An Efficient Scheduling Algorithm for Input-queuing ATM Switches

*Bin Li *, Mounir Hamdi ** and Xi-Ren Cao **

*** Department of Electrical and Electronic Engineering**
and ** Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Email: eebin@ee.ust.hk, hamdi@cs.ust.hk and eecao@ee.ust.hk

Abstract: *In this paper, we propose and evaluate a three-phase scheduling algorithm for input-queuing ATM switches, and we term it WRRLA algorithm. The WRRLA algorithm is an improvement over the conventional Weighted Round Robin (WRR) algorithm by simply combining it together with the Look Ahead (LA) technique. In a WRRLA scheduling operation, the WRR algorithm is first applied to schedule the time-sensitive traffics in order to guarantee their quality of service, and then the LA technique is applied to schedule the data traffics in order to increase the throughput of the ATM switches. Simulation results show that the WRRLA algorithm achieves good performances in simple architecture, less scheduling computation, higher efficiency (throughput) and lower delay bounds. Moreover, WRRLA compares favorably with other famous scheduling algorithms, such as PIM and WRR.*

I. Introduction

The performances of the communication networks progress very fast in the last decade due to the technological advances. Main technique features of a communication network, such as switching speed and bandwidth capacity, have all grown rapidly in following either linearly or exponentially curves. Most communication networks in the near future will be operated over high-speed mediums and be run from several hundred Mbps to several ten Gbps. Such a packet-switched based Broadband Integrated Service Digital Network (B-ISDN) will not only support data communications like today's computer networks, but will also support those broadband multimedia services, including Video on Demand (VOD) and Video Conference (VCF).

Maximizing bandwidth utility and providing service performance guarantees are two most important objectives in designing a communication network. When a network is set up, the bandwidth is always there no matter whether they will be utilized. Try to maximize the network bandwidth utility benefits both network providers and users. On the other hand, the users typically want to have the guaranteed Quality of Service (QOS), and delay bounds guaranteed is the most important feature [1] [2]. Normally, average delay is a common measurement in determining whether or not customers are getting their QOS requirements. This measurement is inadequate to broadband multimedia services other than end-to-end delay bound. A simple but valid method to estimate the end-to-end delay bound is that: to analysis the worst-case local delay at each switch system independently, and bound the end-to-end delay of a connection by summing the local delay bounds at all switches traversed by the connection.

Asynchronous Transfer Mode (ATM) is the core technique in building future's B-ISDN. Except the above network designing objectives, ATM network is also desirable to be able to allocate its bandwidth according to the needs of individual sources. To achieve all these objectives, developing a good switch scheduling algorithm with simple, fast, efficient, and performance bounded properties are very important. In the past ten years, many scheduling algorithms have been proposed to meet one or more objectives that we mentioned above. Examples are FQ [3], WFQ (PGPS) [4], WF^2Q [2], WRR [5], SCFQ [6], PIM [7], WPIM [8], Virtual clock [9], and FIFO+ [10]. Some of them, such as the PIM algorithm, has been implemented to schedule a practical ATM switch in hardware [7].

Most emerged scheduling algorithms have natural drawbacks that decrease the performance. For example, most scheduling algorithms do not distinguish between time-sensitive traffics and data traffics while scheduling. Thus it is difficult for them to guarantee the end-to-end delay bound requirements of time-sensitive traffics. Also, the simple and fast algorithms, such as FQ and its varieties, are operated in the round-robin service order, which will carry out the unfairness and Head Of Line (HOL) blocking problems. On the other hand, the high efficiency algorithms, such as PIM scheme and its varieties, will take a relative complex computation while scheduling, and it is impossible to be applied to large scale ATM switches. Further more, some emerged scheduling algorithms can only provide

The works reported here was supported in part by Hong Kong UGC under grant HKUST RI 93/94. EG10.

0-7803-4443-X/97/$10.00 © 1997 IEEE

probabilistic upper delay bound, or even if the firm upper delay bound can be guaranteed , the value is often too large to accept. We argue a new scheduling algorithm that can guarantee better QOS.

