International Journal of Computer Applications (0975 – 8887)
Volume 72– No.20, June 2013

Efficient QoS-based Scheduling Mechanisms for IEEE 802.16e Networks

M. Deva Priya
Assistant Professor
Department of CSE,
SKCT, Coimbatore,
Tamil Nadu, India.

M.L Valarmathi,Ph.D
Associate Professor
Department of CSE,
GCT, Coimbatore,
Tamil Nadu, India.

V. Sundarameena
Assistant Professor
Department of CSE,
MIT, Pondicherry,
India.

ABSTRACT
Users have become more acquainted to broadband access. IEEE 802.16, a standard for broadband wireless communication in Metropolitan Area Networks (MAN) promises to be one of the best wireless access technologies capable of supporting very high bandwidth applications. The main objective of WiMAX is to deliver wireless communications with high Quality of Service (QoS) guarantees, security and mobility. Many scheduling algorithms compatible with the IEEE 802.16 standards are proposed in the literature with the tenacity of throughput optimization, fairness enhancement and QoS provisioning. However, few scheduling schemes minimize the delay involved. This paper proposes enhancements to the existing scheduling algorithms - Weighted Fair Queuing (WFQ) and Deficit Weighted Round Robin (DWRR). The performance of the existing and proposed algorithms is established by simulating the system under different scenarios. The Novel Weighted Fair Queuing (NWFQ) and Novel Deficit Weighted Round Robin (NDWRR) algorithms yield better Throughput and Packet Delivery Ratio (PDR) and involve less Delay, Jitter and Loss Rate. They are appropriate for scheduling particular types of services.

General Terms
Wireless Networks, WiMAX, Scheduling.

Keywords
WiMAX, DeficitCounter, MAC, QoS, DWRR, NDWRR, WFQ, NWFQ.

1. INTRODUCTION
WiMAX (Worldwide Interoperability for Microwave Access) based on the Wireless MAN (IEEE 802.16) standard is a telecommunication technology providing wireless data over long distances in a variety of ways, from point-to-point links to full mobile cellular type access. The WiMAX wireless broadband access standard provides the missing link for the "last mile" connection in MAN where DSL, Cable and other broadband access methods are not available or too expensive. WiMAX devices with directional antennas offer speeds of 10 Mbits/s covering 10 kms, while WiMAX devices with omni-directional antennas offer only 10 Mbits/s over a range of 2 kms. WiMAX is capable of handling upto 70 Mbits/s [1 - 4].

There is no uniform global licensed spectrum for WiMAX, although three licensed spectrum profiles are being used generally - 2.3 GHz, 2.5 GHz and 3.5 GHz.

The fixed WiMAX standard, IEEE 802.16-2004, 802.16d provides fixed, point-to-multi point Broadband Wireless Access (BWA) service, while mobile WiMAX supports mobility.

The two main areas of research are Call Admission Control and Scheduling. Various scheduling algorithms are proposed in the literature, out of which Weighted fair Queuing (WFQ) and Deficit Weighted Round Robin (DWRR) are discussed in this paper. These algorithms are enhanced and they offer better results when compared to the existing ones.

2. MEDIUM ACCESS CONTROL (MAC)
The MAC layer of the 802.16 protocol forms the foundation of the protocol and all associated implementations. It supports predominantly a Point-to-Multipoint (PMP) architecture with an optional mesh topology. The transmission overhead can be reduced by fragmentation and single burst transmissions. It forms the communication bridge between the physical layer and the transmitting higher layer applications in the network. The MAC information received from the applications is termed as MAC Service Data Unit (MSDU). The MAC layer is responsible for providing appropriate scheduling services.

Each transmitting application can be defined as a Connection and each individual connection has an associated Connection ID (CID).The CID-SFID mapping is done by the scheduler. The MAC layer is formed with three sublayers - the Service Specific Convergence Sublayer (CS), the MAC Common Part Sublayer (CPS) and the Privacy Sublayer (PS). The Service-Specific Convergence Sub-Layer is responsible for interfacing with upper layers while the MAC Common Part Sub Layer caters to the key MAC functions [5]. Privacy Sublayer deals with security.

The IEEE 802.16 standard divides the services into five different classes [6 - 10] - Unsolicited Grant Service (UGS), real-time Polling Service (rtPS), non-real-time Polling Service (nrtPS), Best Effort Service (BE), extended real-time Polling Service (ertPS).

3. SCHEDULING IN WiMAX
There are many algorithms proposed in the literature to schedule users in WiMAX. This paper discusses two algorithms - WFQ and DWRR. The performance of these algorithms for different services is analyzed. Enhancements are also proposed.

3.1 Generalized-Processor-Sharing (GPS)
Generalized-Processor-Sharing (GPS) is an ideal scheduling discipline [11, 12] and is a natural generalization of uniform processor sharing [13]. The packet-based version is proposed in [14] under the name Weighted Fair Queuing.

You- Chiun Wang and Yu-Chee Tseng in [13] proposed many Packet Fair Queuing (PFQ) algorithms to approximate GPS. Weighted Fair Queuing (WFQ) [13, 21] is a representative scheme to approximate GPS.

3.2 Weighted Fair Queuing (WFQ)

WFQ can be referred to as a priority based queuing scheme, where packets are treated based on the priorities assigned to them. It is a generalization of Fair Queuing (FQ) and preferential weighting. This queuing discipline was designed to ensure that each flow has fair access to network resources and bursty flows do not consume more than the shared output bandwidth. As in FQ, each data flow is assigned a separate FIFO queue [16]. In FQ, if the data rate of a link is ‘D’, and there are ‘n’ active data flows (the ones with non-empty queues) at a time, all the flows are serviced concurrently at an average data rate of D/n.

