ECN-Enhanced CoDel AQM

Dhulfiqar A. Alwahab(✉) and Sándor Laki(✉)

Faculty of Informatics, ELTE Eötvös Loránd University, Budapest, Hungary
{aalwahab,lakis}@inf.elte.hu

Abstract. Novel interactive applications require small end-to-end latency for providing end users with good Quality of Experience. To prevent the bufferbloat problem causing increased delay, various AQM solutions have recently emerged. One of the most widely adopted method is called CoDel that detects the increased delay by maintaining the per packet sojourn times and compering them to a desired target delay. Accordingly, congestion situation is indicated when the sojourn time exceeds the target delay. CoDel applies a drop-based feedback to notify the responsive sources about congestion only when the deviation from the target delay is permanent. Dropping packets is then compensated by packet re-transmissions in case of TCP which worsens utilization. In this paper, we propose an ECN-enhanced CoDel that distinguishes between low and high levels of congestion, and ECN-marks or drops the packets, respectively. The performance of the proposed method is analyzed in thorough NS-3 simulations with variable number of flows. The proposed ECN-marking reduces the number of packets re-transmissions and provides similarly good characteristics including convergence time and fairness to the original drop-based CoDel.

Keywords: ECN · DCE · AQM · CoDel · Congestion · Bufferbloat

1 Introduction

End-to-end delay became one of the key issues that attract network researchers and developers since most of the novel applications require fast and reliable Internet service. The end to end delay is affected by many additive factors: transmission, propagation, processing and queuing delays. Processing delay is mostly negligible, while transmission and propagation delays are determined by the physical properties of network paths. Only queuing delay can be managed to reduce the overall end to end delay in heterogeneous networks. The traditional congestion control mechanism of TCP by its nature tries to fill the queue till it notices a packet loss to slow down the sending rate, this mechanism introduces what is known as saw-tooth behavior. Using large buffers to store the packet in routers causes high latency because of increased queuing delay. This may lead to the bufferbloat problem that reduces the network performance by increasing the end-to-end delay specifically; when real time applications such as panoramic real-time video, online games, remote control, etc., or when sharing the same bottleneck link with packet from other nodes or applications. To improve overall

M. Di Felice et al. (Eds.): WWIC 2019, LNCS 11618, pp. 157–169, 2019.
https://doi.org/10.1007/978-3-030-30523-9_13

utilization and handle these applications' sensitivity to throughput fluctuations, fast transmission and fast recovery are used as built-in techniques in TCP to minimize the impact of loss [1]. Bufferbloat problem can be addressed by two approaches: (1) by using end-to-end congestion control; (2) by involving Active Queue Management in the network devices.

Active Queue Management (AQM) had been considered as the best way to deal with the bufferbloat problem. The aims of any AQM algorithm are: absorb packet bursts; prevent packets to spend long time in the queue; deal with aggressive or misbehaving flows, as well as support Explicit Congestion Notification (ECN) [2]. The problem of formal AQM mechanism, like (Tail drop or Random Early Detection (RED)), lies in parameters setting on their algorithms where the parameters needed to be tuned depending on the actual traffic and network condition [3]. The parametrization issue urges researchers to directing their researches toward parameter-less or auto-tuning AQM mechanisms like Controlled Delay Active Queue Management (CoDel) or Proportional Integral Controller Enhanced (PIE). Another problem that limits all the traditional AQM mechanism is that they designed to operate only at layer three [4]. ECN [1] has been introduced as mean for notifying the end node about congestion by marking packets in the ECN-enabled routers based on the decision of the applied AQM mechanism instead of dropping them. This technique may enhance the overall performance of an AQM algorithm [5]. One of the biggest problem with TCP-based communication is that sources only reduce their sending rate after they noticed a packet loss. Retransmission of packets is consuming resources and increasing delay, eventually leading to worst network utilization.

In this paper, we introduce a conservative ECN marking scheme for CoDel that at low congestion level uses ECN-marking while at high and permanent congestion situations applies the original drop-based strategy of CoDel. The key benefits of the proposed method include: (1) It does not affect the good properties of CoDel; (2) It significantly reduces the number of packet retransmissions caused by drops; (3) The implementation is incremental and does not require the deep modification of the original CoDel.

The rest of this paper is organized as follows. In Sect. 2 we summarized the most related research work. Section 3 deals with the technique used to combine ECN with CoDel in more real-way. Next, in Sect. 4 we present simulation results done using NS-3 that prove our design. Finally, we present our conclusion in the last Section.

