A Novel MAC Scheduling Algorithm to Regulate Multimedia Traffic with Promising QoS

R.DEEPALAKSHMI
Department of Computer Science and Engineering, Velammal College of Engineering and Technology, Madurai, Tamil Nadu, India
rdl@vcet.ac.in
DR.S.RAJARAM
Department of Electronics and Communication Engineering, Thiagarajar College of Engineering, Madurai, Tamil Nadu, India
jei.deepa@gmail.com

Abstract: - Dynamic MAC Scheduling Algorithm (DMSA) to regulate packets on optical switches is a heap-based implementation uses a Distributed Traffic and Admission Control Heap (DTACH) characterizes multimedia streams with diverse Quality of Service (QoS) requirements to substantially improve solution quality and reduces the packet drop rate. We present a simulation environment that has been designed combining network simulator ns2 to predict the quality of service the experimental results demonstrate improved scalability compared to a liner implementation in supporting thousands of streams with strict real-time constraints, while causing no loss in accuracy compared to the FIFO algorithm. This DTACH algorithm achieves a low average delay and low packet drop rate as compared to other scheduling algorithms of equivalent complexity while still achieving similar throughput.

Key-Words: - Quality of Service (QoS), Medium Access Control (MAC), Scheduling, Multimedia Traffic, Packet Drop Rate

1.Introduction

Optical Network technology is being pushed forward to satisfy the ever-increasing requirements of future application with diverse quality of service requirements [14]. Real-time media servers need to serve hundreds to thousands of clients, each with its own Quality of Service (QoS) for applications like video, video-on-demand, real time video conferencing etc., [9] along with the development of fiber optics technology, have strained the need for high capacity high speed switching technologies which are capable of providing high quality. Critical QoS properties like packet loss rate, deadline and delay variance need to be maintained without compromising the processing speed of incoming data streams[9][11][12].

Each node receives exogenous demand in form of packets. These nodes communicate these packets through a shared wavelength. Hence their concurrent transmission may contend with each other. The purpose of a scheduling algorithm is to resolve these contentions among transmitting nodes so as to utilize the bandwidth efficiently while keeping the queues at nodes finite [13].

It is common for high performance packet switches to use a crossbar switching fabric and input queues to hold packets during times of congestion. Karol et al. [1][12] showed that input queued switches can suffer from reduced throughput due to head of line blocking. And so it is now common for input queued packet switches to maintain virtual output queues (VOQs) [6]. Such switches, when combined with a suitable scheduling algorithm, have been shown to achieve 100% throughput [3][4] for traffic that is uniformly or non-uniformly distributed over the outputs of the switch. Providing QoS guarantees in a Optical network requires the use of traffic scheduling algorithms in the switches [14][16][20]. The function of a scheduling algorithm is to select, for each outgoing link of the switch, the packet to be transmitted in the next cycle from the available packets belonging to the flows sharing the output link. A data packet arriving at an input port is immediately forwarded to its output, where it is buffered until it can be transmitted over the appropriate output link [6]. The order of transmission of packets waiting at the queue is controlled by a scheduler. The bandwidth and delay guarantees that can be provided entirely on the properties of this scheduler and the nature of traffic at the entry to the buffer[14].

In the Existing scheduling algorithms the first category such as Longest Queue First (LQF) [8][12] and Oldest Cell First (OCF) [13] have impractically high computing complexity, but are of theoretical importance as they deliver 100% throughput under virtually any admissible traffic. The second category includes Round Robin Greedy Scheduling (RRGS) [4][21], Parallel Iterative Matching (PIM) [5] only look for a maximal matching in each time slot hence have practical time complexity. They are therefore preferred in real systems, although they can only give sub-optimal schedules. In addition, the design of the scheduling algorithm should take the fairness into account by giving the ones who are having bad channel conditions more priorities to increase their chance to being served and avoid the problem of starvation[22].