The fairness aspect of WFQ functions resembles Round Robin (RR) queuing since queues are serviced recurrently from the first to the last, in order, until all the queues are empty. This method automatically stabilizes network congestion between individual packet transmission flows. WFQ supports variable-length packets, so that flows with larger packets are not allocated more bandwidth than flows with smaller packets.

Each flow is allocated an equal amount of network bandwidth and hence termed fair. If high-priority queues are not in use, lower-priority traffic uses the bandwidth. This prevents high-bandwidth traffic from seizing an unfair share of resources. All queues are serviced so that none starve, but some queues are serviced more frequently than others. This fair allocation adds significantly to the computational complexity of the queue scheduling algorithm. As it prioritizes flows within a network, multiple flows are able to share the network bandwidth and transmit at the same time. This eliminates starvation of flows due to one ill-behaved flow.

A weight is assigned to each queue to give high priority to some queues. Based on the weights, different percentages of output port bandwidth are allocated. For example, one queue may get half the available bandwidth and the remaining bandwidth may be allocated to other queues. The weight is used to ensure that more significant queues get serviced frequently than other less important ones. Each incoming packet is placed into its corresponding queue based on its type and is timestamped with a finish time. Finish time is the sum of the current time and the time taken to transmit the packet. Current time is zero if there are no packets in the queue.

Queues are first sorted in the order of increasing weights. Each queue is serviced in the order of its weighted proportion to the available resources. Since each data flow has its own queue, an envious flow with larger packets or one that necessitates transmission of more packets per second when compared to others will penalize itself and not other sessions. As mentioned earlier, it schedules interactive traffic to the front of the queue to reduce response time and fairly shares the remaining bandwidth among high bandwidth flows. Low volume traffic streams will benefit as they quickly complete their transmissions without much impact on high volume ones.

3.3 Scheduling by WFQ

WFQ is a packet approximation of GPS, which, as the name suggests, is a Generalization of Processor Sharing (PS). It is a packet scheduling technique allowing guaranteed bandwidth services. The purpose of WFQ is to let several sessions share the same link [15, 17].

WFQ algorithm needs a buffer to queue the incoming packets. The buffer space is divided into many queues, each of which is used to hold the packets of a flow. WFQ permits each flow with different weights to have different percentages of bandwidth. As already stated, this prevents the monopolization of bandwidth by some flows, thus providing fair scheduling for different flows.

It supports variable length packets by approximating the theoretical approach of the GPS system and assigns a finish time to each packet. The packet with the lowest finish time will be scheduled next. The finish number is calculated based on the subscriber’s weight, the finish number of the previous packet scheduled in that connection and the length of the packet. In an OFDMA system, several connections can be served at once during a single frame which would require multiple rounds of the algorithm and hence higher complexity [18, 19]. Given the bit rate of the output port, the number of active queues, the relative weight assigned to each of the queues and the length of each of the packets in each of the queues, the scheduling discipline is established and a finish time is assigned to each arriving packet. The scheduler then selects and forwards the packet that has the earliest (smallest) finish time from among all of the queued packets [15, 17].

The finish time is not the actual transmission time for each packet. Instead, the finish time is the number assigned to each packet that represents the order in which packets should be transmitted on the output port. As mentioned earlier, the virtual finish time for a newly queued packet is given by the finish time of the packet queued ahead of it for its flow plus its own size. If there are no packets queued for the flow, the virtual finish time is given by current virtual time plus the packet size, where the current virtual time is the assigned virtual finish time for the packet which was recently transmitted plus the progress on the current transmission. For each packet, the arrival time, the size, an application payload and a reference to the connection it belongs to are known. All the incoming packets, ordered by arrival time, are analyzed and sorted [20, 21].

The packet selected for output is the packet with the smallest virtual finish time. In [11, 12] Parekh describes a Packet GPS (PGPS) algorithm which is identical to the WFQ algorithm mentioned here.

Round Number represents the progression of virtual time, increased in each scheduling cycle, and is defined as:

$$\text{RoundNumber}(t) = \text{RoundNumber}(t-1) + \text{RoundRate}(t) \quad (1)$$

$$\text{RoundRate}(t) = 1/(\text{Sum of active queues' weights at time } t) \quad (2)$$

A queue is active, if it is not empty and its weight is the normalized Minimum Reserved Traffic Rate (MRTR) expressed as:

$$W(i) = \text{MRTR}(i) / \text{Sum}(\text{MRTR for all queues}) \quad (3)$$

If the queue is unweighted, then the finishing time for a queue with flow ‘f’ is given by,

$$F(f,k) = \max\{F(f,k-1), R(t(f,k))\} + P(f,k) \quad (4)$$

where F (f, k-1) is the finish time of the previous packet.

