International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)), Volume 2, Special Issue 3, February 2014)
International Conference on Trends in Mechanical, Aeronautical, Computer, Civil, Electrical and Electronics Engineering (ICMACE14)

A LOW COMPLEXITY PACKET SCHEDULER FOR EFFICIENT SCHEDULING WITH EFQ ON WIRELESS SENSOR NETWORK

S.Arunkumar[1],*S.Senthil*[2]
[1]*JP College of Engineering, Ayikudy.*
[2]*JP College of Engineering, Ayikudy.*
[1]*arun.me.cse2012@gmail.com*
[2]*jaysen1984@gmail.com*

***Abstract*— The main objective of this project is to create a new packet scheduling technique. Packet scheduling is one of the decision process. It is used to choose which packets should be serviced or dropped. The packet scheduling is needed in network for providing maximum bandwidth, a minimum delay, a minimum packet loss rate at all times. The issues in packet scheduling techniques are space, time complexity, and tight guarantees about packets. To solve the issues in packet scheduling this project uses one method. This method is called Efficient Fair Queuing (EFQ). In this method the space and time complexity is reduced by using the flow grouping and timestamp rounding respectively. This method use a new O(1) scheduler that provides near-optimal tight guarantees. The algorithm has no loops. It has simple data structures and instructions involved makes it well suited to hardware implementations. The execution time is within two times that of DRR and consistently about three times faster than S-KPS. Speed does not sacrifice service guarantees: the WFI of EFQ is slightly better than S-KPS. From the experimental result shows that the EFQ performs well than the other existing methods.**

***Index Terms*—** EFQ**,wireless sensor network,packet scheduling, low complexity.**

I. INTRODUCTION

Packet scheduling refers to the decision process. It is used to choose which packets should be serviced or dropped. The packet scheduling is needed for providing maximum bandwidth, a minimum delay, a minimum packet loss rate at all times.

This packet scheduling policy is simple to implement, and yields good performance in the common case that node schedules are known, and information about node availability is accurate. A potential drawback is that a node crash (or other failure event) can lead to a number of wasted RTSs to the failed node.

The Deficit Round-Robin (DRR) which is a scheduling algorithm devised for providing fair queuing in the presence of variable length packets. The main attractive feature of DRR is its simplicity of implementation: in fact, it can exhibit O(1) complexity, provided that specific allocation constraints are met. However, according to the original DRR implementation, meeting such constraints often implies tolerating high latency and poor fairness[5]. The Self-clocked fair queueing scheme which is feasible for broadband implementation. This scheme is based on the adoption of an internally generated virtual time as the index of work progress, hence the name self-clocked fair queueing[7]. The Generalized Processor Sharing (GPS) algorithm has desirable properties for integrated services networks and many Packet Fair Queueing (PFQ) algorithms have been proposed to approximate GPS. However, there have been few high speed implementations of PFQ algorithms that can support a large number of sessions with diverse rate requirements and at the same time maintain all the important properties of GPS[8]. The per-flow fair queueing has not been deployed in the Internet mainly because of the common belief that such scheduling is not scalable. The objective of this paper is to demonstrate using trace simulations and analytical evaluations that this belief is misguided. This paper show that although the number of flows in progress increases with link speed, the number that needs scheduling at any moment is largely independent of this rate.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)), Volume 2, Special Issue 3, February 2014)
International Conference on Trends in Mechanical, Aeronautical, Computer, Civil, Electrical and Electronics Engineering (ICMACE14)

The number of such active flows is a random process typically measured in hundreds even though there may be tens of thousands of flows in progress[4].

To proposed and developed an O(1) scheduler of the third family, called Efficient Fair Queuing (EFQ). It provides tight service guarantees at an extremely low per-packet cost. A prototype running on a commodity PC takes about 110ns per packet, only twice the time consumed by a Deficit Round Robin scheduler, and about 2.5 to 4 times faster than the fastest competitor providing comparable guarantees.

The remainder of this paper is,section 2 describes the methodologies,section 3 describes the experimental setup and finally conclude this paper.

