Journal of Computer Science 5 (12): 1068-1074, 2009
ISSN 1549-3636

Bandwidth-Adaptive Scheduling for Quality of Service Enhancement of Real-Time Multimedia Applications in Network Processor Based Router

R. Avudaiammal and P. Seethalakshmi
Department of Information and Communication Engineering,
Anna University Tirucirappalli, Tamilnadu, India

Abstract: Problem statement: An explosive growth of multimedia applications in internet has stressed the performance of routers. Hence managing Quality of Service (QoS) enhancement of real-time multimedia applications over IP is a significant and demanding challenge. **Approach:** To address this issue, Bandwidth Adaptive Stratified Round Robin (BASRR) packet scheduling algorithm has been proposed in this paper for enhancing quality of service of real-time multimedia applications. Embedded Network Processors (NP) have recently emerged with flexibility and speed to reduce the stress of the router by effectively processing the packets. The main objective of this study was to implement the proposed packet scheduling algorithm in a Network Processor (NP) based router for enhancing quality of service of real-time multimedia applications **Results:** The effectiveness of the BASRR algorithm has been verified by simulations using Intel's IXP 2400 network processor. The results show that BASRR achieves about 71.25% reduction in jitter compared to SRR when the traffic has uniform distribution of real-time flows and non real-time flows. The reduction in average queuing delay is about 30% compared to SRR for all the types of traffic. **Conclusion:** The QoS for multimedia applications has been achieved by the proposed non-preemptive Bandwidth Adaptive Stratified Round Robin (BASRR) scheduling algorithm and outperforms the three well-known scheduling algorithms including DRR, WDRR and SRR. The results showed that BASRR is efficient with per packet complexity of O(1) and provides better fairness and reduced delay.

Key words: Multimedia, QoS, packet scheduling, BASRR, network processor, IXP2400

INTRODUCTION

Nowadays Internet traffic has become high due to the growth of real-time multimedia applications such as Video on demand, Video telephony, Video streaming and e-learning that requires QoS guarantees. Processing of the packets at router level such as Receiving IP packets from incoming links, classifying the packets, Scheduling, Routing and output porting are to be performed at high speed to satisfy the QoS requirements. Currently, Routers are mainly based on Application Specific Integrated Circuits (ASICs) that are custom made and not flexible enough to support diversified networking services. Earlier General Purpose Processor (GPP) based routers offer flexibility in supporting new features by simply upgrading the software, but have difficulties in supporting higher bandwidthv[1]. Embedded Network Processors have recently emerged to provide both the performance of ASICs and the programmability of GPPs[2]. Powerful Embedded Network Processors have been introduced by many companies that can be placed in routers to execute various network related tasks at packet level that supports QoS functionalities. The design and development of routers using NP has gained significance due to its high performance.

The research presented in this study is based on Intel ® IXP 2400 processor. It implements a high-performance parallel processing architecture on a single chip that is suitable for processing complex algorithms, detailed packet inspection, traffic management and forwarding at the wire speed. It has a set of hierarchically distributed memory devices, a set of on-chip processors (micro engines) to carry out packet level parallel processing operations through multitasking and multithreaded programming. Micro Engines can examine and forward packets independently without using the host processor, bus, or memory. All these features reveal that, one of the Micro Engines in NP can be assigned for efficiently implementing QoS aware scheduler. The Bandwidth Adaptive Stratified Round Robin (BASRR) scheduling

Corresponding Author: R. Avudaiammal, Department of Information and Communication Engineering,
Anna University Tirucirappalli, Tamilnadu, India

algorithm and other router functionalities have been implemented using these micro engines.

INTEL® IXP2400 network processor: Network Processors are designed to perform a wide range of functionalities such as multi-service switches, routers, broadband access devices and wireless infrastructure systems.