2 Related Work

The literature on AQM is vast, a lot of research papers for AQM has been proposed. In this section we will review a few researches that are most related to our work. Authors in [6] propose CoDel-Lifo as a new AQM algorithm to reduce the loss rate in multipath TCP congestion control mechanism, CoDel-Lifo differs from other AQM by treat the most recent packets with higher priority. Also, they compare coDel-lifo with CoDel, and Drop-Tail queue algorithms to show that

CoDel-lifo precedes in reduction the number of packets drop, goodput improvement and keep Round Trip Time (RTT) low. CoDel-lifo improves the performance of the network by depending on end nodes (TCP) and router (AQM). Authors in [3] shows its possible to adapt the setting of CoDel and Fq-CoDel to preserve low queuing delay and high link utilization. The experiments were presented over an emulated test bed and a satellite network in a capacity-limited network. Results show that the modification improves the download time and reduces the latency. In [4] a new class of AQM has been introduced called Active Sense Queue Management (ASQM). A comparison between the proposed mechanism and the traditional AQM (CoDel and PiE) was also presented. Results show the ability of ASQM to manage buffer bloat by decreasing queuing delay more than the traditional AQM. The authors in [7] analyze in theoretical and empirical way to address this question: is there a universal packet scheduling algorithm? After answered this question, they proved theoretically that least slack time first (LSTF) can be a universal solution, and empirically can closely reply to a wide range of scheduling algorithms. Then, they mentioned how to use LSTF with AQM in term of emulation (CoDel and ECN) on the edge nodes. without modifying the network's core. In [8], the benefits of using ECN with AQM, to avoid congestion, were discussed. These benefits can be summarized: (1) improve throughput up to 2% in some type of network. (2) Reduce head-of-line blocking by reduce the number of drop in the router (AQM). (3) Reduce probability of recovery time objective Retransmission Timeout (RTO) expiry by decrease the probability of loss. (4) improve the performance of latency-critical application by decoupling congestion control from loss. (5) Make incipient congestion visible. (6) opportunity for new transport mechanism. In [9], A PID controller with ECN based on neural network is presented, result of this work shows a better performance in compare to RED and typical PID AQM on the queue stability and mean time delay.

3 CoDel and ECN Marking

CoDel allows the sojourn time to be higher than target delay time for one interval, indicating permanent congestion, before it starts dropping packets. In practice this interval parameter is an order of magnitude larger than the target delay and ideally proportional to the observed RTT in the given system. After permanent congestion is detected, CoDel starts dropping packets from the head of the queue and applies a control rule to determine the next drop time. The time of the next drop is decreased gradually to control the TCP behavior in the network. One can see that CoDel's behavior only depends on the congestion level. When the congestion level is high, the sojourn time goes above a threshold value (target delay). When the sojourn time remains above the target for a duration more than one interval, CoDel starts dropping packets from the buffer. After another interval time, if the sojourn time goes back below the target delay, it stops the dropping process. Otherwise, CoDel enters the next drop state for a time interval equal to (interval $\sqrt{(Number\_of\_Drop)}$). On the other hand, it has been shown in [10,11] that CoDel cannot always control the queue delay, while may decrease the bottleneck-link utilization. Additionally, it cannot tune itself for network conditions and objectives without the adaptation of its parameters [3].

Explicit Congestion Notification (ECN) enables routers to manage the amount of cross traffic on the time scale of one RTT by marking the IP packets with a Congestion Experienced (CE) flag. When this flag is received by the TCP end-point, an ECN Echo (ECE) is sent back to the source. The source reacts to ECE flags similarly to packet drops with a much faster feedback loop, reducing its sending rate.

Figure 1(a) [4] overviews the original CoDel algorithm, where m is the interval time value in default (100 ms), τ is the target value (default 5 ms), Spi is the sojourn time of packets to be dequeued. $t1$ is first packet dequeue time in each m long interval, $t2$ is $t1 + m$, and Tpi is the packet's dequeue time when the packet is removed from the queue. In [12], the interval has been chosen with regard to RTT to give endpoint time to react for the congestion, the default value (100 ms) is recommended since this value works well across a range of RTT from 10ms to 100 ms (or even more). The role of the interval is to ensure that CoDel algorithm is up to date and it reacts to delay experience in the last epoch of the length interval [13]. When sojourn time of the packet goes higher than τ (5 ms), the original CoDel waits an interval (m) to check the sojourn time of the next packet. During this interval m, packets are forwarded or dropped. In the early appearance of congestion, before CoDel enter the drop state (in the first interval (m, usually 100 ms), the packet is still forwarded in spite of crossing the target delay by the packet's sojourn time. It is expected that TCP sources, by its nature, will continually increase the traffic rate because they did not know about the congestion appearance (no packets loss) at that moment.

