




Multi-Criteria Priority RED Queuing (MCPRQ): A novel approach to enhance queue management and quality of service in IPv6 networks

Abdullah Orman* 

Ankara Yildirim Beyazit University, Department of Computer Technologies, aorman@aybu.edu.tr

Çetin Elmas 

Gazi University, Department of Electrical - Electronic Engineering, celmas@gazi.edu.tr

İnan Güler 

Gazi University, Department of Electrical - Electronic Engineering, iguler@gazi.edu.tr

Submitted: 01.10.2024

Accepted: 01.11.2024

Published: 31.12.2024



* Corresponding Author

Abstract:

This study aims to develop a new DiffServ queue model and AQM (Active Queue Management) model to improve the quality of service in real-time internet applications. This model, called MCPRQ (Multi-Criteria Priority RED Queuing), aims to provide more effective queue management by evaluating packets according to their priority levels, sizes, and waiting times within the scope of the DiffServ architecture. This evaluation is performed using the Analytical Hierarchy Process (AHP) and integrated with the RED (Random Early Detection) algorithm to provide a solution to increase the quality of service in queue management. The MCPRQ model has been tested with the OMNeT++ simulator in IPv6 networks and has achieved successful results compared to commonly used queue structures. Its low packet loss has attracted attention, especially in low-density networks, and low average delay in high-density networks. This shows that MCPRQ offers a significant advantage in flexibility and scalability. As a result, the MCPRQ model effectively manages congestion in medium and high-density networks while providing better performance by preserving the quality of service in real-time applications.

Keywords: AHP, Differentiated services, IPv6, Quality of service, Queuing model

© 2024 Published by peer-reviewed open access scientific journal, Computers and Informatics (C&I) at DergiPark (dergipark.org.tr/ci)

Cite this paper as: Orman, A., Elmas, Ç., & Güler, İ. RED Queuing (MCPRQ): A novel approach to enhance queue management and quality of service in IPv6 networks. *Computers and Informatics*, 2024; 4(2); 99-111, <https://doi.org/10.62189/ci.1558975>

1. INTRODUCTION

Today, many real-time applications over the Internet have significantly increased bandwidth requirements and quality of service (QoS) expectations. Applications such as IP-based phones, IPTV, internet radio broadcasts, and video conferencing systems demand high bandwidth, low latency, and minimum packet loss. These needs necessitate implementing advanced solutions to optimize data transmission performance in modern network infrastructures. However, congestion, packet loss, and delay variations (jitter) experienced in networks can reduce the quality of these services [1].

QoS is applied to efficient bandwidth use and network performance improvement, especially in applications sensitive to delay and packet loss [2-4]. The differences between protocols such as TCP and UDP necessitate the selection of appropriate strategies for QoS management. In particular, TCP's packet loss retransmission features and UDP's low latency tolerance require separate management of different traffic.

Congestion problems in network traffic are usually caused by router queues filling up and packets being dropped from these queues. Standard drop tail management is insufficient to solve this problem and causes global synchronization problems in the network. At this point, Active Queue Management (AQM) techniques have been developed to control congestion before it starts. RED (Random Early Detection), one of the AQM algorithms, is designed to detect congestion signs in advance by keeping the average queue length within a specific range to prevent queues from filling up[5].

In addition, QoS methods such as DiffServ (Differentiated Services) and IntServ (Integrated Services) offer solutions based on packet prioritization and bandwidth reservation. However, DiffServ methods are insufficient to solve the congestion problem, so they must be integrated with AQM algorithms [6-9].

Recent studies have focused on developing AQM and QoS methods [4, 10-14]. Artificial intelligence-based solutions, intense reinforcement learning (DRL), and in-network telemetry propose new algorithms to meet QoS demands in real-time applications [5, 9, 15-17]. New algorithms such as DESiRED increase performance and minimize congestion problems by dynamically responding to network traffic density [18].

This study investigated the effects of MCPRQ (Multi-Criteria Priority RED Queuing) algorithms developed with DiffServ and AQM methods on QoS. The developed methods were optimized for IPv6 networks and subjected to performance tests in different traffic scenarios. This research shows that new-generation algorithms for optimizing network performance improve essential parameters such as bandwidth and delay.

2. MATERIALS AND METHODS

The algorithm we developed includes both AQM and QoS components. A new packet classification, active queue management, and packet selection algorithm are proposed for QoS. The developed algorithm (MCPRQ) is compared with the Drop tail and RED AQM algorithms, while PQ and MCPQ are compared with QoS algorithms. The comparison is made in the OMNeT++ network simulator under three different scenarios.

2.1. Packet Classification Algorithm

In PQ's packet selection, only packet priority (IPv4: ToS, IPv6: dscp) is taken into account, and packets to be transferred to the output are selected according to the priority order of the queues. However, in the case of high-density traffic, this method causes the queue length to increase and high latency times to occur. RED, conversely, ensures that the queue length remains constant by dropping packets randomly selected from the queue before there is any congestion. In this method we developed, packets are taken into four different virtual queues created at the beginning, and a selection is made among the packets waiting in this queue. Unlike the PQ algorithm, packet priority (dscp value), packet size, and waiting time in the queue are considered. Again, unlike PQ, the length of the created virtual priority queues is not fixed, and they are updated instantly according to the queue density. The RED application is applied only to the queue with the lowest priority value from these virtual queues. Here, the queues created according to the DSCP value of the packets are given in Table 1 [12].

