Performance Evaluation of Active Queue Management Algorithms in Large Network

Mustafa Maad Hamdi[1], Sami AbdulJabbar Rashid[2], Mahamod Ismail[2], Mohammed A. Altahrawi[2], Mohd Fais Mansor[2], and Mohammed K. AbuFoul[3]
[1]Dept. of Computer Engineering Techinques,Al-Maarif University College, Iraq
[2]Centre of Advanced Electronic and Communication Engineering (PAKET)
Universiti Kebangsaan Malaysia, 43600 UKM Bangi Selangor, MALAYSIA
[3]Dept. of computer Technology and Idustrial Professions, University College of Applied Science, 1415 Gaza, PALESTINE
meng.mustafa@yahoo.com, sami25.6.1989@gmail.com, {mahamod, m.mansor@ukm.edu.my}, mohatahrawi@gmail.com, mkabufoul@gmail.com

***Abstract*— The large network architecture of today consists of thousands of computers connected together through many interconnecting router and switches. As many of them communicate concurrently, congestion over the channels may increase. In order to maintain the fairness among the transmission control protocol (TCP) based packets being sent, during congestion each flow of communication must be treated fairly. The most popular scheduling scheme implemented today is the drop-tail, which it cannot guarantee fairness and as the queues get full; the latency increases. Therefore, packet drops cause reduction in throughput of those flows. Research is ongoing to find an Active Queue Management (AQM) scheme that can solve this fairness issue, though fairness is not the only important feature that is looked on. Latency is another feature, that is important as well as the packet loss ratio. In this paper, the performance of network metrics between the drop-tail and the behavior of different Active queue management schemes (PFIFO, RED, ARED, CoDel) in large networks is evaluated using the open source discrete network simulator (NS-3). The results show that the throughput of CoDel and RED is the highest with the minimum latency. Moreover, the simulation results show also that Adaptive Active Queue Management (AAQM) provides postponing congestion and high link utilization whilst maintaining packet loss. The proposed scenario could be extended to evaluate real large network. The results prove the RED and CoDel have the best throughput and less latency comparing to others. Moreover, the ARED presents better performance when it compared to RED.**

Keywords—Queue managment; RED; network; NS-3.

I. INTRODUCTION

Congestion occurs in different types of networks, be it LAN, WAN, MAN. As long as data is transferred within hosts in a network, congestion can happen. Large networks are no exceptions to this problem; in which tens to hundreds of networks nods are interconnected together in one physical communicating manner [1] [11]. With these many hosts, there is bound to be network congestion in such a network. There are significant methods to reduce congestion, either by congestion control or congestion avoidance. This paper comes to evaluate the performance of Drop-tail and different active queue management schemes such as packet first in first out (PFIFO), controlled delay (CoDel), random early detection (RED), and adaptive early detection (ARED). The current router queue management is mainly classified into the two-major classification [2]:

1. Passive Queue Management (PQM)
2. Active Queue Management (AQM).

1 Drop tail

Customary Internet routers employ the Tail Drop feature for managing the buffer queue occupancy [3]. It essentially adjusts a maximum length for each buffer queue and it enqueues packets until the greatest length is achieved, then drops subsequent incoming packets until the queue has decreased below its maximum value as depicted in Figure 1.

Such a mechanism permits the router to keep up high queue inhabitants, which is clearly undesirable since it must discriminate against burst traffic and to drop many packets at the same time producing global synchronization of sources. [4]. When the queue fills, the arriving packet is disposed of.

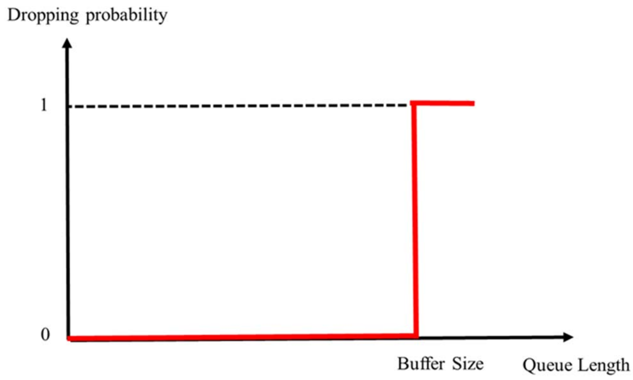

Fig.1 Dropping Probability of Droptail.