II. METHODOLOGIES

Packet scheduling, together with classification, is one of the most expensive processing steps in systems providing tight bandwidth and delay guarantees at high packet rates. This technique makes two contributions:

1) A new O (1) scheduler that provides near-optimal guarantees, and is the first to achieve that goal with a truly constant cost also with respect to the number of groups and the packet length.

2) To develop a production-quality implementations of EFQ and of its closest competitors, which we use to present a detailed comparative performance analysis of the various algorithms.

The EFQ algorithm has no loops, and uses very simple instructions and data structures which contribute to its speed of operation.

Packet scheduling is one of the decision process. It is used to choose which packets should be serviced or dropped. The scheduling algorithm allowed only eligible packets, ineligible packets are blacked. The EFQ scheduler set the every packets in stating time and ending time. So constant cost every packet flow. Finally the algorithm send the eligible packet to group set without any loss.

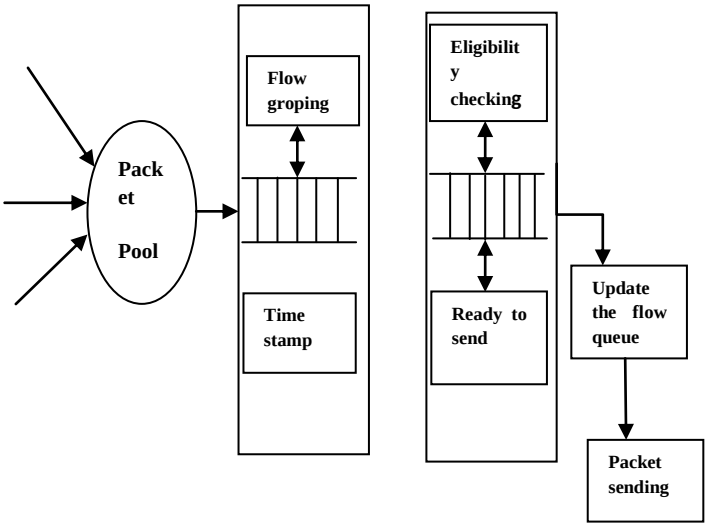

Fig. 1. System Design

A. Model System And Definitions

Here we outline the original WF2Q+ algorithm for a variable-rate system WF2Q+ is a packet scheduler that approximates, on a packet by- packet basis, the service provided by a work-conserving ideal fluid system that delivers the following, almost perfect bandwidth distribution over any time interval

$$W^k(t_1, t_2) \geq \phi^k W(t_1, t_2) - (1 - \phi^k)L$$

The packet and the fluid system serve the same flows and deliver the same total amount of work W(t) (systems with these features are called corresponding in the literature). They differ in that the fluid system may serve multiple packets in parallel, whereas the packet system has to serve one packet at a time, and is non preemptive. Because of these constraints, the allocation of work to the individual flows may differ in the two systems. WF2Q+ has optimal B-/T-WFI and O(logN) complexity, which makes it of practical interest. WF2Q+ operates as follows.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)), Volume 2, Special Issue 3, February 2014)
International Conference on Trends in Mechanical, Aeronautical, Computer, Civil, Electrical and Electronics Engineering (ICMACE14)

Each time the link is ready, the scheduler starts to serve, among the packets that have already started5 in the ideal fluid system, the next one that would be completed in the fluid system; ties are arbitrarily broken.WF2Q+ is a work-conserving on-line algorithm, hence it succeeds in finishing packets in the same order as the ideal fluid system, except when the next packet to serve arrives after one or more out-of-order packets have already started.

B. Packet Enqueue

Function enqueue(), is called on the arrival of a packet. As a first step, the packet is appended to the flow's queue, and nothing else needs to be done if the flow is already backlogged. Otherwise, the flow's timestamps are updated , and checks whether the group's state needs to be updated: this happens if the group was idle, or if the new flow causes the group's timestamp to decrease The update is done by , which possibly remove the group from the ineligible sets, and update the group's timestamps (being the slot size 2i, the start time calculation only needs to clear the last i bits of Sk). Once the group's timestamps are set, a constant-time bucket insert sorts the flow with respect to the other flows in the group. At this point, if needed, V (t) is updated according to the calculation. Finally, function compute group state() computes the new state of the group (which may have changed because of the new values of Si, Fi and V (t)), and puts the group in its new set.