The Intel® IXP2400 is a member of the Intel's second generation network processor family. The architecture[3] of an integrated network processor IXP2400 shown in Fig. 1 has a single 32 bit XScale core processor, eight 32 bit Micro Engines (MEs) organized as two clusters, standard memory interfaces and high speed bus interfaces. Each micro engine has 256 general purpose registers, that are equally shared between eight threads. Micro engines exchange information through an on-chip scratchpad memory or via 128 special purpose next neighbor registers. Data transferring across the MEs and locations external to the ME, (for e.g., DRAMs and SRAMs) are done by the available 512 transfer registers. The Xscale core is responsible for initializing and managing the chip, handling control and management functions and loading the ME instructions. Each ME has eight hardware-assisted threads of execution with no context switch overhead. All threads in a particular ME execute code from the instruction stored on that ME, whose size is 4K. The SRAMs and DRAM are off chips that are shared by all processors. In general, SRAM is used for storing the table information such as routing table and the DRAM is used for packet buffering. Also the IXP2400 chip has a pair of buffers (BUF), Receive BUF (RBUF) and Transmit BUF (TBUF) that are used to send/receive packets to/from the network ports with each of size 8 Kbytes. The data in RBUF and TBUF is divided into sub blocks referred to as elements.

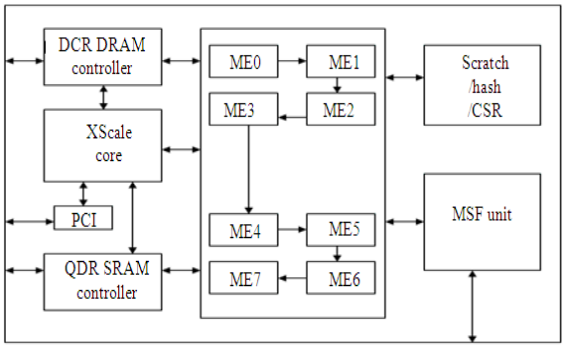

Fig. 1: Architecture of IXP2400

The other noteworthy features of the IXP2400 architecture include a hash unit, a scratchpad memory, a Control Status Register (CSR) Access Proxy, a Peripheral Computer Interface (PCI) controller, a Media Switch Fabric (MSF) Interface and DRAM and SRAM controllers. The packets are injected into the Network Processor from the network through the Media Switch Fabric Interface. Then the packet data and its meta data are kept in DRAM and SRAM respectively. The packets are then forwarded to the Micro Engines for processing. Finally the processed packets are driven into the network by Media Switch Fabric Interface (MSF) at output port.

Related work: Scheduling the packets over a shared network link is very essential in ensuring the QoS for multimedia applications because the queuing delay experienced by each packet at the intermediary router has a greater impact on the quality of multimedia services. The Scheduler should distribute the available network resources such as bandwidth and buffer space to provide a fair service to all flows. Research on design and implementation of optimized scheduling algorithms, has been carried out by many scholars using various techniques[6-23].

The fairest algorithm for packet scheduling is Generalized Processor Sharing (GPS)[9]. However, GPS is not a realistic algorithm, as the processing is carried out on a bit-by-bit basis rather than packet-by-packet. But, GPS serves as a reference scheduler to compare the performance of the practical packet scheduling algorithms that have been broadly classified as Deadline based schedulers, Round-Robin schedulers and Hybrid schedulers. The design of packet scheduler is characterized by an inevitable tradeoff among its features like fairness, delay and complexity of implementation[6]. The fairness of the algorithm can be described by proportional fairness index[14] and Worst case Fairness Index[11].

Deadline based schedulers[9-17] assign transmission deadline (timestamp) for each incoming packet using pre-computation and schedules the packet with most immediate deadline first[8]. But it starves the lower-priority traffic flow at the cost of prioritizing or guaranteeing the delay requirement of the highest priority traffic flow. These schedulers provide O(1) delay bound with O(N) complexity. It also provides constant PFI and WFI, where N is the number of competing flows.