2 Random Early Detection (RED)

RED is an AQM algorithm that was created over a two decades ago to solve the unfairness issue, global synchronization problems, buffer bloat raised in DropTail and is used for congestion avoidance in switches [5]. RED monitors the average queue size in the switch and when a present threshold of number of packets in the queue has been surpassed, it starts to mark all incoming packets based on probability as illustrated in Figure 2; which depends on the average queue size; and drops or marks packets based on the type of transport layer protocol.

The dropping of packets start before a queue fills up so buffer bloat does not occur [6]. The equation is

$$X= (1-w) *a + w*qz \quad (1)$$

X= the average queue size.
qz= the instantaneous queue size.
w = the weight of the queue.

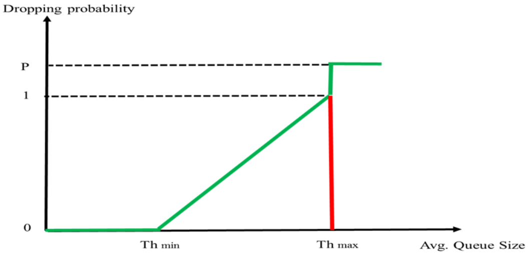

Fig 2. Dropping Probability of RED.

3. Controlled Delay

RED queuing has a problem with it has been needing its parameters set for different sizes of network bandwidths for it to work well, CoDel solves this. It works well with any range of bandwidth size without it needing its settings to be modified [7]. CoDel does not use the size of the queue to manage the queue, it uses the time spent in which packets are buffered called queue time. It sets a queue time called target and cannot drop packets when the queue time is below this set target. CoDel also has a time called interval which is the time in situations of worst-case RTT of connections in the queue. If a packet exceeds target for an at least an interval amount of time, it is dropped [8].

CoDel has periods of good queue and bad queue; where good queue goes drains under one RTT, but bad queue has delays and queues persist over several RTTs. CoDel treats both types of queues differently; it does not have to bother about dropping packets, but during bad queues, CoDel has to react by dropping packets [9].

4. Adaptive random early detection (ARED)

Its improved version of RED. The main features of ARED are [10]:

1. It used the congestion measure function to evaluate the degree of congestion, since it measures returnable value to indicate a proper congestion value of the channel.
2. It used feedback function based on congestion value that calculate how much feedback should be sent to the end system.
3. It implements the congestion feedback action based on the congestion feedback value to determine what action will be taken with the probability calculated by the congestion feedback function.

II. METHODOLOGY

The first main scenario is many clients send packets to one server, so it's called it "Many to one scenario". The second one is many to many where many clients send to many servers. The main procedures to build and simulate the two proposed scenarios are:

1. Decide the number of clients, switches, routers, and server that will be installed in the proposed design.
2. Build the proposed scenarios based on step number one.
3. Write the NS-3 code, and adjust all parameters related.
4. Simulate the NS-code
5. Verifying the results based on the concepts. Adjust the code needed.
6. Using the Microsoft Excel and Plot-Gun softwares, analysis the results and draw the related graphs.

The figure 3 shows the overall methodology of this study

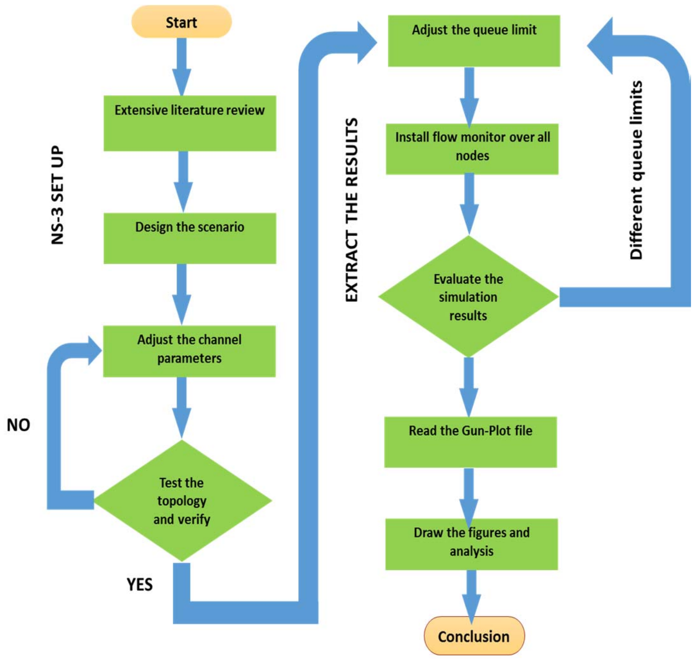

Fig.3 The main methodology

A) Many to one scenario

The first scenario has eight clients and 4 servers, the connection paths come from three clients to one server as shown in Figure 4 below.