According to our proposal, ECN will take place here, in the first interval (m) when the sojourn time first exceeds the target value. To avoid the effect of short temporal bursts experienced in the first interval period before entering packet dropping phase, we introduce a parameter C and mark a packet with probability P if the sojourn time goes above the threshold $C\tau$ (C times the target delay, indicating the beginning of a permanent congestion) Note that we tried several C and P values and found that $C = 1.5$ and $P = 0.3$ can significantly reduce the number of re-transmissions while provide similar flow fairness as original CoDel. The role of ECN markings are to inform the receivers that sojourn time significantly pass the target value in these packets. The receiver then notifies the source of this packet about the congestion appearance through the acknowledgment, resulting in a reduced sending window at the source. Figure 1(b) summarizes the ECN-enhanced CoDel algorithm. If the packets arrived to the AQM router and the queue is full, packet will be drop automatically, in the ECN-enhanced CoDel real packet drops can rarely experienced. In the remaining part of this paper, we will show that this slight modification can reduce the number of unnecessary re-transmissions while reducing the observed average sojourn times by multiple factors.

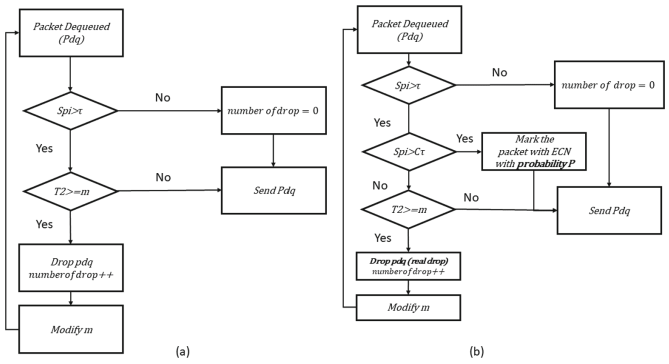

Fig. 1. CoDel (a) and ECN-enhanced CoDel (b) algorithms.

4 Simulation Results

The experiment of ECN-enhanced CoDel has been implemented in NS-3 Network Simulator,using respectively the IP and TCP stacks of Direct Code Extension (DCE). DCE is a framework that offers many advantages, among them are; enable executing network protocols and unmodified applications (i.e TCP) on the traditional library founded in the Linux kernel (libos); and its compatibility with NS-3 network simulator allowing fully reproducible experiments where, in this case, NS-3 is only responsible for network topology and packets issue (forwarding, marking or dropping). Based on these two-main advantages of DCE, our modification only affected the CoDel queue discipline model of NS-3, letting the TCP protocol react to ECN notification as it is implemented in the kernel [14]. We assume that in this way more realistic results can be achieved. The topology (dumbbell topology) of the network is shows in Fig. 2 that has been chosen because it is used in a variety of TCP performance comparison papers [15]. Accordingly, two-pairs of end-nodes are connected through two network devices (routers), and the link between the routers forms the bottleneck link of 50 Mbps in our case. In this way, there will be degradation in the goodput and packet will be enqueued even when there is only one flow. The ECN-enhanced CoDel has been tested under different number of TCP flows (2, 5 and 10 flows) and the results have been compared to the original CoDel.

First we compare the performance of ECN-enhanced CoDel with the normal one, using only two TCP flows. Source 1 starts application at time 0 s till the end of the simulation, and the application in Source 2 starts in the interval 3–40 s, randomly.

Figure 3 compares the number of re-transmitted packets for the ECN-enhanced and the original CoDel. In the original CoDel, the saw-tooth of TCP behavior is obvious in the figure, resulting in periodic re-transmission peaks. Whereas the ECN-enhanced CoDel reduces the re-transmissions in the network by informing the source node to reduce the sending rate earlier, using the previously described ECN mechanism.

