropping mode:
  - if (`count` increased when last in dropping mode AND it's been <1.6s since last drop)
  - it should just use the `count` naturally stored when it last left dropping mode
    - but on this line, CoDel (code and RFC) inexplicably sets: `count = (count - lastcount);`
    - That is, the increase in `count` during the last period in dropping mode (whether initialized to 1 or set to the cached value)
    - The problem: imagine entering dropping mode 4x in succession, and each time `count` reaches the value on the right (the clock icon . . . ⏰ indicates unnecessary delay):

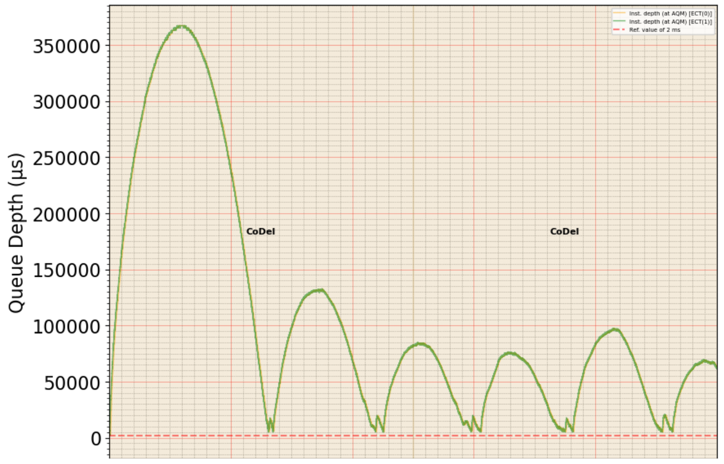
`delta = count - lastcount`	`count`	`lastcount`	...	`count`
	1	1	...	12
11	11	11	...	14
3	3	3	...⏰...	15
1	1	1	...⏰...	15

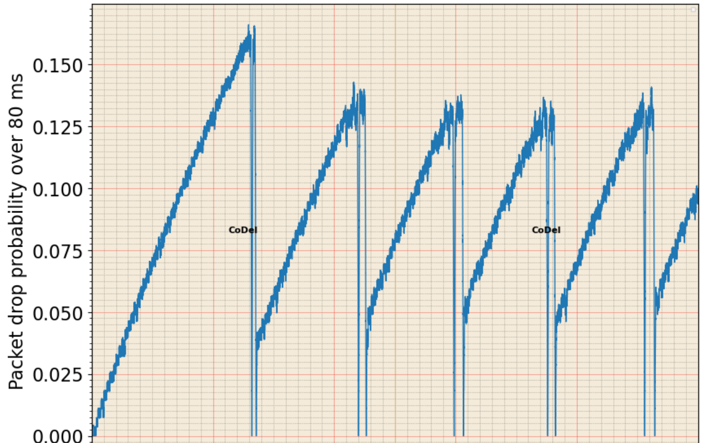
  - Only an increase proves count is fresh so, instead of `count = delta;` I suggest: `count--;` and I also suggest `if (delay > 0)` which is what the code effectively does the first time after `count=1`

Caching control law state: bug symptoms

- Experiment:
  - fq_codel default settings, v5.14.11 kernel
  - 40Mb/s link; 42Mb/s unresponsive flow
  - Slight overload used to cut run time of expt
    - and to stay within buffer, without confusion of tail drop
- Time series plots of salient metrics here:
  - TCP smoothed RTT
  - CoDel queue depth
  - drop prob. averaged over 80ms slots
- For unresponsive traffic,
  - when CoDel exits dropping mode, `count` is huge
- When it re-enters dropping mode
  - it forgets all that work reaching a huge `count`,
    'cos it sets `count` to the last `_delta`, not the last `count`
  - search for `count` starts nearly from scratch again
- Summary:
  - >10s to get under control, even for only 40Mb/s and 5% overload
  - then forgets the answer and repeats
- Thx to Asad Ahmed for running the tests

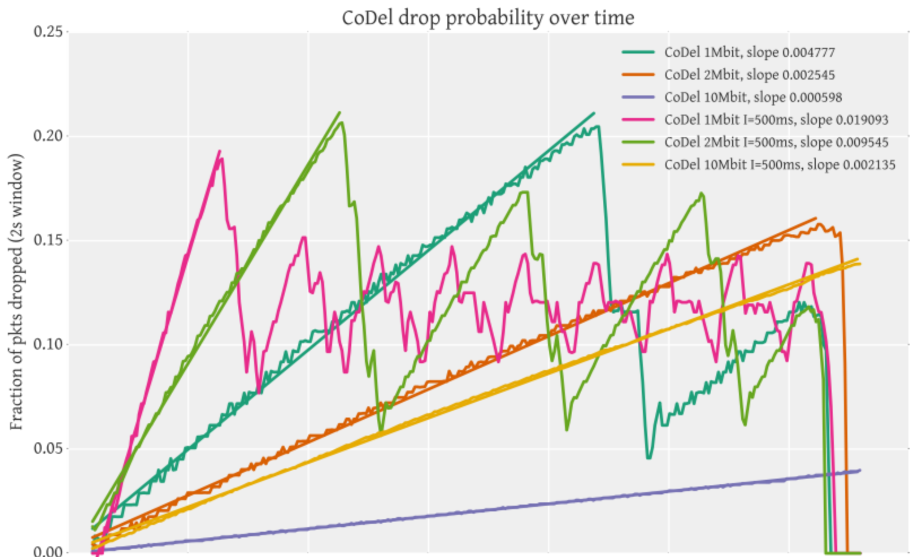






addendum 26-Feb-2022
count caching bug – in context

- Overshadowed by a much bigger bug (design flaw)...
- CoDel's "control" law doesn't measure the queue it is controlling
  - takes hundreds of seconds to bring an unresponsive flow under control
    - so sweating over the count caching bug would be a distraction
  - 12-Nov-2013: I reported the bigger design flaw to Kathie & Van
    - cc'd my posting to aqm@ietf.org
  - 07-Jun-2015: Toke confirmed my analysis empirically (see same thread). Plot paste below
    - also see: source plot, expt definition
  - 30-Sep-2015: Dave Taht (still same thread): "cake uses a better curve for CoDel but we still need to do more testing in the lab"
  - Misses the point
    - CAKE is faster but still extremely slow
    - CAKE's control law still never measures the queue it is meant to be controlling.


CoDel drop probability over time
Fraction of pkts dropped (2s window)
CoDel 1Mbit, slope 0.004777
CoDel 2Mbit, slope 0.002545
CoDel 10Mbit, slope 0.000598
CoDel 1Mbit I=500ms, slope 0.019093
CoDel 2Mbit I=500ms, slope 0.009545
CoDel 10Mbit I=500ms, slope 0.002135