Adaptive In-Network Queue Management using Derivatives of Sojourn Time and Buffer Size

Saad Saleh*†, Sunny Shu*, Boris Koldehofe‡†
*Bernoulli Institute, University of Groningen, Netherlands
†CogniGron (Groningen Cognitive Systems and Materials Center), University of Groningen, Netherlands
‡Department of Computer Science and Automation, Technische Universität Ilmenau, Germany
s.saleh@rug.nl, s.shu@student.rug.nl, boris.koldehofe@tu-ilmenau.de

***Abstract*—Active Queue Management (AQM) algorithms are heavily used in packet processors to maintain an optimal queue size and avoid issues like Bufferbloat. Despite the remarkable performance, the traditional AQM algorithms face a major challenge of estimating the accurate queue congestion due to bursty network conditions. The major reason is the use of baseline queue statistics for congestion estimation like delay and sojourn time for Random Early Detection (RED) and Controlled Delay (CoDel), respectively. In this paper, we propose a novel dAQM algorithm that uses advanced traffic statistics like three higher-order derivatives of sojourn time and buffer size along with the baseline sojourn time and buffer size for accurate congestion estimation. dAQM adjusts its drop rate based on the continuously varying congestion to cater to the needs of bursty traffic. We simulated dAQM in ns-3 and analyzed its performance for FTP traffic by variation in traffic load and packet sizes. The results showed that dAQM provides at least 25% and 39.7% reduction in packet loss ratio and flow completion time, respectively, as compared to the traditional AQM algorithms.**

***Index Terms*—Queue Management; Quality of Service; Congestion; Network Management.**

I. Introduction

The network systems use Active Queue Management (AQM) algorithms to maintain an optimal queue size inside packet processors [1] [2]. A small queue size increases the packet losses resulting in poor Quality of Service (QoS) for end users [3]–[5]. On the contrary, a large queue size increases the end-to-end delay resulting in issues like Bufferbloat [6]–[8]. The role of AQM algorithms is to compute the state of queue congestion and drop the packets. Despite the promising performance, the current AQM algorithms face a major challenge of estimating the accurate congestion in bursty network conditions. The major reason is the use of baseline queue statistics like EWMA delay and sojourn time for Random Early Detection (RED) [9] and Controlled Delay (CoDel) [10], respectively. Moreover, the current AQM algorithms lack adaptability and the packet drop mechanisms are independent of the rate of change of congestion in the queues. These shortcomings motivate the development of algorithms that use advanced traffic statistics, like the rate of change of congestion, for computing the Packet Drop Probability (PDP) and adapting the AQM based on the state of congestion.

Our research focuses on the understanding of advanced queue statistics like higher-order derivatives of sojourn time and buffer size for AQM algorithms. In this paper, we develop a novel Derivative-based Active Queue Management (dAQM) algorithm that computes the PDP based on the sojourn time, buffer size, and the three higher-order derivatives of sojourn time and buffer size. The higher-order derivatives provide the rate of increase of congestion in the queues. It aids in the computation of the PDP based on the state of the congestion. It effectively handles the bursty network traffic by continuously measuring the state of network congestion. The first-order derivative of sojourn time provides the rate of increase of packet delay in the queue. It is directly linked to the increase in queue congestion. A higher increase corresponds to a higher PDP and vice versa. Similarly, the second and third-order derivatives provide the rate of increase of first and second-order derivatives, respectively. They capture the periods of bursty network traffic by showing slight variations in packet delay. The traditional AQM algorithms do not consider the buffer capacity during PDP computation and require separate buffer management algorithms [11]–[14]. Our proposed dAQM algorithm incorporates the buffer size and its three higher-order derivatives to drop packets based on the available buffer capacity. The use of buffer size increases the precision of PDP and avoids packet losses due to buffer overflows.

Contributions and Research Findings. We develop a higher-order derivative based AQM algorithm. Our major contributions are three-fold; (1) Development of a novel dAQM algorithm that uses the sojourn time, buffer size, and the three higher-order derivatives of sojourn time and buffer size for PDP computation, (2) Performance analysis of the proposed dAQM algorithm over variation in traffic loads and packet sizes for FTP traffic, (3) Understanding of the architecture and comparison with prior state-of-the-art AQM algorithms. The analysis over FTP traffic showed that dAQM reduces the Flow Completion Time (FCT) by at least 39.7% (from 25.4 s to 15.3 s) as compared to the prior AQM algorithms i.e., RED, PIE, CoDel, FQ-CoDel, and COBALT. dAQM reduces the Packet Loss Ratio (PLR) by at least 25% as compared to the prior AQM algorithms without any comprimise on throughput. We proposed three configurations of dAQM based on the derivative thresholds. The results showed that dAQM provides the most configurability in PLR, delay, and Queue Length (QL) based on the application requirements.