Table 1 MCPRQ Packet Classification[12]

Category	DSCP Range
Expedited Forwarding (EF)	dscp ≥46
High Queue (HQ)	27 ≤ dscp <46
Low Queue (LQ)	16 ≤ dscp <27
Best Effort Data (BE)+RED	0 ≤ dscp <16

The DSCP value of 46 (EF) indicates that the packet is a voice packet and is the most critical packet. The HQ queue receives real-time application packets with two-way communication (like video conferencing). The LQ queue receives packets with one-way real-time communication (IPTV and Video Stream). The BE queue receives all the remaining packets (such as FTP, SMTP, HTTP, etc.). If a packet has not been assigned a DSCP value, it is still in this queue. The capacities allocated to these queues are given in Equation 1 [12].

$$L_{EF} = L_{HQ} = L_{LQ} = L_T/n \tag{1}$$

$$L_{BE} = \%85 * L_T - (UL_{EF} + UL_{HQ} + UL_{LQ})$$

Here, L_{EF}, L_{HQ} , queue capacity L_T : shows the total queue length, n : the number of queues, and the currently used quantities of the queues $UL_{EF}, UL_{HQ}, UL_{LQ}$.

25% of the total queue length is allocated to EF, HQ, and LQ queues, and 85% of the total idle queue length is assigned to the BE queue. In other words, the capacity of the BE queue is not fixed and is constantly changing. One of the reasons for this distinction is to use the total queue capacity efficiently, and the other is that in regular traffic, the amount of packets called "Best Effort Data" is much higher than all other packet classes. The reason for not assigning the entire idle queue to the BE queue is to prevent a new high-priority packet from being dropped when all virtual queues are full. However, when congestion increases, there is a possibility that the queue will fill up and high-priority packets will be dropped. To prevent this, the RED algorithm is applied to the BE queue to keep the length fixed and ensure that the dropped packets are low-priority. The general structure of MCPRQ is shown in Figure 1.

2.2. MCPRQ Active Queue Management Algorithm

Considering the weaknesses in AQM applications, the packets to which AQM will be applied should be carefully selected. AQM algorithms are packet-dropping algorithms created to prevent congestion. According to the quality of service rules, critical packets must reach the target. For this reason, packets with high importance should not be dropped. The MCPRQ algorithm constantly monitors the total queue length, and congestion is detected in advance with RED. However, the packets to be dropped are selected only from the packets in the BE queue. In this way, five critical problems are solved. These are:

- Since dropped packets are selected from packets with low importance, QoS will increase.
- Since dropped packets are more likely to be TCP packets, the source will regenerate them.
- Global synchronization will be avoided. Since there is no congestion, the possibility of dropping critical packets without being queued will be eliminated.
- When packets in the BE queue, which will have a very high waiting time in the queue, are dropped, the total queue will be used more effectively.
- There is no need to apply any AQM to queues other than the BE queue. In these queues, the last packet will be dropped when the queue is full (Drop Tail).

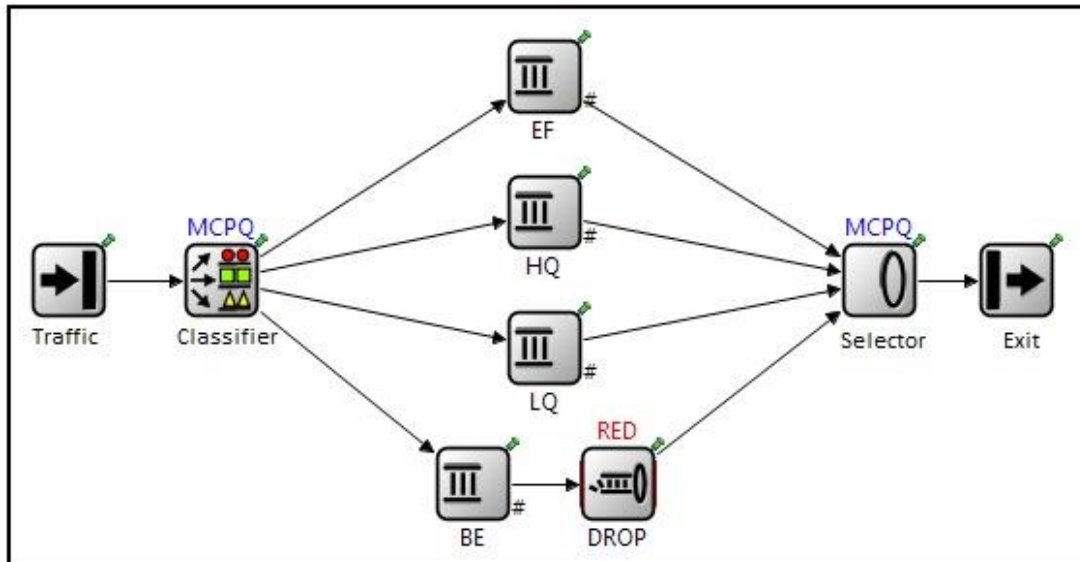


Figure 1. MCRPQ queue structure

```

avg = 0, count = -1, bufferUsed = 0
Calculate the average queue length of avg for each packet arriving in the BE queue
    if queue is not empty
        avg = (1 - wq)avg + wq * bufferUsed
    else
        m = f(time - qtime)
        avg = (1 - wq)m avg
    if minth ≤ avg ≤ maxth
        increase count value
        Calculate the probability of Pa
        Pb = maxp *  $\frac{avg - min_{th}}{max_{th} - min_{th}}$ 
        Pa = Pb / (1 - count * Pb)
        generate a random number
        if random number ≤ Pa probability
            count = 0
    yok eğer maxth ≤ avg
        mark the incoming packet
        count = 0
    else
        count = -1
        queue is empty when
        time = qtime
    if marked packet or packetSize + bufferUsed > Totalbuffer
        drop packet
    else
        add packet to queue
        hit timeStamp (add queued time to packet)
        bufferUsed = bufferUsed + packetSize
    
```

