electronics

Article

Queue-Buffer Optimization Based on Aggressive Random Early Detection in Massive NB-IoT MANET for 5G Applications

Syed Talib Abbas Jafri [1,*], Irfan Ahmed [2] and Sundus Ali [3]

[1] Department of Electronic Engineering, NED University of Engineering and Technology, Karachi 75270, Pakistan
[2] Department of Physics, NED University of Engineering and Technology, Karachi 75270, Pakistan
[3] Department of Telecommunications Engineering, NED University of Engineering and Technology, Karachi 75270, Pakistan
* Correspondence: talibsyed@cloud.neduet.edu.pk; Tel.: +92-333-2645434

Abstract: Elements in massive narrowband Internet of Things (NB-IoT) for 5G networks suffer severely from packet drops due to queue overflow. Active Queue Management (AQM) techniques help in maintaining queue length by dropping packets early, based on certain defined parameters. In this paper, we have proposed an AQM technique, called Aggressive Random Early Detection (AgRED) which, in comparison to previously used Random Early Detection (RED) and exponential RED technique, improves the overall end-to-end delay, throughput, and packet delivery ratio of the massive NB-IoT 5G network while using UDP. This improvement has been achieved due to a sigmoid function used by the AgRED technique, which aggressively and randomly drops the incoming packets preventing them from filling the queue. Because of the incorporation of the AgRED technique, the queue at different nodes will remain available throughout the operation of the network and the probability of delivering the packets will increase. We have analyzed and compared the performance of our proposed AgRED technique and have found that the performance gain for the proposed technique is higher than other techniques (RED and exponential RED) and passive queue management techniques (drop-tail and drop-head). The improvement in results is most significant in congested network deployment scenarios and provides improvements in massive Machine Type Communication, while also supporting ultra-low latency and reliable communication for 5G applications.

Keywords: Internet of Things; 5G; mobile ad hoc networks; narrowband

Citation: Jafri, S.T.A.; Ahmed, I.; Ali, S. Queue-Buffer Optimization Based on Aggressive Random Early Detection in Massive NB-IoT MANET for 5G Applications. *Electronics* **2022**, *11*, 2955. https://doi.org/10.3390/electronics11182955

Academic Editors: Ikram Rehman, Sara Paiva, Nagham Saeed and Waqar Asif

Received: 4 August 2022
Accepted: 15 September 2022
Published: 18 September 2022

1. Introduction

The idea of the Internet of Things (IoT) is to enable all devices to connect with or without any forwarding central station or gateway, which serves as the solution for connecting physical things with the IP-based network in a heterogeneous manner. An IoT network consists of anywhere from several to thousands of nodes communicating with each other over a wireless channel and sometimes partially connected to the internet or cloud through a central station or gateway [1]. The IoT network architecture resembles wireless sensor networks (WSN), but unlike WSNs, they are built for various applications. These applications include device-to-device communication (D2D), WSN, cognitive radio (CR), etc. [2,3], transforming into a giant network that is economically cheap and user-friendly. The portable nature of IoT-enabled devices provides greater mobility to users and a less complex, cost-effective deployment. Due to the variety of supported devices and maintaining heterogeneous connectivity between all nodes, IoT is now replacing old-fashioned wired and wireless networks in the fields of industrial automation, health care, transportation, environment monitoring, sports, and smart homes [4,5]. The similar nature of IoT and Mobile Ad-hoc Network (MANET) allows several MANET protocols to be implemented with some modifications and termed as IoT-based MANET, as this also provides a smooth transfer of technology from MANET, WSN, and similar technologies to IoT.

Some applications of MANET include waste management systems [6], the use of MANET and VANET in smart cities [7], healthcare [8], etc. Smart applications in 5G New Radio (NR) are classified into three major categories, including enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communication (URLLC), and massive Machine Type Communication (mMTC), as mentioned in Figure 1 [9]. The targeted technology of massive NB-IoT or IEEE 802.15.4, though, falls under mMTC but can also be used in URLLC as the communication governed by the technology tends to provide reliable communication and low latency between two nodes connected in a complex network.

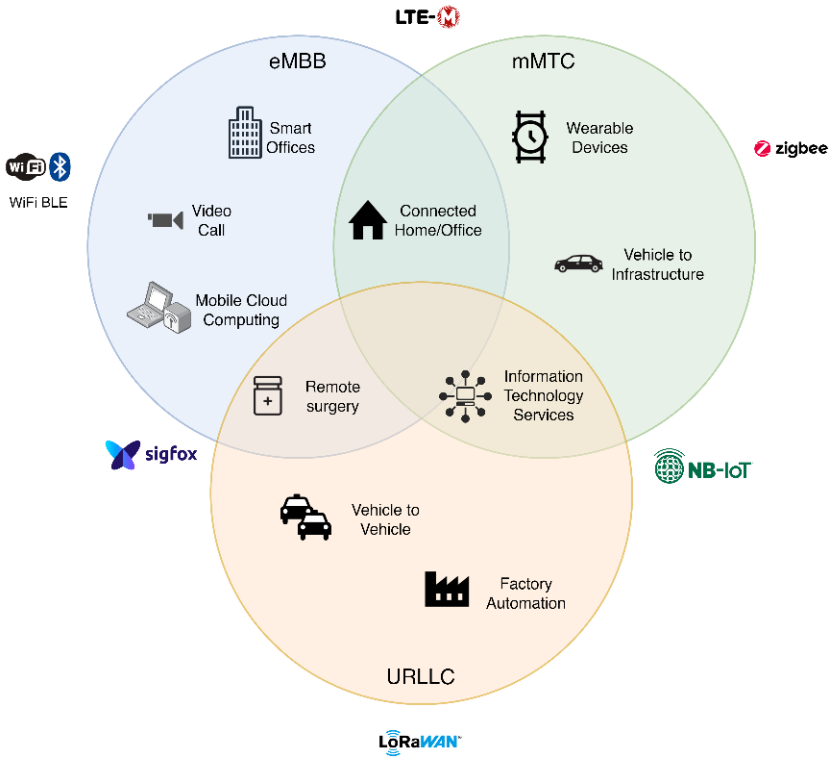


Figure 1. 5G Smart Application use-cases.

Network congestion refers to the arriving packets being dropped or delayed due to a higher number of packets arriving at the queue of the node than it can temporarily save in its buffer, as shown in Figure 2.

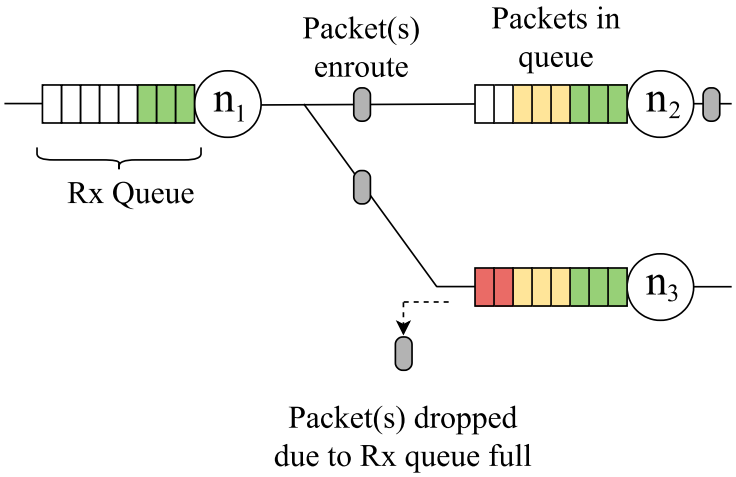


Figure 2. Packet loss/dropped due to full queue buffer.

To compensate for the lost information, some protocols use retransmission after a random period of waiting time in the hope that the dropping node will receive the packet if some queue slots are relieved. Since the buffer has a limited amount of space available for arriving packets, a node starts dropping packets after the queue is full, as illustrated in Figure 2. As the number of network nodes increases, the issue of packet loss or delay is increased exponentially and begs investigation in the massive IoT-based MANET [10].