Saad Saleh, Sunny Shu, and Boris Koldehofe. **Adaptive In-Network Queue Management using Derivatives of Sojourn Time and Buffer Size**. In Proceedings of the 37th Network Operations and Management Symposium (NOMS 2024), IEEE, 6 pages, 2024.

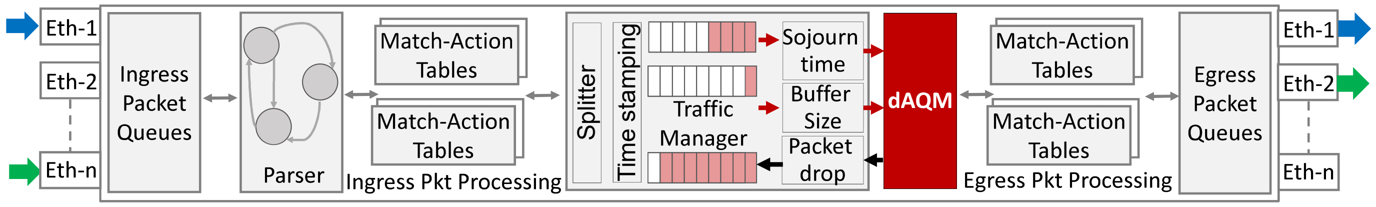

Fig. 1: The switching architecture for the execution of dAQM.

Paper Organization. Sec-II summarizes the related work. The system architecture and dAQM algorithm have been presented in Sec-III. Sec-IV analyzes the performance of dAQM. Finally, Sec-V concludes the paper.

II. Literature Review

Significant research has been conducted on the understanding of optimal traffic features in computing the PDP for AQM algorithms based on the state of congestion.

RED uses the Exponential Weighted Moving Average (EWMA) delay for dropping the packets based on network congestion [9] [15]. An improvement of RED, called FQ-RED, handles different traffic classes separately to avoid unfairness [16]. The major shortcoming of RED is its poor performance for bursty traffic. The programmability of drop probability based on delay is another major challenge [10].

CoDel uses the sojourn time for computing the PDP based on the state of congestion [17]. An improvement called Fair/Flow Queue CoDel (FQ-CoDel), handles flows separately based on traffic classes for fairness [18]. However, the drop mechanism of CoDel is independent of the rate of change of congestion. In fact, the drop rate aggressively increases with time until the threshold limit for sojourn time is reached.

Proportional Integral Controller Enhanced (PIE) uses the PLR for computing the PDP and adjusts its PDP based on the feedback from the queue [19] [20]. BLUE uses both PLR and link idle statistics in its estimation of queue congestion [21]. Recently, multiple features have been used in AQM algorithms to increase the performance. For example, CoDel and BLUE Alternate (COBALT) combines both CoDel and BLUE for accurately estimating the congestion [22]. CAKE [23] combines COBALT with a traffic shaper and flow isolation module to incorporate fair queuing with improved packet handling. Despite promising performance, these techniques rely heavily on raw queue statistics and they cannot adapt based on the continuously varying congestion situation in a bursty network.

Several researches have focused on the use of advanced traffic statistics for the estimation of congestion state. These traffic statistics include traffic load pattern [24], buffer size with delay [25], traffic labels [26], traffic classes (like best effort, real-time, and low latency) [27], and disturbance observer and smith predictor [28]. Some advanced techniques used reinforcement learning-based drop decisions [29], policy-oriented drop probability based on delay and resource utilization [30], and flow statistics-based drop estimation [31]. Many complex approaches have been developed for the tuning of advanced metrics like reinforcement learning-based metric tuning [32] [33], model predictive control theory [34], and reparametrization techniques [35]. A comparison of all the researches shows that the computation of accurate congestion state is an open research problem. The higher-order derivatives of sojourn time and buffer size have not been studied for the computation of PDP in AQM algorithms.

III. Proposed dAQM Algorithm

In this section, we present the system architecture and the details of the higher-order derivative-based dAQM algorithm.