Figure 2. Pseudocode structure of RED applied to MCRPQ.

Here:

avg : Average queue length
q_time : Start of queue empty time
count : Number of packets after the last marked packet
wq : Queue weight
minth : Minimum threshold value for the queue
maxth : Maximum threshold value for the queue
maxp : *pa*Maximum value for
Pa : Current packet marking probability
q : Current queue length
time : Now
m = f(t): Linear function concerning time
bufferUsed: The number of queues currently in use

2.3. MCPRQ Packet Selection Algorithm

Unlike the PQ algorithm, the MCPQ algorithm considers not only the priority of the queue in which the packet is located but also the packet size and the waiting time of the packet in that queue when selecting the packets to be transmitted. The MCPRQ algorithm also uses the same packet selection algorithm as the MCPQ algorithm. A decision-making mechanism has been established to determine the effective rates of the packet priority, packet size, and packet waiting time criteria. The MCPRQ algorithm has also attempted to resolve the packet to be selected by using the analytical hierarchy process (AHP), one of the multi-criteria decision-making mechanisms [19]. As seen in the source MCPQ Equation 2, the packet priority, packet size, and packet priority weights have been calculated [12].

$$F_{EF,HQ,LQ,BE} = 0,49PP_N + 0,20PS_N + 0,31WT_N \quad (2)$$

The values of the weights found in Equation 3 are given after they are normalized. For normalization, the packet priority is divided by the highest dscp value of 63, and the packet size is divided by the largest packet size (MTU) of 1500. The packet duration that waits for the longest in virtual queues is accepted as WT_{max} .

$$\begin{aligned}
 F_{EF} &= 0,49PP_1/63 + 0,20PS_1/1500 + 0,31WT_1/WT_{max} \\
 F_{HQ} &= 0,49PP_2/63 + 0,20PS_2/1500 + 0,31WT_2/WT_{max} \\
 F_{LQ} &= 0,49PP_3/63 + 0,20PS_3/1500 + 0,31WT_3/WT_{max} \\
 F_{BE} &= 0,49PP_4/63 + 0,20PS_4/1500 + 0,31WT_4/WT_{max}
 \end{aligned} \quad (3)$$

```

n = 4,   Fmax = 0,   WTmax = 0,   F[1..n] = 0,   Qmax = 0,   WT = 0
For each queue (l, 1..n)
    If the queue is not empty
        Find the first element of the queue
        Calculate the queue delay WT (l) = CurretTime ()- TimeStamp ()
        If WT (l) > Wtmax
            Wtmax = WT (l)
For each queue (i, 1..n)
    If the queue is not empty
        Find the first element of the queue
        Find DSCP value →PP (i)
        Find packet size →PS (i)
        Calculate the queue delay WT (i) = CurretTime ()- TimeStamp ()
        F(i) = 0,49PPi/63 + 0,20PSi/1500 + 0,31WTi/WTmax
        If F(i) > Fmax
            Fmax =F(i)
            Qmax = i
Transfer the first packet from the Qmaxth queue to the output
Decrease the Qmaxth queue length by PS ( Qmax )
Reduce total queue length by PS ( Qmax ).
    
```

Figure 1Pseudocode of MCPQ packet selection algorithm

Here:

- n : Number of virtual queues
- WT : The amount of time the packet waits in the queue
- $F_{(i)}$: Multi-criteria decision-making function value
- F_{max} : The most significant value of the multi-criteria decision-making function
- WT_{max} : Maximum waiting time of packets in the queue
- Q_{max} : $F_{(i)}$ The queue number of the packet with the most significant value.

3. TESTING THE DEVELOPED ALGORITHM

DiffServ support has been given to applications, TCP, and UDP for all packets. Queue algorithms, statistics, and all capacities have been brought to a structure that calculates packet size. There are many DiffServ test environments built with different simulators. There are studies, especially on NS2 and OPNET [20].

In a real network, more packets belong to TCP applications than UDP applications. For this reason, HTTP, FTP, and SMTP, which can work properly in the test environment, have been adopted. Table 2 shows the established test environment applications and their features.

