Revisiting Old Friends: Is CoDel Really Achieving What RED Cannot?

Nicolas Kuhn [1] Emmanuel Lochin [2] Olivier Mehani [3]

[1]IMT Telecom Bretagne, France

[2]Université de Toulouse, France

[3]National ICT Australia, Australia

Table of content

Context - History of AQM

Deployment of loss-based TCP

- TCP flows competing on a bottleneck would back off at the same moment (tail drops)
- $\Rightarrow$ under utilization of the available capacity
- $\Rightarrow$ lots of loss events

Active Queue Management (AQM)

- a solution to avoid loss synchronization
- queue management schemes that drop packets before tail drops occur
- due to operationnal and deployment issues: $\Rightarrow$ **no AQM** scheme has been turned on

Buffer size in the routers

- to overcome from physical layer impairments (fluctuating bandwidth)
- to avoid loss events

Context - History of AQM

Deployment of loss-based TCP

- TCP flows competing on a bottleneck would back off at the same moment (tail drops)
- ⇒ under utilization of the available capacity
- ⇒ lots of loss events

Active Queue Management (AQM)

- a solution to avoid loss synchronization
- queue management schemes that drop packets before tail drops occur
- due to operationnal and deployment issues: ⇒ **no AQM** scheme has been turned on

Buffer size in the routers

- to overcome from physical layer impairments (fluctuating bandwidth)
- to avoid loss events

Context - History of AQM

Deployment of loss-based TCP

- TCP flows competing on a bottleneck would back off at the same moment (tail drops)
- $\Rightarrow$ under utilization of the available capacity
- $\Rightarrow$ lots of loss events

Active Queue Management (AQM)

- a solution to avoid loss synchronization
- queue management schemes that drop packets before tail drops occur
- due to operationnal and deployment issues: $\Rightarrow$ **no AQM** scheme has been turned on

Buffer size in the routers

- to overcome from physical layer impairments (fluctuating bandwidth)
- to avoid loss events

Context - Bufferbloat

Origins of the *bufferbloat*

- deployment of **aggressive congestion control** (such as TCP CUBIC)
- **large buffers** in the routers
- $\Rightarrow$ permanent queuing in the routers
- $\Rightarrow$ high queuing delay
- $\Rightarrow$ network latency

AQM

In the past proposed to avoid loss synchronisation, is one solution for the *bufferbloat*:

- adapt the knowledge of AQM schemes to control the queuing delay in the routers
- in the 90's: RED was based on the number of packets in the buffer
- recent proposals: PIE and CoDel are based on the queuing delay

Context - Bufferbloat

Origins of the *bufferbloat*

- deployment of **aggressive congestion control** (such as TCP CUBIC)
- **large buffers** in the routers
- ⇒ permanent queuing in the routers
- ⇒ high queuing delay
- ⇒ network latency

AQM

In the past proposed to avoid loss synchronisation, is one solution for the *bufferbloat*:

- adapt the knowledge of AQM schemes to control the queuing delay in the routers
- in the 90's: RED was based on the number of packets in the buffer
- recent proposals: PIE and CoDel are based on the queuing delay

Objectives

Considering that

- ⇒ a performance comparison of RED, CoDel and PIE is missing
- ⇒ their impact on various congestion controls is missing

Our objectives are

- ⇒ compare the performance of RED and CoDel with various TCP variants (delay-based / loss-based)
- ⇒ discuss deployment and auto-tuning issues

What we do not consider:

- PIE: code was missing when running the simulations
- FQ-CoDel (hybrid scheduling/CoDel): did not exist at the time of the study

Objectives

Considering that

- ⇒ a performance comparison of RED, CoDel and PIE is missing
- ⇒ their impact on various congestion controls is missing

Our objectives are

- ⇒ compare the performance of RED and CoDel with various TCP variants (delay-based / loss-based)
- ⇒ discuss deployment and auto-tuning issues

What we do not consider:

- PIE: code was missing when running the simulations
- FQ-CoDel (hybrid scheduling/CoDel): did not exist at the time of the study

