Abstract: The current problem of packets generation and transformation around the world is router congestion, which then leads to a decline in the network performance in term of queuing delay (D) and packet loss (PL). The existing active queue management (AQM) algorithms do not optimize the network performance because these algorithms use static techniques for detecting and reacting to congestion at the router buffer. In this paper, a weight queue active queue management (WQDAQM) based on dynamic monitoring and reacting is proposed. Queue weight and the thresholds are dynamically adjusted based on the traffic load. WQDAQM controls the queue within the router buffer by stabilizing the queue weight between two thresholds dynamically. The WQDAQM algorithm is simulated and compared with the existing active queue management algorithms. The results reveal that the proposed method demonstrates better performance in terms mean queue length, D, PL, and dropping probability, compared to gentle random early detection (GRED), dynamic GRED, and stabilized dynamic GRED in both heavy or no-congestion cases. In detail, in a heavy congestion status, the proposed algorithm overperformed dynamic GRED (DGRED) by 13.3%, GRED by 19.2%, stabilized dynamic GRED (SDGRED) by 6.7% in term of mean queue length (mql). In terms of D in a heavy congestion status, the proposed algorithm overperformed DGRED by 13.3%, GRED by 19.3%, SDGRED by 6.3%. As for PL, the proposed algorithm overperformed DGRED by 15.5%, SDGRED by 19.8%, GRED by 86.3% in term of PL.

Keywords: congestion algorithms; GRED; SDGRED; implementation; queue weight

1. Introduction

Computer network devices are utilized in smart homes, smart buildings, organizations, universities, companies, and countries. The wide distribution of computer networks is a consequence of many factors, and among these is the emerging of the Internet-of-Things (IoT). The ever-increasing use of computer networks has resulted in increasing computer crimes and cyber-attacks. Moreover, the connected terminals throughout these networks add a heavy load on the network resources and devices [1]. As a result, the amount of data transferred across the network devices, such as computers and routers, greatly increases. To share the network resources [2–6], the transferred data are divided into packets. Every device that generates packets must be temporarily stored at the router buffer before forwarding to the next router or its destination. When the router buffer becomes full, congestion occurs, and it will lead to the case of packet loss (Figure 1) [7–10].
Several algorithms were created to manage the queue and control congestion at the router buffer. Ther efore, a dynamic algorithm must be developed to manage the packet dropping based on the dynamic weight queue.

This study proposes a new algorithm called weight queue dynamic active queue management algorithm (WQDAQM) to address the limitations of the existing ones and improve network performance. The proposed algorithm is integrated with the TCP protocol to maintain the established conversation for data exchange in high performance. The performance measures, delay (D), packet loss (P_L), and throughput (T) have a primary concern because they are used to evaluate whether the network performance is satisfactory or not. The existing algorithms do not provide the best performance according to the mean queue length (mql), D and P_L. This is because the existing algorithms used a static technique for detecting and reacting to congestion at the router buffer. Therefore, a dynamic algorithm must be developed to manage the packet dropping based on the dynamic weight queue.

The rest of the paper is organized as follows, Section 2 presents the literature review, and Section 3 discusses the proposed WQDAQM algorithm, Section 4 explains the simulation mechanism, Section 5 discusses the results of all compared algorithms, and Section 6 provides the conclusion.
2. Related Works

Several algorithms were created to manage the queue and control congestion at the router buffer [3,8,11,23]. The default algorithm for congestion control is the drop tail (DT) algorithm [24]. This algorithm uses the first-in first-out (FIFO) principle, in which the arriving packets are dropped once the router buffer overflows, as illustrated in Figure 2 [11]. The DT does not have thresholds, and when the maximum capacity of the router buffer is reached, the new arriving packets are dropped directly. The main disadvantages of DT include increasing bursty traffic and PL, decreasing delay and T, and causing global synchronization [2,11].

![Diagram of Drop Tail (DT) router buffer](image-url)

Figure 2. Drop tail (DT) router buffer.

To overcome the limitations of the DT algorithm, AQM algorithms are introduced [15,25,26]. Unlike the former, which depends on the capacity of the router, the latter use thresholds and parameters to manage and maintain congestion at an early stage. AQM algorithms use the average queue length (aql) and queue length (q) as signs to notify the algorithm to adjust the dropping of arriving packets before the capacity limit of the router buffer is reached, thereby managing the congestion and maintaining the performance measures at the optimal level as much as possible. All AQM algorithms deliver better results than the DT algorithm in terms of mql, D, and P_L.

