DCTCP & CoDel
the Best is the Friend of the Good

Bob Briscoe, BT
Murari Sridharan, Microsoft
IETF-84 TSVAREA Jul 2012
menu

1. why DCTCP is important
2. an (untested) roadmap
   how might DCTCP be deployed and co-exist with current Internet traffic & AQMs
Data Centre TCP (DCTCP)
high utilisation in steady state still leaves room for bursts

Today
TCP on end-systems
RED in queues

if solely change queues

cuts delay but poorer line utilisation

change queues and end-systems

buffer kept for bursts

DCTCP: more smaller saw-teeth

TCP saw-teeth seeking the operating point

shallower operating point

lower queuing delay

good line utilisation
Data Center TCP Algorithm

Switch side:
- Mark packets when Queue Length > K

Sender side (differences from TCP New Reno):
- Maintain *moving average* of fraction of marked packets ($\alpha'$)

  Each RTT: $F = \frac{\# \text{ of marked ACKs}}{\text{Total } \# \text{ of ACKs}} \Rightarrow \alpha \leftarrow (1 - g)\alpha + gF$

- Adaptive congestion window decrease: $W \leftarrow (1 - \frac{\alpha}{2})W$
DCTCP in Action

Setup: Win 7, Broadcom 1Gbps Switch
Scenario: 2 long-lived flows, \( K = 30\text{KB} \)
Throughput-Latency Tradeoff

For TCP:
Throughput $\rightarrow$ 75%

Parameters:
- link capacity = 10Gbps
- RTT = 480μs
- smoothing constant (at source), $g = 0.05.$
DCTCP only for data centres?

- named for a feasible deployment scenario
  - a change to all senders, receivers and switches*
- not intended to be its sole applicability
  - addresses high bandwidth-delay product
  - should be applicable to slow links & long RTTs
- only tested down to 100Mb/s so far†

\[
100\text{Mb/s} \times 500\ \mu\text{s} = 250\text{kb/s} \times 200\text{ms}
\]

* Switches/routers only require reconfig if they support ECN
  senders (and receivers) require implementation change
† An issue with a wide range of RTTs has been addressed
DCTCP activity

- E2e Transport
  - In Windows 8 Server data center template
- I-D for DCTCP feedback (intended EXP)
  [draft-kuehlewind-tcpm-accurate-ecn-01]
- AQM
  - Existing kit: Just a degenerate config of RED
  - Can be implemented as just a step at K packets (single ‘if’ command)
  - For zero-delay can use a virtual queue [RC5670]
    - hardware implementations ["How to Build a Virtual Queue from Two Leaky Buckets"]
    - see HULL for specifics with DCTCP
- Analysis, papers, Linux & ns2 implementation, etc
  - <http://www.stanford.edu/~alizade/Site/DCTCP.html>
  - SIGCOMM paper gives entry point
DCTCP: differences from traditional AQMs
  e.g. CoDel

• source smooths signals
  • not the queue

• source responds to extent of signals
  • not just their existence

• designed for ECN only
which node owns the RTT?

- we want to smooth away queues that disappear within ~1 RTT
  - but which RTT?

the network?

- traditional AQMs hold back signals for the ‘nominal’ worst-case (long RTT)
- DCTCP signals immediately
  - no ‘nominal’ RTT to configure / hard-code / adapt

the host?

- each DCTCP flow smooths over its own RTT
- short RTT flows can fill troughs & avoid peaks on behalf of longer ones (and themselves)
DCTCP & CoDel
the Best is the Friend of the Good

But DCTCP’s approach only makes sense with ECN…
ECN and drop are not equivalent

• ECN is solely a signal
  • no problem sending out a burst of ECN and later smoothing it away at the source

• drop is both an impairment and a signal
  • simultaneously want to avoid it and hear it
  • a burst of loss can’t just be smoothed away
    – collateral damage from timeouts etc.
Can smoothing on the host interoperate with smoothing in the network?

current rule (paraphrased from RFC 3168)
  • Signal ECN when queue would otherwise drop
  • Respond to ECN exactly as a drop
  • intended to prevent starvation of one by the other

proposal: overload the meaning of an ECN-capable pkt
  • for queue, ECN also means SHOULD NOT smooth
  • for transport, ECN also means SHOULD smooth
  • under persistent congestion
    • need to ensure shares stabilise and no-one starves
    • despite different dynamics
interoperability between old & overloaded meanings of ECN

ticks are based on conjecture, not experimental evidence

---

1. don’t get full gain in latency until host upgrades as well
2. doubly delayed response to congestion
message

• zero-config AQMs are good
  • CoDel for drop
  • simple step for ECN
    – far greater potential gains

• in parallel to CoDel field testing
  • work on interop with unsmoothed AQM for ECN
  • otherwise the lazy option (ECN = drop) prevails
  • would be a wasted opportunity
Q&A

Je ne savais pas le mieux était si simple
Data Center TCP Algorithm

Switch side:
- Mark packets when \textbf{Queue Length} > K

Sender side:
- Maintain \textit{moving average} of \textit{fraction} of marked packets \( (\alpha') \)
  
  each RTT: \[ F = \frac{\text{# of marked ACKs}}{\text{Total # of ACKs}} \quad \Rightarrow \quad \alpha \leftarrow (1 - g)\alpha + gF \]
- Adaptive congestion window decrease: \[ W \leftarrow (1 - \frac{\alpha}{2})W \]