If the queue is weighted, then the finishing time for a queue with flow ‘f’ is given by,

$$F(f,k) = \max\{F(f, k-1),R(t(f,k)\}+P(f,k)/W_f \quad (5)$$

```
Algorithm NWFQ
    initialize (i)
1.  for i = 1 to q        /*queue index */
2.          Finish [i] = 0
3.          $R_{link}$ = k
4.          $Round_{current}$ [i] = 0
5.          $R_{active}$ [i] = 0
6.  end /*for*/
    end    /* initialize() */
    enqueue(k, i)
1.  Select the queue that matches the type of the flow to
    which the new packet belongs
2.  if (!InActiveList(i) at time t) then
3.          activate(i)
4.          initialize(i)
5.          q + = 1 // Number of active queues
6.  end /* if*/
7.  if (isempty(i) at time t)        // Queue I is empty
8.          Finish[i] = $R_k(t) + P_k / W_i$
9.  else
10.         Finish[i] = $Finish_{k-1}[i] + P_k / W_i$
11. end /* if*/
12. Insert the packet at end of Queue i
13. $R_{req}$ [k,i] = Finish [i]
    end    /* enqueue()*/
    dequeue( )
1.  while (!isemptyActiveList at time t) then
2.  n = q ,  c = 0
3.  do
4.          $R_{active}$ [i] = $R_{link}$ / q
5.          if ($R_{req}$ [k, i] $\leq$ $R_{active}$[i])
6.                  Service queue i
7.                  $R_{allocated}$[i] = $R_{active}$[i]
8.                  $R_{left}$ =  $R_{active}$[i] - $R_{req}$[k,i]
9.                  $Round_{current}$ [i] += $R_{req}$[k,i]
10.                 $R_{req}$[k,i] = 0
11.                 deactivate(i)
12.                 q - =1 ; c +=1
13.                 for (i = 2; i $\leq$ n && $R_{left}$ !=0; n+=1)
                      if ($R_{req}$ [k,i] $\leq$ $R_{left}$ )
14.                     Sevice queue i
15.                     $R_{left}$ - = $R_{req}$ [k,i]
16.                     $Round_{current}$ [i] += $R_{req}$[k,i]
17.                     $R_{req}$[k,i] = 0
18.                     deactivate(i)
19.                     q - =1 ;  c +=1
20.                   end /*if*/
21.                 end /*for*/
22.                 $R_{link}$ = $R_{link}$ - $R_{allocated}$ [i] + $R_{left}$
23.         else
24.                 Sevice queue i
25.                 $Round_{current}$ [i] += $R_{req}$[k, i]
26.                 $R_{req}$ [k, i] -= $R_{active}$ [i]
27.                 c +=1
28.         end /*if*/
29. while (c $\leq$ n)
30. for each queue
31.     if (isemptyQueue(i)) then   /* Queue I is empty*/
32.         deactivate(i)
33.     else
34.         activate (i)
35.     end /* if */
36. end /*for*/
37. end /*while*/
    end /* dequeue()*/
```

Fig. 1: NWFQ Algorithm

All active queues are maintained in the ActiveList. Whenever a new packet arrives, it is enqueued to the corresponding active queue. Each queue has its own weight [22]. Finish times are calculated according to the equations mentioned above. The link rate is k units/sec. The $R_{req}[k,i]$ is the rate expected by packet k. It is initialized to Finish[i]. The function Enqueue places newly arriving packets into its correct queue and manages the ActiveList. The ActiveList is maintained to avoid examining empty queues. It contains a list of queue indices that contain at least one packet.

To dequeue, $R_{active}[i]$ is first calculated. It varies with the number of active queues, q. R_{link} remains constant. $Round_{current}[i]$ is incremented with $R_{active}[i]$. After a round is completed, the $R_{req}[k,i]$ gets decremented by $R_{active}[i]$, since R_{active} [i] units were served in the previous round. The queue remains active until Finish[i] is greater than $Round_{current}[i]$. If the queue is empty, it is deactivated. Else it is added to the ActiveList.

3.4 Novel Weighted Fair Queuing (NWFQ)

A modified version of WFQ algorithm is given in Fig. 1. In WFQ algorithm, R_{link} remains constant. Further, in some cases, $R_{req}[k,i]$ may be very much less when compared to $R_{active}[i]$. i.e more units are allocated than required. This decreases the number of packets serviced in a single round.

Instead, the unused units may be given to the next queue, so that multiple queues get serviced in a round, thus increasing the throughput. i.e The unused bandwidth of a request (packet) is given to another packet in the next queue, guaranteeing the same QoS services without introducing additional delay. This permits other queues to utilize the unused bandwidth left by the current transmitting queue. Once a queue is serviced, as $R_{req}[k,i]$ becomes zero, the queue is deactivated. The units left after servicing a (multiple) queues, R_{left} is added to the R_{link}, thus increasing the rate of other queues. NWFQ provides these enhancements [7, 23].

4. DEFICIT WEIGHTED ROUND ROBIN (DWRR)

Deficit Weighted Round Robin is the basis of a class of queue scheduling disciplines that is designed to address the limitations of the Weighted Round Robin (WRR) and WFQ models. DWRR a variation of RR visits each non-empty queue and handles packets of variable sizes without knowing their mean size. The packet size is subtracted from the packet length and the packets with sizes that exceed the length are held back until the next visit of the scheduler.

With WRR for each scheduling turn, the number of packets that are granted service is based on a weight that reflects the bandwidth allocation for the queue. Bandwidth allocation can be unfair when the average packet sizes differ for the queues and their flows. This behavior can result in service degradation for queues with smaller average packet sizes. DWRR is a modified WRR scheduling discipline. For certain traffic types, fairness is not the desired behavior. What is needed is a priority scheduling similar to PQ but that preserves the benefits of DWRR. To achieve the predictable service for sensitive, real-time traffic, a priority level for scheduling needs to be introduced. By enabling strict priority or by offering several priority levels and using DWRR to schedule queues with the same priority levels, service assurance with regard to delay and loss protection can be achieved for demanding traffic types, such as voice and real-time broadcasting. DWRR queuing allows grouping traffic into classes. In other words, when WFQ classifies traffic per session, DWRR uses user-defined traffic classes which are less granular but more application-specific. Each class is given its own queue.