AQM is a term associated with queue length management, where methods are deployed to monitor and manage the queue based on predefined criteria. Most AQM techniques probabilistically drop the arriving packets, where probability is calculated based on the current state of the queue [11]. RED [12] is considered an effective technique to apply AQM, which utilizes the calculated average queue length to monitor the current state of the queue. Apart from the advantages RED offers, incorporating RED also brings some issues, which may result in loss of performance gain in highly congested networks and increased delays due to retransmissions when the queue is highly congested [13]. The same performance issues have also been reported in another implementation of RED in [14] named exponential RED (E_RED). E_RED uses exponential behavior to drop the packets, which increases the performance of the network in some scenarios and also eliminates the use of a fixed parameter (maximum dropping probability), which was used in RED and its successors, including greed. Some other popular studies [15,16] also proposed an extension to AQM not limited to RED, which is all focused on infrastructure-based networks and to be especially implemented in routers or gateways, but none were focused on nodes where resource management is essential. Though the bandwidth utilization of CoDel (Controlled Delay AQM) [17] is better as compared to the original RED, but forks of RED have proven that RED can still be competitive when the required optimizations are carried out. Such an example is mentioned in [18], where CoDel performance is good but does not scale as the number of flows increases, as CoDel is based on queueing delay, which increases exponentially [19]. For IoT networks, [20] suggests that the solution "one size fits all" does not exist, as each AQM technique has merits and demerits, as ARED (fork of RED) performs better but suffers from latency issues, while CoDel does not, but underperforms ARED. CoDel is a queuing delay-based approximation of the packets to be dropped, while RED uses queue-length to determine the dropping probability. Measuring queueing delay can be very complex as it requires clock synchronization, sending and analyzing ICMP packets over the network, and infrastructure support from the network [21].

Realizing these limitations, in this paper, we have proposed an optimized extension of the RED algorithm, AgRED, which targets the performance parameters of the network initially left out by both of the algorithms by using a sigmoid function for calculating the dropping probability of the arriving packets.

The remainder of the paper is organized as follows: existing literature, highlighting the research being conducted on AQM, is discussed in Section 2; contributions have been discussed in Section 3; the development of the AgRED technique is discussed in Section 4; simulation setup is presented in Section 5; the results are discussed and analyzed in Section 6; and lastly, the conclusion and future work are mentioned in Section 7.

2. Related Work

AQM is used as a replacement for traditional passive queue management techniques, including drop-tail and drop-head, to control the ever-increasing congestion in networks [22]. Methods developed for AQM can be easily deployed to any switch or router, but current trends have shown its increasing usage in WSN and IoT networks due to an increase in broadcast storms utilizing whole queue buffers of the WSN/IoT nodes [23]. While AQM succeeds in providing free slots in the queue by dropping the arriving packets randomly before the queue overflows, it also results in low-performance gains in terms of throughput, end-to-end delay, latency, and jitter [24–26]. Due to these performance challenges, several AQM techniques have been developed by researchers to overcome these

issues and provide better queue management for a variety of networks. Some of these techniques, especially those based on RED, are discussed later in this section.

RED was proposed by S. Floyd et al. in [12] as an initial AQM technique replacing drop-tail, which drops or marks the arriving packets based on the probability before the queue is full to make the slots available to arriving packets. RED defines parameters such as minimum threshold $\min_{th}$, maximum threshold $\max_{th}$, and average queue length avg, and calculates the probability of the dropped packets if avg is between $\min_{th}$ and $\max_{th}$. RED manages the queues in such a way that the queue is available most of the time, but implementing RED has resulted in reduced performance of the whole network [13].

An architecture for 5G implementation to provide NB-IoT is presented in [27], which discusses a technique to allocate download and uplink resources from the eNodeB base station to the NB-IoT Controller. The paper also presents a queue management system that is made to prioritize IoT-related messages over the 5G network to provide better QoS. A similar study for 5G URRLC and eMBB networks was presented in [28], where the authors present a resource queueing system to improve the performance of 5G NR systems by providing analysis of preemptive priority with random resource requirements. The paper presents core performance metrics, including session drop probability and system resource utilization, to improve the performance of the system. An implementation of RED was proposed by Hussein Abdel-Jaber in [14], which included modification of the behavior of the algorithm for dropping packets. The proposed modified algorithm followed an exponential function to calculate the dropping probability P_d in such a way that P_d is lower when the queue length is closer to $\min_{th}$ and exponentially increases when it gets near to $\max_{th}$. The results show definitive improvement in average queue delays and average queue length for lower values of $\min_{th}$, while maintaining the same threshold as RED. A similar implementation is carried out by Ahmad Adel Abu-Shareha in [29], which proposes a time-window augmented RED technique. This algorithm works by calculating the weighted average queue length by adding arriving packets in a timely distributed manner and then comparing it with the buffer size of the node. The results show better performance in high and moderate traffic scenarios in terms of packet loss and delays, but compared to E_RED, they exhibit poor performance in some other network situations. Maha Salaheldeen used a type-2 fuzzy logic system with weighted random early detection (WRED) to optimize packet drop rates in [30]. The proposed scheme uses fuzzy logic for clustering the packet dropping in three groups (low, medium, and high) and uses maximum membership for each group in the type-2 fuzzy logic system. The results predict increased packet dropping of up to 60% as compared to WRED and a decrease in packet loss by up to 25% more than a type-1 fuzzy logic system. Yonggang Xu presented the importance of RED as compared to the priority-based queuing schemes in [31]. The author analyzed the performance improvement of RED in wireless sensor nodes with varying arrival and service rates. The author concludes that priority-based queuing without RED performs better for high-priority packets. An implementation of RED in the Next Generation Passive Optical Network (NG-PON) is presented in [32] by Xing Xu et al. The paper describes a classifier-based filter inserted at the gates of the Optical Network Unit (ONU) to either drop the burst packets or mark the packets with ECN. The filter defines three classes to perform QoS based on the priority class defined in packets as best-effort, assured forwarding, and expedited forwarding. The proposed results suggest an improved throughput of 35% and a reduced delay for the proposed classifier as compared to base RED and other classifiers for NG-PON. Monisha et al. have proposed a priority-based classifier and used dynamic RED with a Fuzzy Proportional Integral Derivative (FuzzyPID) controller enhanced with a Deep Neural Network (DNN) for different priority-based packets for WSN to be used in healthcare in [33]. The proposed algorithm by the authors utilizes high-priority and low-priority queues and only applies modified dynamic RED to the low-priority queue. The simulation results presented outline the benefits of using the proposed algorithm in WSN with lower packet loss, packet loss probability, and mean queue length as compared to proposed algorithm without DNN. The author states that the reduced transmission rates

help in maintaining low congestion in the network and that this can be further investigated in future. A study was carried out by Guan et al. in [34], where the thresholds are labeled as L1 and L2. The performance is measured in terms of packet loss probability, delay, and threshold, and results were presented by sliding L1 and L2 back and forth. The author finds out that there is a tradeoff between packet loss probability and delay by choosing appropriate L1 and L2 thresholds, and the delay, packet loss, and threshold are the function of the thresholds.

One recent publication suggested multiple threshold levels within RED to better control the queue-length when the average queue length is between the minimum and maximum queue length allowed, after which a hard decision to queue or drop the packet can be taken [35]. As there are multiple levels of thresholds defined, calculations need more computing resources. It is, however, suitable for infrastructure-based devices with an ample amount of power, but due to the small queue buffer in IoT sensors in 5G, multiple threshold levels will not affect the performance as gaps between thresholds will be minimum. Another AQM technique [36] uses the Semi-Markov Decision Process to estimate the dropping probability of the arriving packets before they can be queued into the buffer. The method provides better results as compared to drop-tail and CoDel but needs to set the target delay manually for various network types.

3. Contribution

Over the years, different modifications have been proposed by researchers across the globe to improve the performance of RED [37–40]. In this paper, we have proposed an AQM technique based on RED, named Aggressive-RED (AgRED). This technique has been proposed specifically focusing on NB-IoT-based MANET under 5G umbrella, where network nodes do not have large computing power or energy. AgRED technique supports the implementation of massive nodes existence in a small area supporting 5G mMTC use cases. Our key contributions presented in this paper include:

1. Development and implementation of AgRED algorithm to filter out or block initialization broadcast storm without any classifier (or processing delay). This will help in maintaining queue length and allow operational messages (control and data packets) to be accommodated in queue.
2. Analysis of AgRED technique and comparison with existing techniques like RED [12] and E_RED [14] using UDP traffic. Performance indicators including queue length, end-to-end delay, packet delivery ratio, throughput, packet drop due to queue overflow, and packet drop for intermediate nodes have been considered.
3. In order to validate the proposed scheme AgRED, we have performed a comparative analysis of AgRED against passive queue-management techniques, drop-tail and drop-head. We have considered two performance indicators namely, end-to-end delay and packets delivered to analyze AgRED performance against these passive techniques. The analysis show that AgRED outperforms both the passive queue management techniques.