Objectives

Considering that

- ⇒ a performance comparison of RED, CoDel and PIE is missing
- ⇒ their impact on various congestion controls is missing

Our objectives are

- ⇒ compare the performance of RED and CoDel with various TCP variants (delay-based / loss-based)
- ⇒ discuss deployment and auto-tuning issues

What we do not consider:

- PIE: code was missing when running the simulations
- FQ-CoDel (hybrid scheduling/CoDel): did not exist at the time of the study

Table of content

RED and CoDel

Random Early Detection (RED) from the 90's

- dropping probability, p_{drop}: function of the number of packets in the queue
- depending on p_{drop}, incoming packets might be dropped

Controlled Delay (CoDel) to tackle *bufferbloat*

- measures the queuing delay for each packet, $qdel_p$
- N_{drop} is the cumulative number of drop events
- every *interval* (default is 100 ms), while dequeuing p:

$qdel_p >$ *target delay (5 ms)*	$qdel_p <$ *target delay*
p is dropped $N_{drop}++$ $interval = \frac{interval}{\sqrt{N_{drop}}}$	p is dequed $N_{drop} = 0$ $interval = 100\,ms$

RED and CoDel

Random Early Detection (RED) from the 90's

- dropping probability, p_{drop}: function of the number of packets in the queue
- depending on p_{drop}, incoming packets might be dropped

Controlled Delay (CoDel) to tackle *bufferbloat*

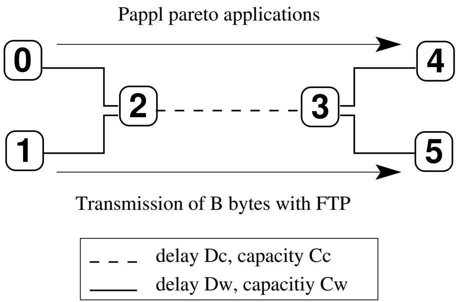
- measures the queuing delay for each packet, $qdel_p$
- N_{drop} is the cumulative number of drop events
- every *interval* (default is 100 ms), while dequeuing p:

$qdel_p > target\ delay\ (5\,ms)$	$qdel_p < target\ delay$
p is dropped	p is dequed
$N_{drop}++$	$N_{drop} = 0$
$interval = \frac{interval}{\sqrt{N_{drop}}}$	$interval = 100\,\text{ms}$

Table of content

Topology and traffic

Topology

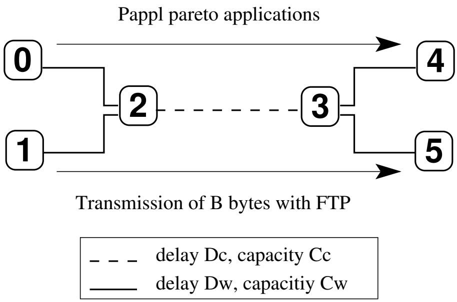


Traffic

- P_{appl} applications transmit a file (size generated following a Pareto law): consistent with the distribution of the flow size measured in the Internet. This traffic is injected to dynamically load the network.
- *FTP* transmission of B bytes to understand the protocols impacts.

Topology and traffic

Topology

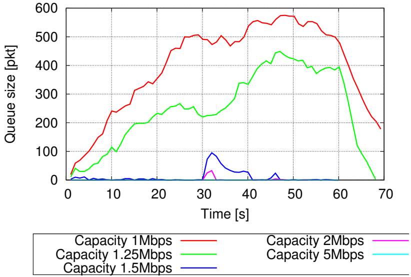


Traffic

- P_{appl} applications transmit a file (size generated following a Pareto law): consistent with the distribution of the flow size measured in the Internet. This traffic is injected to dynamically load the network.
- *FTP* transmission of B bytes to understand the protocols impacts.