C. Packet Dequeue

Function dequeue() is called to return the next packet to send. The packet selection is straightforward. If there are queued flows, at least one flow is eligible, so ER is not empty: a first FFS instruction picks the group with the lowest index in ER, then another FFS is used to locate the first flow in the bucket list, and the head packet from that flow is the next packet to serve. Before returning, the function updates the scheduler's data structures in preparation for further work. The flow's timestamps are updated, and the flow is possibly reinserted in the bucket list. Virtual time is increased in line 19, to reflect the service of the packet selected for transmission. Next ,the group's timestamps and state are updated. If the group has increased its finish time or it has become idle, it is moved to the new set, and function unblock groups() described possibly unblocks other groups.

Finally, make sure that at least one backlogged group is eligible by bumping up V if necessary, and moving groups between sets using function make eligible() which will be discussed next. Support Functions: The remaining support functions, mostly used in the dequeue() code. Function move groups() uses simple bit operations to move groups with indexes in mask from set src to set dest.

Function make eligible() determines which groups become eligible as V (t) grows after serving a flow. The properties of rounded timestamps are used to implement the check in constant time, which gives a graphical representation of the possible values of Si's and V (t), and the binary representations of V (t) (the vertical strings of binary digits). Since slot sizes are powers of two (i = 2i), the binary representation of the timestamps of the i-th group's ends with i − 1 zeros; in any given slot belonging to group i, the value of the i-th bit is constant during the whole slot. Whenever the i-th bit of V (t) changes, the virtual time enters a new slot of size i. As a consequence, on each V (t) update, the highest bit j that changes in V (t) indicates that all backlogged groups Gi, $i \leq j$ are now eligible.

This is exactly the algorithm implemented by function make eligible(): it computes the index j using a XOR followed by a Find Last Set (FLS) operation; then computes the binary mask of all indexes $i \leq j$, and calls function move groups to move groups whose index is in the mask from IR to ER and from IB to EB.

III. EXPERIMENTAL RESULTS

To analyse the performance of the proposed system lots of simulation experiments are conducted. The proposed system is implemented in Network Simulator (NS2). The proposed system is compared with the Droptail and RED. In the simulation experiments several parameters are used. They are listed in the below table.

TABLE I. LIMITATION OF SYSTEM

Number of Nodes	50
Area Size	1000 x 1000
Target Size	[500,500] x [500,500]
Simulation Duration	50
Queue Limit	20
Queue Size	100
Packet Size	552
Packet Interval	2

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)), Volume 2, Special Issue 3, February 2014)
International Conference on Trends in Mechanical, Aeronautical, Computer, Civil, Electrical and Electronics Engineering (ICMACE14)

To analysis the performance the proposed method several performance metrics are used. They are Enqued Packet Ratio, Sended Packet Ratio, Dropped Packet Ratio, Received Packet Ratio, Average packet delay and Average throughput.

A. Enqued Packet Ratio

Enqued Packets defines the total number of enqued packets from the available packets. This is one of the performance metrics. This metric is used to analyse the performance of this proposed method. It is calculated by using the below formula

EP=Total number of enqued packet/Total number of packets

The enqued packet ratio of the EFQ, Drop Tail and RED is shown in the below table

TABLE II. ENQUED PACKET RATIO

Network Size	Drop Tail	RED	EFQ
10	950	1000	1350
20	1200	1600	2000
30	1800	2300	2680
40	2400	2900	3200
50	3500	4000	4500