In this paper, a Dynamic MAC scheduling algorithm with QoS support for hybrid services in optical networks has been introduced. This method is called Dynamic MAC Scheduling (DMSA) algorithm, implemented based on a heap order property. The “Distributed Traffic and Admission Control Heap” (DTACH) algorithm uses a heap structure to store and select packets according to their deadline and loss-tolerance order. The computational complexity of this implementation is proved to be O (n), where n is the number of the simultaneously active streams in the system. In This paper the algorithm addresses the fact that packet scheduling speed in anxiety in interpretations with the bandwidth limitations. The DTACH introduced in this paper reduces the time needed for packet ordering and priority adjustments by reducing the cost of both packet insertion and selection operations. It also eliminates the need to move data within switch by directly inserting packets into the queue that is intentionally kept sparse.

This paper is organized as follows. Section II discusses some of the related work. Section III presents the DMSA algorithm and introduces the DTACH algorithm. Section IV describes the data structures and the implementation of DTACH. Section V presents the performance results and section VI concludes the paper.

2.Dynamic MAC Scheduling Algorithm (DMSA)

2.1. Creating a Multimedia Traffic Class/ Creating a Multimedia Traffic Policy:

The very first step to implement a DMSA algorithm is to differentiate multimedia traffic based on the Traffic Classes and traffic policies to schedule. A Traffic Class specifies a mechanism which we can use to match incoming and/or outgoing packets on a switch's interface.
Traffic Policy is used to (i) collect Traffic Classes together in one object called Traffic Policy (ii) apply to each Traffic Class component the scheduling behavior corresponding to it (iii) mark or re-mark packets belonging to a selected Traffic Class before they are forwarding to switch (iv) assign the packet, i.e., the Traffic Policy, to one or more interfaces of the switch in such a way that the traffic crossing the switch only through the selected interface(s).

2.2. The Multimedia Traffic-Conditioning function:

The Multimedia traffic-conditioning function consists of five elements shown in Figure 1.

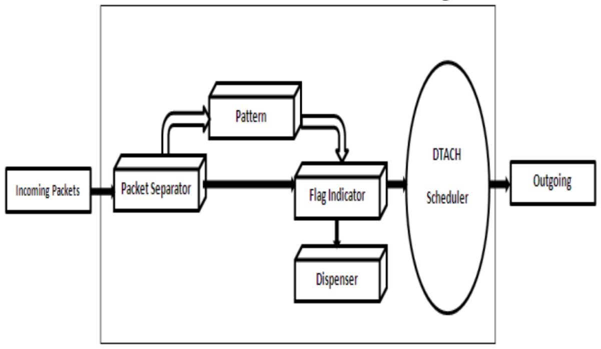

Figure. 1 Multimedia Traffic-Conditioning function consists of Five Elements

Packet Separator: Separates submitted packets into different classes. This is the foundation of providing differentiated services.

Pattern: Measures submitted traffic for conformance to a profile. The meter determines whether a given packet stream class is within or exceeds the service level guaranteed for that class.
Flag Indicator: Polices traffic by re-marking packets with a different flag as needed. This may be done for packets that exceed the profile; for example, if a given throughput is guaranteed for a particular service class, any packets in that class that exceed the throughput in some defined time interval may be re-marked for best-effort handling. Also, re-marking may be required at the boundary between two DS domains. For example, if a given traffic class is to receive the highest supported priority, and this is a value of 3 in one domain and 7 in the next domain, packets with a priority 3 value traversing the first domain are re-marked as priority 7 when entering the second domain.
Dispenser: Drops packets when the rate of packets of a given class exceeds that specified in the profile for that class.
Scheduler: Polices traffic by delaying packets as necessary so that the packet stream in a given class doesn't exceed the traffic rate specified in the profile for that class. In order to improve user and system performance for high speed Multimedia traffic, DMSA introduces new features such as a packet separator.