In this paper, we propose and emulate a new three-phase switch scheduling algorithm by mixing the distributed Weighted Round-Robin (WRR) algorithm together with the Look Ahead (LA) technique, and we name it WRRLA algorithm. Because WRR has both bandwidth and delay guaranteed property, and LA technique is very helpful to increase the throughput when HOL blocking occurs, therefore, the WRRLA algorithm will have better synthetic performances, and is suitable to schedule those input-queuing ATM switches in providing broadband multimedia services. The WRRLA algorithm works in three phases: the initial phase, the WRR scheduling phase, and the LA scheduling phase. Detail description and performance analysis will be studied in the rest of the paper.

This paper is organized as follows: section II gives some necessary backgrounds that we may need in the study. Section III talks about the proposed WRRLA scheduling algorithm in detail. Section IV presents the performance evaluation of the WRRLA algorithm. Some conclusion remarks are listed in section V.

II. Backgrounds

II.1 System and traffic model

We consider a single ATM switch system with arbitrary topology of using the non-blocking crossbar switch fabric. Transmission links, input buffers, traffic shapers, and switch scheduling servers are sequentially connected to it. The system configuration is shown in ***Figure 1***. By using a non-blocking ATM switch architecture, queuing occurs only due to the output blocking of the switch. The ATM switch is also assuming to transmit cells in a slot format, and the length of a time slot equals to the transmitting time of a 53-byte ATM cell. An ATM network can be considered to contain a number of such single ATM switch system. As we mentioned above, once the local performance of each single ATM switch system is guaranteed, the end-to-end performance within the network environment will be guaranteed.

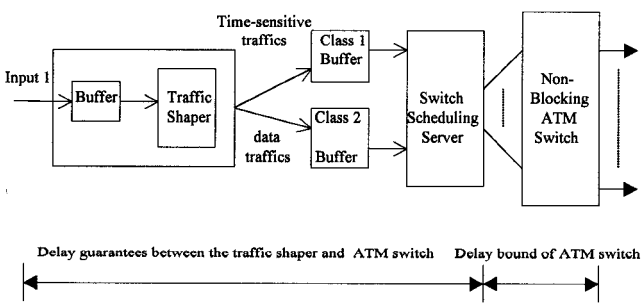

Figure 1. A single ATM switch system model

In our study, the incoming traffic is classified into two classes: Class 1 denotes the time-sensitive real-time traffic, such as voice and video. Class 2 denotes the non-real-time data traffic, such as file transfer. They are distinctively treated in our study. Centrally, time-sensitive traffics will always have the higher priority, which means enough resource is always allocated to it first. A modified Leaky Bucket (LB) shaper is used to attain the desired traffic characteristics after the traffic enters the system. The details are described in Section III.

II.2 A brief overview of emerged scheduling algorithms

- **HOL blocking and LA technique**

An input-queuing packet-switching system always suffers from the Head Of Line (HOL) blocking problem when two or more inputs happen to transfer its traffic to the same output at the same time. HOL blocking increases the end-to-end delay and reduces the throughput of the switch. Normally, the maximum throughput of such a system does not exceed 58% when the traffic load is high [7].

A very powerful technique to reduce the HOL negative effect is the Look Ahead (LA) technique. The principle of LA technique is very simple. The switch checks the cells that behind the HOL cells in the blocked inputs, and sends them to the proper collision free destinations. Our studies show that, when the LA window size reaches 8 or above, the throughput of an input-queuing system can achieve 89% or more.

- **PIM and its varieties**

Parallel Iterative Matching (PIM) is a hardware implemented scheduling algorithm first developed by DEC engineers. PIM and its varieties are a representative class of scheduling algorithm with very high scheduling efficiency. The original PIM is a three-phase scheduling algorithm uses parallelism, randomness, and iteration to accomplish higher efficiency. Running original PIM algorithm in some fixed number of iterations, for example, 4 iterations, an input-queuing ATM switch can achieve a very high throughput, such as more than 97% [7]. Some varieties of PIM are also proposed in the past years. For example, S. Motoyama, D. W. Petr and V. S. Frost designed a scheduling algorithm similar to PIM [11]. The modifications of their approach benefit on eliminating the complex accept phase of original PIM and the iteration convergence is fast. According to their study, the proposed algorithm has almost the same or slightly better throughput than original PIM algorithm, but the implementation is simplified. Also, D. Atiliadis and A. Varma proposed the Weighted Probabilistic Iterative Matching (WPIM) algorithm that allows flexible allocation of bandwidth among the switch inputs sharing a common output link simply [8]. WPIM can provide both bandwidth and delay guaranteed to a connection of a single node. The hardware overhead of its implementation is modest.