Network and application characteristics

Finding central link capacities, C_c, causing Bufferbloat ($P_{appl} = 100$, $C_w = 10$ Mbps)

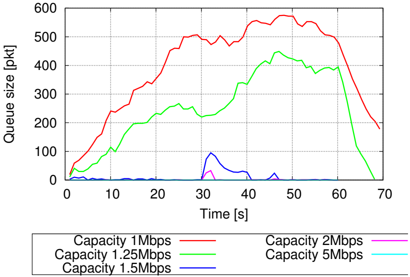


Selecting capacity, P_{app} and buffer size

- $C_c = 1$ Mbps $\Rightarrow$ constant buffering
- $P_{app} = 100$
- buffer sizes: 1) $\ll$ BDP ($q = 10$), 2) $\simeq$ BDP ($q = 45$), 3) $\gg$ BDP

Network and application characteristics

Finding central link capacities, C_c, causing Bufferbloat ($P_{appl} = 100$, $C_w = 10$ Mbps)

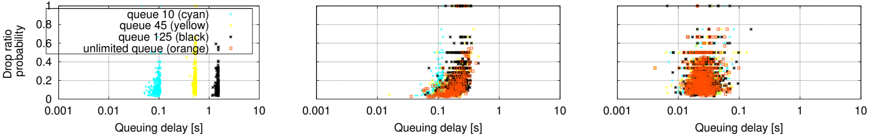


Selecting capacity, P_{app} and buffer size

- $C_c = 1$ Mbps $\Rightarrow$ constant buffering
- $P_{app} = 100$
- buffer sizes: 1) $\ll$ BDP ($q = 10$), 2) $\simeq$ BDP ($q = 45$), 3) $\gg$ BDP

Table of content

Drop ratio *vs.* queuing delay



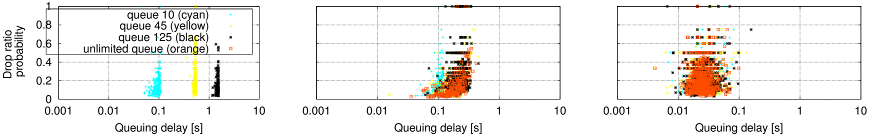
(a) DropTail (b) RED (c) CoDel

Figure: TCP CUBIC: Drop ratio *versus* queuing delay (TCP Vegas shows the same behaviour)

Interpretation

- introduction of RED or CoDel ⇒ drop events whatever the queue size
- with DropTail, the queuing delay is maximised by the size of the queue
- queuing delay is between 0.01 s and 0.1 s with CoDel
- queuing delay is between 0.1 s and 0.5 s with RED

Drop ratio *vs.* queuing delay



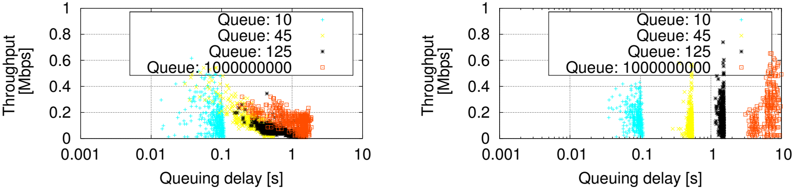
(a) DropTail (b) RED (c) CoDel

Figure: TCP CUBIC: Drop ratio *versus* queuing delay (TCP Vegas shows the same behaviour)

Interpretation

- introduction of RED or CoDel ⇒ drop events whatever the queue size
- with DropTail, the queuing delay is maximised by the size of the queue
- queuing delay is between 0.01 s and 0.1 s with CoDel
- queuing delay is between 0.1 s and 0.5 s with RED

VEGAS and CUBIC with DropTail



(a) VEGAS

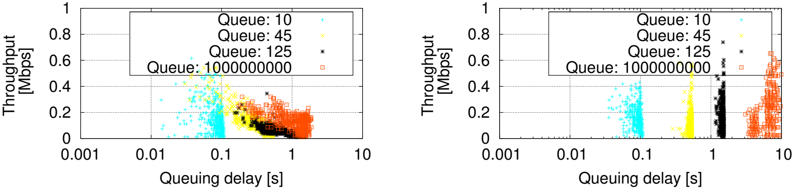
(b) CUBIC

Figure: DropTail: Achieved throughput *versus* queuing delay for varying buffer sizes