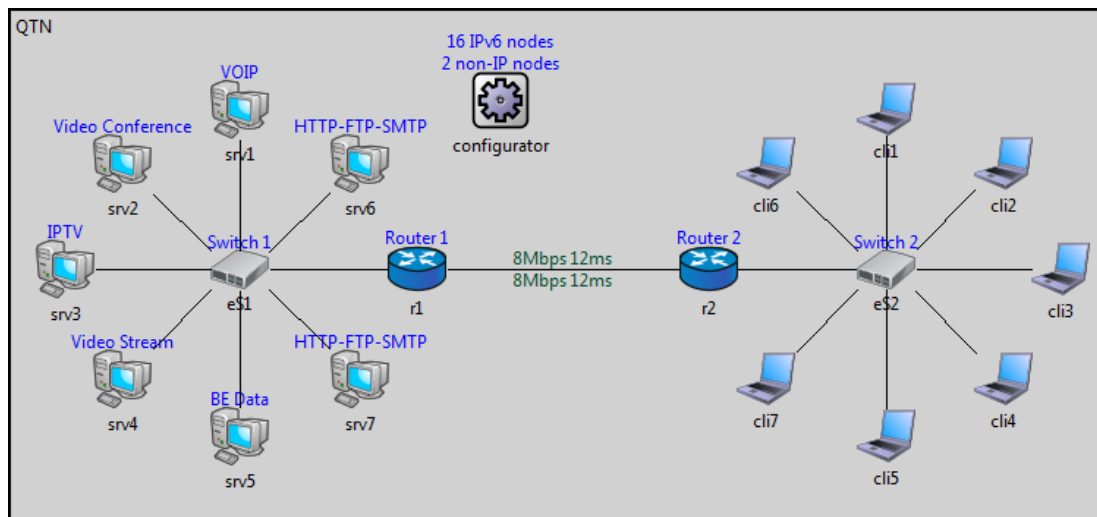


Figure 4. Established test environment

Table 2. Established test environment applications And Features

Application	Datagram	DSCP	Package Size	Production Period	Port	Connection
VOIP	UDP	46	256 B	10ms	2517	srv1↔cli1
Video Conference	UDP	34	1316 B	10ms	3247	srv2↔cli2
IPTV	UDP	32	512 B	20ms	1234	srv3↔cli3
Video Stream	UDP	26	1316 B	10ms	1558	srv4↔cli4
UDP BE Data	UDP	0	1032 B	10ms	6889	srv5↔cli5
HTTP	TCP	12	2024 B	-	80	All
FTP	TCP	11	2024 B	-	20	All
SMTP	TCP	10	2024 B	-	25	All

While two UDP applications run on servers from Sr1 to Sr5, three TCP (Http, Ftp, Smtp) run. Srv6 and srv7 only work on TCP applications created to generate background traffic.

Table 3. Total package sizes used in the simulation.

Package Type	Datagram	Total Packet Size (Byte)
VOIP	UDP	304
Video Conference	UDP	1364
IPTV	UDP	560
Video Stream	UDP	1364
UDP BE Data	UDP	1048
HTTP	TCP	1084
FTP	TCP	1084
SMTP	TCP	1084
ACK	TCP	72
SYN+ACK	TCP	76
FIN	TCP	60

In the developed test environment, MCPQ and MCPRQ algorithms are tested under different scenarios: Drop Tail, RED, and PQ queue structures were compared. The average end-to-end latency, end-to-end jitter, dropped packet amounts, total latency, router queue length, and instantaneous memory usage were used as comparison metrics.

Table 3. Test scenarios and used parameters

Test Name	Packet Loss	Inter-Router Connection	Maximum Queue Capacity	Switch – Node Connection	Simulation Duration
Scenario1	Low	8Mbps 12ms	32Kb	100Mbps 1ms	5sec-60sec
Senate2	High	6Mbps 16ms	25KB	100Mbps 1ms	5sec-60sec
Scenario3	None	10Mbps 8ms	40KB	100Mbps 1ms	5sec-60sec

Three different scenarios were created to compare the developed algorithm. These are the cases where there is queuing. Still, packet losses are low (Scenario 1), the cases where packet losses are high (Scenario 2), and the cases where there is no packet loss but only queuing (Scenario 3).

Scenario 1: It was created to examine the behavior of queuing algorithms and compare them in an environment where congestion occurs in the network, but packet losses are low.

The average queue length and the amount of memory used in the MCPRQ algorithm are shown. The RED algorithm is applied only to the virtual queue with no priority (BE) packets. In this scenario, the MCPRQ algorithm determines $min_{th}=12$ Kb and $max_{th}=24$ Kb. Although the average queue length is variable, it is measured at an average level of 15.83 Kb.

Figure 6 shows the packets and packet sizes dropped in the MCPRQ algorithm. The figure shows that almost all dropped packets are in the same priority group (BE queue where UDP BE Data, SMTP, FTP, and HTTP packets are located). Thanks to the RED algorithm applied to the BE queue, the queue length is kept in the min_{th} - max_{th} range, global synchronization is avoided, and since packet dropping is done only for unimportant packets, the quality of service is increased.