- **FQ and its varieties**

Fair Queuing (FQ) algorithm was first developed by Nagle [12] and then substantially revised and refined by Demers, Keshav, and Shenker [3]. FQ algorithm and its varieties are a representative class of scheduling algorithm with simple architecture and firm performance bounds. Original FQ algorithm gives every host the same fraction of the bandwidth in a round-robin service order, in either packet-by-packet or bit-by-bit method. FQ algorithm can guarantee the firm upper delay bounds. Some extensions of FQ algorithm are also proposed. For example, a simple extension to FQ is to assign a different weight to each source, where the weighting is associated with how long the packet will have to wait in the queue. This extension is called Weighted Fair Queuing (WFQ) algorithm, also known as Packetized Generalized Processor Sharing (PGPS) algorithm. One exciting result of WFQ was done by Parekh [4], who combined Leaky Bucket to work with WFQ and gave out the upper delay bounds, although the value of the bound is quite large. Worst-case Fair Weighted Fair Queuing (WF^2Q) is another extension to FQ developed by J.C.R Bennett and H. Zhang [2]. The different between WFQ and WF^2Q does not affect the end-to-end delay bounds they provide.

- **WRR**

Another representative scheduling algorithm with firm performance guaranteed is the Weighted Round Robin (WRR) algorithm. WRR algorithm is first applied in this area in [13], and then be well studied by K. Mezger, David W. Petr, and T.G. Kelley [5] [14]. This algorithm is designed to allocate bandwidth by controlling the amount of time that a particular queue accesses to the server. Which means it provides average delay guaranteed while bandwidth guaranteed. The principle of WRR algorithm is also very simple. Each customer is guaranteed a certain amount of bandwidth. If a customer does not use its bandwidth at a specific time slot, it is automatically divided up among those customers that need more bandwidth until the original customer needs his allocated bandwidth again. WRR algorithm provides both full minimum bandwidth guaranteed and part of maximum delay bounds to a special connection.

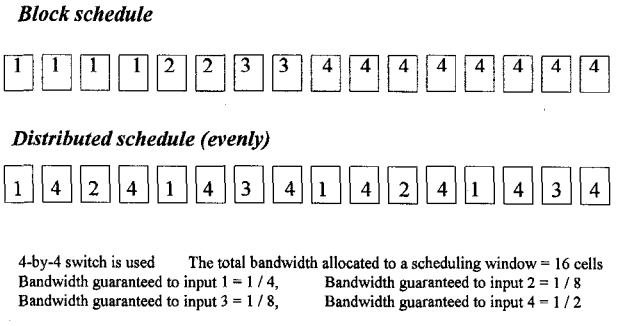

Figure 2 Examples of WRR scheduling algorithm

WRR algorithm works as follows: at the beginning of a transmission slot, the WRR schedule is checked to see which customer is the next one to be served. If that customer's queue is empty, the next customer in the schedule is checked and so on so forth. The schedule order is therefore set up and the bandwidth guarantees are preserved. Once the last customer in the schedule is served, the WRR schedule is reset to the top and the algorithm goes through the schedule again [5]. The WRR schedule is always set up in one of the two methods: the block schedule or the distributed schedule, as be shown in ***Figure 2***. The block schedule gives a particular queue all of its slots in sequence without moving to another queue, and the queue waits to be served again until all the other queues have a chance to get their services. The distributed schedule distributes the dedicated slots for a given queue throughout the schedule. Compared with back-to-back block schedule, this distributed schedule serves few customer cells back-to-back yet waits a shorter amount of time between two service times.

It is easy to image the internal relationships between FQ, WFQ, and WRR algorithm. The main difference between FQ and WFQ is that: WFQ assigns a different weight to each queue according to how long the packet has to wait in the queue, while FQ allocates the same weight to each customer. The main difference between WRR and WFQ is that: WFQ is theorized as minimizing average delay whereas WRR can guarantee both bandwidth and delay to a given queue. Recent studies also show that the different of system delay between WRR and WFQ proves to be insignificant [15]. Since the round-robin service nature, all of them suffer from the low efficiency problem. This is because, although enough bandwidth have been guaranteed to each queue, due to the HOL blocking, some empty time slots (or say unused bandwidth) may left after each matching. Our WRRLA algorithm tries to utilize these left time slots to increase the throughput.