System Architecture. The switching architecture for the execution of dAQM is shown in Fig. 1. The switch builds on the traditional PISA architecture based on Match-Action tables [36]. The incoming packets are stored in the *Ingress Packet Queues*. The *Parser* extracts the packet header fields and passes them to the *Match-Action* tables in the ingress packet processing stages. The *Traffic manager* splits the traffic into multiple queues based on traffic classes. It collects the sojourn time and buffer size for feeding into the dAQM module. dAQM calculates the PDP and feeds it back to the packet drop module. The output of dAQM is processed by a set of egress packet processing stages until ready for processing by the egress packet queues. dAQM can be implemented in four possible methods; (1) Programming dAQM inside the Match-Action tables [37], (2) Using the emerging match-action frameworks that support more expressive line rate network functions [38]–[46], (3) Network Function Virtualization (NFV)-based software implementation, (4) Programming dAQM inside an SDN-based controller.

dAQM Algorithm. The processing stages for the execution of dAQM are shown in Fig. 2. All packets entering the queues are time-stamped to record the sojourn time. At the dequeue, sojourn time is measured for every outgoing packet. Buffer size is measured after fixed intervals to measure queue utilization. Based on the sojourn time ($s(t)$), dAQM calculates the first ($s'(t)$), second ($s''(t)$), and third-order ($s'''(t)$) derivatives of sojourn time at regular intervals (Δt_s) using Eq. 1, Eq. 2 and Eq. 3, respectively. Similarly, the three higher-order derivatives of buffer size are computed at regular intervals to estimate the rate of change of buffer size.

$$s'(t) = \frac{s(t) - s(t-1)}{\Delta t_s} \tag{1}$$

$$s''(t) = \frac{s'(t) - s'(t-1)}{\Delta t_s} \tag{2}$$

TABLE I: The programmed parameters of prior state-of-the-art algorithms and the three configurations of proposed dAQM.

AQM	**Parameters**				**AQM**	**Parameters**		
PIE	Th_{deq} = 20	T_{update}= 15	alpha = 0.125	beta = 1.25	CoDel	Int= 100 ms	Target= 5 ms	
COBALT	P_{drop}= 0	Inc.= 0.08	Dec.= 0.04	BlueTh.= 400	FQ-CoDel	Int= 100 ms	Target= 5 ms	Flows= 1024
RED	Th_{max}= 1000	Th_{min}= 500	QW= 0.002	Target= 5 ms	alpha = 0.01	beta = 0.9		
dAQM$_1$	s= 1000 $D_r[2]$= 0.05 $D_d[2]$= 400 ms	s'= 10 $D_r[4]$= 0.05 $D_d[4]$= 400 ms	s''= 10 $D_r[6]$= 0.05 $D_d[6]$= 400 ms	s'''= 10 $D_r[8]$= 0.05 $D_d[8]$= 400 ms	b= 120 p $D_r[1]$= 0.05 $D_d[1]$= 400 ms	b'= 1 $D_r[3]$= 0.05 $D_d[3]$= 400 ms	b''= 10 $D_r[5]$= 0.05 $D_d[5]$= 400 ms	b'''= 10 $D_r[7]$= 0.05 $D_d[7]$= 400 ms
dAQM$_2$	s= 600 $D_r[2]$= 0.5 $D_d[2]$= 400 ms	s'= 0.1 $D_r[4]$= 0.5 $D_d[4]$= 400 ms	s''= 0.1 $D_r[6]$= 0.5 $D_d[6]$= 400 ms	s'''= 0.1 $D_r[8]$= 0.5 $D_d[8]$= 400 ms	b= 80 p $D_r[1]$= 0.5 $D_d[1]$= 400 ms	b'= 0.1 $D_r[3]$= 0.5 $D_d[3]$= 400 ms	b''=0.1 $D_r[5]$= 0.5 $D_d[5]$= 400 ms	b'''=0.1 $D_r[7]$= 0.5 $D_d[7]$= 400 ms
dAQM$_3$	s=100 $D_r[2]$= 0.98 $D_d[2]$= 400 ms	s'=0.01 $D_r[4]$= 0.98 $D_d[4]$= 400 ms	s''= 0.01 $D_r[6]$= 0.98 $D_d[6]$= 400 ms	s'''= 0.01 $D_r[8]$= 0.98 $D_d[8]$= 400 ms	b= 20 p $D_r[1]$= 0.98 $D_d[1]$= 400 ms	b'= 0.01 $D_r[3]$= 0.98 $D_d[3]$= 400 ms	b''= 0.01 $D_r[5]$= 0.98 $D_d[5]$= 400 ms	b'''= 0.01 $D_r[7]$= 0.98 $D_d[7]$= 400 ms