4. Methodology

Broadcasts are an essential part of any network and are used in discovering nodes and routes across the topology. However, broadcast storms often lead to congestion, loops, and suboptimal performance of the network. AgRED targets excessive broadcast storms without implementing any QoS classifier or packet identification. The communication that takes place in a network is divided into two streams, one is the initialization stream where nodes discover neighbors or exchange control messages for routing updates and other critical information, and the second is the operational stream, where data along with control messages are sent to or received from other nodes of the network. The initial stream is the one where excessive broadcast storms are generated and need to be addressed to maintain a low queue length count. In order to better understand the problem, consider an NB-IoT MANET, having n nodes within a defined region. If any broadcast is generated, e.g., an

ARP (Address Resolution Protocol) request or AODV RREQ (Route Request) message, each node will receive at least one copy of the broadcast. Similarly, if a broadcast is generated by each node in a short period of time, each node will receive at least $n-1$ copies of the broadcasts. Now, let us consider that queue length of the node is less than $n-1$. As a result, the queue will start to drop all the packets until space is available in the queue. However, RREQ and ARP are just examples of how many control messages are broadcasted onto the network, as many other protocols from layer 2 and layer 3 rely on broadcast storms to gather and distribute network information. Due to this behavior, during the initialization phase of the wireless network, nodes receive multiple copies of broadcast messages due to a shared medium. Previous considerations of RED employ a linear, step, or exponential function to drop these packets, which does not differentiate with the initialization and operational mode of the network.

4.1. Dropping Probability for RED, E_RED, and AgRED

After the queuing model of the network was defined, we computed the dropping probabilities for the RED, E_RED, and AgRED techniques. A sample queue for a node, with thresholds and average queue length, is presented in Figure 3.

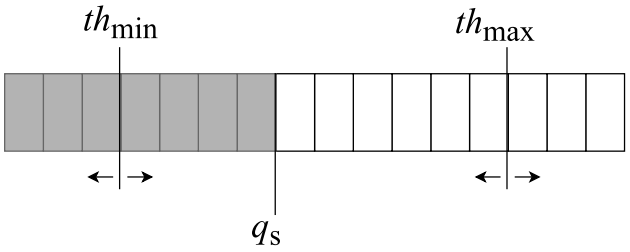


Figure 3. Threshold levels within a queue-buffer for RED.

For RED technique, two threshold levels ($th_{\min}$ and $th_{\max}$) have been considered with an average queue length of q_{avg}, to calculate the dropping probability P_d of arriving packets using the equation defined in [12]:

$$P_d \leftarrow P_{\max} \times q_s \tag{1}$$

where, P_{max} is the maximum probability for packet dropping with nominal values ranging from (0, 0.1] [12], and q_s is defined as [41]:

$$q_s \leftarrow \frac{q_{avg} - th_{\min}}{th_{\max} - th_{\min}} \tag{2}$$

In (2), the parameter q_s is the percentage of the queue, which is filled with respect to the difference of $th_{\min}$ and $th_{\max}$. This helps in keeping the probability of dropping packets between $0 \leq P_d \leq P_{\max}$. For the exponential RED technique [14], an exponential function for q_s is considered as mentioned below to relax the dropping probability initially and then increase it exponentially until it reaches $P_{\max}$, where q_s is defined as in [14]:

$$q_s \leftarrow e^{\frac{qavg - th_{\min}}{th_{\max} - th_{\min}}} \tag{3}$$

For AgRED, which is our proposed technique, we define an activation function, called sigmoid, commonly used in deep learning. Using this function, we probabilistically target messages in the initialization phase, with a higher probability of targeting broadcast messages. The sigmoid function is presented below:

$$f(x) \leftarrow \frac{1}{1 + e^{-x}} \tag{4}$$

In order to better understand the dropping/blocking function of RED, E_RED and AgRED, Figure 4 presents the dropping probability behavior, defined in (1)–(4). Threshold levels ($th_{\min}$ and $th_{\max}$) control the dropping function such that all the arriving packets are accepted if the queue length is less than $th_{\min}$, while discarding the arriving packets if the queue length is more than the defined parameter $th_{\max}$. Bounding the packet dropping probability function will ensure that the defined function will only be applied to the incoming packets when the average queue length is between minimum and maximum thresholds defined. Therefore, P_d can be defined as the same as in RED.

$$P_d = \begin{cases} l_q < th_{\min} & 0 \\ th_{\min} < l_q < th_{\max} & P_d \\ l_q > th_{\max} & 1 \end{cases} \tag{5}$$

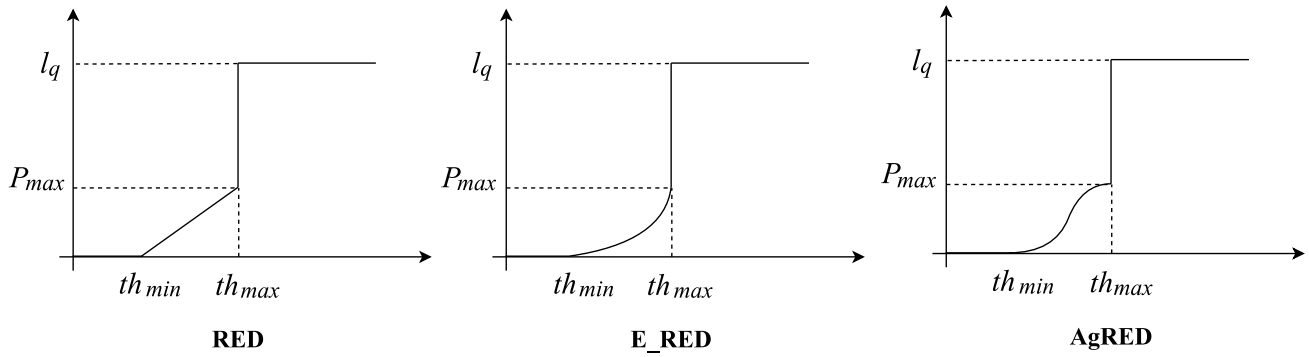


Figure 4. Dropping probability curve comparison between RED, E_RED, and AgRED, the dropping probability is presented on the *y*-axis, while queue length is shown on the *x*-axis.

The algorithm for RED predicts a linear dropping function depicted in Figure 4, which increases the dropping probability of arriving packets with the increase in average-queue length. This function does not account for the behavior of arriving packets in bursty networks. Therefore, the queue may get congested prematurely in the network initialization phase and result in the dropping of all other arriving packets until the queue length reaches an average value of less than $th_{\max}$.

This effect is somewhat adjusted in Figure 4 E_RED algorithm uses an exponential dropping function, so that the dropping probability increases exponentially with the arriving packets and therefore, fewer packets are dropped initially when average queue length is greater than but near to $th_{\min}$ as compared to $th_{\max}$. E_RED dropping probability then starts increasing exponentially so that P_d increases when average queue length is close to $th_{\max}$. On one hand, E_RED allows fewer packets to drop from the queue due to its exponential behavior, but on the other hand, it also affects overall packets collected at the sink, resulting in reduction in end-to-end delay.

The algorithm proposed in this paper uses a sigmoid function to incorporate the benefits of both previously existing techniques, i.e., RED and E_RED. The idea is to allow arriving packets into the queue initially, with lowest dropping probability at $th_{\min}$, while discarding packets aggressively when the average queue length is between $th_{\min}$ and $th_{\max}$. Due to the ad-hoc nature of MANETs, where broadcast messages are received multiple times, AgRED helps in discarding most of the early initialization messages. This technique is implemented without any classifier, hence using fewer resources and providing better battery life. Therefore, AgRED starts discarding arriving packets more rigorously, resulting in a higher value of P_{AgRED}. By doing so, it discards some or most of the burst packets, which might or might not contain duplicate messages. P_{AgRED} is defined as follows.

$$P_{AgRED} = \frac{1}{1 + P_d} \tag{6}$$

where P_d is:

$$P_d = \left\{ e^{-MAX(P_{AgRED})} \times \frac{e^{-q_{avg}} - e^{-th_{\min}}}{e^{-th_{\max}} - e^{-th_{\min}}} \right\} \quad (7)$$

4.2. AgRED Algorithm

Algorithm 1 presents the AgRED dropper filter, which checks the average queue length for each arriving packet and decides whether to keep the packet in the queue or drop the packet based on the Equation (7).