Floyd and Jacobson [27] proposed an algorithm called random early detection (RED) to manage the congestion at an early stage and overcome the drawbacks and limitations of the DT algorithm. The main mechanism of this algorithm involves computing the aql of the incoming packets and dropping them before the router buffer becomes full. To calculate the dropping probability (DP), the calculated aql and the router buffer thresholds (min and max thresholds) are maintained to implement packet dropping based on the following scenarios: (1) if aql < min threshold, no packet is dropped; (2) if min threshold ≤ aql < max threshold, packets are dropped following a specific equation; and (3) if aql ≥ max threshold, all packets are dropped, and DP = 1 because the incoming packet will be eliminated [27]. Therefore, the random dropping in the RED algorithm is a good solution for avoiding bursty traffic and global synchronization. In addition, this algorithm achieves better performance than the DT algorithm. Figure 3 depicts the router buffer in the former [25].
As the RED algorithm enhances the performance measures of congestion control and addresses the limitations of the DT algorithm, researchers begin to improve AQM algorithms with reference to the RED mechanism. For instance, Floyd [25] created GRED, which aims to manage full congestion. The objective of GRED is to stabilize the aql among the min, max, and doublemax thresholds (Figure 4). Unlike RED, GRED uses three thresholds that provide the opportunity to control congestion at an early stage by dropping the packets in three stages to prevent the router buffer from the building itself. The mechanism of GRED is executed as follows. First, the router buffer receives the packet when aql reaches the min threshold. If the min threshold < aql ≤ max threshold, GRED drops the packet randomly following a specific equation. Secondly, Dp is increased following a specific equation when aql is between the max and doublemax thresholds. Lastly, Dp becomes 1 when the aql exceeds the doublemax threshold. As a result, every packet will overflow.

![Diagram of RED Algorithm](Symmetry 2020, 12, 2077)

**Figure 3.** Router buffer in the random early detection (RED) algorithm.

However, GRED has a problem regarding the number of thresholds that it contains, which results in a parameterization issue [20]. The parameter should be set to a defined value to obtain a good performance. In addition, the behavior of aql cannot be suddenly changed. When aql relies on the min threshold area, and heavy congestion occurs, aql needs a long time to modify its value, and thus causes the router buffer to overflow. The GRED algorithm is one of the important AQM algorithms. DGRED is an enhanced version of GRED [20], which is developed to improve the congestion investigation [20]. This algorithm is based on Taql; the value of Taql should be between the min and max thresholds following a specific equation to stabilize aql at a specific position and prevent the router buffer from building itself and overflowing [20]. Figure 5 shows the Taql at the router buffer. The DGRED algorithm
achieves a better performance than other AQM algorithms and stabilizes the $Taql$ at a specific level to prevent the buffer from overflowing rapidly.

Similar to GRED, DGRED suffers from a parameterization problem because of its several thresholds and $Taql$. Therefore, the SDGRED algorithm is developed to improve and manage congestion [20]. This algorithm uses dynamic thresholds based on the value of $aql$ to detect congestion at an early stage. Figure 6 presents the SDGRED router buffer.

![Figure 5. Dynamic GRED (DGRED) buffer.](image)

Unlike in DGRED, all thresholds in the SDGRED algorithm change following the status of $aql$ at the router buffer [16]. Moreover, SDGRED eliminates $Taql$ to decrease the parameterization in the router buffer. This algorithm reduces congestion in either congested or non-congested cases. The SDGRED algorithm improves the network performance in terms of $mql$, $D$, $P_L$ and decreases the dependency on a specific parameter. However, this algorithm requires further improvement to reduce the bursty traffic when the number of arriving packets increases, especially during high traffic. This study investigates the congestion detection mechanism, advantages, and disadvantages of the DT, RED, GRED, DGRED, and SDGRED algorithms. These algorithms aim to control the congestion before the router buffers become occupied. In general, the drawbacks of the algorithms that use a non-adaptive mechanism, such as DT, RED, and GRED algorithms, include static parameters, fixed threshold values, and fixed dropping calculation. The adaptive mechanism, such as DGRED and SDGRED algorithms, are just
for thresholds. Table 1 compares the existing AQM algorithms and highlights the gaps that will be covered in this paper.