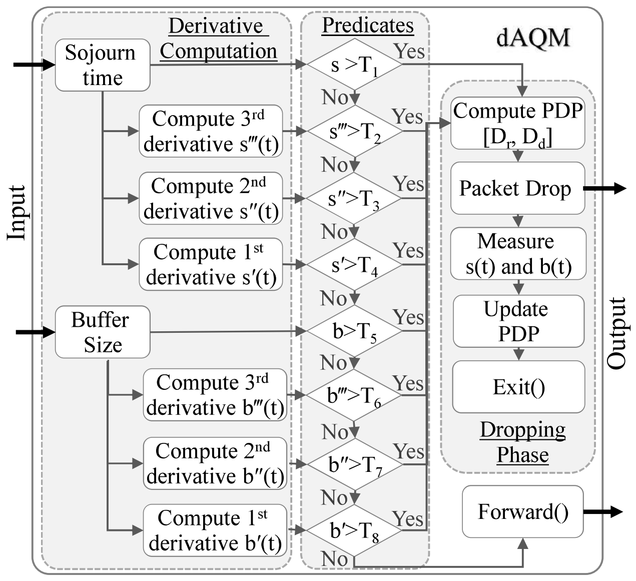

Fig. 2: The flowchart of the proposed dAQM algorithm.

$$s'''(t) = \frac{s''(t) - s''(t-1)}{\Delta t_s} \tag{3}$$

In the next stage, dAQM matches the higher-order derivatives with the programmed predicates which contain the thresholds of derivative limits (T_1, T_2, etc.). Since higher-order derivatives show major fluctuations in delay (unlike the lower-order derivatives), the match process assigns higher priority to higher-order derivatives than lower-order ones. The packets matching the programmed predicates are marked for packet drops. The rest of the packets are forwarded to the output port. The packets are dropped based on the matched derivative thresholds. dAQM contains the programmed drop rates (D_r) and drop durations (D_d) against all matched predicates. After packet drops, the state of congestion is measured again to update the drop rates and durations.

IV. Modeling and Performance Analysis

In this section, we present the modeling and performance analysis of the proposed dAQM algorithm.

TABLE II: Simulation parameters of the network.

Parameter	**Value**	**Parameter**	**Value**
Clients/Servers	100/5	Pkt Size	1400 B
Link Bandwidth	2 Gbps	Link Latency	1 ms
Trans. Rate (pc.)	10/5/1.5/0.128/0.08 Mbps	Buffer Size	2000 p
CBR	ON/OFF=1/0s	VBR	ON/OFF=1/1.5s

A. dAQM ns-3 model

The simulation model for dAQM has been built in ns-3 [47]. The network consists of 100 clients communicating with 5 servers through a common switch. The switch uses the dAQM with a fair-queuing mechanism to handle various flows in separate queues. Constant Bit Rate (CBR) distribution model was used for simulating FTP traffic. In order to understand the performance improvements, the traditional AQM algorithms including RED, PIE, CoDel, FQ-CoDel, and COBALT were also simulated. All AQM algorithms were optimized for maximum throughput. In consistent with prior researches [48]–[53], the performance parameters of the various AQM algorithms and simulation setup are shown in Tab-I and Tab- II, respectively. dAQM was programmed in 3 configurations i.e., dAQM$_1$, dAQM$_2$, and dAQM$_3$. dAQM$_3$ was programmed with the lowest derivative thresholds. It drops packets with slight variations in delays. dAQM$_2$ and dAQM$_1$ had higher derivative thresholds to cater to the traffic that can handle higher delays.

B. Performance analysis

The performance of dAQM was analyzed by variation in traffic loads and packet sizes for FTP traffic.