Algorithm 1. AgRED Dropper

```
Init:
    l_q ← 0
    count ← −1
for each packet arriving:
        calculate the new l_q
        if the queue is nonempty
            l_q ← (1-w_q)l_q + w_q. q
        Else
            m ← f(time—q_time)
            l_q ← (1-wq)m. lq
          If th_min ≤ lq ≤ th_max
            count++
            calculate probability P_d:
            P_d ← { e^{-MAX(P_AgRED)} × (e^{-q_avg} − e^{-th_min}) / (e^{-th_max} − e^{-th_min}) }
            calculate PAgRED ← 1/(1 + Pd)
          Drop the packet with probability PAgRED
          else if th_max < l_q
            drop the arriving packet
            Count ← 0
          Else
            count ← −1
          when queue becomes empty
            q_time ← time
End
```

5. Simulation Setup

The simulation of the proposed algorithm is carried out using the OMNeT++ discrete event simulator and the INET framework. OMNeT++ measures and evaluates the status of simple and compound modules whose functionality can be extended with C++. Furthermore, INET extends the functionality of OMNeT++ with real-time models approved by IEEE and IETF. For analyzing the performance of the proposed algorithm AgRED and comparison with RED and E_RED, a graph G of 25 km^2 area has been generated, where nodes are positioned using the normal distribution, and movement has been modeled using the Gauss Markov mobility model [42], where node speed and direction are functions of average speed and direction as well as random variables (s_{rt} and d_{rt}) following the Gaussian distribution:

$$s_{t+1} \leftarrow \alpha s_t + (1-\alpha)s_{avg} + \sqrt{(1-\alpha^2)}s_{rt} \quad (8)$$

$$d_{t+1} \leftarrow \alpha s_t + (1-\alpha)d_{avg} + \sqrt{(1-\alpha^2)}d_{rt} \quad (9)$$

where s_{t+1} and d_{t+1} are the speed and direction of the node for next time-slot. The value α having bounds $0 \leq \alpha \leq 1$ is defined as follows

$$\alpha = \begin{cases} 0 & \text{totally random motion} \\ 0 < \alpha < 1 & \text{random motion with index } \alpha \\ 1 & \text{linear motion} \end{cases} \quad (10)$$

The radio used in the simulation is an IEEE 802.15.4 narrowband interface, which falls under the umbrella of 5G, enabling a massive number of devices to communicate with each other while utilizing a small portion of the available spectrum [43], emulating an NB-IoT network. RED was implemented to be used with TCP as major traffic on the internet is based on TCP. However, with the growing demand for real-time applications, including multimedia streaming. UDP is becoming more popular [44] and is an essential part of MANET and other similar technologies, such as TCP, could drastically increase the number of packets transmitted due to retransmissions and control messages. It is also mentioned in [45] that UDP utilizes major bandwidth share as compared to TCP when using RED, while TCP was able to utilize more bandwidth initially (due to no congestion) but bandwidth slowly falls as it tries to reduce the congestion window by half each time congestion is observed, limiting the overall bandwidth. Important parameters of the simulation have been listed in Table 1.

Table 1. Simulation Parameters.

Description	Parameter
Covered Area	5000 m × 5000 m
Number of Runs per setup	5
Simulation time	30 sec
Number of Nodes	200, 250, 300, 350, 400, 450, 500
Mean Speed	5, 10, 15, 20, 25 (m/s)
Speed Std. Dev	2 m/s
Routing Protocol	AODV RFC 3561
WLAN Interface type	IEEE 802.15.4 Narrowband
Node Bitrate	250 Kbps
Node Communication Range	500 m
Node random mobility model	Gauss Markov Mobility
Data Transport Protocol	UDP
Packet length	1000 Bytes
Packet sending interval	Exponential (10 ms)
Buffer size of nodes	100
Minimum Threshold (RED, E_RED, AgRED)	5
Maximum Threshold (RED, E_RED, AgRED)	50
Maximum probability of packet dropping (P_{max}) for RED and AgRED	0.02

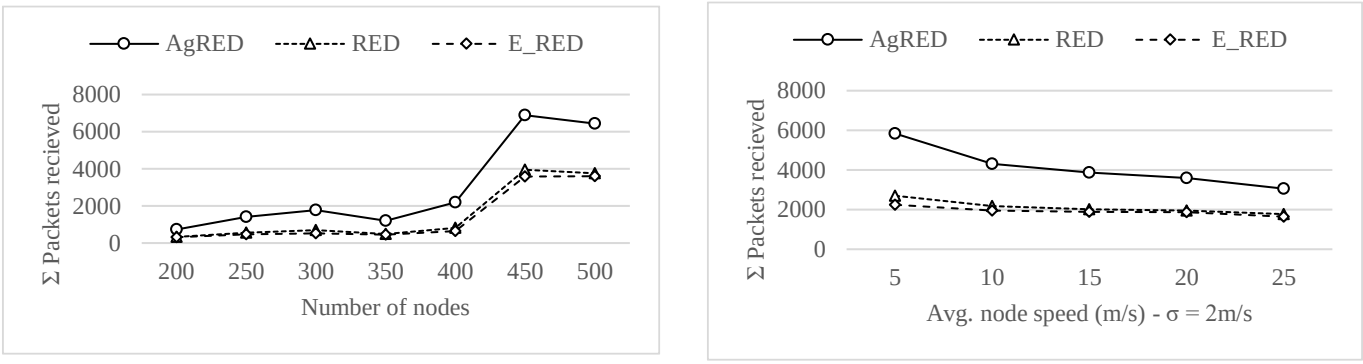
6. Simulation and Evaluation

The simulations have been carried out on two computers, both running Core i7 4th generation processors with at least 8GB of RAM. The results presented in this section have been obtained directly from the OMNeT++ built-in result collector. AgRED, E_RED, and RED performances have been compared with different sets of speeds and numbers of nodes for the narrowband interface to support massive NB-IoT networks. The results have been

compiled and basic operations, e.g., sum and average, have been performed on the results, and the output has been plotted.

The traffic scenario chosen for the simulation is a highly congested network that suffers greatly from packet drops due to congestion and the low buffer size of the participating nodes. In order to eliminate the varying distance between the transmitter (Tx) and receiver (Rx), which may affect the accuracy of the comparison between three methods, both nodes, Tx and Rx, are placed at the far corners of the topology, while all other nodes are positioned randomly following the normal distribution.

Figure 5a presents the total number of packets that are received by the receiving node in the total number of simulation runs against the different numbers of nodes randomly distributed with varying speeds and directions. The graph shows that the AgRED algorithm provides a higher overall number of received packets as compared to RED (50% more packets received) and E_RED (55% more packets received). The graph also shows that a higher number of nodes provides better results due to the shorter distance between two adjacent nodes and the high node density, whereas AgRED yields better results, with an average difference of 465 more packets as compared to RED and 512 more packets as compared to E_RED. Another comparison with the number of packets received is shown in Figure 5b with different sets of average speed and a standard deviation of 2 m/s. Overall, AgRED delivers a higher number of packets for all the speeds tested, but deliverables are more consistent with RED and E_RED, as depicted in Figure 5. At a low speed of 5 m/s (average), AgRED performs 53% and 61% better as compared to RED and E_RED, respectively. On the other hand, at an average speed of 25 m/s, AgRED provides 41% and 46% more packets as compared to RED and E_RED, respectively.



(**a**) (**b**)

Figure 5. Sum of packets received with UDP traffic for (**a**) varying number of nodes and (**b**) average node speed.

The overall packet delivery ratio has been plotted in Figure 6 by averaging packets received from the results of Figure 5 and dividing them by the total number of packets generated in each run. It is evident by the plots presented in Figure 6 that AgRED performs 51% and 46% better as compared to RED and E_RED. After some packets were dropped from the queue with a higher probability, it made more space for the upcoming packets, therefore minimizing the future dropping of the packets with a lower probability of dropping as compared to the probability when the queue was more occupied.