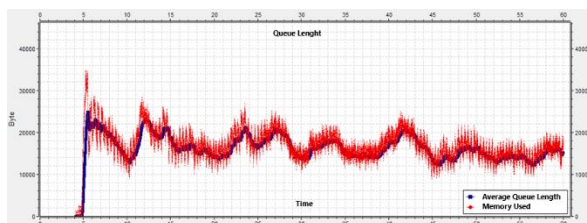


Figure 5. Scenario1 MCPRQ mean queue length

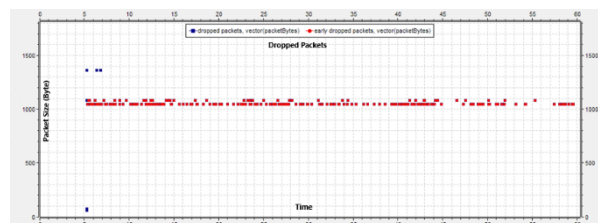


Figure 6. Scenario1 MCPRQ dropped package graph

Figure 7 shows the end-to-end delay graph with the MCPRQ queue algorithm. Since the priorities of the packets were taken into account and placed in different queues, different delays occurred. The fact that the queue length was not constant due to the RED algorithm caused the delays of BE data packets to be non-constant. While VOIP, Video conference IPTV, and Video stream applications are transmitted with a delay of 15-25 ms, UDP BE data packets are transmitted with a delay of 50-100 ms. Although the delay in BE data is relatively low according to the PQ and MCPQ algorithms, the change in delay (jitter) is higher than the others.

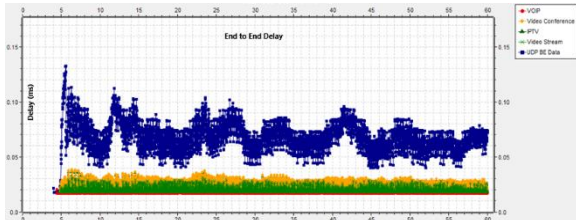


Figure 7. Scenario 1 MCPRQ the end-end delay graph

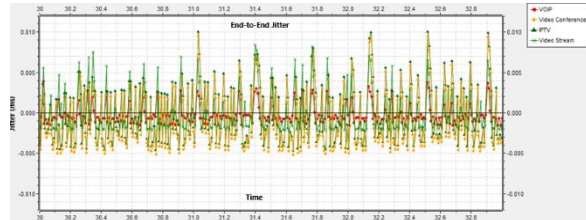


Figure 8. Scenario 1 MCPRQ end-end jitter graph

The MCPRQ algorithm also creates a very low jitter ($\sim \pm 2$ ms) in VOIP, while it creates a jitter of $\sim \pm 5$ ms for video conferencing, $\sim \pm 4$ ms for IPTV, and $\sim \pm 3$ ms for video stream packets. It consists of. A similar jitter graph was obtained with the MCPQ algorithm, except for tiny jumps. Although the MCPQ and MCPRQ algorithms have a worse jitter value than the PQ algorithm, they are pretty low compared to the allowed jitter values. It does not negatively affect the service quality.

Equation 4 calculates the processing speed of the transmitted packets in the router queue to examine the impact of all these algorithms on the router's performance.

$$\overline{Qs} = \frac{\sum Transmitted\ Packet}{\sum Delay\ Time} \tag{4}$$

Here, the Transmitted Packet is the sum of the sizes of the packets transferred from the queue to the exit. The Delay Time is expressed as the total waiting time for the packet to enter the queue.

Table 4. Router packet analysis for scenario 1.

Queue Type	Received Package (Kb)	Sent Package (Kb)	Dropped Package (Kb)	Total Delay (s)	Average Tail Length (Kb)	\overline{Qs} (Kb/s)
Drop Tail	54301	53537	733	2079.96	30.12	25.73
RED	54272	53539	730	1064.22	15.29	50.30
PQ	53747	53543	176	2222.48	29.11	24.09
MCPQ	53625	53545	55	1857.44	23.84	28.82
MCPRQ	53785	53543	227	1243.34	15.83	43.06

Table 4 shows that the lowest packet loss is obtained in the MCPQ algorithm, while the highest packet loss is obtained in the drop tail algorithm. While 32% less packet loss occurs in the MCPQ algorithm than in the PQ algorithm, the PQ algorithm also causes 22% more packet loss than the MCPRQ algorithm. Again, as seen in Table 4, RED has the fastest queue processing speed and is ranked as MCPRQ, MCPQ, and drop tail. PQ is measured as the slowest working queue.

Table 5. For Scenario 1, the average from the end-end delay durations (ms)

Queue Type	VOIP	Video Conference	IPTV	Video Stream	UDP BE Data
Drop Tail	48	49	48	49	48
RED	33	34	32	33	33
PQ	17	19	18	18	127
MCPQ	17	24	23	19	99
MCPRQ	18	23	22	20	68

The developed MCPRQ algorithm, like the MCPQ algorithm, considers packet priorities, packet size, and the waiting time for the packet in the queue. Therefore, the packets are not transferred according to the transmission order of the queues but according to the Fmax value. The packet in the queue with the most considerable Fmax value is transmitted. In the MCPRQ algorithm, unlike MCPQ, the queue length is tried to be kept in the minth-maxth range, and the dropped packets are selected from the BE queue with RED. While the packets in the priority queue other than VOIP are transmitted 2-4 ms later than the same PQ algorithm as MCPQ, the packets in the non-priority queue (BE) are transmitted ~30 ms earlier than MCPQ and ~60 ms earlier than PQ.

Scenario 2: This simulation environment was created to examine the behavior of queuing algorithms and compare them with each other in an environment where network congestion occurs and packet losses are high.