**Table 1. Summary of the existing active queue management (AQM) algorithms.**

| Algorithms | Metric(s) | Adaptive/ Non Adaptive | Dropping Mechanism | Advantages | Disadvantages |
|------------|-----------|------------------------|-------------------|------------|---------------|
| Drop tail  | Queue length (q) | Non Adaptive | Drop all packets only after the buffer gets overloaded | Simple and low computation overhead requirements | High PL, high delay and low throughput, and leads to global synchronization |
| RED        | (Qavg)    | Non Adaptive | Drop packets stochastically | Eliminate global synchronization problems | Sensitive to sudden congestion |
| GRED       | (Qavg)    | adaptive    | Drop packets stochastically | More robust to sudden congestion | Several threshold values, parameterization and overflow |
| DGRED      | Qavg      | adaptive    | Drop packets stochastically | Stabilize Qavg partially | Several threshold values, parameterization and overflow |
| SDGRED     | Qavg      | adaptive    | Uses dynamic thresholds based on the value of aql to detect congestion at an early stage | Stabilize Qavg partially | Stabilize Qavg partially |

Managing the congestion in dynamic router buffers is urgent. The algorithms mentioned previously fail to address the issue regarding the bursty traffic congestion caused by the numbers of packets in the router buffer. This problem leads to an increase in average time and PL. Therefore, the WQDAQM algorithm is proposed to improve the PL and D of dynamic router buffers. This algorithm aims to identify and avoid congestion in the buffer by stabilizing aql at a level that is less than the router buffer capacity by using a dynamic q_{aw}.

**3. Proposed Algorithm**

The WQDAQM algorithm adopts several processes used in the SDGRED mechanism and utilizes a dynamic q_{aw} to timely manage the congestion. Unlike the latter, which uses a static value that is intended for its buffer technique, the aim of the proposed algorithm is to control the dropping of the packets by dynamically stabilizing the q_{aw} between the min and max thresholds. Another aim of the proposed algorithm is to enhance the network performance in terms of mql, PL, and D compared with other AQM algorithms when full congestion occurs.

As illustrated in Figure 7, the router buffer in the proposed algorithm is separated into three parts, namely, min, max, and doublemax thresholds. Congestion is measured based on aql.

![Figure 7. Weight queue active queue management (WQDAQM) router buffer.](image-url)
occurs in the router buffer. However, when the number of incoming packets increases, the arrival quantity ranges between the min threshold and max threshold, and the buffer drops the packets randomly in accordance with Equation (1):

\[
D_p = \frac{D_{\text{max}} \times (\text{aql} - \text{min threshold})}{\text{max threshold} - \text{min threshold}} (1 - C \times \text{D}_{\text{init}})
\]

(1)

When the traffic increases and the number of packets exceeds the max threshold, \(D_p\) increases following Equation (2) to prevent the router buffer from the building itself. When the number of:

\[
D_p = D_{\text{max}} + \frac{[1-D_{\text{max}}] \times (\text{aql} - \text{max threshold})}{\text{max threshold}} (1 - C \times \text{D}_{\text{init}})
\]

(2)

arriving packets exceeds the doublemax threshold, \(D_p\) is set to 1, which means that all incoming packets will be dropped and the router buffer is occupied.

WQDAQM also updates the queue weight parameter at the router buffer to control the congestion. WQDAQM uses the SDGRED algorithm’s policy in dropping packets with probability when the \(\text{aql}\) is either between the min threshold and max threshold or between the max threshold and doublemax threshold.

**Phases of the WQDAQM Algorithm**

The WQDAQM algorithm is implemented in seven stages, as illustrated in Figure 8. In stage one, the parameters are initialized as the packets start to arrive at the router buffer. The thresholds are set to 3, 9, and 18, which are the same as the threshold values of the GRED, DGRED and SDGRED algorithms [25]. The value of aql is set to zero, which indicates that no packet has reached the router buffer yet. \(D_{\text{max}}\), which represents the highest value of the primary packet that will be dropped, is set to 0.1 [25]. The value of \(qw\) is dynamic and dependent on the station of \(aql\) at the router buffer.

In stage two, the router buffer of the proposed algorithm starts receiving the packets to store and forward them to their destinations. The Bernoulli model is used to manage the receiving packets, which is suitable for the buffer that has a static length slot [28,29]. In the third stage, \(qw\) prominently drops the packets because \(D_p\) depends on the values of aql. When aql is extremely small and q, which represents the real packet in the router buffer, is excessively high, the aql needs a long time to update its value. Therefore, \(qw\) must be dynamic to balance aql and q to drop the packet and prevent sudden congestion or bursty traffic. The detailed process flow is displayed in Figure 9.
Accordingly, in the proposed algorithm, when the value of aql is less than the min threshold, qw is set to 0.002, which is the same value as that set in the GRED algorithm [25]. No dropping occurs in this situation to allow more packets to enter the router buffer. When the value of aql is between the min and max thresholds, qw is determined as

\[ qw = qw + 0.001 \] (3)

The aql is updated, and packet dropping starts randomly according to Equation (1) to prevent the router buffer from building itself. In this scenario, aql exceeds the min threshold, and the dynamic qw allows the value of aql to be updated. In this situation, more packets are dropping to prevent the router buffer arriving at max threshold to maintain the aql as small as possible.