III. The WRRLA algorithm

As we mentioned above, WRR is a good scheduling algorithm except its lower efficiency due to HOL blocking; and LA technique is a powerful technique to increase the throughput when HOL blocking occurs. Thus, the combination of these two algorithms, which we name it WRRLA scheduling algorithm, will eliminate their drawbacks, take their advantages, and becomes a scheduling algorithm with better synthetic performance.

A WRRLA scheduling server contains a WRR sub-server, a LA sub-server, and some buffers. The WRRLA scheduling operation consists of three phases: the initial phase; the WRR scheduling phase; and the LA scheduling phase. A modified Leaky Bucket shaper is placed in front of WRRLA server to do two things: traffic identification and traffic shaping. The system configuration is shown in ***Figure 3***. The parameters of the shapers are selected based on the following rules: the parameters of LB 1 are selected according to what should be guaranteed to the input links

and the time-sensitive flows; and the parameters of LB 2 are selected according to the parameters of LB 1. For example, suppose, B_{Nm} and B_{Nd} is the depth of bucket of LB shaper 1 and 2 at input link N, respectively; R_N is the total bandwidth allocated to input link N; R_{Nm} is the guaranteed bandwidth to the time-sensitive traffics of input link N, then the bandwidth to be allocated to data traffic of input link N is $R_{Nd} = R_N - R_{Nm}$.

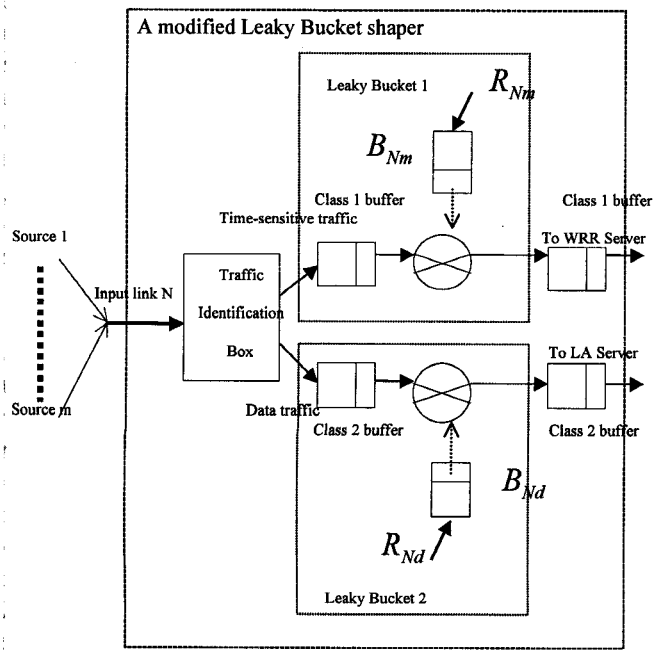

Figure 3 A WRRLA server

The three-phase operation of the WRRLA scheduling algorithm is outlined as follows:

1. **The initial phase**, in which is to set up the parameters of the WRRLA server. Since the WRR algorithm here is using evenly distributed scheduling window, the WRR sub-server will distribute the dedicated slots to each Class 1 traffic throughout the scheduler according to their weight (or say the guaranteed bandwidth), and the order of WRR service will follow the scheduling window.

2. **The WRR scheduling phase**, in which a WRR sub-server schedule the feasible connections to all queued Class 1 cells. Because of using evenly scheduling window approach, once the WRR window threshold is reached, a WRR scheduling phase is finished and start the third phase. Also, once all slots in the scheduling window are severed, a new window will be set up. The detail of this phase is that:

> 2a) At the beginning of this phase, the WRR sub-server checks the 1st Class 1 buffer according to the scheduling window. If there is any queued cell, take the first one and try to match it with the correct outputs. After such processing or if there is not any queued cell, the WRR sub-server moves to check the 2nd Class 1 buffer.
>
> 2b) The WRR sub-server do the same operation to every Class 1 buffer one by one according to the scheduling window until it reaches the WRR window threshold. If at this moment, there does not exist any empty output, stop the operation, do routing and then start a new operation. Otherwise, starts the third phase at the 1st unmatched input.