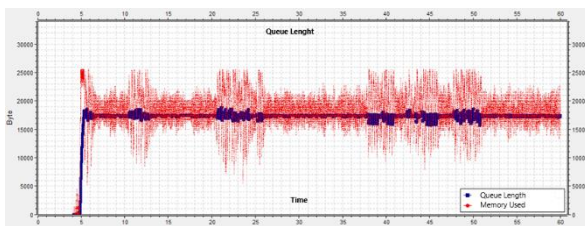


Figure 9. Scenario2 MCPRQ average tail length

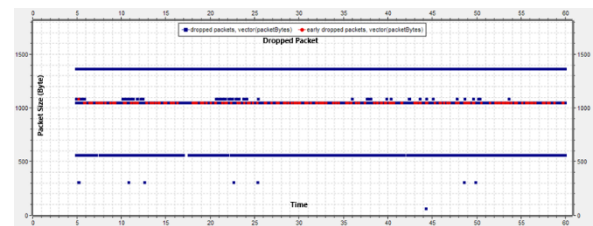


Figure 10. Scenario2 MCPRQ dropped packet graph

Figure 9 shows the average queue length and the amount of memory used in the MCPRQ algorithm. The RED algorithm in the MThe CPRQ algorithm is applied only to the virtual queue with no priority (BE) packets. In this scenario, the MCPRQ algorithm determines minth=8Kb and maxth=17Kb. Although the average queue length is variable, it is measured at an average of 16.71 Kb.

Figure 10 shows the graph of packets dropped in the MCPRQ algorithm and packet sizes. Almost all packets dropped with RED consist of packets in the same priority group (BE queue where UDP BE Data, SMTP, FTP, and HTTP packets are located). However, due to excessive traffic density, there were losses in video conference, IPTV, and video stream packets. Thanks to the RED algorithm applied to the BE queue, the queue length is kept in the minth-maxth range, global synchronization is avoided, and since packet dropping is done only from unimportant packets, the service quality is increased.

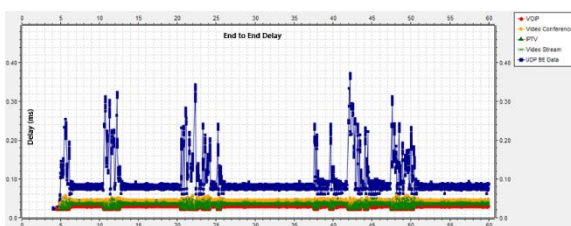


Figure 11. Scenario2 MCPRQ end-to-end delay graph

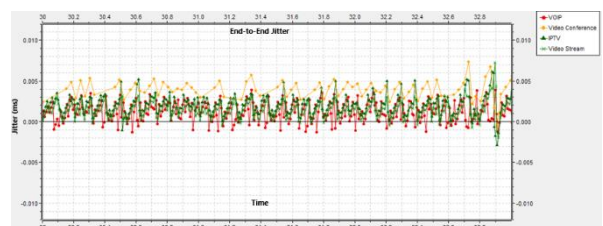


Figure 12. Scenario2 MCPRQ end-to-end jitter graph

The end-to-end delay graph is shown using the MCPRQ queue algorithm. Different delays occurred since the packets' priorities were considered and placed in other queues. The fact that the BE queue length was not constant due to the RED algorithm caused the delays of BE data packets to be inconsistent. While VOIP, Video conference IPTV, and Video stream applications are transmitted with a delay of 25-40 ms, UDP BE data packets are transmitted with a delay of 80-120 ms.

The MCPRQ algorithm creates a very low ($\sim \pm 3$ ms) jitter in VOIP, 3 ms for video conferencing, and ~ 4 ms for IPTV and video streaming. In this scenario, while MCPQ gives a worse result in terms of jitter, MCPRQ provides a result that is very close to the PQ algorithm.

Table 6. Scenario 2 router package analysis

Queue Type	Received Package (KB)	Sent Package (KB)	Dropped Package (KB)	Total Delay (s)	Avg. Queue Length (Kb)	\overline{Qs} (Kbps)
Drop Tail	50459	40238	10197	1501.25	22.90	26.80
RED	50600	40248	10337	1072.66	16.75	37.52
PQ	50443	40230	10189	1327.11	23.09	30.31
MCPQ	50406	40238	10147	1189.22	19.12	33.83
MCPRQ	50328	40238	10073	1043.90	16.71	38.54

When looking at the sum of the delay times at the router for all packets sent from the queue (excluding those dropped), the most considerable waiting time is Drop. The shortest delay time was achieved in the MCPRQ algorithm, followed by the RED, MCPQ, and PQ algorithms, respectively. The MCPRQ algorithm gave a 12% better result than MCPQ and a 21% better result than PQ.

Table 7. For Scenario 2 average from the end-end delay durations (ms)

Queue Type	VOIP	Video Conference.	IPTV	Video Stream	UDP BE Data
Drop Tail	52	54	55	55	54
RED	45	45	45	45	45
PQ	21	25	22	23	138
MCPQ	24	36	37	30	141
MCPRQ	27	39	40	34	90

The developed MCPRQ algorithm, like the MCPQ algorithm, considers the packet priorities, the packet size, and the waiting time of the packet in the queue, so packets are transferred according to the Fmax value, not the transmission order of the queues. The packet in the queue with the most considerable Fmax value is transmitted. Unlike MCPQ, the queue length in the MCPRQ algorithm is tried to be kept in the minth-maxth range, and the dropped packets are selected from the BE queue with RED. Packets in the priority queue other than VOIP are transmitted 6-18 ms later than MCPQ and the same PQ algorithm. In comparison, packets in the non-priority queue (BE) are transmitted ~50 ms earlier than PQ and MCPQ.