When the queue is visited by the scheduler, packets are transmitted from the queue as long as there are sufficient tokens in the bucket. When the next packet in the queue exceeds the remaining tokens in the bucket, the scheduler moves to the next queue [24, 25]. There are 3 quantities – weight, DeficitCounter, Quantum. Weight determines the percentage of the output port bandwidth allocated to the queue. DeficitCounter specifies the total number of bytes that the queue is permitted to transmit in a service round. Quantum is based on the weight of the queue and is mentioned in terms of bytes.

$$\text{Quantum} = w_i * \text{Bandwidthport} \tag{7}$$

Introducing DeficitCounter permits the DWRR algorithm to be aware of bandwidth and improves the fairness. WRR serves every non-empty queue whereas DWRR serves packets at the head of every non-empty queue whose DeficitCounter is greater than the packet's size at the Head of the Queue (HoQ). Initially the DeficitCounter value is set to zero. When the scheduler visits the queue for the first time, the DeficitCounter for that queue is incremented by the quantum. When one of the queues becomes empty, it is removed from the ActiveList and the token rates for the other buckets are adjusted proportionately, ensuring that all of the bandwidth is fully utilized. If only one queue has traffic, it gets the total bandwidth.

4.1 Scheduling in DWRR

In the classic DWRR algorithm, the scheduler visits each non-empty queue and determines the number of bytes in the packet at the HoQ. The variable DeficitCounter is incremented by the value quantum. If the size of the packet at the HoQ is greater that the DeficitCounter, then the scheduler moves to service the next queue. DWRR avoids packet fragmentation by scheduling only full packets. If the packet is large, it cannot be sent in a single round. If the DeficitCounter value is too small to send even a portion of a packet, then the packet is held for transmission in the next round [26]. Therefore the maximum permitted packet size is subtracted from the available DeficitCounter and the packet is sent during the next visit of the scheduler. In other words, the queue is skipped and its credit is increased by some given value called quantum.

$$\text{DeficitCounter} = \text{Remaining Deficit Counter value of the previous round} + \text{Quantum} \tag{8}$$

This increased value is used to calculate the DeficitCounter for the next round, when the scheduler examines the queue for serving its head-of-line packet. If the queue is served, then the Credit is decremented by the size of packet being served. If the size of the packet at the HoQ is less than or equal to the variable DeficitCounter, then the variable DeficitCounter is reduced by the number of bytes in the packet and the packet is sent to the output port. The scheduler continues to dequeue packets and decrement the variable DeficitCounter by the size of the transmitted packet until either the size of the packet at the HoQ is greater than the variable DeficitCounter or the queue is empty.

$$\text{DeficitCounter-= size (packet (HoQ))} \tag{9}$$

If the queue is empty, the value of DeficitCounter is set to zero and the queue is deactivated. When this occurs, the scheduler moves to service the next non-empty queue.

4.2 Novel Deficit Weighted Round Robin (NDWRR)

In DWRR, if the size of the first packet in a queue cannot be serviced as the DeficitCounter is less than the size of the first packet. The packet should wait for the next round. No packets from the queue will be serviced in this round. The queue may contain a smaller packet that is denied service. This increases the delay of packets.

Instead, at any instant, the queues may be sorted separately based on the packet sizes. Now there are high chances for the size of the first packet in a queue to be less than its DeficitCounter. The DWRR algorithm is modified to support this enhancement (Fig. 2).

The other modification is the movement of the DeficitCounter, in case the queue becomes empty. In DWRR, once a queue becomes empty, the DeficitCounter of that particular queue is made zero. Instead, in NDWRR, it is moved to the next active queue. The DeficitCounter of the queue that is currently serviced is increased, which in turn aids in servicing more packets in a single round.

```
Algorithm NDWRR
    initialize (i)
1.  for i = 1 to n        /*queue index */
2.      DeficitCounter[i] = 0
1.      Quantum = $w_i$ * $\text{Bandwidth}_{\text{port}}$
3.  end /*for*/
    end   /* initialize() */
    enqueue(k, i)
1.  Select the queue that matches the type of the flow to
    which the new packet belongs
2.  if (!InActiveList(i)) then  /*i is not in ActiveList*/
3.          activate(i)
4.          initialize(i)
5.  end /* if*/
6.  Insert the packet k to the end of Queue i
7.  Sort the packets in the ascending order of  Quantum
    end   /* enqueue()*/
    dequeue(i)
1.  while (!isemptyActiveList) then
2.    select the queue i from the head of the ActiveList
3.    DeficitCounter[i] += Quantum[i]
4.    while(DeficitCounter[i] > 0 && (!isemptyQueue[i]))
5.        PacketSize = Size(packet at the HoQ i )
6.        if (PacketSize $\le$ DeficitCounter[i]) then
7.              Transmit the packet at the HoQ of queue i
8.              DeficitCounter[i] - = PacketSize
9.        else
10.             break /*exits this while loop*/
11.       end  /*if*/
12. end  /*while*/
13. if (isEmptyQueue(i)) then
14.     Move the residual DeficitCounter[i] to the next
        Queue  in the ActiveList
15.     deactivate(i)
16. else
17.     activate(i)
18. end  /*if*/
19. end   /* while*/
    end  /*dequeue*/
```

Fig. 2: NDWRR Algorithm

5. IMPLEMENTATION

As stated earlier, all the four algorithms are implemented. PDR, Throughput, Loss rate, Delay and Jitter are analyzed by scheduling the requests using all the four algorithms. From the results, it is obvious that NDWRR and NWFQ outperform the existing algorithms. In Fig. 3, NDWRR and NWFQ are only shown as they are sufficient to schedule the different services offered by WiMAX.