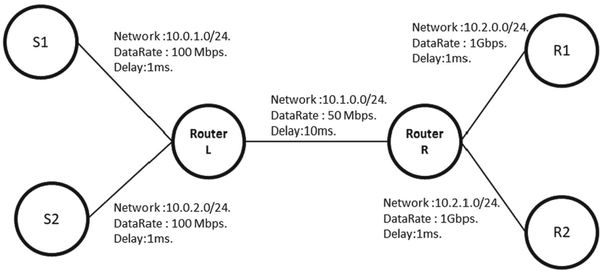
S1
S2
Router L
Router R
R1
R2
Network :10.0.1.0/24. DataRate : 100 Mbps. Delay:1ms.
Network :10.0.2.0/24. DataRate : 100 Mbps. Delay:1ms.
Network :10.1.0.0/24. DataRate : 50 Mbps. Delay:10ms.
Network :10.2.0.0/24. DataRate : 1Gbps. Delay:1ms.
Network :10.2.1.0/24. DataRate : 1Gbps. Delay:1ms.

Fig. 2. Network topology.

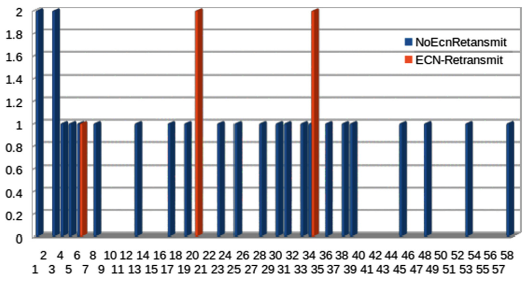

Fig. 3. Number of packets re-transmissions over time (2-TCP flows).

In case of normal CoDel, 32 segments had been lost whereas 10 segments in case of ECN-enhanced CoDel. Figure 4 shows the throughput of each flow in the network between the two nodes using the same application and port numbers. ECN-enhanced CoDel (with parameter $C = 1.5$ and $P = 0.3$) and original CoDel show similar performance but the fairness between the two TCP flows is slightly better for the original CoDel.

Figure 5 depicts the queue-length (byte) in this network within the two TCP flows. ECN-enhanced CoDel shows better result in the usage of buffer by keeping the buffer size low during most of the simulation time, compared to normal CoDel. Figure 6 displays the sojourn time for both variants in nanoseconds. One can observe that the modified CoDel could achieve more stability in the queue during the simulation time.

To test how ECN-enhanced CoDel works in more realistic scenarios, we increased the number of applications to 5 and 10. Each flow represents an application (with pre-specified port number) with different starting and ending time in the simulation. Table 1 details the operation time, source and destination nodes and port number of each application. Figure 7 shows the throughput

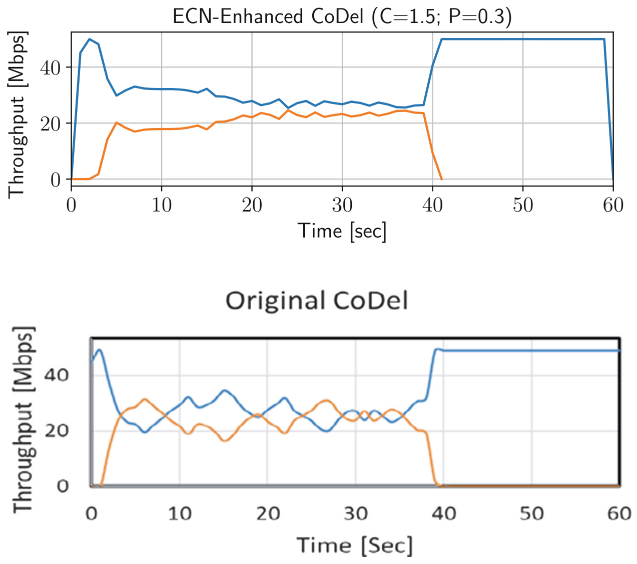

Fig. 4. Throughput of CoDel with/wo ECN (2-TCP flows)

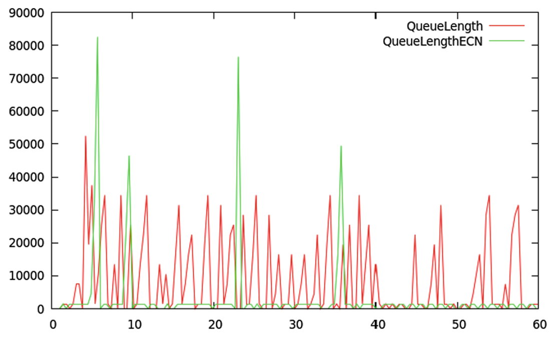

Fig. 5. Queue length in bytes over time (2-TCP Flows).