Increasing the traffic load. Fig. 3 presents the performance of dAQM in comparison to prior AQM algorithms by increasing the traffic load from 50 kbps to 10 Mbps (100%) for FTP traffic. The results show that dAQM$_1$ reduces the PLR by at least 25% as compared to the state-of-the-art AQM algorithms. All AQM algorithms have been programmed for maximum throughput. COBALT shows high throughput but it provides high PLR at heavy traffic loads. High PLR makes its deployment infeasible for traffic classes demanding low PLR. Among other protocols, RED has the highest QL and sojourn time that also increase the end-to-end delay. A comparison

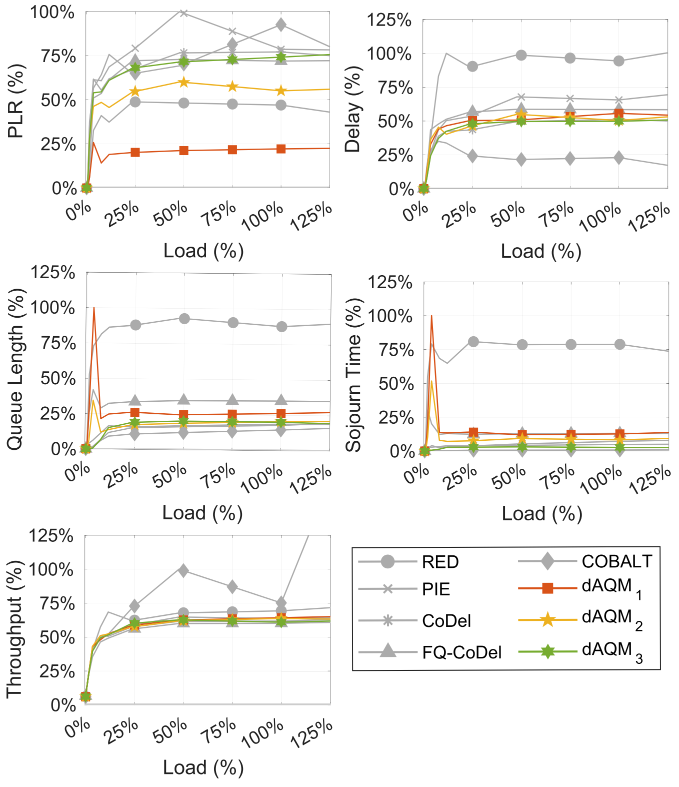

Fig. 3: Performance of dAQM with an increase in load for FTP traffic with CBR distribution model.

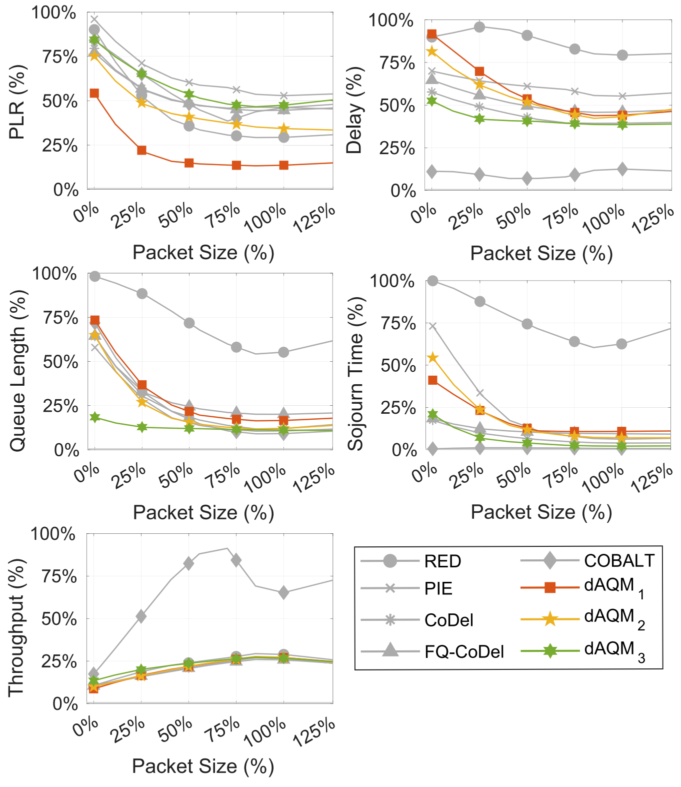

Fig. 4: Performance of dAQM with a decrease in packet size for FTP traffic with CBR distribution model.

shows that all dAQM configurations have minimal QL and sojourn time. Moreover, dAQM provides the capability of multiple configurations that can handle different traffic classes for improving the QoS of end users.

Decreasing the packet sizes. The packet sizes of network traffic vary depending on the applications. In order to understand the impact of variation in packet sizes, the performance of dAQM was analyzed for packet sizes from 50 B to 1500 B. The performance analysis is shown in Fig. 4. The results show that dAQM$_1$ provides the lowest PLR as compared to all other protocols. Since all AQM algorithms have been programmed for maximum throughput, there is very minimal variation in the throughput of various protocols except for COBALT. COBALT provides high throughput for optimal load but its PLR is higher than the dAQM. RED has the highest QL and sojourn time which also increases the end-to-end delay.