3. **The LA scheduling phase**, in which the LA sub-server tries to find feasible connection between the left empty inputs and outputs to the queued Class 2 cells. The LA sub-server begins its search at the 1st unmatched input after the second phase. Once all left unmatched input is reached or no empty output exists, stop the operation, do routing and start a new operation. The detail of this phase is that:

> 3a) The LA sub-server checks the 1st unmatched input at its Class 2 buffer, and try to pair the queued cells in a FIFO order to the empty outputs. If a connection can be found during the search or the LA sub-server reaches its window threshold, stops the operation and moves to check the next unmatched input.
>
> 3b) The LA sub-server do the same operation to all unmatched inputs until it finishes to check the last unmatched input or no empty output exists.

Now the WRRLA server finds out all feasible connection between the inputs and the outputs, and one scheduling operation finishes. Routing can be done according to the matched connection table of the WRRLA server.

IV. Performance Evaluation and Discussion

In this section, we evaluate the performance of WRRLA by simulation. We also compare it with some famous scheduling algorithms, such as PIM and WRR. We will show that, under the same traffic load, 1) The scheduling computation of WRRLA is much less than PIM, but is a little bit more than WRR. 2) The throughput provided by WRRLA is very close to PIM when LA window size reaches 8 or above, and which have much more improvements compared with WRR. 3) The average delay of a connection in WRRLA depends on the number of bandwidth allocated to the different classes of traffic. The more the residual bandwidth is utilized by Class 2 traffics (LA sub-server), the lower the average delay will be, and the higher throughput is. 4) Since WRRLA is an improvement over WRR, it can provide lower upper delay to time-sensitive traffics. Detailed descriptions are presented as follows.

IV. 1 Scheduling computation

In this sub-section, we present the simulation results with a focus on the scheduling computation. All simulations were performed on the independent uniform traffic load, and the

total offered loads equals to 1. PIM algorithm is implemented in running 4 iterations; and the LA window size of WRRLA algorithm equals to the size of the switch. Simulations were run long enough to eliminate the effect of any initial transient.

Figure 4 compares the scheduling computation of different algorithms with each other. ***Figure 5*** shows the scheduling computation of WRRLA under different LA window size. The simulation result is a relative measurement and independent of the processors. In ***Figure 4***, if we denote the scheduling computation of a 2-by-2 switch using WRR algorithm equals to 1 CPU time unit, then to schedule a 32-by-32 switch using WRR algorithm needs about 12 CPU time units, and the value increases to be about 27 and 55 CPU time units by using WRRLA and PIM algorithm, respectively. We can see that the scheduling computation increases significantly with the growth of switch size. Under the same conditions, the scheduling computation of WRRLA is much less than PIM, but a little bit more than WRR algorithm.

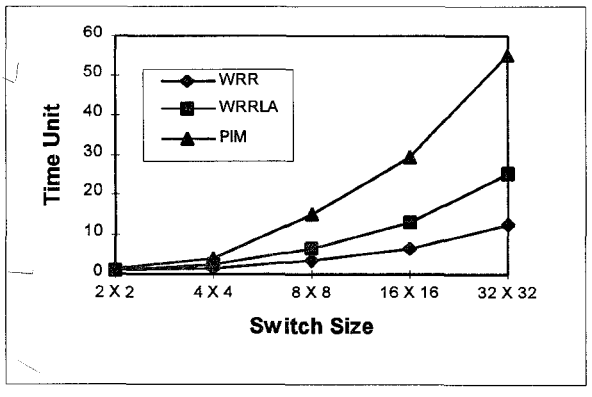

Figure 4 Scheduling computation Vs. Switch size

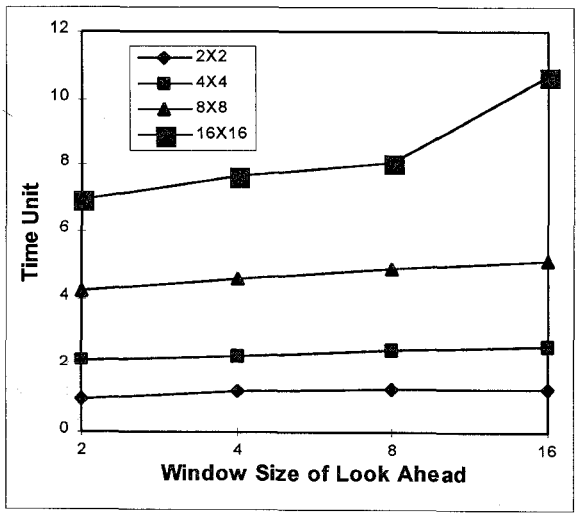

Figure 5 Scheduling computation Vs. Different LA window size in WRRLA

Similarly, in ***Figure 5***, if we denote a 2-by-2 switch scheduled by WRRLA with LA window size of 2 equals to 1 CPU time unit, then to schedule a 2-by-2 switch with LA window size of 16 needs 1.5 CPU time units, and it increases to be about 11 CPU time units when a 16-by-16 switch is scheduled. We can see that the scheduling computation increases with the growth of switch size and the LA window size. The distinction is insignificant when LA window size is smaller than 8 in all cases.