When the value of aql is between the max and doublemax thresholds, qw is calculated as

\[ qw = qw + 0.002 \] (4)

As the value of aql increases, DP increases following Equation (2) that used in the GRED, DGRED, and SDGRED algorithms. Moreover, the dropping probability is increased quickly to control the congestion at the router buffer by dropping more packets before the packets arrive at the doublemax threshold. As a result, every arriving packet will be overflow, and heavy congestion will occur.

Figure 10 shows the pseudocode for implementing the fourth stage. This code depends on the case of the buffer. If the router buffer is unfilled, aql is calculated using the equation in line 4. Meanwhile, if the packets arrive at the router buffer, aql is determined using the equation in line six.

1. start
2. calculate the value of aql
3. if aql is empty
4. aql = (1 – queue weight) * n
5. else
6. aql = (1 – queue weight) + queue weight * q instantaneous
7. the aql is available
8. end

Figure 10. Pseudocode for the calculation of aql.
In stage five, a dynamic threshold is utilized with the router buffer with each packet arrival. The value of \( aql \) in the proposed algorithm is compared with the three thresholds (min, max and doublemax threshold). The proposed algorithm updates both the max threshold and doublemax threshold, according to Equations (5)–(7). The updating processes stabilize the value of \( aql \) near min threshold to prevent the router buffer growing rapidly and overflowing, which leads to little packets being dropped:

\[
\text{max\_threshold} = \text{max\_threshold} - 2 \times \text{min\_threshold} \quad (5)
\]

\[
\text{Doublemax\_threshold} = \text{Doublemax\_threshold} - 2 \times \text{min\_threshold} \quad (6)
\]

\[
\text{max\_threshold} = \text{max\_threshold} + \text{Min\_threshold} \quad (7)
\]

As such, if \( aql \) is less than the min threshold, no update occurs for the value of max threshold and doublemax threshold. Also, if the value of \( aql \) is still less than min threshold \( \times 2 \) and the max threshold position arrives at the value \( \geq \text{min\_threshold} \times 3 \), the proposed algorithm starts to update the position of max threshold and doublemax threshold using the following equations.

The proposed algorithm controls the congestion by examining the value of \( aql \). As the number of packets is below the min threshold, an update occurs, no packets are dropped in the router buffer, and the dropping probability is represented by zero. In case, more packets arrive at the router buffer and the value of \( aql \) increased and reached the min threshold, the WQDAQM algorithm starts dropping the packets based in Equation (1), to allow the router buffer accommodating the arrival packets and prevent the router overflowing. When the value of \( aql \) increases to the value that exceeds the max threshold, the proposed algorithm starts dropping the packets according to Equation (2). Lastly, when the value of \( aql \) increases, suddenly the doublemax threshold is reached, the dropping probability is set to one, which means no arriving packets are accommodated as the router buffer is fully occupied and overflowing.

The proposed algorithm is simulated in the seven stages. The FIFO technique is adopted, and only one router is present. As previously mentioned, the Bernoulli model is used to represent the arrival of the packets [30]. The Bernoulli phase is better when the router buffer is divided into static time length time or slot [30].

The proposed and existing algorithms are simulated through a discrete-time methodology. The packets depart from the router buffer with geometrically distributed service times [17]. The details of the simulation are discussed in the next section.

The simulation results are collected in the final stage, and the outputs of the compared algorithm are calculated. The results of the proposed algorithm are compared with those of the existing ones to determine which algorithm achieves the best performance.

### 4. Simulation

The proposed and existing algorithms are simulated, and the results are evaluated using \( mql \), \( P_L \), \( D \), \( T \), and \( D_P \). The simulation is implemented in Java using NetBeans Integrated Development Environment (IDE) 8.1 under 64bits Windows 10, in Intel Core i3 2.10GHz processor and 6 GB RAM. A single router was modulated using first-in first-out (FIFO) with a variable arrival rate and departure rate. The system architecture is illustrated in Figure 11.