Round Robin schedulers[18-21] assign time slots to flow in multiple ways of round robin technique and services the flows to achieve fairness. Deficit Round-Robin scheduling, algorithm[20] handles variable packet

sizes without knowing the mean packet size in advance. It provides strong rate differentiation as well as protection between flows by assigning quantum. Packets from different flows are queued in separate FIFO queues. A Deficit Counter (DC) is maintained by each queue to preserve the amount of quantum used for a service round. A queue is activated on the arrival of its first packet and its DC is set to its allocated fixed quantum. If the packet length of HoL is lesser than the quantum size then the packet will be forwarded and its DC is decremented according to the size of the packet sent. If the packet length is greater than the quantum size then its value will be stored in deficit counter to hold the current unused portion of allocated bandwidth which can be used to send a packet of large size in the next round. Once serviced, a flow has to wait for (N-1) other flows. DRR services next queue when the head of line packet in the current queue becomes empty or there is insufficient quantum for serving. When all the queues have been serviced a new service round begins. In the next round the value of the deficit counter of all the queues is the sum of the credit of the previous round and the allocated fixed quantum. Thus in DRR, the unused portion of quantum value is carried over to the next round as the value of DC. This preservation method ensures fairness. But during each round, each flow transmits its entire quantum, which leads to Poor delay and Output burstiness. This minimizes the delay of packets being serviced in the queue at the expense of increasing the delay of other queues waiting to be served which leads packet of other queues to miss their deadlines. In other words, it does not provide delay guarantee Its Computational complexity is O(1) and delay bound is O(N). It provides constant PFI and non-constant WFI.

Weighted Deficit Round robin (WDRR)[21] works like Deficit Round Robin except that each queue is assigned with different weights. It performs well when all packets have the same size. It provides unfair bandwidth allocation. It also doesn't guarantee delay differentiation.

Hybrid schedulers combine the best features of Round Robin schedulers and deadline based schedulers. The deadline based approach is used to schedule between queues and round robin approach is used to schedule within a queue.

Stratified round robin scheduling[22,23] is one of the Hybrid schedulers. It bounds the unfairness by stratifying the traffic into different flow classes based on their bandwidth requirements. It chooses one class from 'n' flow classes to schedule next based on deadline mechanism and within a flow class, performs service based on WDRR. The another advantage of this stratification is that it simplifies the scheduling decision that is to be taken among the flows of a flow class through interleaved weighted round robin because all flows belonging to a particular flow class have approximately equal weights. The weight assigned in such a way that at least one packet is guaranteed to be sent every time a flow is assigned a slot. It also faces serving bottleneck of deadline based schedules such as WFQ[18]. The important difference is that while WFQ selects the earliest deadline flow among N flows, SRR selects among 'n' flow classes. SRR has a constant packet delay of O(1), whose computational complexity is also close to O(1). But SRR does not provide delay guarantee for real-time multimedia applications.

Summarizing schedulers, any algorithm that has involved the calculation of a deadline is inherently complex. Still for any scheduling algorithm to support QoS as well as fairness, some degree of time dependent parameter needs to be introduced.

The proposed Bandwidth Adaptive Stratified Round Robin (BASRR) is a weight based service discipline that has been developed to minimize the queuing delay of real-time multimedia applications and to utilize the bandwidth effectively. To achieve this, it classifies the packets into different flows according to a time sensitive measure as well as bandwidth requirement and then schedules it.

MATERIALS AND METHODS

Bandwidth adaptive SRR algorithm: Bandwidth Adaptive Stratified Round Robin scheduling technique is a non-preemptive weight based scheduling algorithm that addresses three different goals such as rate differentiation and delay differentiation among flows as well as delay guarantee for real-time multimedia applications. The main objective of BASRR is to minimize the queuing delay for real time multimedia applications. Internet traffic has different mixes of elastic applications and QoS applications. The service requirements of these applications on an Internet are mainly along two dimensions namely delay and bandwidth. Based on these service requirements, the flows are stratified into three Flowclasses namely F1 referring real time multimedia traffic flows (RTP/UDP/IP), F2 referring non real time traffic flows (UDP/IP) and F3 referring Best-Effort flows (TCP/IP). Each flowclass has a number of queues to hold the packets of different flows.

Flowclass F1 has been assigned highest priority and Flowclass F3 the lowest priority to achieve delay differentiation among the Flowclasses. BASRR provides rat differentiation among the Flowclasses by assigning the weight R_j:

$$\sum_{j=1}^{3} R_j \leq R$$

where, R, is the total capacity of the link that is shared by the Flowclasses.