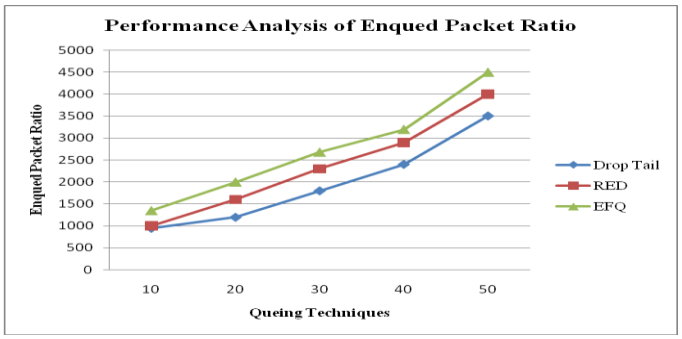

Fig. 2. Performance Analysis of Enqued Packet Ratio

The above figure shows that the EFQ method performs well than the other two methods such as RED and Drop Tail. Because the enqued packet ratio is higher for EFQ only. So its performance overcome the performance the Drop Tail and RED.

B. Sended Packet Ratio (SP)

Sended Packet Ratio defines the total no of sended packets from the available packets. This is one of the performance metrics. This metric is used to analyse the performance of this proposed method. It is calculated by using the below formula

SP= Total number of sended packet/Total number of packet

The sended packet ratio of the EFQ, Drop Tail and RED is shown in the below table

TABLE III. SENDED PACKET RATIO

Network Size	Drop Tail	RED	EFQ
10	840	900	1150
20	920	1100	1640
30	1600	2090	2420
40	2100	2390	2980
50	2300	2860	3200

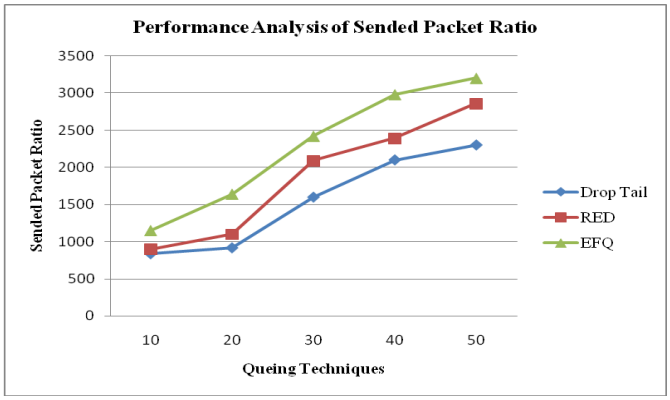

Fig. 3. Performance Analysis of Sended Packet Ratio

The above figure shows that the EFQ method performs well than the other two methods such as RED and Drop Tail. Because the sended packet ratio is higher for EFQ only. So its performance overcome the performance the Drop Tail and RED.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)), Volume 2, Special Issue 3, February 2014)
International Conference on Trends in Mechanical, Aeronautical, Computer, Civil, Electrical and Electronics Engineering (ICMACE14)

C. Dropped Packet Ratio (DP)

Dropped Packet Ratio defines the total no of dropped packets from the available packets. This is one of the performance metrics. This metric is used to analyse the performance of this proposed method. It is calculated by using the below formula

DP=Total number of dropped Packet/Total number of packet

The dropped packet ratio of the EFQ, Drop Tail and RED is shown in the below table

TABLE IV. DROPPED PACKET RATIO

Network Size	Drop Tail	RED	EFQ
10	640	587	200
20	932	863	400
30	1300	1023	650
40	2000	1090	890
50	2200	2380	1200

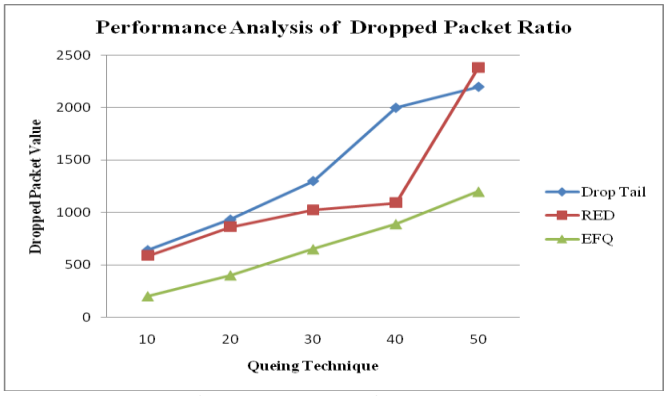

Fig. 4. Performance Analysis of Dropped Packet Ratio

The above figure shows that the EFQ method performs well than the other two methods such as RED and Drop Tail. Because the dropped packet ratio is lower for EFQ only. So its performance overcome the performance the Drop Tail and RED.