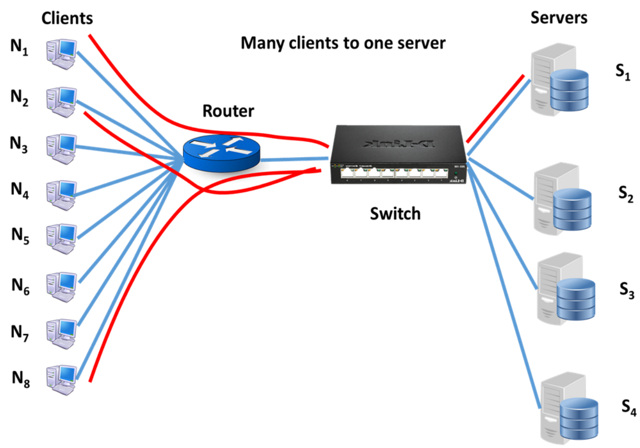

Fig.4 One to many scenario

Parameters	Setting/ value
Total nodes	14
# of clients	8
# of router	1
# of switch	1
# of servers	4
Channel Bandwidth	1.5Mbps
Channel delay	20ms
Routing type	NixVector
Pfifo Queue Limit	1000 Packets
RED Queue Limit	1000 packets
RED MIN threshold	20
RED MAX threshold	80
Application Type	ON/OFF
Simulation time	20 S
Data size	512 bytes

Table 1 Simulation Parameters for the first scenario.

B) Many to Many scenarios

The proposed scenario aims to send different packets from multi clients over different channel properties. The scenario is figured out in Figure 5, and scenario parameters are presented in Table 2.

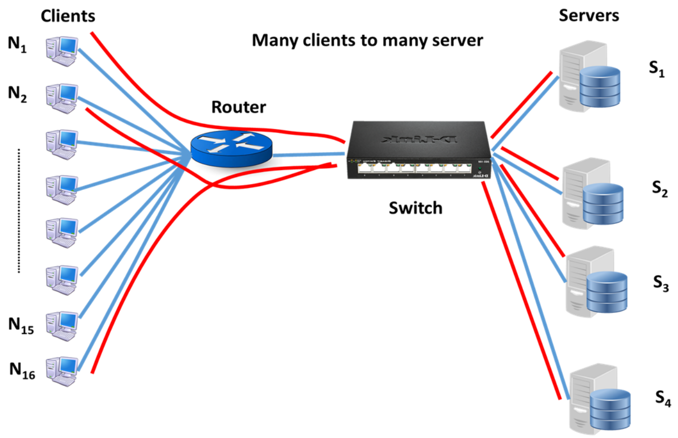

Fig.5 Many to many scenario

Parameters	Setting/ value
Total nodes	25
# of clients	16
# of router	4
# of switch	1
# of servers	4
Channel Bandwidth	1.5 Mbps
Channel delay	20 ms
Routing type	NixVector
Pfifo Queue Limit	1000 Packets
RED Queue Limit	1000 packets
RED MIN threshold	20
RED MAX threshold	80
Application Type	ON/OFF
Simulation time	20 s
Data size	512 bytes

Table 2 Simulation Parameters for the second scenario.

III. RESULTS AND DISCUSSION

This section focusses on presenting all figures related to queue management schemes like DropTail, CodDel, pfifo, and RED, and comparing between them in two different scenarios; one from multi to one scenario; the other is multi to multi scenario. Network performance metrics like throughput, end to end delay, and packet loss ratio has been evaluated over these two scenarios.

The scope of the paper focus on types of active queue management, so RED and new queue scheme ARED has been investigated also regarding to many aspects. All scenarios are simulated by means of discreet event network simulator NS-3.

A. MANY-TO–ONE SCENARIO

The money to one scenario is a common scenario in the daily life, since many clients can send request to one server. Such as different user tries to establish connection from their devices to www.google .com server. In the following sessions, the performance of network metrics regarding to the throughput, latency, and packet loss ratio is presented.