2.3. Scheduler Queue Service:

Scheduler should implement precedence-ordered queue service. Precedence-ordered queue service means that when a packet is selected for output on a (logical) link, the packet of highest precedence that has been queued for that link is sent. Distributed Traffic and Admission Control Heap (DTACH) scheduler implemented procedures that result in strict precedence ordering.

2.4. Packet Dropping Mechanism:

Distributed Traffic and Admission Control Heap (DTACH) scheduler receives a packet beyond its storage capacity it must discard it or some other packet. A recommended policy in packet transmission environments using FIFO queues is to discard a packet randomly selected from the queue. The DTACH selects a packet from one of the sessions most heavily abusing the link, given that the applicable QoS policy permits this. If Distributed Traffic and Admission Control Heap service is implemented and enabled, the scheduler discard a packet whose precedence is higher than that of a packet that is not discarded. To help prevent scheduling perturbations or disruption of management functions, the DTACH protects the packets used for routing control, link control, or network management from being discarded.

3. Distributed Traffic and Admission Control Heap (DTACH) Scheduler Architecture:

3.1. The prototype

The DTACH scheduler has several independent functional blocks, with the most important ones located at the packet forwarding path in the network node. Together they provide the necessary mechanisms for the desired forwarding behavior. Basically, this set of modules represents a queuing architecture and the elements of the traffic control. These elements include the packet classifier, the traffic meters, the very important packet scheduler, and the correspondent queuing mechanism.

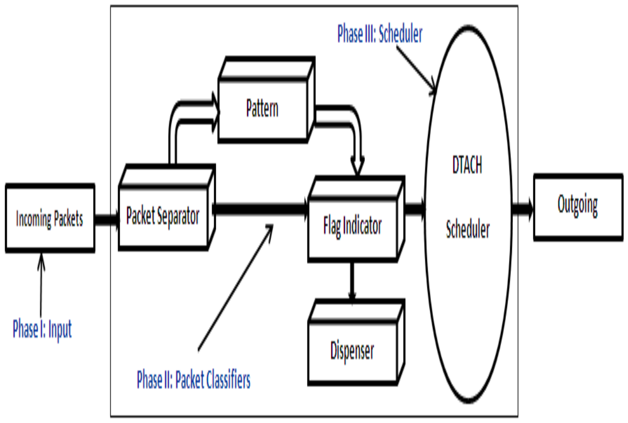

Figure. 2 Distributed Traffic and Admission Control Heap (DTACH) Scheduler

3.2. Packet Classifier phase:

This module is conceptually very large, and includes the packet separator, the pattern and the dispenser mechanisms. In terms of packet separator the classification of packets into classes is done by comparing information present at the packet header, such as the source and destination addresses, and the rules present in a table configured by the user a search is made through the table of rules until whether a match is done or no match at all is found. The packet is sent to the default class if not present. The packet is then marked or remarked if necessary with the corresponding Set. Dispenser mechanism simply drops the packet when the corresponding queue is full.

3.3. The Scheduler Phase: Scheduler algorithm tries to do the best regarding fairness and protection, and takes into consideration the dynamics of the underlying system. The algorithm is a set-based, temporal heap. There is a heap for each class and each set has its priority based on the higher or lower importance of the packet average delay in the forwarding path of the node. The arriving packets are classified into the appropriate set, and the scheduler outputs the packet from the current set being serviced if it is already the time to do so. Otherwise the scheduler skips into the next set. The time that each set wait to be serviced depends on its degradation slope and on the current the highest priority set. The lowest priority set, also known as the default set, is only serviced when there are no other packets waiting to be serviced on the scheduler.

3.4. Heap Scheduling

The DTACH scheduler in optical buffer selects packets to send out from all flows that have queued packets. DTACH maintains a service list to keep the flow sequence being served and to avoid examining empty queues. If a flow has no packets in its queue, its identifier will be deleted from the service list. The next time a packet arrives to the flow that has an empty queue, the identifier of the flow will be added to the tail of the list.