Figure 12 shows the simulation cycle for the algorithms. The initialized parameters of the compared algorithms are summarized in Table 2.
The packet arrival and departure rates are set. The $q$ value of the router buffer is an indication that the router buffer prevents congestion and the increase in $q$. The maximum $mql$ values of the compared algorithms are calculated. The results of the proposed algorithm are compared with those of the existing ones to determine which algorithm achieves the best performance.

The thresholds in the GRED algorithm are set to 3, 9, and 18, whereas those in DGRED, SDGRED, and WQDAQM are assigned dynamic values. The buffer capacity is set to 20 packets to detect congestion at small buffer sizes. The processes in the router buffer are divided into time units called slots, and the warm period occurs when 2,000,000 slots are generated [31]. After such a period, the packet departure is evaluated, that is if the departure is terminated or not. If the generated number is less than beta (0.5), the number of departing packets is increased by one; otherwise, no packet will depart. Subsequently, the number of arriving packets is checked. If this quantity is less than the number of departing packets, no congestion is present, and the slot number is increased by one unit; otherwise, no packet will depart. The slot number and the buffer capacity are assigned with dynamic and static values, respectively. The slot number is calculated as given in Equation (8):

$$\text{Slot} = \begin{cases} \text{Slot} + 1, & \text{if } \text{Arrival} < \text{Departure} \\ \text{No packet will depart} \\ \text{else} \\ \text{no packet will be departure} \\ \text{check the arrival packet} \\ \text{if the random number } \leq \beta \\ \text{Increase the departure by 1} \\ \text{else} \\ \text{no packet will arrive} \\ \text{still increase the arrival packet until reach to slot 799,999,} \\ \text{start calculate the measure performance } mql, D_P, P_L, \text{ and } D_F \\ \text{check the simulation runs } \leq 10 \\ \text{go to step 1} \\ \text{else} \\ \text{finish} \\ \end{cases}$$

1. Parameter initialization
2. Check the packet departure
3. Generate random number
4. If the random number $\leq \beta$
5. Increase the departure by 1
6. else
7. no packet will be departure
8. check the arrival packet
9. if the random number $\leq \beta$
10. Increase the departure by 1
11. else
12. no packet will arrive
13. still increase the arrival packet until reach to slot 799,999,
14. start calculate the measure performance $mql, D_P, P_L,$ and $D_F$
15. check the simulation runs $\leq 10$
16. go to step 1
17. else
18. finish

**Figure 11.** System architecture.

**Figure 12.** Simulation cycle.

**Table 2.** Parameter initialization.

| Parameters         | GRED Algorithm | DGRED Algorithm | SDGRED Algorithm | WQDAQM Algorithm |
|--------------------|----------------|-----------------|------------------|------------------|
| Alpha              | 0.33–0.93      | 0.33–0.93       | 0.33–0.93        | 0.33–0.93        |
| $q_w$              | 0.002          | 0.002           | 0.002            | Dynamic          |
| Beta               | 0.5            | 0.5             | 0.5              | 0.5              |
| Maximum probability| 0.1            | 0.1             | 0.1              | 0.1              |
| Slot               | 2 million      | 2 million       | 2 million        | 2 million        |
| Buffer size        | 20             | 20              | 20               | 20               |
| Minimum T          | 3              | ...             | ...              | ...              |
| Maximum T          | 9              | ...             | ...              | ...              |
| Double MT          | 18             | ...             | ...              | ...              |
The packet arrival and departure rates are set. The \( q_\text{ur} \) of the proposed and existing algorithms are assigned with dynamic and static values, respectively. The slot number and the buffer capacity are the same as those in the GRED algorithm [25]. The probability of the packet arrival is set within 0.18–0.93 to guarantee that either congestion or non-congestion will occur.

The thresholds in the GRED algorithm are set to 3, 9, and 18, whereas those in DGRED, SDGRED, and WQDAQM are assigned dynamic values. \( D_\text{max} \) is set to 0.1 following the corresponding values in the DT and GRED algorithms [25]. The buffer capacity is set to 20 packets to detect congestion at small buffer sizes. The processes in the router buffer are divided into time units called slots, and the packets may arrive or depart during a specific slot. The warm period occurs when 2,000,000 slots are generated [31]. After such a period, the packet departure is evaluated, that is if the departure is terminated or not. If the generated number is less than beta (0.5), the number of departing packets is increased by one unit; otherwise, no packet will depart. Subsequently, the number of arriving packets is checked. If this quantity is less than the number of departing packets, no congestion is present, and the former is increased by one unit. Moreover, no packet is allowed to enter the router buffer. The router buffer calculates the performance measures of all algorithms if the slot number reaches 7,999,999. The steps are repeated 10 times to obtain accurate results.