IV. 2 Throughput

In this sub-section, we present the simulation results with a focus on the throughput. All simulations in this sub-section are done under the same conditions of the last sub-section, and the LA window size equals to the size of the switch.

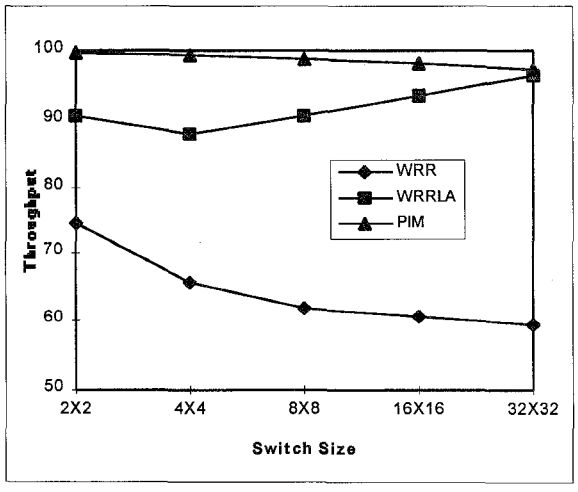

Figure 6 Throughput Vs. Switch size

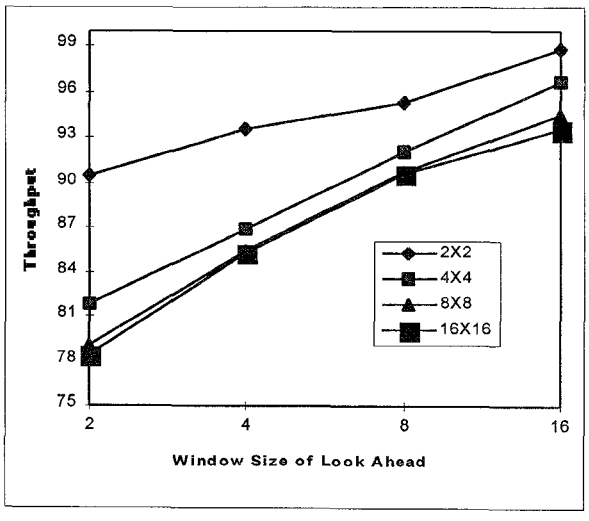

Figure 7 Throughput Vs. Different LA window size in WRRLA

Figure 6 compares the throughput of different algorithms with each other. ***Figure 7*** shows the throughput of WRRLA under different LA window size. We can see that, in ***Figure 6***, the throughput of WRR is always lower than 58% when the switch size becomes large. WRRLA achieves almost the same or slight lower throughput of PIM, say around 90-95%

when LA window size reaches 8 or above. These improvement benefits from the LA technique, which increases 30-35% of throughput compared with the pure WRR. In ***Figure 7***, we can see that the increase of LA window size is much helpful to improve the throughput of WRRLA.

IV.3 Average Delay

In this sub-section, we compare the average queuing delay versus offered load with each other by using the different algorithms. All simulations in this sub-section were performed on a 16-by-16 switch, and the offered traffic load varies from 0.1 to 1.0. The destinations of arriving cells are uniformly distributed among the outputs. Simulations were run long enough to eliminate the effect of any initial transient.

Figure 8 illustrates average delay of different algorithms under uniform traffic loads. At the low workloads, which means the total traffic load is lower than 0.4, there is a little difference among these four algorithms. All of them will have almost the same average delay performance because the number of queued cells are few. At the moderately high loads, which means the total traffic load is from 0.4 to 0.8, the input queuing is suffering from HOL blocking, and the distinction of output queuing is not significant. There is a little bit increase for both PIM and WRRLA. The reasons are that: in the output queuing, a queued cell is delayed only by other cells at the same output; in PIM, a queued cell must compete for the crossbar with both cells queued at the same input and cells destined for the same output; in WRRLA, the increasing queuing delay for WRRLA is because of its round-robin nature. In addition, we can see that, in WRRLA, the more the bandwidth is utilized by Class 2 traffics (LA sub-server), the lower the average delay will be. When 80% out of the total bandwidth is utilized by LA sub-server, WRRLA has almost the same average local delay as PIM. This is also very reasonable. Because the bandwidth allocated to LA sub-server is the residual bandwidth after WRR scheduling. The more the residual bandwidth is, the faster the queued cells are served.