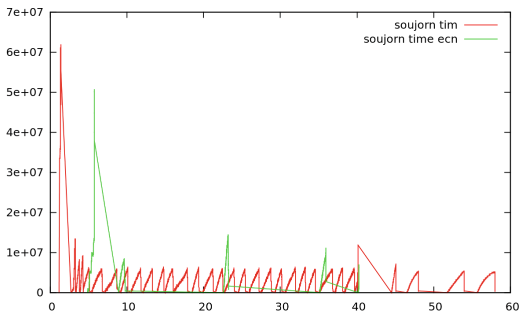

Fig. 6. Sojourn time in ns over time (2-TCP Flows).

Table 1. TCP flows details

Application No.	StartTime	EndTime	Port	Between
1	1	70	9	S1-R1
2	3	40	10	S2-R2
3	2	55	19	S1-R1
4	6	55	20	S2-R2
5	10	40	17	S1-R1
6	10	40	60	S2-R2
7	3	40	100	S1-R1
8	3	40	101	S2-R2
9	3	40	102	S1-R1
10	3	40	103	S2-R2

values of 5 applications generating TCP traffic at the bottleneck link for both ECN-enhanced and normal CoDel, respectively. One can observer that ECN-enhanced CoDel is slightly slower in the rumping up phase of TCP and results in slightly better fairness in stable regions (e.g. in range between 40 and 55 s). However, the difference in flow throughput and fairness is negligible, we can say that from the performance point of view ECN-enhanced CoDel can reach similar properties with less re-transmissions.

Figure 8 shows the total number of re-transmitted packets in the scenario of 5 flows. Accordingly, ECN-Enhanced CoDel reduces the packet loss in compare with the normal one. The total number of lost segment in the ECN-enhanced CoDel is 22 whereas its 99 in the normal CoDel. Figures 9 and 10 show the queue length (byte) and the sojourn time (nanosecond), respectively.

Figure 11 shows the throughput curves when 10 TCP flows are in the system. The observed fairness is similar for ECN-enhanced CoDel and original CoDel while the number of real packet drops is 1 and 441 for ECN-enhanced and original CoDel respectively.

Figure 12 depicts a similar scenario with a bottleneck of 500 Mbps (10 times more than in previous scenarios). The number of TCP flows is 10 while the parameters C and P are also the same as previously. One can note that the characteristics of ECN-Enhanced and the original CoDel are similar and in stable ranges (e.g. from 10 s to 40 s) the modified CoDel provides slightly better fairness and more stable throughput while the number of real packet drops is 1 and 183 for ECN-Enhanced and original CoDel variants, resp.

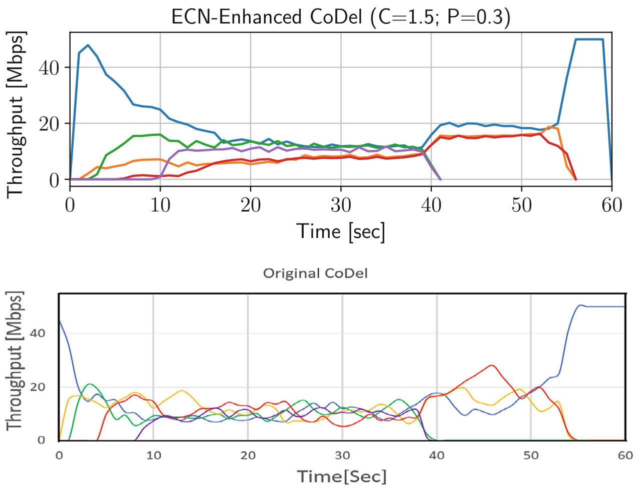

Fig. 7. Throughput with ECN-enhanced/Original CoDel (5-TCP flows)

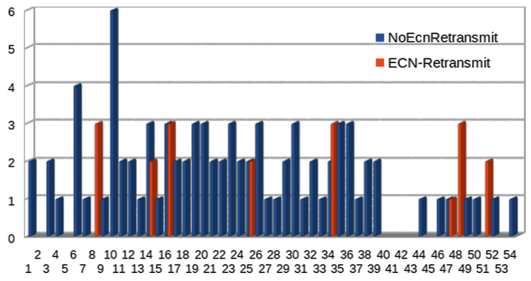

Fig. 8. Packets re-transmission process (5-TCP flows).