5. Evaluation Results

The results of the proposed WQDAQM algorithm are compared with those of GRED, DGRED, and SDGRED algorithms, which were discussed in Section 2. All experiments are run ten times to eliminate potential bias, and yield confidence intervals of the performance run. The algorithms were assessed based on their \( mql, T, D, P_L, \) and \( D_P \) values. To facilitate easy comparison between the proposed and existing algorithms, the improvement ratio is calculated as given in Equation (8):

\[
\text{IR} = \frac{\text{ReferencePerformance} - \text{ComparedPerformance}}{\text{ReferencePerformance}} \times 100\%
\]  

(8)

5.1. \( mql \)

The results of the GRED, DGRED, SDGRED, and WQDAQM algorithms are evaluated using various packet arrival probabilities (0.18, 0.33, 0.48, 0.67, 0.78, and 0.93) to ensure that either congestion or non-congestion will occur at the router buffer. Figure 13 shows the \( mql \) values of the compared algorithms. The value of \( mql \) is an important performance measure of router buffers. The small value of \( mql \) is an indication that the router buffer prevents congestion and the increase in q.

![Figure 13. Mean queue length (mql) vs. alpha.](image-url)
As depicted in Figure 13, the \( mql \) values of the proposed and existing algorithms are the same at alpha’s value of 0.18 and 0.33. When the probability of packet arrival is less than that of packet departure (\( a < B \)), each packet that arrives will depart. When the value of alpha is 0.48, the WQDAQM, DGRED, and SDGRED algorithms demonstrate better performances than the GRED algorithm, and bright congestion is present. When \( mql \) increases and the probability of packet arrival exceeds that of packet departure, the proposed algorithm overperforms the other algorithms. The overflow obtained by the WQDAQM algorithm during heavy congestion is less than those obtained by the existing ones. Consequently, the proposed algorithm maintains the \( mql \) at a level that is lower than the doublemax threshold to allow the arrival of many packets and prevent the rapid increase in the size of the router buffer. In detail, in a heavy congestion status at a probability of packet arrival of 0.93, the proposed algorithm overperformed DGRED by 13.3%, GRED by 19.2%, SDGRED by 6.7% in terms of \( mql \).

5.2. \( D \)

The delay performance results of the compared algorithms are illustrated in Figure 14.

![Figure 14. D vs. alpha.](image)

The proposed and existing algorithms have identical outputs according to D, when the probability of packet arrival is 0.18 or 0.33 as the number of arriving packets is less than that of departing ones. The proposed algorithm exhibits better performance than the existing ones when heavy congestion is present in the router buffer because the former has fewer departing packets than the other algorithms. The results further indicate that \( mql \) plays a key role in the delay equation \( D = \frac{mql}{T} \). This equation implies that \( D \) is small when \( mql \) is small. In detail, in a heavy congestion status at a packet arrival of 0.93, the proposed algorithm overperformed DGRED by 13.3%, GRED by 19.3%, SDGRED by 6.3% in term of D.

5.3. \( T \)

The plot of \( T \) vs. the probability of packet arrival is illustrated in Figure 15.

The proposed and existing algorithms obtained identical results when the alpha was less than beta. In addition, the light congestion occurs in the router buffer at packet arrival probabilities of 0.18, 0.33, and 0.48. When the value of alpha approaches that of beta and heavy congestion is present, the compared algorithms stabilize at the value of the latter (0.5). In detail, in a heavy congestion status at a packet arrival of 0.93, the proposed algorithm overperformed DGRED by 13.3%, GRED by 19.3%, SDGRED by 6.3% in term of D.
The number of packets dropped by the proposed and the existing algorithms are analyzed in this section. The \( P_L \) and \( D_P \) values of the GRED, DGRED, SDGRED, and WQDAQM algorithms are presented in Figures 16 and 17, respectively.

Figure 15 shows that the proposed algorithm achieves the lowest \( P_L \) when heavy congestion is present in the router buffer (packet arrival and packet departure). This phenomenon occurs because the router buffer that is managed by this algorithm takes a long time to overflow, and the number of arriving packets reaches the doublemax threshold. When alpha is less than beta, all algorithms produced similar results. In details, with low traffic load, no \( P_L \) occurs as a result of using any of the algorithms. As slight congestion occurs at a packet arrival of 0.48, the DGRED overperformed the proposed algorithm by 2\% only. The proposed algorithm and SDGRED perform equally. At the same time, the proposed algorithm overperformed GRED by 39.7\%. In a heavy congestion status at a packet arrival of 0.93, the proposed algorithm overperformed DGRED by 15.5\%, SDGRED by 19.8\%, GRED by 86.3\% in term of \( P_L \).