BASRR services all the Flowclasses in an interleaved manner to reduce the time taken to do the scheduling decision which reduces the overall delay experienced by each packet. BASRR assigns time slots to Flowclasses as {F1 F2 F1 F2 F1 F3} which shows that $R_1 = R/2$, $R_2 = R/3$ and $R_3 = R/6$.

The bandwidth allotted to the Flowclass is shared among the flows to achieve rate differentiation among the flows. BASRR chooses one of the backlogged flows from the Flowclass F1 based on the dead line mechanism to meet delay guarantee, the most important goal. Each queue of Flowclass F1 has a time interval associated with it. Packets of real-time multimedia applications are stored in the queue that corresponds to the time slice containing the virtual departure time (deadline) of that packet in FIFO order.

Let Flowclass F1 has two queues namely F1Q1 and F1Q2 to store the packets with different deadline time interval say (d1, d2) and (d2, d3) respectively. If there is a backlog then BASRR transmits the packets in F1Q1 first during the timeslot of FlowclassF1. If F1Q1 becomes empty and the reserved rate of FlowclassF1 has not been exhausted then BASRR starts serving F1Q2. BASRR moves to the next Flowclass when both the Queues F1Q1 and F1Q2 are empty or the reserved rate for that timeslot is over.

Within Flowclass of F2 and F3, each flow reserves a amount of its bandwidth requirement by occupying a number of slots in a frame in an interleaved manner.

The weight 'w_i' of flow 'f_i' of Flowclass F_j is defined as its reserved bandwidth normalized with respect to the total bandwidth of each Flowclass 'R_j'. If there are 'm' flows altogether in F2 and in F3:

$$w_i = r_i/R_j \; i = 1 \text{ to } m: j \neq 1 \qquad (2)$$

If there is no backlog for a Flowclass F1 or F2 during their current time slot, the backlogged flows of Flowclass F3 are transferred in the same order then the handling of flowclasses is same as that of WRR. Thus BASSR utilizes the bandwidth effectively and supports throughput guarantee even for best-effort flows.

Implementation: The framework for the implementation of the Non-preemptive Bandwidth Adaptive Stratified Round Robin scheduling algorithm (BASRR) is shown in Fig. 2. The framework has four microblocks namely Packet Receive, Classifier, Scheduler and Packet Transmit.

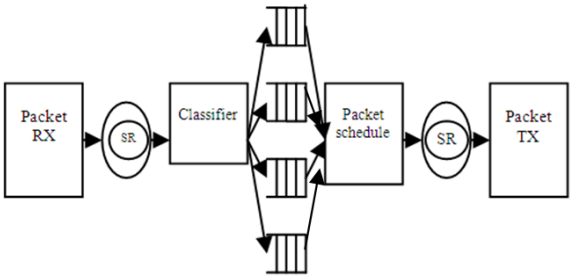

Fig. 2: Software framework design

Receive Microengine has been interfaced with the Media Switch Fabric Interface (MSF). The packets are injected through Media switch Fabric Interface and Receive Buffer (RBUF) reassemble incoming m-packets (packets with length 64 bytes). For each packet, the packet data is written to DRAM, the packet meta-data (offset, size) is written to SRAM. The Receive Process is executed by Micro Engine (ME0) using all the eight threads available in that Micro Engine (ME0)[4]. The packet sequencing is maintained by executing the threads in strict order. The packet buffer information is written to a scratch ring for use by the packet processing stage. Communication between pipeline stages is implemented through controlled access to shared ring buffers.

Classifier microblock is executed by Micro Engine (ME1) that removes the layer-3 header from the packet by updating the offset and size fields in the packet meta descriptor. The packets are classified into different traffic flows based on the IP header. Packet meta data of packets from different flows are queued in separate FIFO queues.

When multiple queues are backlogged at the output of Classifier, Packet Scheduler microblock is executed by Micro Engine (ME2) which schedules the queue based on the proposed algorithm and places the scheduled packets in the output virtual queue in order to reduce queuing delay.