1) Aggregate Throughput

From the scenario's figure 4 the client N0 and N1 are sending different packets over the TCP connections to the server N4. The queue management is settled respectively to RED, DropTail, CodDel, pfifo, and all network metrics are evaluated. The total amount of throughput that was earned during the network simulation is the aggregate throughput. Hence, every flow's throughput is accumulated in order to evaluate them.

When comparing the four schemes as shown in Figure 6, CoDel shows clearly the highest aggregate throughput around 1220 kbps. The RED still has roughly the same throughput as the previous. The pfifo and Droptail have the worst aggregate throughput. Although DropTail has better throughput, but it considers unfair one. The Droptail and pfifo have about 700 kbps in serial.

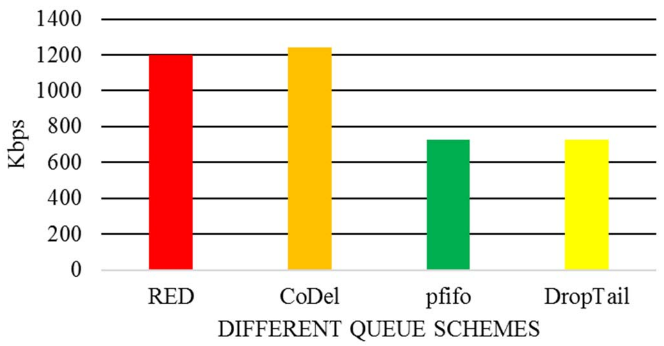

Fig 6 The throughput of the first scenario.

2) *Packet loss ratio (PLR)*

As discussed earlier, the packet loss ratio shows how the packets dropped or losses during the transferring of packets from source to destination. From Figure 7, the queue schemes RED and CoDel show very little packet loss ratio comparing to the others. RED has 8% and CoDel has 6%. The Droptail and pfifo have the highest value, so they are not suitable for priority consideration both has 32% of PLR.

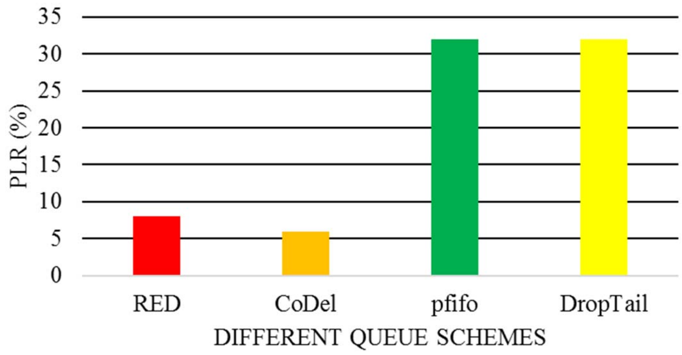

Fig.7 The PLR of the first scenario.

3) *End- End Delay*

The end to end delay or the latency shows the sum of delay over all transitions from source to destination. From Figure 8, the CoDel has a very low end to end delay which is considered as one of the advantages of CoDel. It's just 100 µs. The RED performs the same.

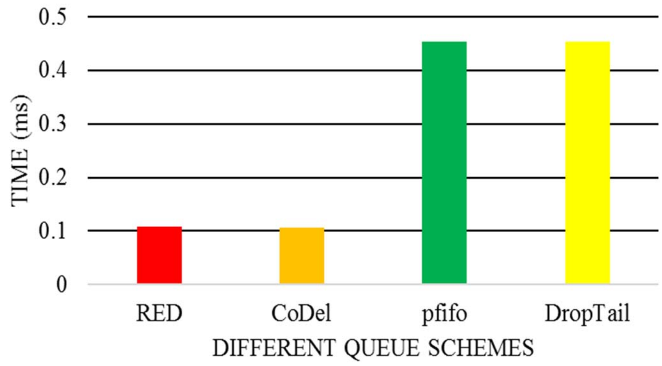

Fig.8 The Latency of the first scenario.

On the other hand, both pfifo and Drop Tail delivers high latency and not suitable for emergency and high-speed network. The latency of pfifo is probably due to managing the priority of the packets (the processing delay). Drop Tail comes from high congestion over the channel.

B. *MANY-TO–MANY SCENARIO*

The scenario is many to many, hence N0, N4, N13 are sending packets over their channels to multi server in three different cases as discussed below.

Case 1: N0 send to server N11 and N12.

Case 2: N4 send to server N10 and N9.

Case 3: N13 send to server N11, N12, N10 and N9 (All servers).