We consider N types of traffic flows, referred as coalition and the link capacity α. Each coalition-i traffic flow has apex value ωi and the traffic flow arrives according to a random process and leaves the system once some random volume of data has been transferred. We denote the associated traffic intensity by γi defined as the product of the flow arrival rate by the mean flow size, and the corresponding load $\beta i = \gamma i / \alpha$.

Then the overall load is defined by

$$\beta = \sum_{i=1}^{N} \beta i$$

The packet drop rate estimation depend on the complex packet level dynamics induced by the optical switch and packet losses are mostly due to the presence of a too high number of simultaneous flows. And the approximation considering that each flow is active at its peak rate the state is referred as δ the intensity of lost packet is equal to $\delta.\omega i - \alpha$.

$$\text{Packet Loss} = \begin{cases} \delta.\omega i > \alpha & \ell = 1 \\ \text{Otherwise} & \ell = 0 \end{cases}$$

We will now derive the heap based model for the optical buffer. The Optical heap based packet dropping algorithm keeps track of a weighted moving average queue length ρ .Which is updated upon each packet arrival. If the optical heap buffer is non-empty upon packet arrival the average queue length is updated according to $\rho \leftarrow (1 - \omega i)\rho + \delta$

If the packet arrives to an empty buffer, the average is scaled down, proportional to the time the queue has been empty, τ as

$$\rho \leftarrow (1 - \omega i)^{\tau} \rho$$

The moving average queue length is used to decide if an arriving packet should be dropped or not .Packets are dropped the drop function

$$di = d\rho = \begin{cases} 0, \rho \le \beta i \\ 1, \rho < \gamma i \\ \dfrac{\rho - \beta i}{\gamma i - \beta i} & \beta i < \rho < \gamma \end{cases}$$

For our simplified optical heap buffer with some arrival intensity, service intensity and buffer size as for the packet drop rate we will experience packet loss already before the buffer is full. With the correctly configured Heap these losses should force a significant proportion of flow controlled sources to slow down and hence it should be rare to experience the dropping packet. The problem of concluding which the heap based queue will work best is due to the interaction between flow controlled sources and packet loss events that exists.

4. DTACH Scheduler Algorithm

The DTACH Algorithm can work as a network packet scheduler. It schedules packets from multiple streams by limiting the number of late or lost packets over a finite number of consecutive packets. This is most favorable for video or audio streams, where a limited amount of packet loss is tolerable over a fixed transfer window. DTACH Algorithm can also be applied to the scheduling of processes for time-sensitive real-time tasks. The DTACH Algorithm uses two parameters for each stream:

- **Bound Limit (BL)-** Bound Limit is the newest time a packet can originate service.
- **Packet Drop Tolerance (PDT) -** This is specified as a value PDT= N_{lp}/ N_{in}, Where N_{lp} is the number of packets that can be lost or transmitted late for every input N_{in} of consecutive packet arrivals in the same stream i.

```
// packets transmitted before bound limit
if ($N_{in}$ Current> $N_{lp}$ Current) then
    $N_{in}$ Current = $N_{lp}$-1
else if ($N_{in}$ Current== $N_{lp}$ Current==0) then
    $N_{lp}$ Current= $N_{lp}$ and $N_{in}$ Current = $N_{in}$
// packets missing during the bound limit
If($N_{lp}$ Current >0) then
    $N_{lp}$ Current = $N_{lp}$ Current -1; $N_{in}$ Current = $N_{in}$ Current -1;
If ($N_{lp}$ Current = =$N_{in}$ Current =0) then
    $N_{lp}$ Current = xi; $N_{in}$ Current = $N_{in}$;
else if ($N_{lp}$ Current = 0)then
    if ($N_{lp}$ > 0) then
        $N_{in}$ Current = $N_{in}$ Current + {($N_{in}$ - $N_{lp}$)/ $N_{lp}$};
if ($N_{lp}$ = 0) then
        $N_{in}$ Current = $N_{in}$ Current + $N_{in}$;
```

Figure 3. Pseudo code for packet processing

DTACH transmits the packets in an order according to their PDT and BL. Packets of the same stream have the same original and current PDT and will be scheduled in the order of their arrival. When scheduled by DWCS, the priorities of each stream can be adjusted dynamically. Whenever a packet misses its BL, the PDT for all packets in the same stream is reduced to reflect the increased importance of transmitting a packet from this stream. This approach prevents starvation and tries to keep the stream from violating its original PDT.