Initially, in Tier -1, intra-class scheduling is done. In case of WFQ and NWFQ, intra class scheduling is performed based on finish times. In case of DWRR, requests are queued in the order of arrival. In NDWRR all the requests are first sorted based on the packet sizes.

In Tier-2, inter-class scheduling is performed and the requests from different queues are serviced by using all the four algorithms taken into consideration and the results are analyzed.

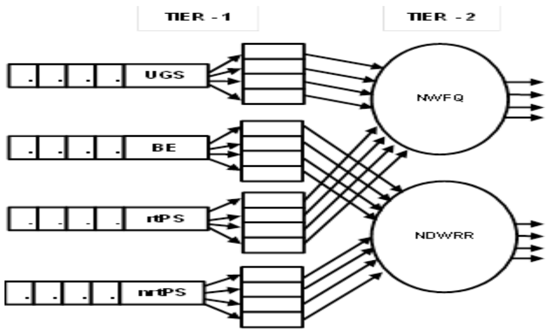

Fig. 3: Architecture of the proposed system

6. PERFORMANCE ANALYSIS

The system was simulated using ns2. 25 nodes were deployed. For each type of traffic, standard packet sizes were taken into consideration. The algorithms were analyzed for different services and QoS parameters like PDR, Loss rate, Throughput, Delay & Jitter. The following graphs show that NWFQ outperforms all the other algorithms for UGS and rtPS services. Similarly NDWRR produced better results for BE and nrtPS services. Table. 1 summarizes the simulation parameters.

Table 1: Simulation parameters

PARAMETERS	UGS	BE	rtPS	nrtPS
Packet size	**1024**	**512**	**512**	**1024**
MAC protocol	802.16			
Bandwidth	2 Mbps			
Routing Protocol	DSDV			
Queue Type	Queue/DropTail			
Queue Length	50			
Start time	20 ms			
Stop time	100 ms			
Modulation Scheme	OFDM_QPSK			
Frame Duration	0.020 ms			

The following section shows how each service (BE, UGS, rtPS and nrtPS) adapts to different scheduling schemes (DWRR, NDWRR, WFQ, NWFQ).

6.1 Performance for BE services

For BE, NDWRR yields better PDR and Throughput when compared to NWFQ. NDWRR involves the least Delay, Jitter and Loss rate. WFQ shows poor performance when compared to DWRR (Fig. 4 to Fig. 8).

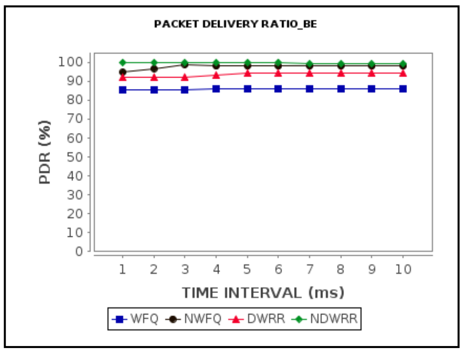

Fig. 4: PDR for BE traffic

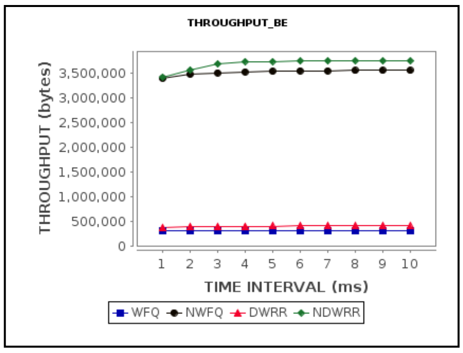

Fig. 5: Throughput for BE traffic

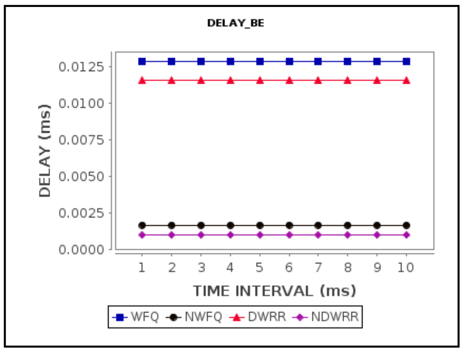

Fig. 6: Delay for BE traffic

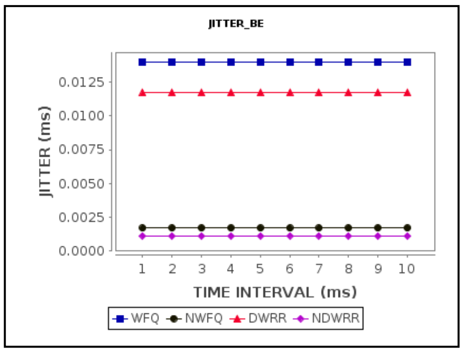

Fig. 7: Jitter for BE traffic

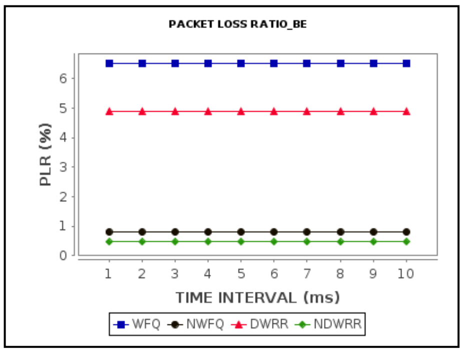

Fig. 8: PLR for BE traffic

6.2 Performance for UGS services

For UGS, NWFQ outperforms the rest, while DWRR shows the least performance. WFQ is better when compared to DWRR. NDWRR does not outdo NWFQ when compared to its performance for BE service (Fig. 9 to Fig. 13).