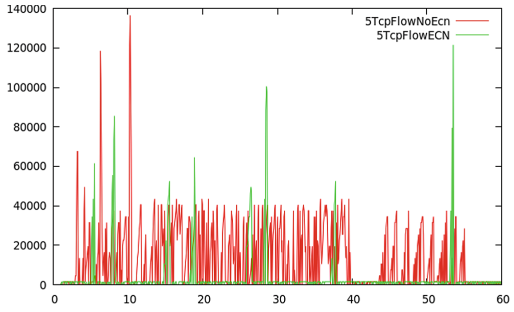

Fig. 9. Queue length (5-TCP flows).

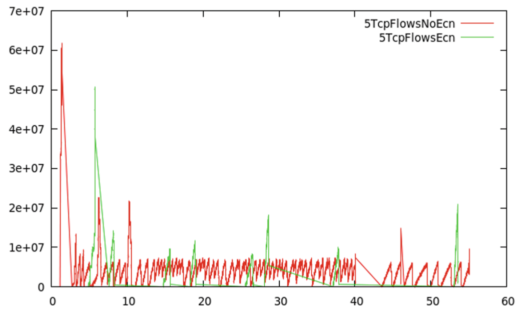

Fig. 10. Sojourn time (5-TCP flows).

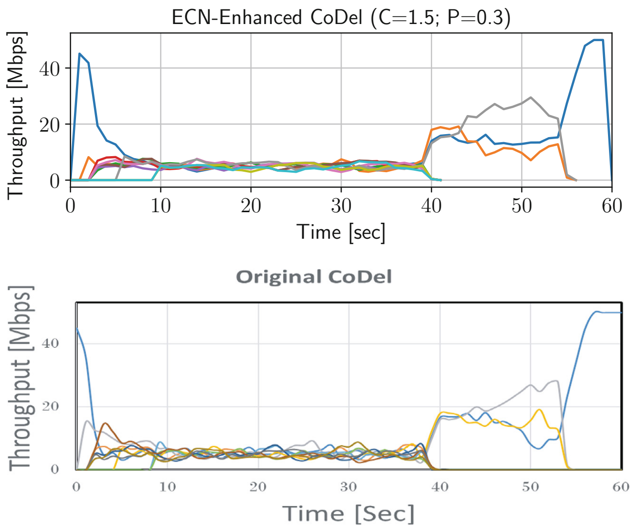

Fig. 11. Throughput with ECN-enhanced/Original CoDel (10-TCP flows)

Figure 13 summarizes the total number of segments loss for all the scenarios. HS is for high speed where links set to 1 Gbit/s. One can observe that ECN-enhanced CoDel reduces the number of packets re-transmission and resources consumption by lowering the number of packets drops.

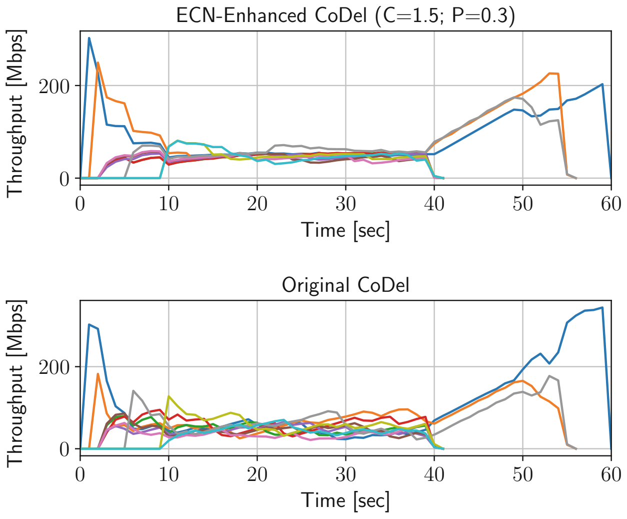

Fig. 12. Throughput with ECN-enhanced/Original CoDel (10-TCP flows) and a 500 Mbps bottleneck

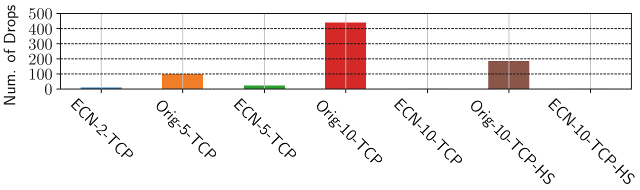

Fig. 13. Packet drop statistics for all scenarios.