This implementation (figure 3) uses a heap for each stream. Packets of the same stream are added to the end of the heap and removed from the root of the heap. Two heaps are set up for bound limit and Packet Drop Tolerance. With the heaps, packet insertion can be done using heap sorting. Packet precedence comparison will follow the rules given in the algorithm. DTACH checks streams for packets that have missed their bound limit every time a packet has been serviced. In the worst case, every stream can have late packets when the scheduler completes the service of one packet. The deletion of these packets from the packet drop tolerance heap and the heap adjustment will cost O (n log n). Also, O (log n) time is required to find the next packet to be sent from the n streams. Thus, the current implementation of heap structure is scalable as the number of active streams in the system increases. When the stream rate is low (more active streams co-exist in the system) or the bandwidth is high compared to the stream rate, the two heaps will be adjusted frequently. Since more time is spent on insertion and selection, there will be further degradation in the throughput as the system is overloaded.

5. Performance Evaluation and Analysis

We have implemented the DTACH Scheduler algorithm with heap data structure. All experiments are conducted on the ns2 simulation environment. We demonstrate that our DTACH Scheduler performs well with high -speed optical cross connects. All the experiments are simulated.

5.1. Number of Iterations vs. Number of streams comparison

In Table 1, we compare the number of iteration required in our algorithm it takes only one iteration to find out the next packet to be sent from the optical buffer. However, maintaining the heap requires O (logn) steps, and there are two heaps in the original DTACH Scheduler implementation. The proposed algorithm minimizes the number of iterations as compared to the WFQ implementation. The number of iterations per stream is close to a constant for DTACH Scheduler, while it increases logarithmically with the number of active streams for the heap based DTACH. Combining results in Table 1, we can conclude that a simple FIFO would not scale as well as for a large number of streams.

No. of. Streams		10	50	100	200	500	1000	1500	2000
No of	WFQ	45.8	73.7	85.7	97.7	113.	125.	127.	137.

iteration		6	3	3	3	59	59	67	58
	DTACH	19.8	20.36	25.68	30.34	32.135	35.068	38.147	41.034

Table 1. No. of iteration comparison for different Number of streams

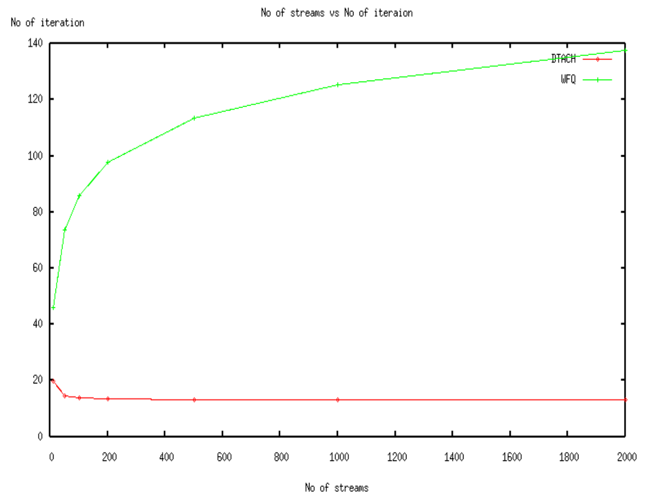

Figure 4. Number of Iterations vs. Number of streams comparison

5.2. Maximum Load vs Packet Drop rate Comparison

The simulation performed on the model based on the packet drop rate for the special case of a buffer size 100 for different loads. The buffer size and the batch size parameters are chosen to reflect buffer and batch sizes investigated, as shown in table 2 as the load increase above 1.1 the packet drop rate estimates shows tremendous different in DTACH compared to WFQ. The first approach considers packets to be independent, where the second approach assumes that packet loss, after the first packet dropped, is strongly associated so that once a packet is dropped every packet is dropped for a short period of time.