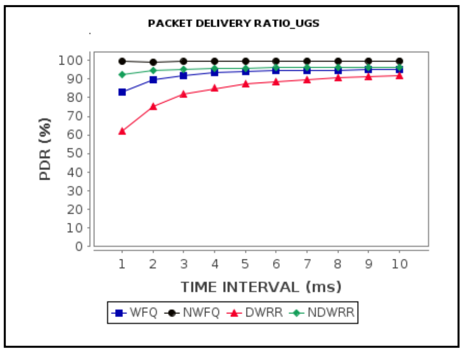

Fig.9: PDR for UGS traffic

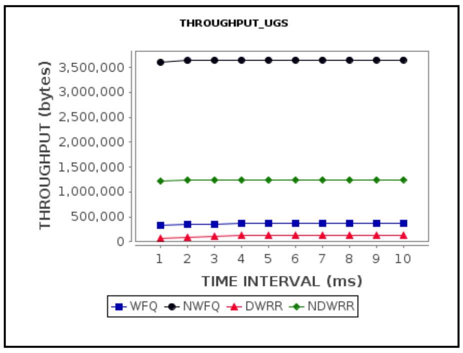

Fig. 10: Throughput for UGS traffic

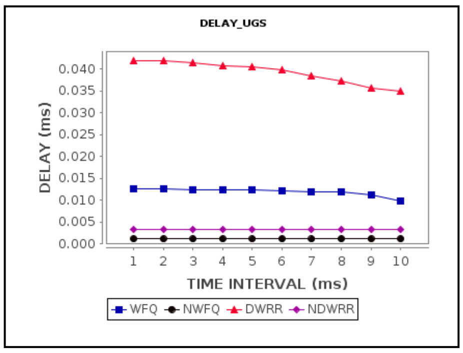

Fig. 11: Delay for UGS traffic

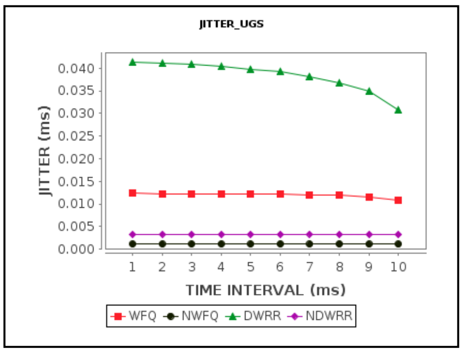

Fig. 12: Jitter for UGS traffic

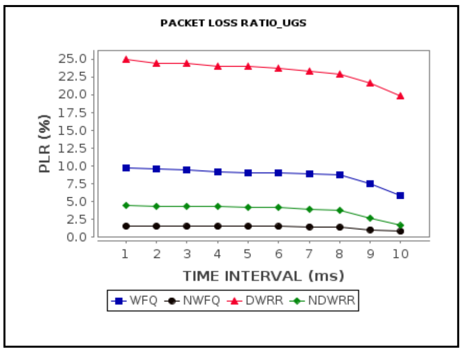

Fig. 13: PLR for UGS traffic

6.3 Performance for rtPS services

For delay sensitive rtPS, NWFQ performs well. NDWRR is better when compared to DWRR and WFQ. DWRR shows the least performance involving larger delay and jitter (Fig. 14 to Fig. 18).