Statistical Analysis of dAQM. Tab-III presents the statistical analysis of dAQM with prior AQM algorithms. The results show that dAQM reduced the FCT from 39.7%-49.3% as compared to the state-of-the-art protocols. dAQM$_2$ has an FCT of 15.3 s, while FCTs of traditional protocols vary from 25.4 s to 30.2 s. Moreover, dAQM also has the lowest PLR. Only COBALT provides a lower delay than dAQM but it has twice the PLR and a very large FCT (25.7 s). The statistical analysis shows that dAQM can avoid excessive packet losses and manage the congestion by providing the lowest FCT.

TABLE III: Performance statistics for FTP-based dAQM.

AQM	Delay (ms)	TP (kbps)	QL (pkts)	ST (ms)	PLR %	FCT (s)
RED	2403.5	263.6	677.5	13166.8	0.4	30.2
PIE	1908.4	225.9	94.5	809.9	0.6	29.2
CoDel	1349.2	221.2	103.8	593.5	0.5	27.8
FQ-CoDel	1955	206.7	303.7	2911.1	0.6	25.4
COBALT	668.2	302.3	67.2	78.7	0.6	25.7
dAQM$_1$	**1082.5**	**292.7**	122.9	394.5	**0.3**	**15.4**
dAQM$_2$	**1138.1**	**278.2**	144.5	1191.8	**0.5**	**15.3**
dAQM$_3$	**1341**	**291.5**	124.5	443.2	**0.5**	**15.8**

V. Conclusion and Future Work

In this paper, we proposed a novel dAQM algorithm for queue management in packet processors. dAQM uses the sojourn time, buffer size, and the three higher-order derivatives of sojourn time and buffer size for PDP computation. The performance analysis showed that dAQM reduced the FCT and PLR by up to 49.3% and 50%, respectively, as compared to the traditional AQM algorithms. In the future, we will focus on the understanding of the performance of dAQM by variation in traffic distribution models and traffic classes. Moreover, we will study the programmability of optimal features of dAQM.

Acknowledgment

The authors would like to acknowledge the financial support of the CogniGron research center and the Ubbo Emmius Funds (University of Groningen).

References

[1] R. Adams, "Active Queue Management: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1425–1476, 2013.

[2] S. Jung, J. Kim, and J.-H. Kim, "Intelligent Active Queue Management for Stabilized QoS Guarantees in 5G Mobile Networks," *IEEE Systems Journal*, vol. 15, no. 3, pp. 4293–4302, 2020.

[3] S. Saleh, Z. Shah, and A. Baig, "Improving QoS of IPTV and VoIP over IEEE 802.11n," *Elsevier Computers & Electrical Engineering*, vol. 43, pp. 92–111, 2015.

[4] S. Saleh, Z. Shah, and A. Baig, "Capacity Analysis of Combined IPTV and VoIP over IEEE 802.11n," in *Proceedings of the Annual Conference on Local Computer Networks*. IEEE, 2013, pp. 785–792.

[5] S. Saleh, Z. Shah, and A. Baig, "IPTV Capacity Analysis using DCCP over IEEE 802.11n," in *Proceedings of the Vehicular Technology Conference*. IEEE, 2013, pp. 1–5.

[6] J. Ye, K.-C. Leung, and S. H. Low, "Combating Bufferbloat in Multi-Bottleneck Networks: Theory and Algorithms," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1477–1493, 2021.

[7] J. Gettys, "Bufferbloat: Dark Buffers in the Internet," *IEEE Internet Computing*, vol. 15, no. 3, pp. 96–96, 2011.

[8] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet: Networks Without Effective AQM May Again be Vulnerable to Congestion Collapse," *ACM Queue*, vol. 9, no. 11, pp. 40–54, 2011.

[9] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[10] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," RFC 8289, Jan. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8289

[11] S. Das and R. Sankar, "Broadcom Smart-Buffer Technology in Data Center Switches for Cost-Effective Performance Scaling of Cloud Applications," *Broadcom White Paper*, 2012.

[12] M. Apostolaki, L. Vanbever, and M. Ghobadi, "FAB: Toward Flow-Aware Buffer Sharing on Programmable Switches," in *Proceedings of the Workshop on Buffer Sizing*, 2019, pp. 1–6.