D. Received Packet Ratio (RP)

Received Packet Ratio defines the total no of received packets from the available packets. This is one of the performance metrics. This metric is used to analyse the performance of this proposed method. It is calculated by using the below formula

RP=Total number of Received Packet/Total number of packet

The received packet ratio of the EFQ, Drop Tail and RED is shown in the below table

TABLE V. RECEIVED PACKET RATIO

Network Size	Drop Tail	RED	EFQ
10	610	640	1148
20	820	876	1345
30	934	1013	1832
40	1213	1320	2400
50	1421	1632	2800

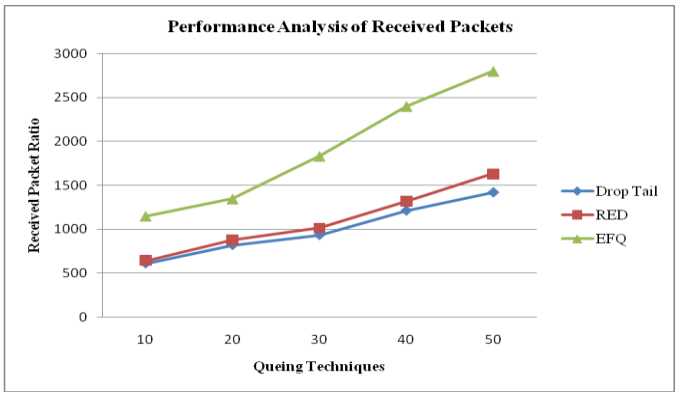

Fig. 5. Performance Analysis of Received Packet Ratio

The above figure shows that the EFQ method performs well than the other two methods such as RED and Drop Tail. Because the received packet ratio is higher for EFQ only. So its performance overcome the performance the Drop Tail and RED.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online)), Volume 2, Special Issue 3, February 2014)
International Conference on Trends in Mechanical, Aeronautical, Computer, Civil, Electrical and Electronics Engineering (ICMACE14)

E. Average Packet Delay (APD)

Average Packet Delay defines the total no of time to receive the packets to destination. This is one of the performance metrics. This metric is used to analyse the performance of this proposed method. It is calculated by using the below formula

APD= Ending Time - Starting Time

The average packet delay of the EFQ, Drop Tail and RED is shown in the below table

TABLE VI. AVERAGE PACKET DELAY

Network Size	Drop Tail	RED	EFQ
10	0.6428	0.3008	0.297
20	0.8325	0.4128	0.321
30	0.9267	0.6285	0.451
40	1.3727	0.7283	0.532
50	1.6283	0.8921	0.621

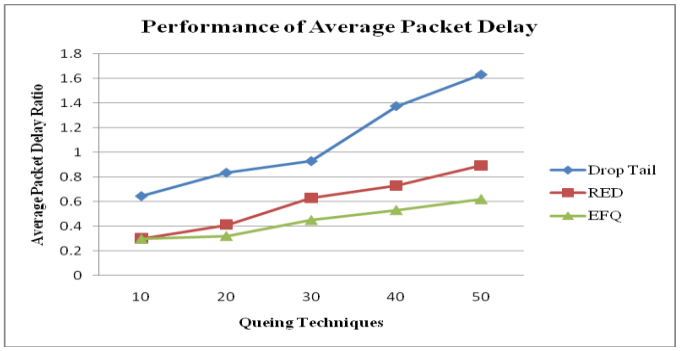

Fig. 6. Performance Analysis of Average Packet Delay

The above figure shows that the EFQ method performs well than the other two methods such as RED and Drop Tail. Because the average packet delay ratio value is lower for EFQ only. So its performance overcome the performance the DropTail and RED.