The relationship between \( D_P \) and alpha is illustrated in Figure 16. The result reveals that the proposed WQDAQM algorithm drops many packets in the router buffer when the beta is less than the alpha and when the number of arriving packets is less than that of the departing ones.
6. Conclusions

In this paper, the WQDAQM algorithm was proposed by enhancing the SDGRED algorithm with dynamic dropping rates. This algorithm aims to maintain the aql between two dynamically predetermined thresholds to prevent the buffer from exceeding the latter and overflowing. The WQDAQM algorithm controls the congestion at the router buffer and prevents the loss of the packets. The proposed algorithm is compared with GRED, DGRED, and SDGRED algorithms in terms of D, aql, Pt, T, and Dp to determine which among them achieves the best results based on the value of alpha. The WQDAQM algorithm resulted in better aql, D, and Pt compared to SDGRED, DGRED and GRED algorithms when heavy congestion exists in the buffer. However, when the alpha is either less than or greater than beta, all algorithms produce the same throughputs. Moreover, the proposed algorithm drops fewer packets at the router buffers compared to the SDGRED, DGRED and GRED algorithms. Accordingly, WQDAQM enhances and improves the congestion consequences at the router buffer. The limitation appears as the proposed algorithm with slight congestion as the DGRED overperformed the proposed algorithm by 2% only. Accordingly, in future work, the focus will be on enhancing the performance of the proposed algorithm in a slightly congested network.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Al-Zubi, M.; Abu Shareha, A.A. Efficient signcryption scheme based on El-Gamal and Schnorr. Multimed. Tools Appl. 2018, 78, 11091–11104. [CrossRef]
2. Welzl, M. Network Congestion Control: Managing Internet Traffic, 1st ed.; Wiley: Hoboken, NJ, USA, 2005; pp. 10–12.
3. Baklizi, M.; Ababneh, J.M.; Abdallah, N. Performance investigations of fired and aged active queue management methods. In Proceedings of the Academicsera 13th International Conference, Istanbul, Turkey, 23–24 February 2018; p. 14.
4. Khatari, M.; Samara, G. Congestion control approach based on effective random early detection and fuzzy logic. MAGNT 2015, 3, 180–193.
5. Sun, D.; Zhao, K.; Fang, Y.; Cui, J. Dynamic Traffic Scheduling and Congestion Control across Data Centers Based on SDN. Futur. Internet 2018, 10, 64. [CrossRef]
6. Abu Shareha, A.A.; Mandava, R.; Khan, L.; Ramachandram, D. Multimodal concept fusion using semantic closeness for image concept disambiguation. Multimed. Tools Appl. 2011, 61, 69–86. [CrossRef]
7. Baklizi, M.; Ababneh, J. A Survey in Active Queue Management Methods According to Performance Measures. *Int. J. Comput. Trends Technol.* 2016, 38, 145–152. [CrossRef]

8. Abu-Shareha, A.A. Enhanced Random Early Detection using Responsive Congestion Indicators. *Int. J. Adv. Comput. Sci. Appl.* 2019, 10, 358–367. [CrossRef]

9. Pedrycz, W.; Vasilakos, A. *Computational Intelligence in Telecommunications Networks*; CRC Press: Boca Raton, FL, USA, 2000; p. 528.

10. Abdeljaber, H.; Ababneh, J.; Daoud, A.; Baklizi, M. Performance Analysis of the Proposed Adaptive Gentle Random Early Detection Method under NonCongestion and Congestion Situations. In *International Conference on Digital Enterprise and Information Systems (DEIS)*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 592–603.

11. Abualhaj, M.M.; Abu Shareha, A.A.; Al-Tahrawi, M.M. FLRED: An efficient fuzzy logic based network congestion control method. *Neural Comput. Appl.* 2016, 30, 925–935. [CrossRef]

12. Baklizi, M.; Abdel-Jaber, H.; Abu-Shareha, A.A.; Abualhaj, M.M.; Ramadass, S. Fuzzy Logic Controller of Gentle Random Early Detection Based on Average Queue Length and Delay Rate. *Int. J. Fuzzy Syst.* 2014, 16, 9–19.

13. Ababneh, J.; Abdel-Jaber, H.; Thabtah, F.; Hadi, W.; Badarneh, E. Derivation of Three Queue Nodes Discrete-Time Analytical Model Based on DRED Algorithm. In Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations; Institute of Electrical and Electronics Engineers (IEEE), Las Vegas, NV, USA, 12–14 April 2010; pp. 885–890.