All performance metrics will be evaluated in this proposed scenario to show which queue management introduced the best regarding to network throughput, latency, and packet loss ratio.

1) *Aggregate Throughput*

From Figure 9, the RED throughput shows very high value, it's around 11 Mbps, so its clear, RED queue management is the best for many to many networks. The CoDel stand in the second rank; it's around 9.3 Mbps. Both pfifo and Droptail has roughly the same value. Droptail as mentioned before is unfair throughput.

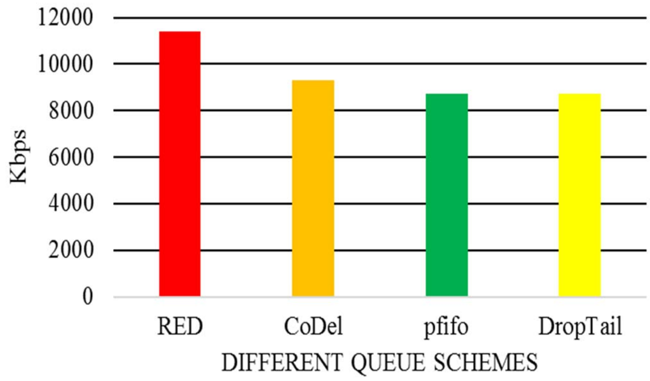

Fig.9 The throughput of the second scenario.

2) *Packet loss ratio (PLR)*

From Figure 10, the Random early detection algorithm, RED shows high packet losses, the RED parameters needs to be adjusted and calibrated. RED has 28% PLR. The CoDel clearly presented very low packet loss ratio comparing to others, it's around 14% PLR, since no adaptation required for it, the PLR truly appeared. Both pfifo and Droptail has 25% PLR as normally expected. Pfifo presented high PLR according to its mechanism which high priority packets can deliver faster than lower ones.

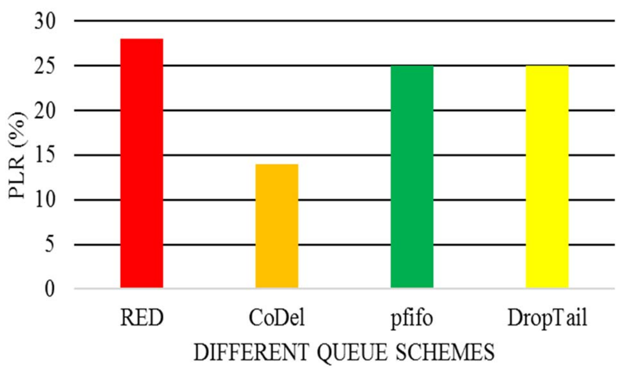

Fig.10 The PLR of the second scenario.

3) *End- End Delay*

The delay is an important metric in the network. From Figure 11, the CoDel has a very low end to end delay which is considered as one of the advantages of CoDel. It's just 100 µs. The RED shows the same. On the other hand, both pfifo and DropTail shows high latency, thus they are not suitable for emergency and high-speed network. The latency of pfifo is probably due to managing the priority of the packets (the processing delay). For Droptail comes from high congestion over the channel.

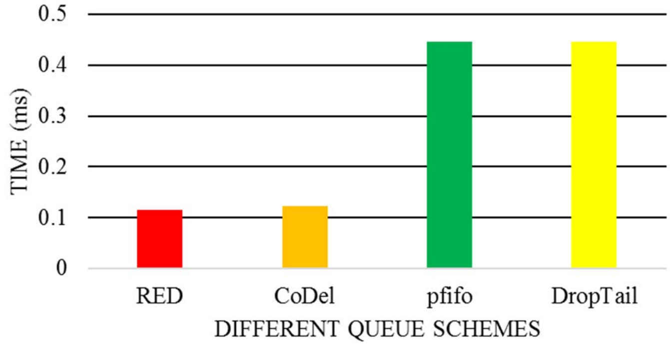

Fig.11 The latency of the second scenario.

C. *RED VERSUS ARED*

Adaptive RED focuses on the problem of parameterizing the RED algorithm in order to reach perfect performance over the network. So, it adjusts its parameters regarding the channel queue limit capacity base on the average of queue size as mentioned before. The following sessions evaluate the performance of RED vs ARED to investigate its performance over the channel regarding to throughput, latency, and packet loss ratio. The behaviour through the simulation time of both is figured also.