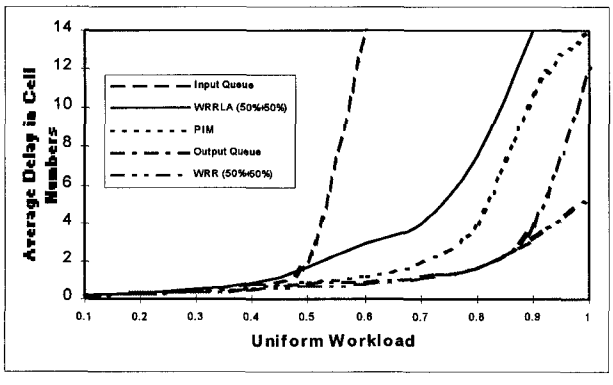

Figure 8 Average delay Vs traffic load with different algorithms

IV.4 Bounded Delay for Real-time Traffic

Since WRRLA is an improvement over WRR, it can guarantee the maximum delay to real-time traffics between the traffic shaper and the switch. Which means the part 1 delay guaranteed be shown in ***Figure 1***. Most of the analysis results at this sub-section are based on [14].

Here we denote that R is the bandwidth guaranteed to the real-time traffic of a special connection, also R is the token generation rate of LB 1 in the modified LB of that connection; We also denote that B is the depth of that LB 1 bucket; t is the window size of the WRR scheduling window; s is the number of slots guaranteed to the flows in the WRR schedule; and X is the traffic burst characteristics after it passes through the shaper. Then we have:

$$R = s/t \quad (1)$$

$$X = \lfloor (B * t / (t - s)) + 0.5 \rfloor \quad (2)$$

and $$Maximum\ delay = (X+1) * ((t/s) - 1) \quad (3)$$

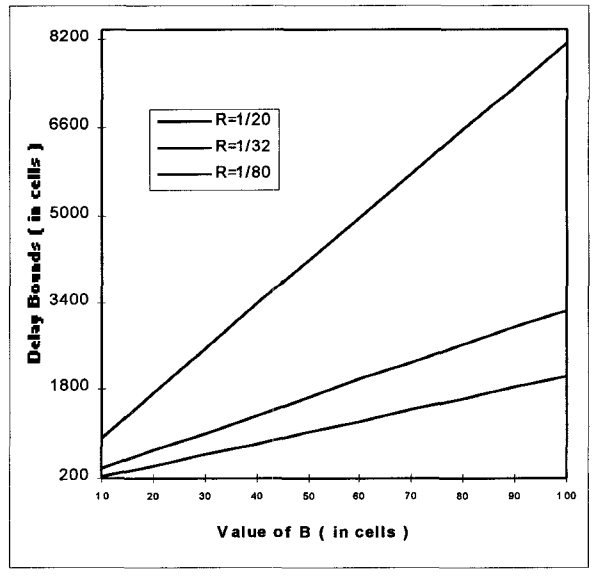

Figure 9 Delay bounds of multimedia traffics Vs. value B and value R

Figure 9 and ***10*** shows the delay bounds of the multimedia traffics. In ***Figure 9***, the value of t is set to 320, we can see that, by given the value of R, the delay bound increases as the value of B increases. By given the value of B, the delay bounds decreases as the value of R increases. Which means that by the given B, the more bandwidth is guaranteed to the real-time multimedia traffics, the lower the delay bounds will be. In ***Figure 10***, the value of R is set to 1/32, we can see that, since the evenly distributed scheduling window approach is used, no variance as the value of t are changed due to the distributed nature of service.