To handle the multiple backlogged queues at the output of Classifier, Packet Scheduler microblock has been implemented. Since the packet scheduler is a context pipe-stage that is implemented as a microblock that runs on Micro Engine (ME2) to schedule the packets based on the proposed BASRR algorithm. It places the scheduled packets in the output virtual queue. The packet scheduler is a frame-based scheduler which means that it schedules a complete packet at a time. Scheduler thread is responsible for finding a eligible queue and sending a dequeue request to the micro engine handling the queues to schedule the packet.

Micro Engine (ME3) executes the Packet Transmit microblock to segment a packet into m-packets and moves them into Transmit Buffer (TBUFs) for transmitting the packets over the media interface. In this study, microblocks are implemented using microcode. Implementation assigns individual blocks from the fast path pipeline to separate micro engines on the IXP2400 NPs.

IXA 3.51 SDK is a cycle based simulator in which IXP2400 is set to run under the following conditions: PLL output frequency: 1200 MHz, ME frequency: 600 MHz, Xscale Frequency: 600 MHz. SRAM frequency: 200 MHz, two channels, 64 MB per channel, DRAM Frequency: 150 MHz, 64 MB.

Real-time Multimedia stream is simulated with UDP flows as voice and video applications use UDP rather than TCP. Traffic generated in this study consists of RTP/UDP/IP packets, UDP/IP packets with different ToS and TCP packets. Real-time flows will deliver the packets in bursty in one time and zero number of packets in other time whereas non real-time packets will come continuously. Hence different data stream files are created by changing the percentage of these packets in the traffic namely Type 1-3 to analyze the algorithm. Type 1 Traffic has been characterized as the uniform distribution of the flows. Type2 Traffic has 50% of Best Effort Flows where as in Type 3, it has been reduced to 10%. Traffic is generated at a constant rate of 1000 Mb sec^{-1}. The data is injected to the Network Processor through Media Switch Fabric (MSF) interface using network traffic assignment functionality of workbench simulator. Then the flows are analyzed by measuring the parameters delay, Jitter and throughput.

RESULTS AND DISCUSSION

In this study, the performance evaluation of the Bandwidth Adaptive Stratified Round Robin scheduling algorithm (BASRR) has been carried out and the metrics delay, Jitter and throughput have been compared with DRR, WDRR and SRR algorithm under different traffic types 1-3. The average queuing delay incurred by SRR and BASRR scheduling algorithms for these three traffic types is shown in Table 1.

The results show that the delay is slightly high for traffic type 1 compared to other traffic type 2 and 3 because in the traffic type 1 flows are uniformly distributed. If the number of real-time flows is increased as in traffic type 3 compared to traffic type 2 then the delay reduction is more because the real-time flows are competing only with the other real-time flows that have been arrived earlier. As the traffic type 2 comprises of 50% real-time traffic and the remaining of non real-time traffic, its delay is 32.14% lesser than SRR. It is evident from the delay results that BASRR achieves superior delay guarantees compared to SRR for all the traffic types.

The Jitter experienced by the flows of various Traffic types for the DRR, WDRR, SRR and BASRR scheduling algorithms is shown in Fig. 3. The obtained results show that the level of jitter is same for both the algorithms in the case of Traffic type 2 and Traffic type 3 where as it is reduced by 71.25% for the Traffic Type 1 in BASRR compared to SRR. The huge reduction in jitter, is one of the major achievements of BASRR algorithm.

The work complexity of a scheduler is defined as the order of time complexity with respect to enqueuing and dequeuing a packet for transmission. In this algorithm, the dequeuing procedure determines the next flow to be served by calculating the credit counter and removing the packet from that flow. As it can be done in constant time, the time complexity of dequeue operation is O(1). After removing the packet from the flow it enqueues it in the egress ring for transmission. It is also done with time complexity of O(1). As the complexity of both enqueuing and dequeuing is O(1), the work complexity of BASRR scheduler is O(1).

The number of packets serviced by the DRR, WDRR, SRR and BASRR scheduling algorithms have been observed for the period of 50,000 cycles and is shown in Fig. 4. BASRR scheduling algorithm is able to service more number of packets in a given period of time compared to DRR, WDRR and SRR under all traffic types because of its low complexity.