Scenario 3: This scenario aims to examine the behavior of queuing algorithms and compare them with each other in an environment where congestion occurs in the network, but there is no packet loss.

The average queue length and the amount of memory used for the MCPRQ algorithm are shown. Since the average queue length of the RED algorithm in the MCPRQ algorithm is less than minth=14Kb, no packet loss occurred. As a result, the MCPRQ algorithm works the same way as the MCPQ algorithm and gives the same results (Table 8).

Table 8. Scenario 3 router packet analysis

Queue Type	Received Package (Kb)	Sent Package (Kb)	Dropped Package (Kb)	Total Delay (Sec)	Avg Queue Length (Kb)	\overline{Qs} (Kbps)
Drop Tail	66786	66771	0	1344.35	16.10	46.66
RED	66814	66770	27	1133.51	13.46	58.9
PQ	66785	66775	0	766.55	6.67	87.11
MCPQ	66784	66774	0	800.64	7.15	83.4
MCPRQ	66784	66774	0	800.64	7.15	83.4

The end-to-end delay graph with the MCPRQ queue algorithm is shown. Since the queue length is lower than the minimum threshold value of RED, there is no packet loss, which caused it to work in the same way as the MCPQ algorithm. While VOIP, Video conference IPTV, and Video stream applications are transmitted with a delay of 18-32 ms, UDP BE data packets are transmitted with a delay of 18-32 ms (Table 9).

Table 9. For Scenario3 average from the end-end delay durations (ms)

Queue Type	VOIP	Video Conference	IPTV	Video Stream	UDP BE Data
Drop Tail	26	27	25	26	26
RED	24	25	23	25	24
PQ	13	14	13	14	25
MCPQ	13	15	14	15	25
MCPRQ	13	15	14	15	25

Since the priorities of the packets, as well as the packet size and the waiting time of the packet in the queue, are taken into consideration in the developed MCPQ and MCPRQ algorithms, packets are transferred according to the Fmax value, not according to the transfer order of the queues. For this reason, packets in the priority queue other than VOIP are transmitted 1-2 ms later than the PQ algorithm, while packets in the non-priority queue (BE) are transmitted simultaneously.

Since there is no congestion in this scenario, no situation can create a high jitter in the queue. While a ± 1 ms jitter occurs in the PQ algorithm, a ± 2 ms jitter was measured in the drop tail, MCPQ, and MCPRQ algorithms, and a ± 3 ms jitter was measured in the RED algorithm.

4. PERFORMANCE COMPARISON

Each scenario was analyzed regarding average queue length, packet losses, average queue speed, end-to-end delay, and end-to-end jitter criteria. Table 10 shows the first two queue methods that gave the best results.

Table 10. Result comparison

Scenario	Avg. Queue Length	End to End Delay	Total Delay	Package The loss	Jitter	Avg. Queue Speed
Regular Traffic	RED,	MCPRQ,	RED,	MCPQ,	PQ,	RED,
	MCPRQ	MCPQ	MCPQ	PQ	Drop Tail	MCPRQ
Heavy traffic	MCPRQ,	MCPRQ,	MCPRQ,	MCPRQ,	PQ,	MCPRQ,
	RED	PQ	MCPQ	MCPQ	Drop Tail	RED
Light Traffic	PQ,	MCPRQ,	PQ,	MCPRQ,	PQ,	PQ,
	MCPRQ	PQ	MCPRQ	PQ	MCPRQ	MCPRQ

As seen in Table 10, the algorithms with the lowest average queue length are RED and MCPRQ. However, the queue model with the most stable average queue length is MCPRQ. RED exhibits an unstable graph because it could not solve cases where packet sizes differ. In light traffic, all algorithms in the test environment gave very close values.

5. CONCLUSION

DiffServ methods at the IP layer were developed to improve the quality of service in IPv6 networks. These methods are drop-tailed compared with other AQM and Diffserv methods, such as RED and PQ. The comparison is made on a test network built on the OMNeT ++ simulator.

Another method developed is Multi-Criteria Priority RED Queuing (MCPRQ), a DiffServ and AQM method. It is created by combining MCPQ and RED algorithms. In packet scheduling, MCPQ uses the RED model to drop packets randomly in case of congestion or near congestion. Dropping packets from the least essential packets prevents a decrease in service quality and global synchronization.

Although using MCPQ in end-to-end delay slightly increased the delay in real-time applications, it significantly reduced the average delay and the delay in BE packets (UDPBE data, HTTP, FTP, SMTP).

MCPQ and MCPQ again stand out for their low packet loss rates, which are desired in real-time applications.

MCPQ is that a packet arriving at the queue leaves the queue quickly. In particular, the MCPQ algorithm makes the queue work faster by reducing the average queue length and not allowing congestion.

In future studies, the weights calculated in the MCPQ algorithm can be improved with a machine-learning algorithm that changes according to the traffic density. To combat jitter more successfully, a change in delay can be added to the timing criteria, or a queuing algorithm that combats jitter can be added. A structure like SRED can eliminate the unstable RED structure in the MCPQ algorithm against sudden queue changes and different packet sizes.