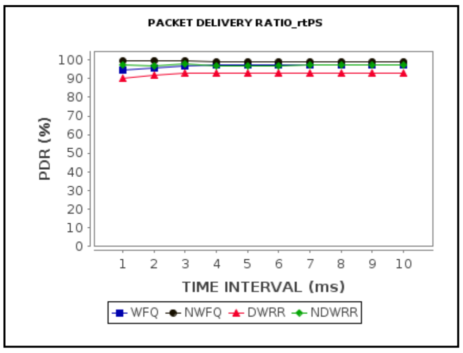

Fig. 14: PDR for rtPS traffic

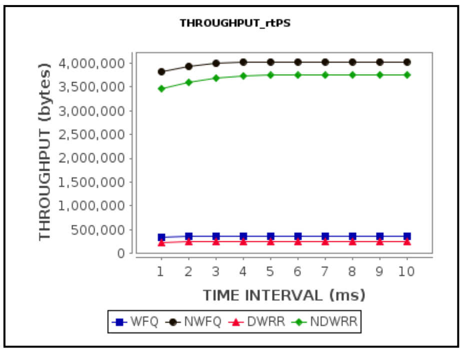

Fig. 15: Throughput for rtPS traffic

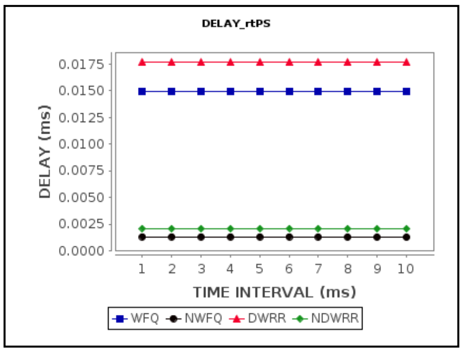

Fig. 16: Delay for rtPS traffic

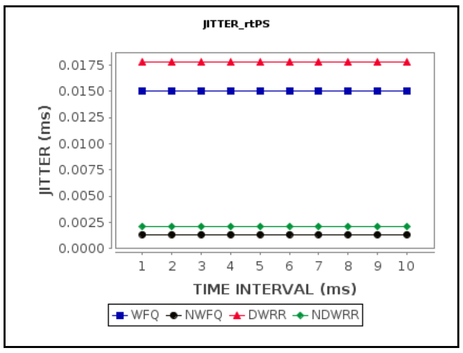

Fig. 17: Jitter for rtPS traffic

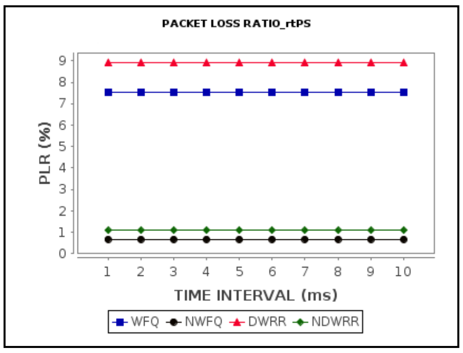

Fig. 18: PLR for rtPS traffic

6.4 Performance for nrtPS services

For nrtPS, NDWRR is applicable as it involves less Delay and Jitter, yielding high PDR and Throughput, whereas WFQ offers the least values. Performance of DWRR is better when compared to WFQ (Fig. 19 to Fig. 23).

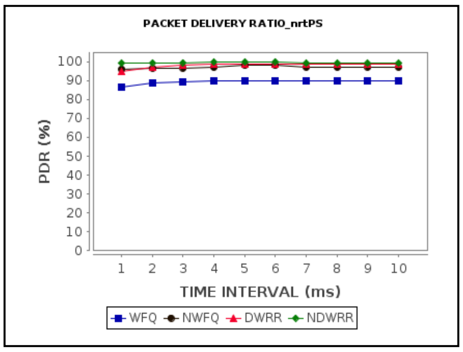

Fig. 19: PDR for nrtPS traffic

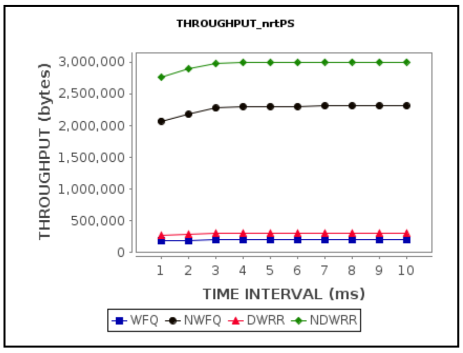

Fig. 20: Throughput for nrtPS traffic

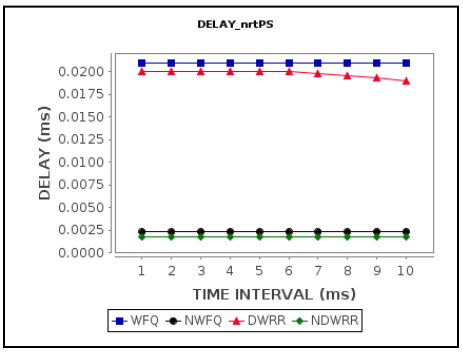

Fig. 21: Delay for nrtPS traffic

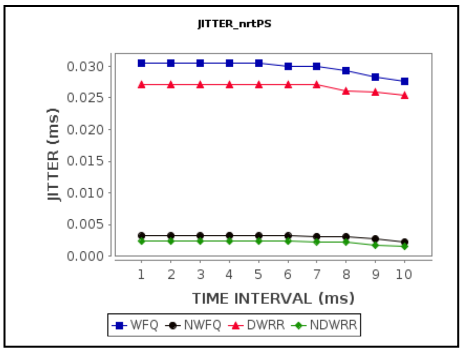

Fig. 22: Jitter for nrtPS traffic

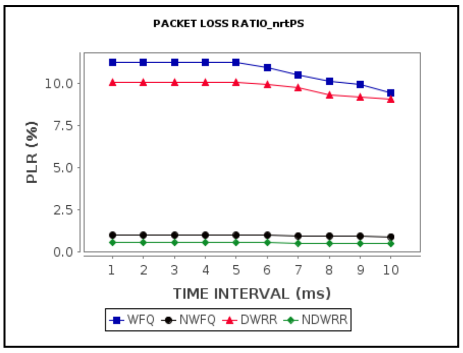

Fig. 23: PLR for nrtPS traffic

For BE and nrtPS, NDWRR is best suitable. On the other hand, for UGS and rtPS, NWFQ yields a better performance.

7. CONCLUSION

This work apparently shows that NDWRR and NWFQ are better when compared to DWRR and WFQ. BE services require the least level of service guarantees, while nrtPS have loose delay requirements. For these two types of services, NDWRR scheduling algorithm can be used, in which the residual DeficitCounter is shifted to the next queue, thus increasing the number of queues serviced in a round. On the other hand, since unexploited bandwidth is efficiently utilized in NWFQ, it can be involved in scheduling delay sensitive rtPS services and UGS type of requests that support real time traffic.

8. REFERENCES

[1] Shankar A.R, Hegde R, WiMAX on the road to future, Proceedings of IET International Conference on Wireless, Mobile and Multimedia Network, 2008, pp. 275-278.

[2] IEEE P802.16Rev2/D2, DRAFT Standard for Local and metropolitan area networks, Part 16: Air Interface for Broadband Wireless Access Systems, 2007, pp. 2094.