Table 1: Comparative queuing delays

	Average queuing delay (μ sec)			
Traffic type	DRR algorithm	WDRR algorithm	SRR algorithm	BASRR algorithm
Type 1	14.08	15.97	21.90	15.58
Type 2	20.80	14.40	12.60	8.55
Type 3	15.67	12.80	12.78	9.50

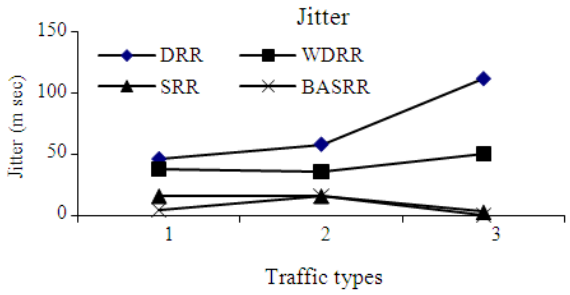

Fig. 3: Jitter of real-time flow with various traffic types

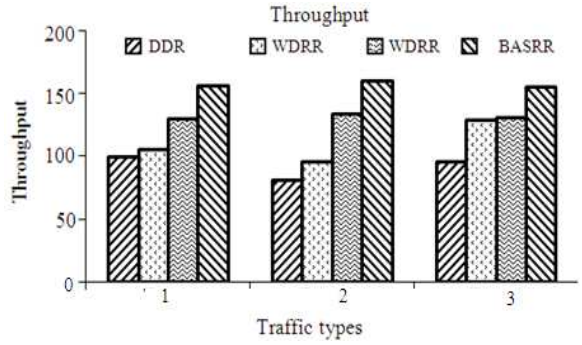

Fig. 4: Throughput of different traffic types

The results show that the BASRR algorithm performs better in terms of Average Queuing Delay, Jitter and Throughput compared to DRR, WDRR and SRR for the various types of traffic.

CONCLUSION

Bandwidth Adaptive Stratified Round Robin scheduling technique is a non-preemptive weight based scheduling algorithm that achieves three different goals such as rate differentiation and delay differentiation among flows as well as delay guarantee for real-time multimedia applications. Bandwidth Adaptive Stratified Round Robin scheduling algorithm (BASRR) has been implemented successfully using Intel IXP2400 Network Processor. The results show that BASRR achieves about 71.25% reduction in jitter compared to SRR when the traffic has uniform distribution of real-time flows and non real-time flows. The reduction in average queuing delay is about 30% compared to SRR for all the types of traffic. On analyzing the performance metrics, it is clear that BASRR provides better fairness and reduced delay with minimal work complexity of O(1) compared to the contemporary scheduling algorithms. Thus BASRR is efficient in enhancingQoS for multimedia applications and can be practically deployed in Embedded Network Processor based high speed routers.