Interpretation

- DropTail and VEGAS: throughput decreases when the queue size increases. When the queue is large, VEGAS reacts to queuing delay increases.
- DropTail and CUBIC: throughput increases with larger queues. The

VEGAS and CUBIC with DropTail



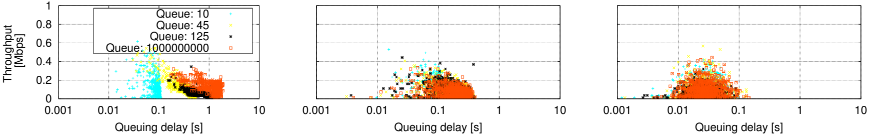
(a) VEGAS (b) CUBIC

Figure: DropTail: Achieved throughput *versus* queuing delay for varying buffer sizes

Interpretation

- DropTail and VEGAS: throughput decreases when the queue size increases. When the queue is large, VEGAS reacts to queuing delay increases.
- DropTail and CUBIC: throughput increases with larger queues. The

VEGAS with RED or CoDel



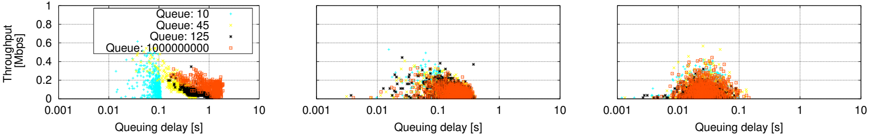
(a) DropTail (b) RED (c) CoDel

Figure: VEGAS w/ AQM: Achieved throughput *versus* queuing delay

Interpretation

- the queuing delay is between 0.01 s and 0.1 s with CoDel
- the queuing delay is between 0.1 s and 0.5 s with RED
- the throughput is the same whatever the choice of the AQM is.

VEGAS with RED or CoDel



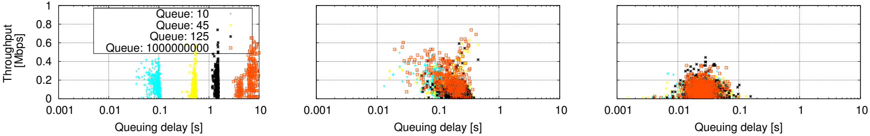
(a) DropTail (b) RED (c) CoDel

Figure: VEGAS w/ AQM: Achieved throughput *versus* queuing delay

Interpretation

- the queuing delay is between 0.01 s and 0.1 s with CoDel
- the queuing delay is between 0.1 s and 0.5 s with RED
- the throughput is the same whatever the choice of the AQM is.

CUBIC with RED or CoDel



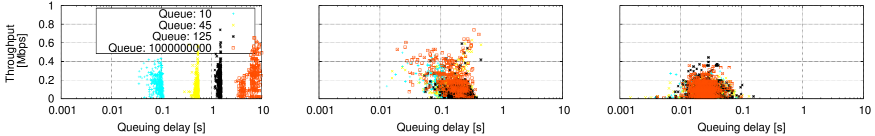
(a) DropTail (b) RED (c) CoDel

Figure: CUBIC w/ AQM: Achieved throughput *versus* queuing delay

Interpretation

- the queuing delay is between 0.01 s and 0.1 s with CoDel
- the queuing delay is between 0.1 s and 0.5 s with RED
- the throughput is larger with RED (up to 0.75 Mbps) than with CoDel (up to 0.45 Mbps)

CUBIC with RED or CoDel



(a) DropTail (b) RED (c) CoDel

Figure: CUBIC w/ AQM: Achieved throughput *versus* queuing delay

Interpretation

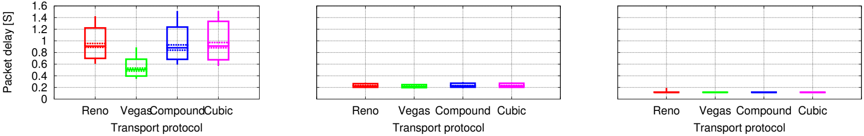
- the queuing delay is between 0.01 s and 0.1 s with CoDel
- the queuing delay is between 0.1 s and 0.5 s with RED
- the throughput is larger with RED (up to 0.75 Mbps) than with CoDel (up to 0.45 Mbps)