1) *Actual Queue Size versus Time*

The ARED serve the link utilization, better than RED by mean of packet loss rate and queue size. Figure 12 below shows RED algorithm results and ARED algorithm. Results. It's clear that ARED has lower queue variation than RED algorithm results.

This means ARED algorithm introduces a better channel utilization and create lower queuing delays. But the extracted average queue size of both RED and ARED grows, slower than actual queue size. The algorithms response of AQM algorithms is slow, as it's detected from the slow variation of the estimated queue length. This phenomenon shows up in each queue size based AQM algorithms.

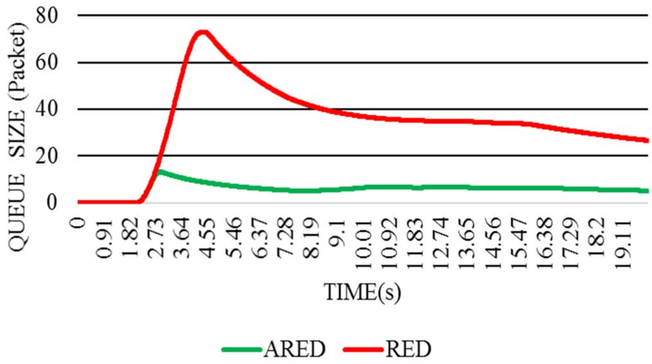

Fig.12. RED, RED actual queue versus simulation time.

2) *The average throughput*

As it is shown from the figure 13, as the queue limit increases the throughput increase slightly, when the queue limit was 20 queues, the throughput was around 600 and 700 for RED and ARED respectively. While when it reaches 140 queue limit size, the throughput for both of them become roughly above 820. When comparing RED with ARED regarding to throughput, ARED show higher than RED, which mean making better channel utilization.

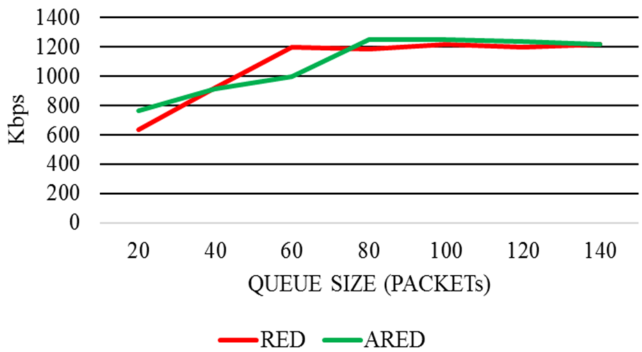

Fig.13 The throughput of the RED vs ARED scenario.

3) *Packet loss ratio (PLR)*

One of the most network metrics is the packet loss ratio which is equal to one minus packet delivery ratio PLR (1-PDR). The PLR shows how the packet losses vary during the simulation. From the figure 14, its clear the packet loss ratio has different values over different queue limits. Moreover, as the queue limit is increased the packet loss ratio decrease, which mean better network performance. And for large networks, utilization of big queue size over the channel is needed.

From the graph, when the queue size was 20 queue limits, the PLR was 18% and 16 for both RED, and ARED respectively. Nevertheless, when the queue, max limit is increased to 140 queue sizes, it became very much less below 8%, which mean serve the network performance better.

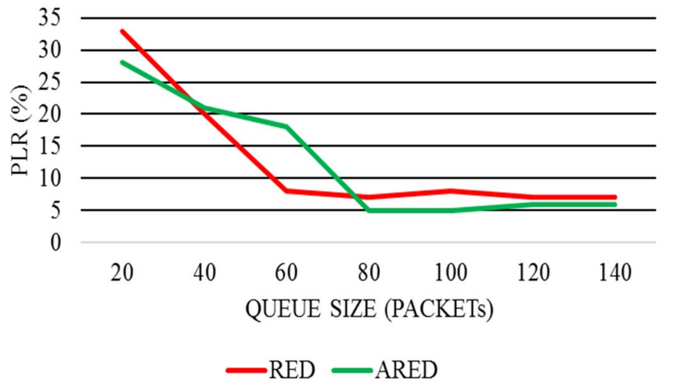

Fig.14 The PLR of the RED vs ARED scenario.