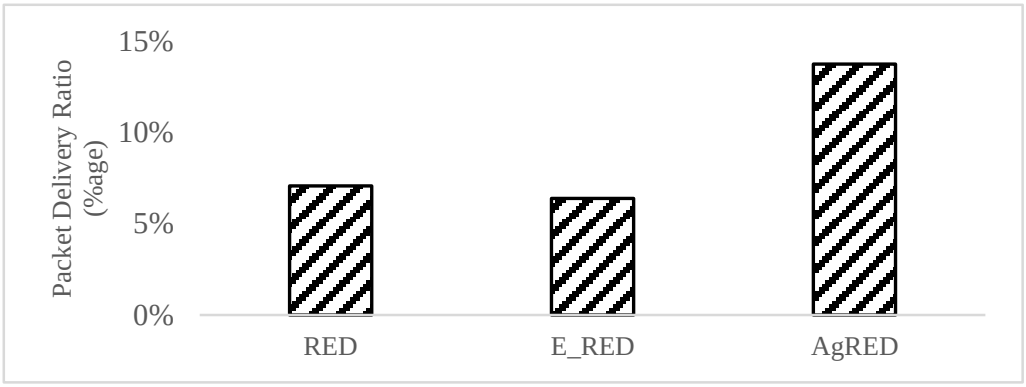
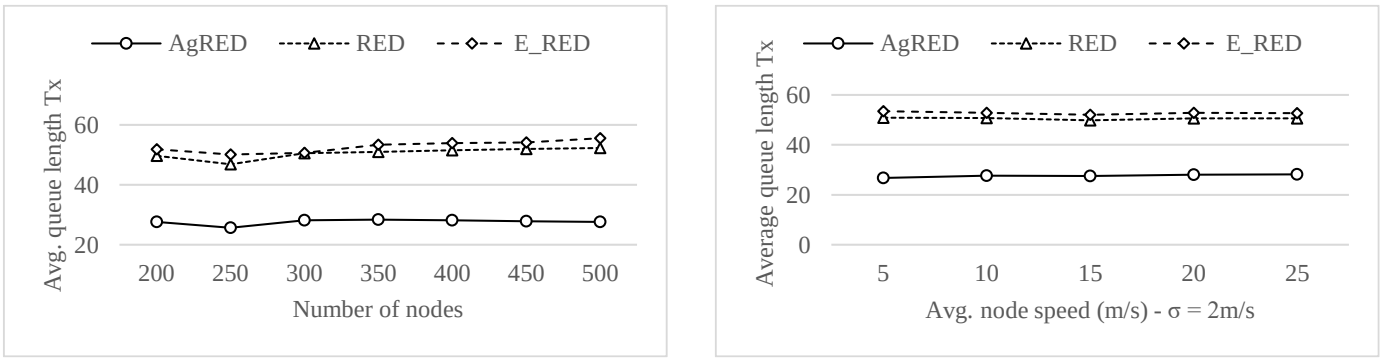


Figure 6. Average packet delivery ratio using UDP for RED, E_RED, and AgRED.

Figure 7 shows the average utilization of the queue of the transmitter for RED, E_RED, and AgRED, where AgRED keeps the average value below 30% of the number and speed of nodes. This allows AgRED to maintain a proper ratio of packet acceptance and probabilistically drop the arriving packets randomly. Figure 8 depicts the average queue length of the nodes that are participating in the construction of a path from Tx towards Rx and forwarding packets. Because all the nodes present in the topology may not be participating in forwarding data packets and only communicating with neighbor nodes to maintain connectivity, the queue length of such nodes would be considerably smaller, and their average queue lengths will account for non-forwarding nodes and hence bias the results. Therefore, Figure 8 only shows the average queue length for nodes that have active AODV routes towards the destination and actively forward data packets, hence eliminating any biased results. As evident from Figure 8, the average queue length observed for AgRED is mostly consistent and significantly smaller as compared to RED and E_RED, regardless of the increase in the number of nodes or node speed.



(**a**) (**b**)

Figure 7. Average queue length of transmitter vs. (**a**) number of nodes and (**b**) average node speed.

Figure 9 shows the average queue length with one standard deviation away from the mean for RED, E_RED, and AgRED. Assuming the normal or Gaussian distribution of arriving packets, around 68% of the arriving packets will not fill the queue later than 55%. Hence, packets arriving at a 2-standard deviation away from the mean will also have the opportunity to be allowed into the node queue. In comparison to AgRED, RED and E_RED already have one standard deviation, reaching 82% and 88% of the available queue length, respectively. This illustrates that RED and E_RED will fill the queue prematurely, and after the queue is full, all arriving packets will be discarded. All the packets accumulating at the input of the buffer will fill the buffer. Though RED and E_RED try to minimize this effect by randomly dropping the packets, ultimately, the arriving packets will completely fill up the queue. At this point in time, any new packets arriving at the input of the buffer will be dropped.

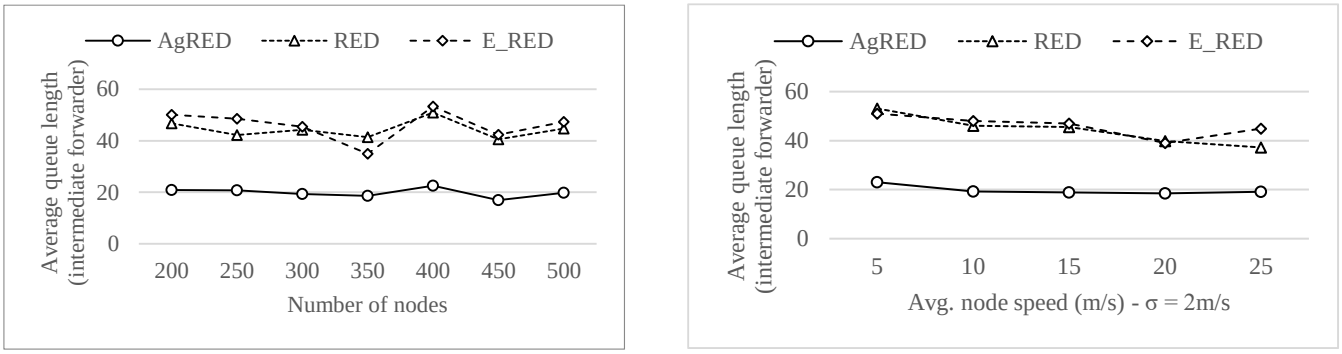


Figure 8. Average queue length of nodes participating in forwarding packets vs. (**a**) number of nodes and (**b**) average node speed.

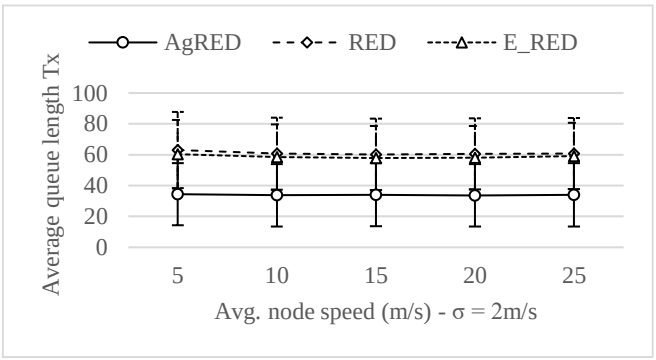


Figure 9. Average and standard deviation of queue length Tx vs. average node speed.

Figure 10 shows the average number of packets that are dropped due to queue overflow. As shown, the AgRED packet drop is consistently zero, regardless of the speed and number of nodes, allowing the arriving packets a place in the queue and, hence, the possibility of successfully delivering the packet at the destination. The most focused parameter of this study remains the delay metric of the network, which highlights the performance gain of AgRED against the RED and E_RED approaches. In Figure 11a, mean end-to-end delay is plotted against the number of nodes, representing the dominance of greed by an average margin of 49.75% against RED and 50% against E_RED. The rate of change in overall delay can be estimated by taking the difference between delays at the minimum number of nodes, i.e., 200, and the maximum number of nodes, i.e., 500, and that is found to be 297.2 milliseconds (ms) for Agreed, while E_RED and RED differences stand at quite larger values of 739.2 ms and 921.4 ms, respectively.

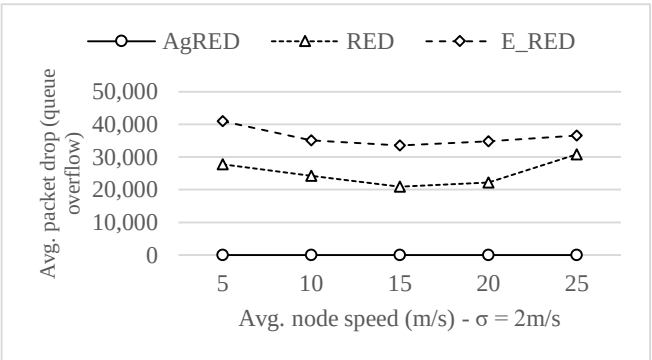
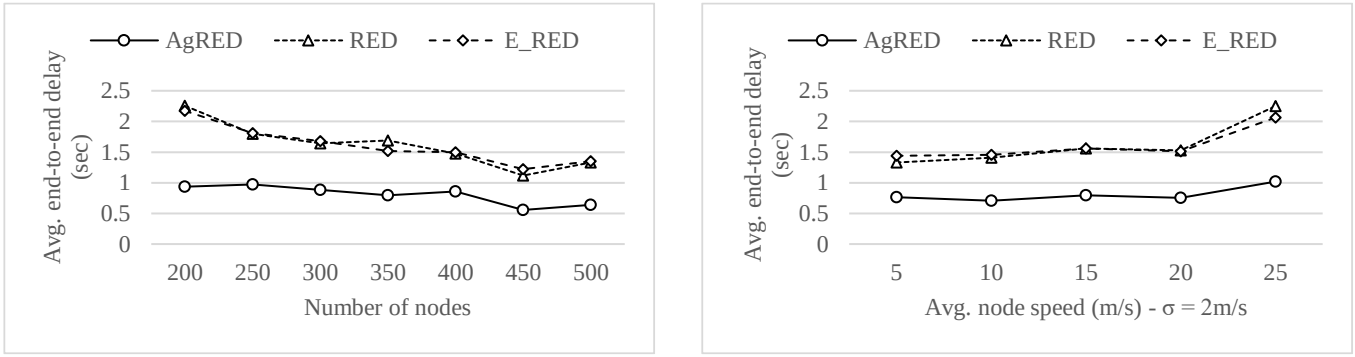


Figure 10. Average packets dropped due to queue overflow in UDP vs. Average node speed.