Early conclusions

- CoDel is a good candidate to reduce latency
- RED may reduce the latency as well
- RED allows to transmit more traffic and better exploit the capacity of the bottleneck
- $\Rightarrow$ a better trade-off might exist between latency reduction and more efficient capacity use than the one of CoDel

Table of content

Application Delay



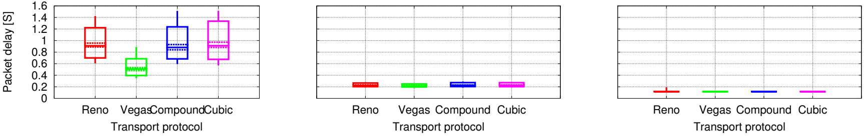
(a) DropTail (b) RED (c) CoDel

Figure: Packet transmission times

Interpretation

- RED and CoDel enable reduction of the latency compared to DropTail
- CUBIC the packet transmission time is reduced by 87% with CoDel and by 75% with RED
- the median packet transmission time with CUBIC and CoDel is 115 ms compared to 226 ms with RED
- latency is reduced by 44% when the congestion control is VEGAS

Application Delay



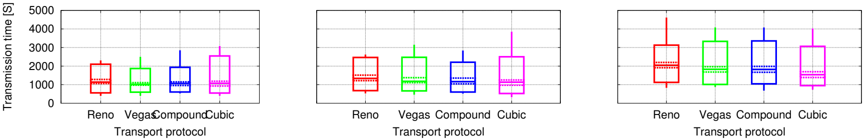
(a) DropTail (b) RED (c) CoDel

Figure: Packet transmission times

Interpretation

- RED and CoDel enable reduction of the latency compared to DropTail
- CUBIC the packet transmission time is reduced by 87% with CoDel and by 75% with RED
- the median packet transmission time with CUBIC and CoDel is 115 ms compared to 226 ms with RED
- latency is reduced by 44% when the congestion control is VEGAS

Application Goodput



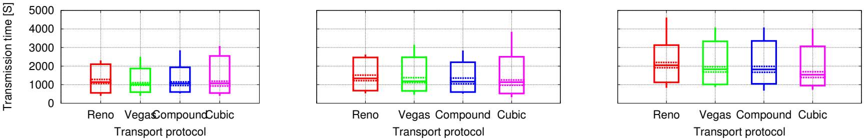
(a) DropTail (b) RED (c) CoDel

Figure: Time needed to transmit 10 MB

Interpretation

- dropping events generated by RED do not impact this transmission time much
- with CUBIC, introducing RED increases the median transmission time of 10 MB by 5% compared to DropTail
- with CUBIC, introducing CoDel results in an increase of 42% of this transmission time.

Application Goodput



(a) DropTail (b) RED (c) CoDel

Figure: Time needed to transmit 10 MB

Interpretation

- dropping events generated by RED do not impact this transmission time much
- with CUBIC, introducing RED increases the median transmission time of 10 MB by 5% compared to DropTail
- with CUBIC, introducing CoDel results in an increase of 42% of this transmission time.

Table of content

Deployment of CoDel and RED

- AQM: a solution to tackle the *bufferbloat* that SHOULD be deployed. RED and CoDel enable to reduce the latency: in our simulations, CoDel reduced the latency by 87% and RED by 75%
- a trade-off must be found between reducing the latency and degrading the end-to-end performance: CoDel increased the time needed to transmit 10 MB by 42%, while RED only introduced a 5% increase
- deployment issues of RED: RED was not tuned on because it is hard to configure for a given network. Adaptive RED (proposed after Gentle RED) has less deployment issues but was not deployed
- deployment issues with CoDel: in a document published by CableLabs, the authors explain that they had to adjust CoDel's target value to account for MAC/PHY delays even for packets reaching an empty queue. There is a need for a large parameters sensitivity
- consider the intended traffic to be carried: as an example, conjoint deployment of LEDBAT and AQM is a problem as this protocol would not be "low-than-best-effort" anymore.