[13] A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds for Shared-Memory Packet Switches," *IEEE/ACM Transactions On Networking*, vol. 6, no. 2, pp. 130–140, 1998.

[14] S. Krishnan, A. K. Choudhury, and F. M. Chiussi, "Dynamic Partitioning: A Mechanism for Shared Memory Management," in *Proceedings of the International Conference on Computer Communications*, vol. 1. IEEE, 1999, pp. 144–152.

[15] S. Floyd, "TCP and Explicit Congestion Notification," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, 1994.

[16] B. Suter, T. Lakshman, D. Stiliadis, and A. K. Choudhury, "Design Considerations for Supporting TCP with Per-Flow Queueing," in *Proceedings of the International Conference on Computer Communications*, vol. 1. IEEE, 1998, pp. 299–306.

[17] K. Nichols and V. Jacobson, "Controlling Queue Delay," *Communications of the ACM*, vol. 55, no. 7, p. 42–50, jul 2012.

[18] T. Høiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm," RFC 8290, Jan. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8290

[19] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," RFC 8033, Feb. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8033

[20] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem," in *International Conference on High Performance Switching and Routing*. IEEE, 2013, pp. 148–155.

[21] W.-c. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The BLUE Active Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 513–528, 2002.

[22] J. Palmei, S. Gupta, P. Imputato, J. Morton, M. P. Tahiliani, S. Avallone, and D. Täht, "Design and Evaluation of COBALT Queue Discipline," in *International Symposium on Local and Metropolitan Area Networks*. IEEE, 2019, pp. 1–6.

[23] T. Høiland-Jørgensen, D. Täht, and J. Morton, "Piece of CAKE: A Comprehensive Queue Management Solution for Home Gateways," in *International Symposium on Local and Metropolitan Area Networks*. IEEE, 2018, pp. 37–42.

[24] A. Adamu, V. Shorgin, S. Melnikov, and Y. Gaidamaka, "Flexible Random Early Detection Algorithm for Queue Management in Routers," in *Proceedings of the International Conference on Distributed Computer and Communication Networks*. Springer, 2020, pp. 196–208.

[25] V. Addanki, M. Apostolaki, M. Ghobadi, S. Schmid, and L. Vanbever, "ABM: Active Buffer Management in Datacenters," in *Proceedings of the SIGCOMM Conference*. ACM, 2022, pp. 36–52.

[26] G. White and D. Rice, "Active Queue Management in DOCSIS 3.x Cable Modems," *Technical report, CableLabs*, 2014.

[27] G. Park, B. Jeon, and G. M. Lee, "QoS Implementation with Triple-Metric-Based Active Queue Management for Military Networks," *Electronics*, vol. 12, no. 1, p. 23, 2022.

[28] R. Hotchi, H. Chibana, T. Iwai, and R. Kubo, "Active Queue Management Supporting TCP Flows using Disturbance Observer and Smith Predictor," *IEEE Access*, vol. 8, pp. 173 401–173 413, 2020.

[29] M. Kim, M. Jaseemuddin, and A. Anpalagan, "Deep Reinforcement Learning based Active Queue Management for IoT Networks," *Journal of Network and Systems Management*, vol. 29, no. 3, p. 34, 2021.

[30] R. Bless, M. Hock, and M. Zitterbart, "Policy-oriented AQM Steering," in *IFIP Networking Conference*. IEEE, 2018, pp. 1–9.

[31] X. Chen, S. L. Feibish, Y. Koral, J. Rexford, O. Rottenstreich, S. A. Monetti, and T.-Y. Wang, "Fine-Grained Queue Measurement in the Data Plane," in *Proceedings of the International Conference on Emerging Networking Experiments And Technologies*. ACM, 2019, pp. 15–29.

[32] D. A. Alwahab, G. Gombos, and S. Laki, "On a Deep Q-Network-based Approach for Active Queue Management," in *Proceedings of the Joint European Conference on Networks and Communications & 6G Summit*. IEEE, 2021, pp. 371–376.

[33] D. A. Alwahab and S. Laki, "A Simulation-based Survey of Active Queue Management Algorithms," in *Proceedings of the International Conference on Communications and Broadband Networking*. ACM, 2018, pp. 71–77.

[34] Q. Xu, G. Ma, K. Ding, and B. Xu, "An Adaptive Active Queue Management based on Model Predictive Control," *IEEE Access*, vol. 8, pp. 174 489–174 494, 2020.