(**a**) (**b**)

Figure 11. Average end-to-end delay for UDP with (**a**) varying number of nodes and (**b**) average node speed.

Figure 11b shows the change in end-to-end delay with increasing average node speed (s), again highlighting the advantage of AgRED. Similar to Figure 11a, an overall improvement of 49.75% and 50% is observed in AgRED as compared to RED and E_RED, respectively. In other words, for the average speed of 25 m/s, the AgRED delay was found to be 1.018 sec as compared to 2.25 sec and 2.06 sec for RED and E_RED, respectively. Since end-to-end delay varies over time, presenting average values only might present an incomplete result by eliminating the outlier values. Therefore, a box plot is presented in Figure 12 representing the median and quartiles from the box, and the whisker represents values lower than quartile 1 and higher than quartile 3.

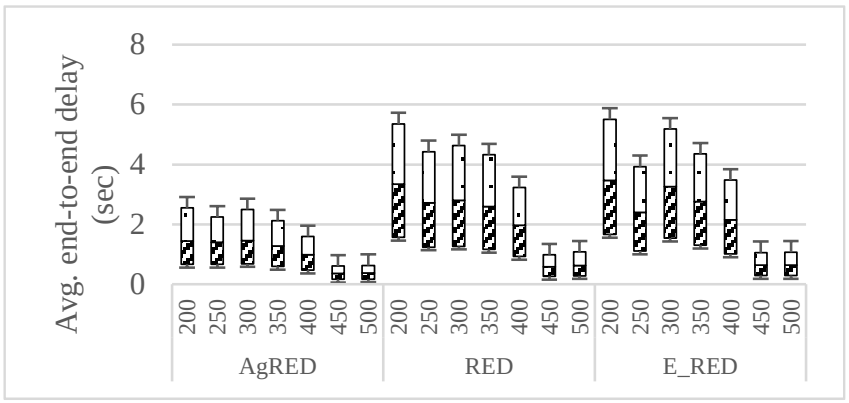


Figure 12. Box plot for average end-to-end delay (UDP) vs. number of nodes in AgRED.

This graph is evidence that overall delay performance is almost doubled in the majority of scenarios expanding from 200 nodes to 400 nodes. The higher number of nodes, e.g., 450 and 500, represents a major improvement in all three techniques caused by the routes created by the routing protocol AODV, which benefits from the larger number of nodes in the vicinity. Thus, a shorter route is formed and delay is minimized. It is believed that the routing protocol selected will play an important role in estimating delays regardless of the AQM technique used [46], but regardless of the routing protocol used, AQM techniques will offer improved end-to-end delays with the same routing protocol in congested networks by managing queues to help queue overflows.

The throughput of AgRED, RED, and E_RED is presented in Figure 13a, highlighting that the overall throughput of AgRED surpasses RED and E_RED by a great margin. In the mentioned graph, AgRED performs 49.5% better as compared to RED and 44.8% better as compared to E_RED. Generally, higher throughput is achieved in all three techniques where the number of nodes is between 450 and 500.

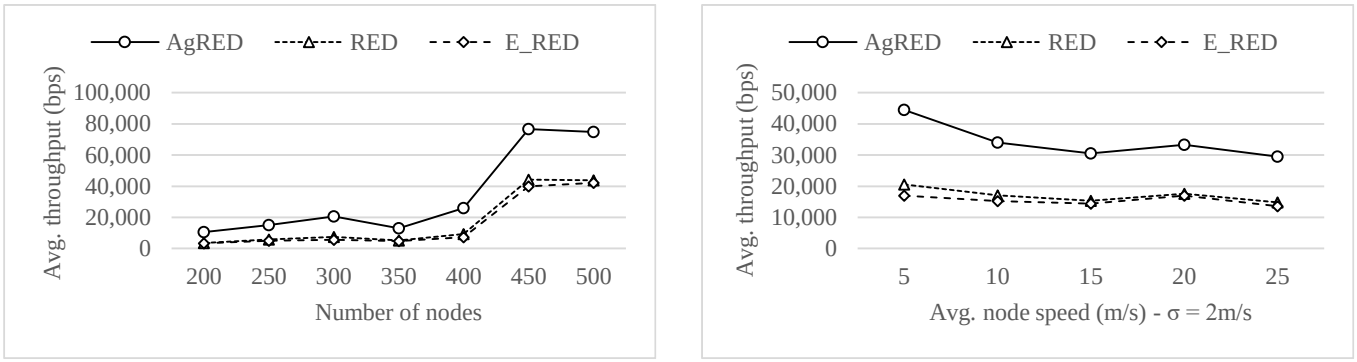


Figure 13. Average throughput (UDP) vs. (**a**) number of nodes and (**b**) average node speed.

The gain insured in throughput going from 400 nodes to 450 nodes is 33.8% in AgRED, while RED gain in throughput is 20.9% and E_RED has the least gain of only 18.2% as compared to other two techniques. This improvement in gain also favors the AgRED implementation, which offers at least 12.8% and 15.5% more performance as compared to RED and E_RED, respectively. Though the improvement in throughput is evident in Figure 13a, Figure 13b shows the throughput trend against the average speed of the nodes. Apart from the improvements of AgRED, it is important to note here that all three techniques suffer from lower throughput as average speed is increased. Though, AgRED here as well performs such that the lowest throughput at 25 m/s (29,544.44 bps) is still higher than the highest throughput of RED at the speed of 5 m/s (20,548.57 bps).

Similar to end-to-end delay, throughput is measured for each receiving packet and therefore is a random variable. Figure 14 represents the box plot for the throughput of all three techniques, highlighting major improvements for AgRED against RED and E_RED. It can be seen that the highest throughput achieved is by AgRED for the 450 nodes, with a peak throughput of more than 120,000 bps.

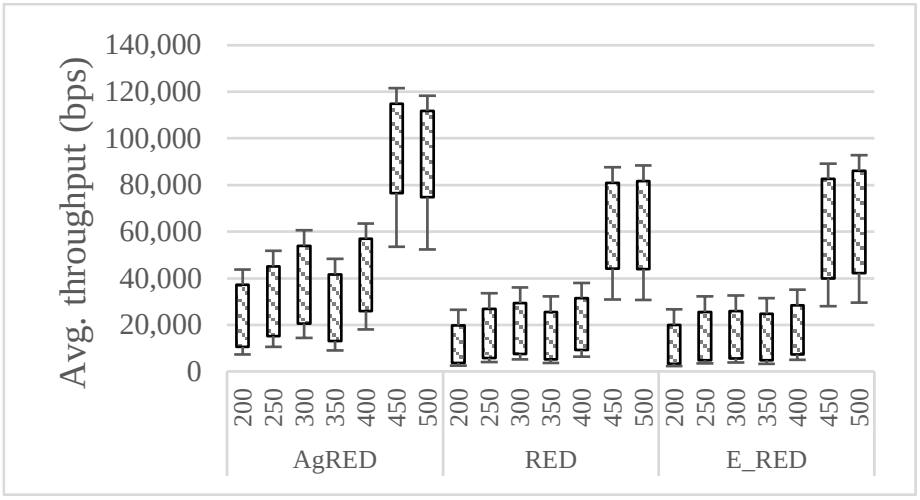


Figure 14. Box-plot of throughput (UDP) vs. number of nodes.

In order to highlight the importance of AQM techniques in NB IoT MANET, it is important to compare AgRED (an AQM technique) with two passive queue management schemes. Figure 15 presents the comparison of the proposed scheme AgRED with passive queue management schemes, namely drop-tail and drop-head. Average end-to-end delay has been presented against the increasing number of nodes in Figure 15a, where drop-tail shows losses in delay due to no management of the queue-length. On the other hand, drop-head or drop-from-the-front shows incredible gains (less delay) as compared to drop-tail, while falling

short in comparison to AgRED. The average recorded values for drop-head point to a gain of 2.15 times less delay as compared to drop-tail but 1.64 times higher delay as compared to AgRED. Figure 15b shows that regardless of the queue management scheme used, the significance of speed is negligible as all the schemes remain consistent, but in comparison to each other, AgRED performs better against both passive queue management techniques.