Deployment of CoDel and RED

- AQM: a solution to tackle the *bufferbloat* that SHOULD be deployed. RED and CoDel enable to reduce the latency: in our simulations, CoDel reduced the latency by 87% and RED by 75%
- a trade-off must be found between reducing the latency and degrading the end-to-end performance: CoDel increased the time needed to transmit 10 MB by 42%, while RED only introduced a 5% increase
- deployment issues of RED: RED was not tuned on because it is hard to configure for a given network. Adaptive RED (proposed after Gentle RED) has less deployment issues but was not deployed
- deployment issues with CoDel: in a document published by CableLabs, the authors explain that they had to adjust CoDel's target value to account for MAC/PHY delays even for packets reaching an empty queue. There is a need for a large parameters sensitivity
- consider the intended traffic to be carried: as an example, conjoint deployment of LEDBAT and AQM is a problem as this protocol would not be "low-than-best-effort" anymore.

Deployment of CoDel and RED

- AQM: a solution to tackle the *bufferbloat* that SHOULD be deployed. RED and CoDel enable to reduce the latency: in our simulations, CoDel reduced the latency by 87% and RED by 75%
- a trade-off must be found between reducing the latency and degrading the end-to-end performance: CoDel increased the time needed to transmit 10 MB by 42%, while RED only introduced a 5% increase
- deployment issues of RED: RED was not tuned on because it is hard to configure for a given network. Adaptive RED (proposed after Gentle RED) has less deployment issues but was not deployed
- deployment issues with CoDel: in a document published by CableLabs, the authors explain that they had to adjust CoDel's target value to account for MAC/PHY delays even for packets reaching an empty queue. There is a need for a large parameters sensitivity
- consider the intended traffic to be carried: as an example, conjoint deployment of LEDBAT and AQM is a problem as this protocol would not be "low-than-best-effort" anymore.

Deployment of CoDel and RED

- AQM: a solution to tackle the *bufferbloat* that SHOULD be deployed. RED and CoDel enable to reduce the latency: in our simulations, CoDel reduced the latency by 87% and RED by 75%
- a trade-off must be found between reducing the latency and degrading the end-to-end performance: CoDel increased the time needed to transmit 10 MB by 42%, while RED only introduced a 5% increase
- deployment issues of RED: RED was not tuned on because it is hard to configure for a given network. Adaptive RED (proposed after Gentle RED) has less deployment issues but was not deployed
- deployment issues with CoDel: in a document published by CableLabs, the authors explain that they had to adjust CoDel's target value to account for MAC/PHY delays even for packets reaching an empty queue. There is a need for a large parameters sensitivity
- consider the intended traffic to be carried: as an example, conjoint deployment of LEDBAT and AQM is a problem as this protocol would not be "low-than-best-effort" anymore.

Appendix

- On CoDel's target value:[1]

  > *The default target value is 5 ms, but this value SHOULD be tuned to be at least the transmission time of a single MTU-sized packet at the prevalent egress link speed (which for e.g. 3 Mbps and MTU 1500 is ∼15 ms).*

- On LEDBAT not being LBE over AQMs:[2]

  > *[. . .] RED invalidates LEDBAT low priority [with] similar throughput of TCP and LEDBAT, both at flow and aggregate levels*

---

[1] T. Hoeiland-Joergensen et al. *FlowQueue-CoDel*. Internet-Draft draft-hoeiland-joergensen-aqm-fq-codel-00.txt. Mar. 2014. URL: http://www.rfc-editor.org/internet-drafts/draft-hoeiland-joergensen-aqm-fq-codel-00.txt, sec. 5.1.2.

[2] Y. Gong et al. "Interaction or Interference: Can AQM and Low Priority Congestion Control Successfully Collaborate?" In: *CoNEXT 2012*. Nice, France, 2012, pp. 25–26. DOI: 10.1145/2413247.2413263. URL: http://conferences.sigcomm.org/co-next/2012/eproceedings/student/p25.pdf, sec. 2.