IV. CONCLUSION

In this paper EFQ, an approximate implementation of WF2Q+ is presented which can run in true constant time, with very low constants and using extremely simple data structures. The algorithm is based on very simple instructions, and uses very small and localized data structures, which make it amenable to a hardware implementation. Together with a detailed description of the algorithm, we provide a theoretical analysis of its service properties, and present an accurate performance analysis, comparing EFQ with a variety of other schedulers. The experimental results show that EFQ lives up to its promises: it is faster than other schedulers with optimal service guarantees, only two times slower than DRR, and operates, even in software, at a rate compatible with 10Gbit/s interfaces.

In this project the packet scheduling algorithm is worked out on the wired and wireless sensor network. In future this packet scheduling algorithm will apply on several networks and then the performance will analyzed. Not only that, In this project the performance of EFQ is compared with the Drop Tail and RED. In future several other queuing techniques will be considered for performance.

V. REFERENCES

[1] Fabio Checconi, Paolo Valente, and Luigi Rizzo. QFQ: Efficient Packet Scheduling with Tight Bandwidth Distribution Guarantees.

[2] Chuanxiong Guo. SRR: An O(1) time complexity packet scheduler for flows in multi-service packet networks. Proceedings of ACM SIGCOMM 2001, pages 211–222, August 2001.

[3] Martin Karsten. Approximation of generalized processor sharing with stratified interleaved timer wheels. IEEE/ACM Transactions on Network- ing, 18(3):708–721, 2010.

[4] Abdesselem Kortebi, Luca Muscariello, Sara Oueslati, and James Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. SIGMETRICS Performance Evaluation Review, 33(1):217–228, 2005.

[5] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. IEEE/ACM Transactions on Networking, 12(4):681–693, 2004.

[6] M. Shreedhar and George Varghese. Efficient fair queuing using deficit round robin. IEEE/ACM Transactions on Networking, 4(3):375–385, 1996.

[7] S.J.Golestani. A self-clocked fair queueing scheme for broadband applications. Proceedings of IEEE INFOCOM ’94, pages 636–646, June 1994.

[8] Donpaul C. Stephens, Jon C.R. Bennett, and Hui Zhang. Implementing scheduling algorithms in high-speed networks. IEEE Journal on Selected Areas in Communications, 17(6):1145–1158, June 1999.

[9] Dimitrios Stiliadis and Anujan Varma. A general methodology for designing efficient traffic scheduling and shaping algorithms. Proceedings of IEEE INFOCOM ’97, pages 326–335, April 1997.

[10] Jun Xu and Richard J. Lipton. On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms. IEEE/ACM Transactions on Networking, 13(1):15–28, 2005.

[11] P.Goyal, H.M. Vin, and H. Chen. Start-time Fair Queuing: A scheduling algorithm for integrated services. In Proceed-ings of the ACM-SIGCOMM 96, pages 157{168, Palo Alto, CA, August 1996.

[12] S.Keshav. A control-theoretic approach to ow control.In Proceedings of ACM SIGCOMM'91, pages 3{15, Zurich,Switzerland, September 1991.

[13] P. McKenney. Stochastic fair queueing. In Proceedings ofIEEE INFOCOM'90, San Francisco, CA, June 1990.

[14] O.Ndiaye. An e_cient implementation of a hierarchical weighted fair queue packet scheduler. Master's thesis, Mas- sachusetts Institute of Technology, May 1994.

[15] A.Parekh and R. Gallager. A generalized processor sharingapproach to ow control - the single node case. ACM/IEEETransactions on Networking, 1(3):344{357, June 1993.

[16] M.Shreedhar and G. Varghese. E_cient fair queueing using de_cit round robin. In Proceedings of SIGCOMM'95, pages 231{243, Boston, MA, September 1995.