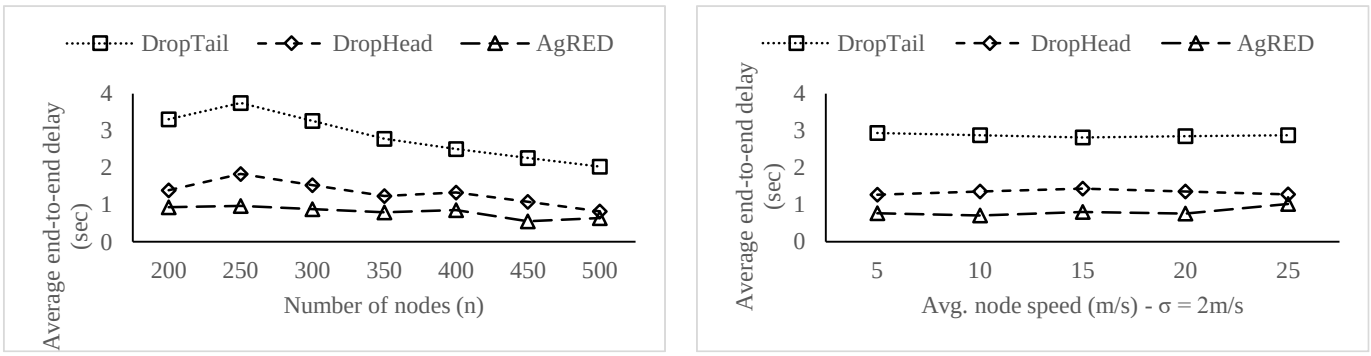


Figure 15. Average end-to-end delay (UDP) for Drop-tail, Drop-head and AgRED with (**a**) varying number of nodes and (**b**) varying speed.

The improvement in delay for drop-head is evident against drop-tail in Figure 15 (though falling short as compared to AgRED), but apart from delay, it is vital to look into the number of packets successfully delivered by each queue management scheme. Therefore, Figure 16 presents the sum of all the packets received during the simulation against (a) an increasing number of nodes and (b) increasing node speed. All three schemes in Figure 16a show the increase in the number of packets delivered as the number of nodes increases. This is because there are now more paths and queues available to accommodate higher numbers of packets. It is interesting to observe that as the number of nodes crosses 400, the difference in packets delivered between AQM (AgRED) and passive queue management schemes becomes significantly large, which proves that for massive deployment of nodes, the AQM (AgRED) technique will outperform passive queue management schemes (drop-tail and drop-head). Figure 16b shows the total number of packets received for AgRED, drop-tail, and drop-head techniques against node speed. As evident from Figure 16b, the AgRED technique outperforms both passive queue management schemes in terms of the total number of packets received at various node speeds. Moreover, the difference between the number of packets received for AgRED versus drop-tail and drop-head techniques decreases at high node speeds.

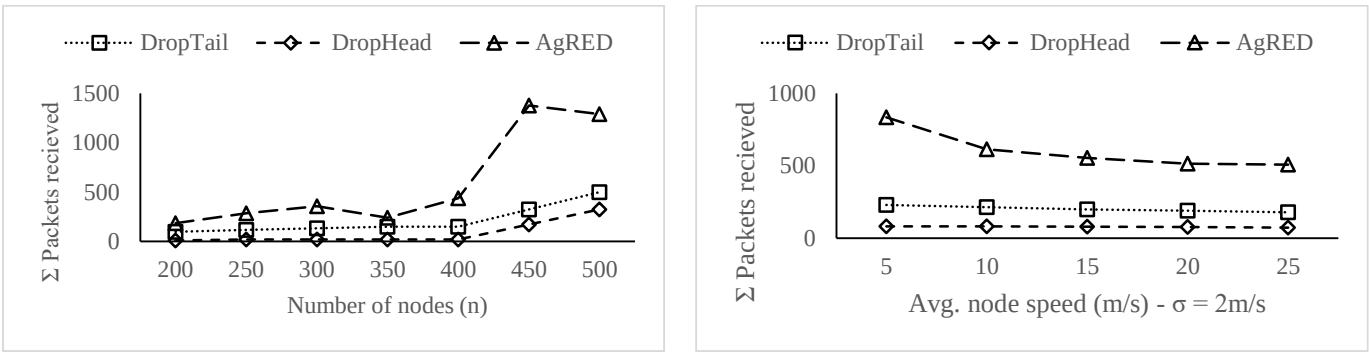


Figure 16. Sum of data packets received (UDP) for Drop-tail, Drop-head and AgRED with (**a**) varying number of nodes and (**b**) varying speed.

7. Conclusions

The performance of massive IoT networks in 5G suffers greatly as compared to typical wireless networks due to small queue buffers. RED mitigates this by optimizing the queue buffer to allow the probabilistic dropping of the arriving packets, but this yields suboptimal performance as the average queue length increases, eventually leading to queue overflow. AgRED ensures that overall performance improves in NB-IoT congested networks in 5G as compared to RED and E_RED and passive queue management schemes, drop-tail and drop-head, while using UDP as the transport protocol. Aggressive behavior of the proposed AgRED algorithm enables queue buffer optimization, such that no packets are dropped due to buffer overflow. Therefore, it is safe to conclude that as a consequence of incorporating the AgRED technique, possible flooding attacks can be prevented where an attacker tries to fill queue buffers with massive amounts of falsified traffic directed to render the nodes useless and force nodes to drop all the genuine incoming messages. The obtained results show the better overall performance of AgRED in comparison to RED and exponential RED (E_RED), tested in a discrete event simulator. AgRED can be utilized in many real-time MANET applications and is not limited to 5G mMTC applications as it supports high mobility and small-sized nodes with restricted resources because there is no complex implementation needed, which not only offers better battery life but also keeps the node's processing requirements at a minimum.

Author Contributions: Conceptualization, methodology and software, S.T.A.J. and I.A.; validation and formal analysis, S.T.A.J. and S.A.; investigation, S.T.A.J., I.A. and S.A.; writing—original draft preparation, S.T.A.J.; writing—review and editing, I.A. and S.A.; project administration, I.A. and S.A.; funding acquisition, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology (MoST), Pakistan (Project # 01/2020).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available as data is owned by parent institution and funding agency.

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

References

1. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. [CrossRef]
2. Kuo, Y.-W.; Li, C.-L.; Jhang, J.-H.; Lin, S. Design of a wireless sensor network-based IoT platform for wide area and heterogeneous applications. *IEEE Sens. J.* **2018**, *18*, 5187–5197. [CrossRef]
3. Shafique, K.; Khawaja, B.A.; Sabir, F.; Qazi, S.; Mustaqim, M. Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios. *IEEE Access* **2020**, *8*, 23022–23040. [CrossRef]
4. Ungurean, L.; Brezulianu, A. An internet of things framework for remote monitoring of the healthcare parameters. *Adv. Electr. Comput. Eng.* **2017**, *17*, 11–16. [CrossRef]
5. Sodhro, A.H.; Gurtov, A.; Zhaid, N.; Pirbhulal, S.; Wang, L.; Rahman, M.M.U.; Imran, M.A.; Abbasi, Q.H. Toward convergence of AI and IoT for energy-efficient communication in smart homes. *IEEE Internet Things J.* **2020**, *8*, 9664–9671. [CrossRef]
6. Abdullah, N.; Al-wesabi, O.A.; Mohammed, B.A.; Al-Mekhlafi, Z.G.; Alazmi, M.; Alsaffar, M.S.; Aljaloud, A.S.; Baklizi, M.; Sumari, P. Improving Waste Management System Efficiency and Mobility with Efficient Path MANET. *Appl. Sci.* **2021**, *11*, 11039. [CrossRef]
7. Sultanuddin, S.; Ali Hussain, M. Token system-based efficient route optimization in mobile ad hoc network for vehicular ad hoc network in smart city. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3853. [CrossRef]
8. Mohamed, H.E.-D.; Azmi, H.; Taha, S. Using MANET in IoT Healthcare Applications: A Survey. *Int. J. Comput. Digit. Syst.* **2021**.
9. Barzegar, H.R.; el Ioini, N.; Pahl, C. Wireless network evolution towards service continuity in 5G enabled mobile edge computing. In Proceedings of the Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020; pp. 78–85. [CrossRef]
10. Kanellopoulos, D. Congestion control for MANETs: An overview. *ICT Express* **2019**, *5*, 77–83. [CrossRef]
11. Hilquias, V.C.C. Queueing systems with different types of renovation mechanism and thresholds as the mathematical models of active queue management mechanism. *Discret. Contin. Models Appl. Comput. Sci.* **2020**, *28*, 305–318. [CrossRef]
12. Floyd, S.; Jacobson, V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413. [CrossRef]