5 Conclusions

In this work, we propose a slight modification of CoDel called ECN-enhanced CoDel that exploits the benefits of ECN marking to notify TCP sources faster without segment loss about congestion than traditional CoDel. The key benefit of the proposed method are: (1) It does not affect the good properties of CoDel; (2) It significantly reduces the number of packet re-transmissions caused by bufferbloat drops; (3) It reduces the observed sojourn times; (4) The implementation is incremental and does not require the deep modification of the original CoDel.

The performance of the proposed method has been investigated in simulations under various traffic mixes. In some scenarios TCP flows with ECN-enhanced CoDel show a slightly slower ramping up behaviour that seems to be affected by the introduced C parameter. This work proposes the concept of combining the advantage of using AQM (CoDel) for measuring the packets waiting time inside the inter-networks devices and the benefit of notifying the end-nodes devices in the network about the packets delay by ECN. Testing the proposed algorithm in a more expanded scenario (Real scenario) as well as implementing the test-bed for it are considered in our future work. Additionally, a deeper investigation of this problem and consideration of flow queue concept (FQ-CoDel) is also included in our future work.

Acknowledgments. The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

References

1. Ramakrishnan, K., Floyd, S., Black, D.: The addition of explicit congestion notification (ECN) to IP. No. RFC 3168 (2001)
2. White, G. Pan, R.: A PIE-based AQM for DOCSIS cable modems. Internet-Draft draft-ietf-aqm-docsis-pie-02, Internet Engineering Task Force (2016)
3. Kulatunga, C. et al.: Tackling Bufferbloat in capacity-limited networks. In: 2015 European Conference on Networks and Communications (EuCNC), pp. 381–385. IEEE (2015)
4. Havey, D.M., Almeroth, K.C.: Active sense queue management (asqm). In: IFIP Networking Conference (IFIP Networking), 2015, pp. 1–9. IEEE (2015)
5. Hamann, T., Walrand, J.: A new fair window algorithm for ECN capable TCP (new-ECN). In: INFOCOM, vol. 3, pp. 1528–1536, March 2000
6. Felix, B. et al.: A New Queue Discipline for Reducing Bufferbloat Effects in HetNet Concurrent Multipath Transfer. arXiv preprint arXiv:1609.09314 (2016)
7. Mittal, R. et al.: Universal packet scheduling. In: Proceedings of the 14th ACM Workshop on Hot Topics in Networks, p. 24. ACM (2015)
8. Fairhurst, G., Welzl, M.: The Benefits of using Explicit Congestion Notification (ECN). No. RFC 8087 (2017)
9. Zhou, C., Zhang, L., Chen, Q.: An adaptive PID controller for AQM with ECN marks based on neural networks. In: 7th Asian Control Conference, 2009, ASCC 2009, pp. 779–783. IEEE (2009)
10. Järvinen, I., Kojo, M.: Evaluating CoDel, PIE, and HRED AQM techniques with load transients. In: 2014 IEEE 39th Conference on Local Computer Networks (LCN), pp. 159–167. IEEE (2014)
11. Schwardmann, J., Wagner, D., Kühlewind, M.: Evaluation of ARED, CoDel and PIE. In: Kermarrec, Y. (ed.) EUNICE 2014. LNCS, vol. 8846, pp. 185–191. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13488-8_17
12. Nichols, K., Jacobson, V.: A Modern AQM is just one piece of the solution to bufferbloat. ACM Queue Netw. **10**(5) (2012)
13. Hoeiland-Joergensen, T., McKenney, O., Taht, D., Gettys, J., Dumazet, E.: The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. No. RFC 8290 (2018)

14. Tazaki, H. et al.: Direct code execution: revisiting library OS architecture for reproducible network experiments. In: Proceedings of the ninth ACM Conference on Emerging Networking Experiments and Technologies, pp. 217–228. ACM (2013)
15. Alwahab, D.A., Laki, S.: A simulation-based survey of active queue management algorithms. In: Proceedings of the 6th International Conference on Communications and Broadband Networking, pp. 71–77. ACM (2018)