REFERENCES

1. Coss, M. and R. Sharp, 2004. The network processor decision. Bell Labs Tech. J., 9: 177-189. DOI: 10.1002/bltj.20012
2. Comer, D.E., 2003. Network Systems Design Using Network Processors. First Indian Reprint, Pearson Education, ISBN: 81-7808-994-7.
3. Intel IXP 2400/IXP2800, 2003. Network processor "hardware reference manual". Intel Corporation. http//www.bupt-bcnl.com/ixa/doc/IXP2400HRM.pdf
4. Intel IXP 2400/IXP2800, 2003. Network processor "programmer's reference manual". Intel Corporation. http://www.bupt-bcnl.com/ixa/doc
5. Johnson, E. and A. Kunze, 2003. Programming the Complete Microengine Coding Guide. Intel Press, ISBN: 097178616X, pp: 350.
6. Zhang, H., 1995. Service disciplines for guaranteed performance service in packet-switching networks. IEEE, 83: 1374-1396. DOI: 10.1109/5.469298
7. Stiliadis, D. and A. Varma, 1998. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. IEEE Trans. Network., 6: 611-624. DOI: 1063-6692(98)08403-9
8. Figueira, N.R. and J. Pasquale, 1997. A schedulability condition for deadline-ordered service disciplines. IEEE/ACM Trans. Network., 5: 232-244. DOI: 10.1109/90.588088
9. Parekh, A.K. and R.G. Gallager, 1993. A Generalized processor sharing approach to flow control in integrated services networks: The single node case. IEEE/ACM Trans. Network., 1: 344-357. http://portal.acm.org/citation.cfm?id=159914
10. Demers, A., S. Keshav and S. Shenker, 1989. Analysis and simulation of fair-queuing algorithm. Proc. ACM Symposium on Communications architectures and Protocols, Sept. 25-27, ACM Press, Austin, Texas, United States, pp: 1-12. http://portal.acm.org/citation.cfm?id=75248
11. Bennett, J.C.R. and Zhang, 1996. WF^2Q: Worst-case fair weighted fair queueing. Proceedings of the IEEE 15th Annual Joint Conference of the IEEE Computer Societies: Networking the Next Generation, Apr. 24-24, ResearchGATE, pp: 120-128. DOI:10.1109/INFCOM.1996.497885.
12. Zhang, L., 1990. Virtual clock: A new traffic control algorithm for packet switching networks. ACM Trans. Comput. Syst., 9: 19-29. DOI: 10.1145/103720.103721
13. Mckenney, P., 1991. Stochastic fairness queuing: In Internetworking. Res. Exp., 2: 113-131. http://www2.rdrop.com/users/paulmck/scalability/paper/sfq.2002.06.04.pdf
14. Golestani, S.J., 1994. A self clocked fair queuing scheme for broadband applications. Proceeding of the IEEE Conference on Networking for Global Communications, June 12-16, IEEE Xplore Press, Toronto, Ont., Canada, pp: 636-646. DOI: 10.1109/INFCOM1994.337677

15. Suri, S., G. Varghese and G. Chandranmenon, 1997. Leap forward virtual clock: A new fair queuing scheme with guaranteed delays and throughput fairness. Proceeding of the 16th Annual Symposium on Principles of Distributed Computing, Aug. 21-24, ACM Press, Santa Barbara, California, United States, pp: 281. http://portal.acm.org/citation.cfm?id=259380.259482
16. Stankovic, J.A.. M. Spuri, K. Ramamritham and G.C. Buttazzo, 1998. Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms. 1st Edn., Springer, ISBN: 10: 0792382692, pp: 296.
17. Stiliadis, D. and A. Varma, 1998. Efficient fair queuing algorithm for packet-switched networks. IEEE/ACM Trans. Network., 6: 175-185. DOI: 10.1109/90.664266
18. Greenberg, A.G. and N. Madras, 1990. How fair is fair queuing? J. ACM, 39: 568-598. http://portal.acm.org/citation.cfm?id=146658
19. Katevenis, M., S. Sidiropouls and C. Courcoubetis, 1991. Weighted Round Robin cell multiplexing in a general-purpose ATM switch chip. IEEE J. Select. Areas Commun., 9: 1265-1279. DOI :10.1109/49.105173
20. Shreedhar, M. and G. Varghese, 1996. Efficient fair queuing using deficit round robin. ACM SIGCOMM Computer Commun. Rev., 25: 231-242. DOI: 10.1145/217391.217453
21. Wong, M.Y.L. and C.K. Li, 2002. Low computational complexity weighted queuing using weighted deficit round robin. Proceedings of the IASTED applications Information, Feb. 18-21, Innsburg, Austria. http://www.actapress.com/Content_of_Proceeding.aspx?proceedingid=382
22. Chaunxiong, G., 2001. SRR: An O(1) time complexity packet scheduler for flows in multi-service packet networks. Proceeding of the 2001 conference on Applications, Technologies, Architectures and Protocols for Computer Communications, Aug. 2001, ACM Press, San Diego, California, United States, pp: 211-222. http://portal.acm.org/citation.cfm?id=383076
23. Ramabhadran, S. and J. Pasquale, 2006. The stratified round robin scheduler: Design, analysis and implementation. IEEE/ACM Trans. Network., 14: 1362-1373. DOI: 10.1109/TNET 2006.886287.