Maximum Load		**.01**	**.02**	**.03**	**.04**	**.05**	**.06**	**.07**	**.08**	**.09**	**0.1**
Packet Drop	**WFQ**	0.7	0.85	0.89	0.91	0.98	1.02	1.05	1.09	1.1	1.15
	DTA	0.	0.	0.	0.	0.	0.8	0.8	0.	0.8	0.8

Rate	**CH**	68	82	83	84	86	65	68	87	72	75

Table 2.Maximum Load vs Packet Drop rate Comparison

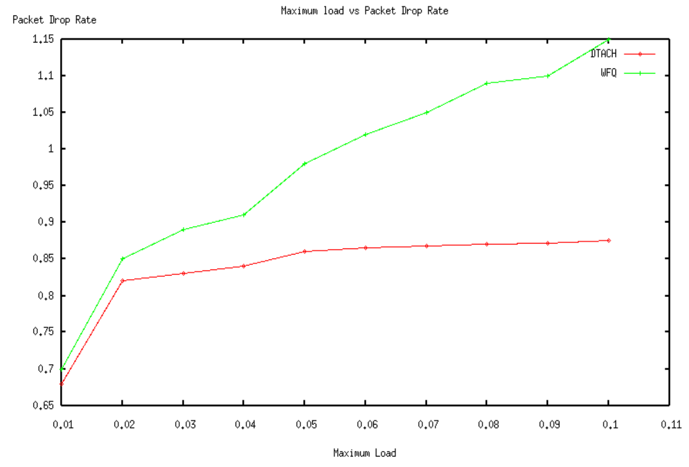

Figure 5. Maximum Load vs Packet Drop rate Comparison

5.3. Throughput vs. Scheduling Cycles for Different Packet Sizes

We linked our DTACH schedule with the optical switch that performs packet receiving and transmission. The packet from the input transferred to the scheduler queue .the DTACH scans each segment linearly and sends out the waiting packets on the output queue. The transmission thread reads from the output queue and puts the packet on the switch. To sustain a certain data rate, the scheduler has to complete a packet scheduling cycle for different sizes of packets. The figure 6 indicates that large packets allow a higher time for a packet scheduler. When the packet size is 1K, a packet scheduler with about 900 cycle schedule overhead is still capable of supporting a data link. However, for small packet size like 128 bytes and 256 bytes throughput starts to drop even with small scheduling overhead. For smaller size packet such as 256 bytes, the throughput is about 650MB/ps. This result is consistent with the scheduler time shown in figure 6.For 512-byte packets, the scheduler is able to schedule about 2,60,000 packets /ps.If the stream has only about 40-60 packets per second ,our DTACH scheduler would be able to support about 4,000-6,000 streams.

5.2.Scalability - No. of Active Transmission/per second vs. Average Scheduling Delay

We test the scalability of the DTACH scheduler in simulation. There are two sets of tests, one without input packet traffic, and the other with input/output packet traffic. The objective of the first testing is to chart the actual scheduling overhead under different numbers of active transmission. To obtain a large number of active transmissions, we use a separate variety of pseudo-transmission. In total one sends transmission to scheduling requests and the other runs the DTACH scheduler.