[3] IEEE Std 802.16 - 2004, Air Interface for Fixed Broadband Wireless Access Systems, 2004, pp. 895.

[4] Zhang X, Wang Y and Wang W, Capacity analysis of adaptive multiuser frequency-time domain radio resource allocation in OFDMA systems, Proceedings of IEEE International Symposium on Circuits and Systems, 2006, pp. 4-7.

[5] Eklund C, Marks R, Stanwood K and Wang S, IEEE standard 802.16: a technical overview of the WirelessMAN/sup TM/ air interface for broadband wireless access, IEEE Communications Magazine, Vol. 40 , No.6, 2002, pp.98-107.

[6] Jani Lakkakorpi, Alexander Sayenko and Jani Moilanen, Comparison of Different Scheduling Algorithms for WiMAX Base Station Deficit Round-Robin vs. Proportional Fair vs. Weighted Deficit Round-Robin, Proceedings of the Wireless Communications Networking Conference WCNC, 2008, pp. 1991-1996.

[7] Chakchai So-In, Raj Jain, and Abdel-Karim Tamimi, Scheduling in IEEE 802.16e Mobile WiMAX Networks: Key Issues and a Survey, IEEE Journal on selected areas in Communications, Vol. 27, No. 2, 2009, pp. 156-171.

[8] Mikael Gidlund and Gang Wang, Uplink Scheduling Algorithms for QoS Support in Broadband Wireless Access Networks, Journal of Communications, Vol. 4, No. 2, 2009, pp. 133-142.

[9] Ahmed H. Rashwan, Hesham M. ElBadawy and Hazem H. Ali, Comparative Assessments for Different WiMAX Scheduling Algorithms, Proceedings of the World Congress on Engineering and Computer Science, Vol. I, 2009, pp. 362-366.

[10] Najah Abu Ali, Pratik Dhrona and Hossam Hassanein, A performance study of uplink scheduling algorithms in point to- multipoint WiMAX networks, Computer Communications, Vol. 32, No. 3, 2009, pp. 511-521.

[11] Abhay K. Parekh, Robert G. Gallager, A generalized processor sharing approach to flow control in integrated services networks: the single-node case, IEEE/ACM Transactions on Networking, Vol.1, No.3, 1993, pp.344-357.

[12] Abhay K. Parekh, Robert G. Gallager, A generalized processor sharing approach to flow control in integrated services networks : the multiple node case, IEEE/ACM Transactions on Networking, Vol.2, No.2, 1994, pp.137-150.

[13] You-Chiun Wang and Yu-Chee Tseng, Packet Fair Queuing Algorithms for Wireless Networks, Design and Analysis of Wireless Networks, Nova Science Publishers, 2005.

[14] Demers, Keshav S and Shenkar S, Analysis and simulation of a fair queuing algorithm, Internetworking Research and Experience, 1990.

[15] Richard Kautz, Raymond Keh, Kee Chaing Chua and Alberto Leon-Garcia, A Distributed Fair Queuing (DFQ) Architecture for Wireless ATM Local Access Networks, International Journal of Wireless Information Networks, Vol. 7, No. 4, 2000, pp. 221-229.

[16] Taniguchi S, Kawate R, Sato K, Horiuchi E, Yokotani T, Performance evaluation of the simplified WFQ to multiplex a huge number of queues, IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR), 2012, pp. 1-6.

[17] Jean-Philippe Georges, Thierry Divoux and Eric Rondeau, Strict Priority versus Weighted Fair Queuing in Switched Ethernet networks for time critical applications. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005, pp. 141-148.

[18] Iera A, Molinaro A, Pizzi S and Calabria R, Channel-Aware Scheduling for QoS and Fairness Provisioning in IEEE 802.16/WiMAX Broadband Wireless Access Systems, IEEE Network, Vol. 21, No. 5, 2007, pp. 34-41.

[19] Dianati M, Shen X, Naik S, A new fairness index for radio resource allocation in wireless networks, Proceedings of IEEE Wireless Communications and Networking Conference, 2005, Vol. 2, pp. 712-717.

[20] Jin-Cherng Lin, Chun-Lun Chou, Cheng-Hsiung Liu, Performance Evaluation for Scheduling Algorithms in WiMAX Network, Proceedings of 22nd International Conference on Advanced Information Networking and Applications-Workshops, 2008, pp. 68-74.

[21] Zhang H, Service disciplines for guaranteed performance service in packet-switching networks, Proceedings of the IEEE, Vol. 83, No. 10, 1995, pp. 1374-1396.

[22] www.sics.se/~ianm/WFQ/wfq_descrip/node23.html.

[23] David Chuck and Morris Chang J, Bandwidth Recycling in IEEE 802.16 Networks, IEEE Transactions on Mobile Computing, Vol. 9, No. 10, 2010, 1451-1464.

[24] Shreedhar M, Varghese G, Efficient Fair Queuing using Deficit Round Robin, IEEE/ACM Transactions on Networking, Vol.1, No.3, 1996, pp.375-385.

[25] Cicconetti C, Erta A, Lenzini L and Mingozzi E, Performance Evaluation of the IEEE 802.16 MAC for QoS Support, IEEE Transactions on Mobile Computing, Vol.6, No.1, 2007, pp.26-38.

[26] Ravichandiran C, Pethuru Raj C, Vaidhyanathan V, Analysis, Modification, and Implementation (AMI) of Scheduling Algorithm for the IEEE 802.116e (Mobile WiMAX), International Journal of Computer Science and Information Security, Vol. 7, No. 2, 2010.