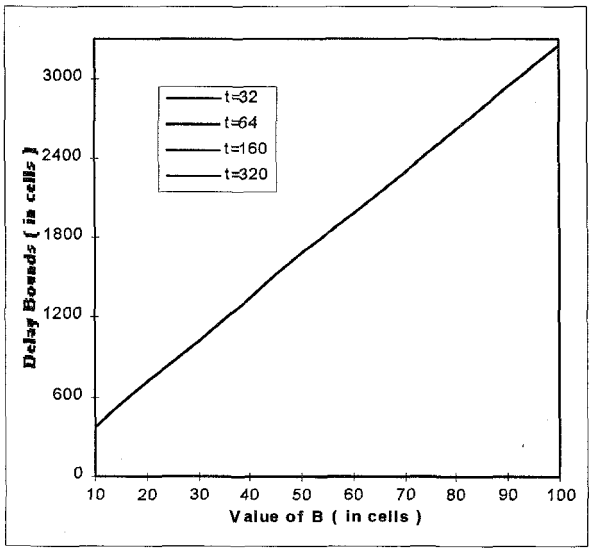

Figure 10 Delay bounds of multimedia traffics Vs. Value B and value t

V. Conclusion remarks

Our contribution in this paper is to propose and emulate a new scheduling algorithm, which is named WRRLA algorithm. The WRRLA algorithm works in three phases: the initial phase, the WRR scheduling phase, and the LA scheduling phase. A WRRLA scheduling server schedules time-sensitive traffics by a WRR sub-server to guarantee their timing performance, and data traffics are scheduled by a LA sub-server to increase the throughput of the switches. Simulation studies show that, WRRLA can achieve almost the same throughput of PIM, but with much less scheduling computation. On the other hand, although the scheduling computation of WRRLA is a little bit more than pure WRR, but WRRLA can improve the throughput to a very high level, say 30-35% more than pure WRR. Further more, WRRLA is able to provide delay bounds to multimedia traffics. Therefore, WRRLA has the best synthetic performance. WRRLA can be used to schedule a broadband high-speed switch to provide multimedia services. In this case, the residual bandwidth after scheduling the multimedia traffics will be allocated to data traffics in the LA manner, which increases the bandwidth utilization to a high level.

References

[1] C. Partridge, *Gigabit Networking,* Addison-Wesley 1994.

[2] Hui Zhang, "Service Disciplines for guaranteed performance service in packet-switching networks," Invited paper, *Proceedings of the IEEE*, Vol. 83, No. 10, pp. 1374-1396, October 1995.

[3] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *J. Internetworking Research and Experience*, pp. 3-26, October 1990.

[4] A. K. Parekh and R. 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, June 1993. Also in Proceedings of IEEE INFOCOM'93.

[5] Brian Lang, Kert Mezger and David W. Petr, "Delay Performance of Weighted Round Robin Scheduling," *TISL Technical Report TISL-10230-03*, September 1994.

[6] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proceedings of IEEE INFOCOM'94*, pp. 636-646, June 1994.

[7] T. S. Anderson, S. S. Owicki, J. B. Saxe and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, Vol. 11, No. 4, pp. 319-352, November 1993.

[8] D. Stiliadis and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," *Proceedings of IEEE INFOCOM'95*, Vol. 3, pp. 960-968, 1995.

[9] L. Zhang, "Virtual clock: a new traffic control algorithm for packet switching networks," *Proceedings of ACM SIGCOMM'90*. pp. 19-29, September 1990.

[10] Clark. D. D., S. Shenker and L. Zhang, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," *Proceedings of ACM SIGCOMM'92*, pp. 14-26, August 1992.

[11] S. Motoyama, D. W. Petr and V. S. Frost, "Input-queued switch based on a scheduling algorithm," *IEE Electronics Letters*, Vol. 31, No. 4, pp. 1127-1129, 6th July 1994.

[12] J. Nagle, "On packet switches with infinite storage," *IEEE Transactions on communications*, Vol. 35, No. 4, pp. 435-438, April 1987.

[13] M. Katevenis, S. Sidiropoulos and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE JSAC*, Vol. 9, No. 8, pp. 1265-79, October 1991.

[14] Kert Mezger and David W. Petr, "Bounded Delay for Weighted Round Robin," *TISL Technical Report TISL-10230-07*, May 1995.

[15] K. Mezger, D. W. Petr and T. G. Kelley, "Weighted Fair Queueing Vs. Weighted Round-Robin: a Comparative Analysis," *Proceedings of the IEEE WICHITA Conference on Communications, Networks and Signal Processing*, pp. 3-8, April 26-27, 1994.