No. of Active Transmission		**0**	**500**	**1000**	**1500**	**2000**	**2500**	**3000**
Scheduling Delay/Transmission	**DTACH**	0	600	650	780	850	980	1200

Table 3. Scalability - No. of Active Transmission/per second vs. Average Scheduling Delay

Figure 7 shows the scheduling delay per active transmission as we vary the number of active transmission. It indicates that our DTACH scales well for a large number of active transmissions. As the number of transmissions increases, the average scheduling delay per active transmission approaches 500 cycles/per stream. The high scheduling delay when the number of active transmission is small (below

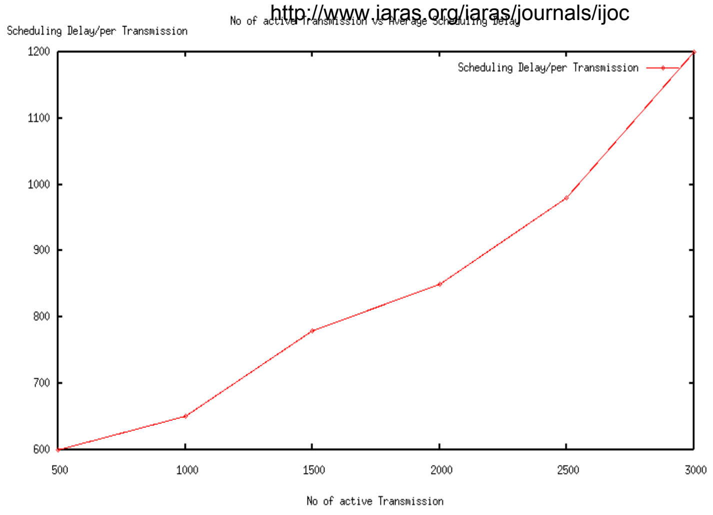

Figure 6. Throughput vs. Scheduling Cycles for Different Packet Sizes

20). This is caused by a fixed amount of overhead in our DTACH scheduler. The scheduler has to read and check all indexes during a segment sweep even when there are only few active streams. Secondly, we want to show that the DTACH scheduler can actually produce high data throughput. We can adjust the packet speed and the number of transmission through the packet generator and the configuration option in the simulator.

6. CONCLUSION

This paper proposes a fast algorithm to implement a multimedia stream scheduler on the optical networks. A data structure called Distributed Traffic and Admission Control Heap (DTACH) meets the requirements by exploiting certain architectural attributes with improving burst dropping performance keeping the computational complexity low. Simulation results have shown that the proposed DMSA algorithms perform significantly better than standard algorithm in terms of burst dropping probability. By using heap based scheduling, provides fair access to output lines and prevents starvation and achieves high throughput.

By careful control of the DTACH, the algorithm can achieve 100% throughput for uniform traffic. When the traffic is non-uniform, the algorithm quickly adapts to an efficient heap based policy among the busy queues. High performance is gained by integrating send/receive transmission in the order of heap. Our implementation can achieve O (1) packet insertion and selection time using optical heap queue and to separate the garbage collection transmissions.

References:

[1] R.S. Tucker, P.-C Ku and C.J. Chang-Hasnain, “Fundamental limitations of slow-light optical buffers,” *Optical Fiber Communication Conference*, 2005. Technical Digest. OFC/NFOEC, vol. 3, Mar. (2005).

[2] L. Liu and Y. Yang, “Achieving 100% throughput in input-buffered WDM optical packet interconnects,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 237-286, February (2011).

[3] S. Shimizu, Y. Arakawa and N. Yamanaka, “Wavelength assignment scheme for WDM networks with limited-range wavelength converters,” *Journal of Optical Networking*, vol. 5, issue 5, pp. 410-421, (2006).

[4] X. Qin and Y. Yang, “No blocking WDM switching networks with full and limited wavelength conversion,” *IEEE Transactions on Communications*, vol. 50, no. 12, pp. 2032-2041, (2002).

[5] Roe Hemenway and Richard R. Grzybowski “Optical-packet-switched interconnect for supercomputer applications,” *Journal of Optical Networking*, Vol. 3, No. 12,December (2004).

[6] Zhenghao Zhang, Lin Liu, Yuanyuan Yang “Slotted Optical Burst Switching (SOBS) networks,” *Computer Communications,* 30 (2007).