*4) **End -End delay***

It shows how the packet delivers from the source to destination regarding to the time required. From the figure 15, it's clearly shown, the RED has a higher delay comparing to RED. Since RED queue algorithms can adapt its mechanism to reduce the queue over the network channel, which mean less delay. Both of them presented low delay over all proposed queue sizes. As a sample, when the queue size was 20, the End- End delay was 60 µs, 80 µs respectively.

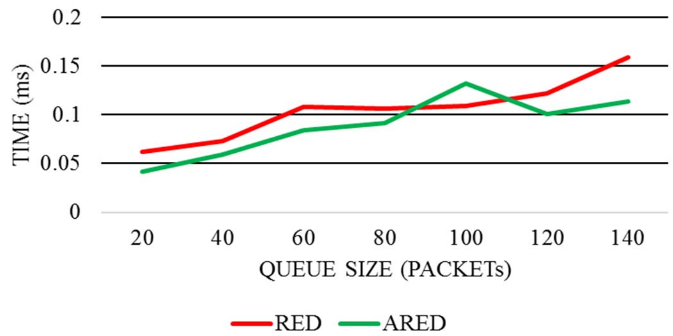

Fig.15 The Latency of the RED vs ARED scenario.

IV. CONCLUSION

In this paper, a simulated performance is successfully evaluated that is able to simulate a large network. This simulation model could be able to apply different AQM schemes and generate their output files that can be used to investigate the performance of the network regarding to throughput, latency, packet loss ratio (PLR). The Droptail, RED, CoDel, and PFIFO AQM schemes are simulated using discrete event simulator NS-3. The results of RED and CoDel show significantly better throughput with less packet dropping ratio comparing to the PFIFO and the Droptail for the two proposed scenarios. Moreover, the RED versus ARED also investigated to come up with the advances of ARED and its features. The results show the RED queue algorithm and code has better performance in the two proposed scenarios, many to one and many to many. The ARED has been designed to adapt the RED parameters based on the congestion over the channel, so its figures are presented high performance comparing to the RED algorithms. The proposed system preserves expensive resources cost during large network emulation because everything is virtual and does not include the use of real servers.

REFERENCES

[1] Athuraliya, S., Low, S. H., Li, V. H. & Yin, Q. 2001. Rem: Active Queue Management. IEEE Network 15(3): 48-53.

[2] Baklizi, M. & Ababneh, J. 2016. Performance Evaluation of the Proposed Enhanced Adaptive Gentle Random Early Detection Algorithm in Congestion Situations. International Journal of Current Engineering and Technology 6(5):1658-1664.

[3] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G. & Partridge, C. 1998. Recommendations on Queue Management and Congestion Avoidance in the Internet. IETF RFC 2309. pp. 1-17.

[4] Chavan, S., Malangadan, N. & Raina, G. 2016. TCP with Virtual Queue Management Policies: Stability and Bifurcation Analysis. IEEE/ACM Transactions on Networking 25(2):1020-1033.

[5] Chawla, J. & Kumari, S. 2016. Performance Evaluation of DropTail and Random Early Detection. International Research Journal of Engineering and Technology 3(5):721-727.

[6] Dijkstra, S., Haverkort, B., De Boer, P. & Van Foreest, N. 2004. Modeling Active Queue Management Algorithms Using Stochastic Petri Nets. Faculty of Electrical Engineering, Mathematical and Computer Science, University of Twente. Master Thesis.

[7] La, R. J., Ranjan, P. & Abed, E. H. 2003. Analysis of Adaptive Random Early Detection (ARED). Teletraffic Science and Engineering 5(2003):1271-1280.

[8] Low, S. H. 2003. A Duality Model of Tcp and Queue Management Algorithms. IEEE/ACM Transactions on Networking 11(4): 525-536.

[9] Narsimhulu, G. & Rao, D. S. 2016. Random Early Detection-Techniques for Differentiated Services Networks Based on Packet Droppings. PARIPEX-Indian Journal of Research 4(5):283-286.

[10] Nguyen, T. A. N., Gangadhar, S. & Sterbenz, J. P. 2016. Performance Evaluation of Tcp Congestion Control Algorithms in Data Center Networks. Proceedings of the 11th International Conference on Future Internet Technologies, pp. 21-28.

[11] Al Chaab, W., Ismail, M., Altahrawi, M. A., Mahdi, H., & Ramli, N. (2017, November). Efficient rate adaptation algorithm in high-dense vehicular ad hoc network. In Communications (MICC), 2017 IEEE 13th Malaysia International Conference on (pp. 23-28).