REFERENCES

- [1] Naeem EA, Abdelaal AEA, Eyssa AA, Al Azrak FM, Ahmed RA, Hassan ES, et al. Efficient signal and protocol level security for network communication. *International Journal of Speech Technology*. 2020; 23(2):399-424.
- [2] Ahmed S, Ali M, Baz A, Alhakami H, Akbar B, Khan IA, et al. A Design of Packet Scheduling Algorithm to Enhance QoS in High-Speed Downlink Packet Access (HSDPA) Core Network. *International Journal of Advanced Computer Science and Applications*. 2020;11(4):596-602.
- [3] Aureli D, Cianfrani A, Diamanti A, Vilchez JMS, Secci S, Ieee. Going Beyond DiffServ in IP Traffic Classification. *NOMS 2020 - PROCEEDINGS OF THE 2020 IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM 2020: MANAGEMENT IN THE AGE OF SOFTWAREZATION AND ARTIFICIAL INTELLIGENCE2020*.
- [4] Hassan ES, Abdelaal AEA, Oshaba AS, El-Emary A, Dessouky MI, El-Samie FEA. Optimizing bandwidth utilization and traffic control in ISP networks for enhanced smart agriculture. *PLoS One*. 2024; 19(3):e0300650.
- [5] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*. 1993; 1(4):397-413.
- [6] Oleiwi HW, Saeed N, Al-Taie HL, Mhawi DN. Evaluation of Differentiated Services Policies in Multihomed Networks Based on an Interface-Selection Mechanism. *Sustainability*. 2022;14(20).
- [7] Strzeciwiłk D. Timed Petri Nets for Modeling and Performance Evaluation of a Priority Queueing System. *Energies*. 2023;16(23).
- [8] Xue JH, Wu Y, Tao J, Zhang YL, Ieee. Research on Campus Network Based on QoS Technology. *2020 IEEE 3RD INTERNATIONAL CONFERENCE ON INFORMATION COMMUNICATION AND SIGNAL PROCESSING (ICICSP 2020)2020*. p. 418-23.
- [9] Zhang R, Liu L, Lu XD, Yan ZM, Li H. Performance Modeling of a General GPS Scheduling Under Long Range Dependent Traffic. *2020 IEEE INTL SYMP ON PARALLEL & DISTRIBUTED PROCESSING WITH APPLICATIONS, INTL CONF ON BIG DATA & CLOUD COMPUTING, INTL SYMP SOCIAL COMPUTING & NETWORKING, INTL CONF ON SUSTAINABLE COMPUTING & COMMUNICATIONS (ISPA/BD CLOUD/SOCIALCOM/SUSTAINCOM 2020)2020*. p. 683-9.
- [10] Demir S, Özçelik I. A priority-based queuing model approach using destination parameters for real-time applications on IPv6 networks. *Turk J Electr Eng Comput Sci*. 2020;28(2):727-42.
- [11] Orman A. Improvement of the quality of service in IPv6 networks by active packet management [PhD thesis]. YÖK Thesis Center: Gazi University; 2012.
- [12] Orman A, Elmas C, Güler I, editors. Increasing Quality of Services with Priority Active Package Management. *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT); 2018 19-21 Oct. 2018*.
- [13] Park G, Jeon B, Lee GM. QoS Implementation with Triple-Metric-Based Active Queue Management for Military Networks. *Electronics*. 2023;12(1).
- [14] Shi HF, Pan CS, Wang YZ. BS-HTIS: Buffer Sizing for Heterogeneous Traffic and Integrated System. *Ieee Access*. 2021;9:115237-45.

- [15] Barry MA, Tamgno JK, Lishou C, Ieee. Influence of quality service in IP/MPLS network load with IPTV and VoD services. 2020 22ND INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY (ICACT): DIGITAL SECURITY GLOBAL AGENDA FOR SAFE SOCIETY!2020. p. 378-85.
- [16] Chen J, Chen J, Zhang H. DRL-QOR: Deep reinforcement learning-based QoS/QoE-aware adaptive online orchestration in NFV-enabled networks. *IEEE Transactions on Network and Service Management*. 2021;18(2):1758-74.
- [17] Sathyanarayana SD, Sankaradas M, Chakradhar S, Ieee. 5GLoR: 5G LAN Orchestration for enterprise IoT applications. 2022 IEEE FUTURE NETWORKS WORLD FORUM, FNWF2022. p. 28-35.
- [18] de Almeida LC, da Silva WRD, Tavares TC, Pasquini R, Papagianni C, Verdi FL. DESiRED — Dynamic, Enhanced, and Smart iRED: A P4-AQM with Deep Reinforcement Learning and In-band Network Telemetry. *Computer Networks*. 2024;244:110326.
- [19] ORMAN A, DÜZKAYA H. Ulaşım Planlama Çalışmalarında Veri Analiz Yöntemleri: Çok-Disiplinli Bir Mühendislik Yaklaşımı ve Ankara Ulaşım Ana Planı Örneği [Data Analysis Methods in Transportation Planning Studies: A Multidisciplinary Engineering Approach and the Case of Ankara Transportation Master Plan]: Atlas Akademi; 2019.
- [20] Chandavarkar BR. Media Independent Handover and Mobile IPv6-Based UDP Performance Evaluation Suite for Heterogeneous Wireless Networks. *Wirel Pers Commun*. 2023;129(2):1197-228.