[7] Nick McKeown “The iSLIP Scheduling Algorithm for Input-Queued Switches,” IEEE/ACM Transactions on Networking, VOL. 7, NO. 2, APRIL (1999).

[8] A. Shami, X. Bai, N. Ghani, C. Assi, H.T. Mouftah, “QoS control schemes for two-stage ethernet passive optical access networks,” *IEEE Journal on Selected Areas in Communications (JSAC)* ,23 1467–1478. (2005).

[9] G. Kramer, B. Mukherjee, S. Dixit, Y. Ye, R. Hirth, “Supporting differentiated classes of service in Ethernet passive optical networks,”, *Journal of Optical Networks* , 280–298,(2002).

[10] H. Lee, T. Yoo, J. Moon, H. Lee, A two-step scheduling algorithm to support dual bandwidth allocation policies in an ethernet passive optical network, *ETRI Journa*l ,26 185–188, (2004).

[11] F. An, Y. Hsueh, K. Kim, I. White, L. Kazovsky, A New Dynamic Bandwidth Allocation Protocol with Quality of Service in Ethernet-based Passive Optical Networks, in: *International Conference on Wireless and Optical Communication (WOC)*, Banff, July (2003).

[12] Mei Yang and S.Q. Zheng, Dominique Verchere “A QoS Supporting Scheduling Algorithm for Optical Burst Switching DWDM Networks,” *Global Telecommunications Conference, GLOBECOM '01*. IEEE (2001).

[13] Biao Chen, Jiajia Chen ; Sailing He “Efficient and fine scheduling algorithm for bandwidth allocation in ethernet passive optical networks*,” IEEE Journal of Selected Topics in Quantum Electronics*, July-Aug. (2006).

[14] Glen Kramer, Biswanath Mukherjee, Sudhir Dixit, Yinghua Ye, and Ryan Hirth “ Supporting differentiated classes of service in Ethernet passive optical networks,” *Journal of Optical Communications and Networking*, Vol. 1, Iss. 9 pp: 280–298 Aug. 20,(2002).

[15] KrishnaM. Sivalingam,Jie Wang,Xiangjun Wu,Manav Mishra “An Interval-Based Scheduling Algorithm for Optical WDM Star Networks,” *Photonic Network Communications*, Volume 4, Issue 1, pp 73-87, January (2002).

[16] D. Levine, I. Akyildiz, PROTON: A media access control protocol for optical networks with star topology, *IEEE/ACM Transactions on Networking*, vol. 3,no. 2, pp. 158–168, April (1995).

[17] R. Deepalakshmi, Dr. S. Rajaram “New MAC Protocol for Traffic Routing in Optical Networks by exploiting Delays in Dynamic Bandwidth Allocation,” *International Conference on Computer Applications & Industrial Electronics*,pp.307-312,Dec.(2011).

[18] R. Deepalakshmi, Dr. S. Rajaram “A Novel Medium Access Control Protocol for

Routing Multimedia Traffic in Optical Networks by exploiting Delays with improved Dynamic Bandwidth Allocation," *ARPN Journal of Systems and Software*, Volume 1 No. 7, October (2011).

[19] R. Deepalakshmi, Dr. S. Rajaram "Multimedia Traffic Routing Algorithm for Optical Networks with Enhanced Performance Characteristics," *International Conference on Internet Multimedia Systems Architecture and Applications*, December (2010).

[20] R. Deepalakshmi, Dr. S. Rajaram "A Dynamic Cost Optimized Provisioning Algorithm- (DCOPA) for Optical Networks with less Rejection Ratio," *International Conference on Communication and Computational Intelligence – 2010*,.pp.11-16, December(2010).

[21] R. Deepalakshmi, Dr. S. Rajaram "New Enhanced Performance MAC Routing Algorithm to Improve Reliability in Multimedia Data Transmission Based on Mutual Diversity for Optical Networks," *European Journal of Scientific Research*, Vol.72 No.2, pp. 285-297,(2012)