[35] C. Kulatunga, N. Kuhn, G. Fairhurst, and D. Ros, "Tackling Bufferbloat in Capacity-limited Networks," in *Proceedings of the European Conference on Networks and Communications*. IEEE, 2015, pp. 381–385.

[36] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[37] R. Kundel, A. Rizk, J. Blendin, B. Koldehofe, R. Hark, and R. Steinmetz, "P4-CoDel: Experiences on Programmable Data Plane Hardware," in *Proceedings of the International Conference on Communications*. IEEE, 2021, pp. 1–6.

[38] S. Saleh, A. S. Goossens, S. Shu, T. Banerjee, and B. Koldehofe, "Analog In-Network Computing through Memristor-based Match-Compute Processing," in *Proceedings of the International Conference on Computer Communications*. IEEE, 2024, p. 10 pages.

[39] S. Saleh and B. Koldehofe, "The Future is Analog: Energy-Efficient Cognitive Network Functions over Memristor-Based Analog Computations," in *Proceedings of the Workshop on Hot Topics in Networks*. ACM, 2023, p. 254–262.

[40] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "TCA*m*M$^{\text{CogniGron}}$: Energy Efficient Memristor-Based TCAM for Match-Action Processing," in *Proceedings of the International Conference on Rebooting Computing*. IEEE, 2022, pp. 89–99.

[41] S. Saleh and B. Koldehofe, "Memristor-based Network Switching Architecture for Energy Efficient Cognitive Computational Models," in *Proceedings of the International Symposium on Nanoscale Architectures*. ACM, 2023, p. 4 pages.

[42] S. Saleh and B. Koldehofe, "On Memristors for Enabling Energy Efficient and Enhanced Cognitive Network Functions," *IEEE Access*, vol. 10, pp. 129 279–129 312, 2022.

[43] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "Towards Energy Efficient Memristor-based TCAM for Match-Action Processing," in *Proceedings of the International Green and Sustainable Computing Conference*. IEEE, 2022, pp. 1–4.

[44] S. Saleh, A. S. Goossens, T. Banerjee, and B. Koldehofe, "PA*m*M: Memristor-based Probabilistic Associative Memory for Neuromorphic Network Functions," in *Proceedings of the Non-Volatile Memory Technology Symposium*. IEEE, 2023, pp. 1–5, in Press.

[45] C. Li, C. E. Graves, X. Sheng, D. Miller, M. Foltin, G. Pedretti, and J. P. Strachan, "Analog Content-Addressable Memories with Memristors," *Nature Communications*, vol. 11, no. 1, pp. 1–8, 2020.

[46] G. Pedretti, C. E. Graves, T. Van Vaerenbergh, S. Serebryakov, M. Foltin, X. Sheng, R. Mao, C. Li, and J. P. Strachan, "Differentiable Content Addressable Memory with Memristors," *Advanced Electronic Materials*, vol. 8, no. 8, p. 2101198, 2022.

[47] Ns3 network simulator. [Online]. Available: https://www.nsnam.org/

[48] P. L. Dorlan, *An Introduction to Computer Networks*, 2nd ed. Autoedición, 2020.

[49] J. Zheng, C. Wu, T. Lan, C. Tian, and G. Chen, "Revisiting Weighted AIMD-based Congestion Control: A Comprehensive Perspective," in *International Symposium on Quality of Service*. IEEE, 2023, pp. 1–10.

[50] M. Hock, R. Bless, and M. Zitterbart, "Experimental Evaluation of BBR Congestion Control," in *Proceedings of the International Conference on Network Protocols*. IEEE, 2017, pp. 1–10.

[51] X. Du, K. Xu, L. Xu, K. Zheng, M. Shen, B. Wu, and T. Li, "R-AQM: Reverse ACK Active Queue Management in Multitenant Data Centers," *IEEE/ACM Transactions on Networking*, vol. 31, no. 2, pp. 526–541, 2023.

[52] A. Iqbal, U. Javed, S. Saleh, J. Kim, J. S. Alowibdi, and M. U. Ilyas, "Analytical Modeling of End-to-End Delay in OpenFlow Based Networks," *IEEE Access*, vol. 5, pp. 6859–6871, 2016.

[53] U. Javed, A. Iqbal, S. Saleh, S. A. Haider, and M. U. Ilyas, "A Stochastic Model for Transit Latency in OpenFlow SDNs," *Elsevier Computer Networks*, vol. 113, pp. 218–229, 2017.