13. Khatari, M.; Zaidan, A.; Zaidan, B.; Albahri, O.S.; Alsalem, M. Multi-criteria evaluation and benchmarking for active queue management methods: Open issues, challenges and recommended pathway solutions. *Int. J. Inf. Technol. Decis. Mak.* **2019**, *18*, 1187–1242. [CrossRef]
14. Abdel-Jaber, H. An exponential active queue management method based on random early detection. *J. Comput. Netw. Commun.* **2020**, *2020*, 8090468. [CrossRef]
15. Athuraliya, S.; Li, V.H.; Low, S.H.; Yin, Q. REM: Active queue management. In *Teletraffic Science and Engineering*; Elsevier: Amsterdam, The Netherlands, 2001; Volume 4, pp. 817–828. [CrossRef]
16. Feng, W.-C.; Shin, K.G.; Kandlur, D.D.; Saha, D. The BLUE active queue management algorithms. *IEEE/ACM Trans. Netw.* **2002**, *10*, 513–528. [CrossRef]
17. Nichols, K.; Jacobson, V.; McGregor, A.; Iyengar, J. Controlled delay active queue management. *Experimental* **2018**, 1–25.
18. Järvinen, I.; Kojo, M. Evaluating CoDel, PIE, HRED AQM techniques with load transients. In Proceedings of the 39th Annual IEEE Conference on Local Computer Networks, Edmonton, AB, Canada, 8–11 September 2014; pp. 159–167. [CrossRef]
19. Keith, J.F.K.; Ross, W. Delay and Loss in Packet-Switched Networks. Available online: http://www2.ic.uff.br/~{}michael/kr1999/1-introduction/1_06-delay.htm (accessed on 15 September 2022).
20. Grazia, C.A.; Patriciello, N.; Klapez, M.; Casoni, M. Which AQM fits IoT better? In Proceedings of the IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), Modena, Italy, 11–13 September 2017; pp. 1–6. [CrossRef]
21. Salehin, K.M.; Rojas-Cessa, R. Scheme for Measuring Queueing Delay of a Router Using Probe-Gap Model: The Single-Hop Case. *IEEE Commun. Lett.* **2014**, *18*, 696–699. [CrossRef]
22. Hotchi, R.; Chibana, H.; Iwai, T.; Kubo, R. Active queue management supporting TCP flows using disturbance observer and smith predictor. *IEEE Access* **2020**, *8*, 173401–173413. [CrossRef]
23. Kim, M.; Jaseemuddin, M.; Anpalagan, A. Deep reinforcement learning based active queue management for iot networks. *J. Netw. Syst. Manag.* **2021**, *29*, 34. [CrossRef]
24. Do, H.-K.; Nguyen, A.-T.T.; Nguyen, T.-N.P. Evaluate the Performance of RED and DropTail Algorithm in NS2. *Int. J. Sci. Res.* **2019**, *8*, 1514–1517.
25. Patel, S.; Gupta, P.; Singh, G. Performance measure of Drop tail and RED algorithm. In Proceedings of the 2nd International Conference on Electronic Computer Technology, Kuala Lumpur, Malaysia, 7–10 May 2010; pp. 35–38. [CrossRef]
26. Chawla, J.; Kumari, S. Performance Evaluation of DropTail and Random Early Detection. *Int. Res. J. Eng. Technol.* **2016**, *3*, 721–727.
27. Beshley, M.; Kryvinska, N.; Seliuchenko, M.; Beshley, H.; Shakshuki, E.M.; Yasar, A.-U.-H. End-to-End QoS "smart queue" management algorithms and traffic prioritization mechanisms for narrow-band internet of things services in 4G/5G networks. *Sensors* **2020**, *20*, 2324. [CrossRef] [PubMed]
28. Sopin, E.; Begishev, V.; Moltchanov, D.; Samuylov, A. Resource queuing system with preemptive priority for performance analysis of 5g nr systems. In Proceedings of the International Conference on Distributed Computer and Communication Networks, Moscow, Russia, 14–18 September 2020; pp. 87–99. [CrossRef]
29. Abu-Shareha, A.A. Enhanced Random Early Detection using Responsive Congestion Indicators. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 358–367. [CrossRef]
30. Alhassan, M.S.E.; Hagras, H. A congestion control approach based on weighted random early detection and Type-2 fuzzy logic system. *Int. J. Comput. Sci. Trends Technol.* **2020**, *8*, 83–94. [CrossRef]
31. Xu, Y.; Qi, H.; Xu, T.; Hua, Q.; Yin, H.; Hua, G. Queue models for wireless sensor networks based on random early detection. *Peer-to-Peer Netw. Appl.* **2019**, *12*, 1539–1549. [CrossRef]
32. Xu, X.; Liu, B.; Zhang, L.; Mao, Y.; Wu, X.; Ren, J.; Han, S.; Jiang, L.; Xin, X. Self-adaptive Bandwidth Scheduling based on Improved Random Early Detection for NG-PON. In Proceedings of the 18th International Conference on Optical Communications and Networks (ICOCN), Huangshan, China, 5–8 August 2019; pp. 1–3. [CrossRef]
33. Monisha, V.; Ranganayaki, T. A Service Differentiation Aware Dynamic Random Early Detection and Optimized Fuzzy Proportional Integral Derivative for Active Queue Management Congestion Control in Mobile Wireless Sensor Network. *Int. J. Comput. Sci. Netw.* **2018**, *16*, 30–40.
34. Guan, L.; Woodward, M.; Awan, I. A Discrete-Time Performance Model for Congestion Control Based on Random Early Detection Using Queue Thresholds. *Int. J. Simul. Syst. Sci. Technol.* **2021**, *22*, 47–55. [CrossRef]
35. Pan, C.; Zhang, S.; Zhao, C.; Shi, H.; Kong, Z.; Cui, X. A Novel Active Queue Management Algorithm Based on Average Queue Length Change Rate. *IEEE Access* **2022**, *10*, 75558–75570. [CrossRef]
36. Kar, S.; Alt, B.; Koeppl, H.; Rizk, A. PAQMAN: A Principled Approach to Active Queue Management. *arXiv* **2022**, arXiv:2202.10352.
37. Rezaee, A.A.; Pasandideh, F. A fuzzy congestion control protocol based on active queue management in wireless sensor networks with medical applications. *Wirel. Pers. Commun.* **2018**, *98*, 815–842. [CrossRef]
38. Danladi, S.B.; Ambursa, F.U. DyRED: An enhanced random early detection based on a new adaptive congestion control. In Proceedings of the 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 10–12 December 2019; pp. 1–5. [CrossRef]
39. Ismail, A.H.; El-Sayed, A.; Elsaghir, Z.; Morsi, I.Z. Enhanced random early detection (ENRED). *Int. J. Comput. Appl.* **2014**, *92*, 20–24. [CrossRef]

40. Hamidian, H.; Beheshti, M.T. A robust fractional-order PID controller design based on active queue management for TCP network. *Int. J. Syst. Sci.* **2018**, *49*, 211–216. [CrossRef]
41. Hassan, S.O.; Nwaocha, V.O.; Rufai, A.U.; Odule, T.J.; Enem, T.A.; Ogundele, L.A.; Usman, S.A. Random early detection-quadratic linear: An enhanced active queue management algorithm. *Bull. Electr. Eng. Inform.* **2022**, *11*, 2262–2272. [CrossRef]
42. Khan, M.F.; Das, I. Analysis of Various Mobility Models and Their Impact on QoS in MANET. In *Computationally Intelligent Systems and their Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 131–141. [CrossRef]
43. Chen, J.; Hu, K.; Wang, Q.; Sun, Y.; Shi, Z.; He, S. Narrowband internet of things: Implementations and applications. *IEEE Internet Things J.* **2017**, *4*, 2309–2314. [CrossRef]
44. Ali, S.H.; Nasir, S.A.; Qazi, S. Impact of router buffer size on TCP/UDP performance. In Proceedings of the 3rd IEEE International Conference on Computer, Control and Communication (IC4), Karachi, Pakistan, 25–26 September 2013; pp. 1–6. [CrossRef]
45. Lin, D.; Morris, R. Dynamics of random early detection. In Proceedings of the ACM SIGCOMM'97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Cannes, France, 14–18 September 1997; pp. 127–137. [CrossRef]
46. Sahu, P.K.; Acharya, B.M.; Panda, N. QoS-Aware Unicasting Hybrid Routing Protocol for MANETs. In *Intelligent and Cloud Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 631–640. [CrossRef]