14. Chintam, J.R.; Daniel, M. Real-Power Rescheduling of Generators for Congestion Management Using a Novel Satin Bowerbird Optimization Algorithm. *Energies* 2018, 11, 183. [CrossRef]

15. Baklizi, M.; Ababneh, J. Performance Evaluation of the Proposed Enhanced Adaptive Gentle Random Early Detection Algorithm in Congestion Situations. *Int. J. Curr. Eng. Technol.* 2016, 6, 1658–1664.

16. Baklizi, M. Stabilizing Average Queue Length in Active Queue Management Method. *Int. J. Adv. Comput. Sci. Appl.* 2019, 10. [CrossRef]

17. Baklizi, M.; Ababneh, J.; Abualhaj, M.M.; Abdullah, N.; Abdulllah, R. Markov-Modulated Bernoulli Dynamic Gentle, Random Early Detection. *J. Theor. Appl. Inf. Technol.* 2018, 9, 6688–6698.

18. Liu, J.; Yang, O.W.W. Using Fuzzy Logic Control to Provide Intelligent Traffic Management Service for High-Speed Networks. In *IEEE Transactions on Network and Service Management*; June 2013; Volume 10, pp. 148–161. Available online: https://ieeexplore.ieee.org/document/6514996 (accessed on 20 June 2020).

19. Baklizi, M. FLACC: Fuzzy Logic Approach for Congestion Control. *Int. J. Adv. Comput. Sci. Appl.* 2019, 10, 43–50. [CrossRef]

20. Baklizi, M.; Abdeljaber, H.; Abualhaj, M.M.; Abdullah, N.; Ramadass, S.; Almomani, A. Dynamic Stochastic Early Discovery: A New Congestion Control Technique to Improve Networks Performance. *Int. J. Innov. Comput. Inf. Control* 2015, 9, 1113–1126.

21. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A Survey on Software-Defined Networking. *IEEE Commun. Surv. Tutor.* 2015, 17, 27–51. [CrossRef]

22. Abu-Shareha, A.A. Controlling Delay at the Router Buffer Using Modified Random Early Detection. *Int. J. Comput. Netw. Commun.* 2019, 11, 63–75. [CrossRef]

23. Alsaidah, A.; Zalisham, M.; Fadli, M.; Abdeljaber, H. Markov-Modulated Bernoulli-Based PErformance Analysis for Gentle Blue and Blue Algorithms under Bursty and Correlated traffic. *J. Comput. Sci.* 2016, 12, 289–299. [CrossRef]

24. Brandauer, C.; Iannaccone, G.; Diot, C.; Fdida, S. Comparison of Tail Drop and Active Queue Management Performance for Bulk-Data and Web-Like Internet Traffic. In Proceedings of the Sixth IEEE Symposium on Computers and Communications, Hammamat, Tunisia, 5 July 2001.

25. Floyd, S. Recommendations on Using the Gentle Variant of RED. Available online: http://www.aciri.org/floyd/red/gentle.html (accessed on 20 June 2020).

26. Loukas, R.; Kohler, S.; Andreas, P.; Phuoc, T.G. Fuzzy RED: Congestion control for TCP/IP Diff-Serv. In Proceedings of the 10th Mediterranean Electrotechnical Conference, Information Technology and Electrotechnology for the Mediterranean Countries, Lemesos, Cyprus, 29–31 May 2020.

27. Floyd, S.; Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* 1993, 1, 397–413. [CrossRef]

28. Ng, C.H.; Yuan, L.; Fu, L.; Zhang, L. Research note: Methodology for traffic modeling using two-state Markov-modulated Bernoulli process. *Comput. Commun.* 1999, 22, 1266–1273. [CrossRef]
29. Woodward, M.E. *Communication and Computer Networks: Modelling with Discrete-Time Queues*; Wiley-IEEE Computer Society Press: Washington, DC, USA, 1993.

30. Abdel-jaber, H.; Thabtah, F.; Woodward, M. Traffic Management for the Gentle Random Early Detection Using Discrete-Time Queueing. In Proceedings of the International Business Information Management Conference (9th IBIMA), Marrakech, Morocco, 4–6 January 2008; pp. 289–298.

31. Abdeljaber, H.; Thabtah, F.; Woodward, M.; Ababneh, J.; Bazar, H. Random Early Dynamic Detection Approach for Congestion Control Baltic. *J. Mod. Comput.* 2014, 2, 16–31.

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).