2003

Priority-based Lambda Scheduler

Kalpana Ganesan
University of Nebraska-Lincoln, kganesan@cse.unl.edu

Byrav Ramamurthy
University of Nebraska-Lincoln, bramamurthy2@unl.edu

ICC '03. IEEE International Conference on Communications, 2003.
Volume: 2 Digital Object Identifier: 10.1109/ICC.2003.1204637
Publication Year: 2003 , Page(s): 1479 - 1483 vol.2

Priority-based Lambda Scheduler*

Kalpana Ganesan
Dept. of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588-0115, U.S.A
kganesan@cse.unl.edu

Byrav Ramamurthy
Dept. of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588-0115, U.S.A
byrav@cse.unl.edu

Abstract-**Optical networks provide a new dimension to meet the demands of exponentially growing traffic. Optical packet switching requires a good switch architecture, which eliminates the O/E/O conversion as much as possible. Wavelength Division Multiplexing (WDM) provides a breakthrough to exploit the huge bandwidth of the optical fiber. Different applications have different requirements, which necessitate employing differentiated services. This paper presents the idea of a priority-based λ-scheduler, where the packets are differentiated into different classes and services are provided accordingly. For example, class 0 can correspond to non real time applications like email and ftp, while class 1 can correspond to real-time audio and video communications. The architecture is based on that of the λ-scheduler and hence it has the added advantage of reduced component cost by using WDM internally.**

Index terms-- **packet switching, optical networks, priority scheduling, simulation, architecture.**

I. INTRODUCTION

Fiber-optic communication links provide an extremely large (multi-THz) bandwidth potential with very low loss. Optical networks[1][2] composed of almost-all optical switches, where the data packets remain in the optical domain and only the packet header or control information is processed electronically, can offer large bandwidth gains with extremely fast switching speeds while maintaining data transparency. This is because almost-all optical switches eliminate the need for optical - electronic - optical (O/E/O) conversion of the data, the so-called electronic bottleneck.

The design of almost-all optical switches has traditionally been based on emulating electronic switches, for which there are two basic components: the space switch, which connects the input ports to the output ports, and the buffering strategy, which is used to temporarily store data packets if contention occurs for a common resource (e.g., if multiple packets require the same output port). Depending on its design, the space switch can be either blocking, where certain permutations of input-output connections cannot be made, or non-blocking, where all permutations of input-output connections can be made.

The buffering strategy can be categorized into five designs: (i) input buffering - separate buffer for each input; (ii) Scheduling or input smoothing - a frame of T packets is examined at each input before being launched into the switching fabric; (iii) output buffering - separate buffer for each output; (iv) shared buffering - buffers are shared among multiple inputs or outputs and (v) no buffering. Almost all optical switches can be either single wavelength systems or multi wavelength (WDM) systems.

Starlite shared-buffer switch [3] was one of the first architectures to address high-bandwidth and buffering issues. It uses a self-routing space switch to route packets and re-circulating loops to resolve contention. This increases the switch-block complexity and to maintain a reasonable packet-loss probability many loops are required.

Haas' Staggering switch [4] uses fiber delay lines for temporarily buffering packets. The staggering switch uses a set of parallel delay lines of different lengths to appropriately delay the optical signals. It is based on two stages: scheduling and switching. Each one of the stages is a reconfigurably nonblocking switching fabric, implemented with electronically controlled optical devices. The scheduling stage ($n \times m$) is connected to the switching stage ($m \times n$, where $m >= n$) by m delay lines, d_i, $i = 1$ to m. The delay of the delay line d_i equals i packets. Among the salient features of the staggering switch are its transparency, lack of recirculation, and flexibility in operation and in performance.

The scheduling switch, proposed in [5], is comprised of a scheduler followed by a $N \times N$ non-blocking Space Switch, where the purpose of the scheduler is to rearrange the incoming packets so that packets appearing during the same slot at the outputs of the scheduler require different outgoing links of the Space Switch. It is designed to operate as a single-wavelength system. At each input, a splitter is used for header detection, and packets are synchronized to a local clock so that scheduling and switching are performed synchronously. Both the scheduling and packing switch architectures are single-wavelength switches that perform

* This work was supported by NSF grants (ANI-0074121 and EPS-0091900).

the buffering function using the feed-forward fiber delay lines.

This λ-scheduler [6] is a multi-wavelength scheduling switch, which uses WDM internally to fold the switch architecture in both the space and time domains to reduce the number of elementary components, and the total fiber length required to implement delays. This is based on the scheduling switch architecture and it uses novel scheduling and wavelength assignment algorithms to avoid packet collisions within the switch and at the switch output.

Though the architecture suggests a novel way to reduce the number of switch components, it does not offer differentiated services to the incoming packets. In this paper, we propose to incorporate the Quality of Services (QoS) feature to the λ-scheduler [6], to find out how it affects the switch performance. Also the λ-scheduler assumes certain smoothness (explained later in this paper) properties, and satisfying them always is not possible. So in this paper, simulations are conducted to test the performance of the switch under different circumstances and the results are presented.

For the situation in which smoothness property is not satisfied, we suggest using the leaky bucket scheme [7] to shape traffic at the source. So this regulates the packet traffic, with the degree of burstiness allowed dependent on the bucket size. The bucket size is normally assumed as equal to frame size. Since it is priority-sensitive, the packets are discarded based on their priority levels.

II. SWITCH ARCHITECTURE

The proposed architecture is a non-blocking switch of size N. There is a tunable wavelength converter at each switch input, which can convert to a different wavelength and a fixed wavelength converter at each switch output, which converts packets back to the original wavelength. Presently this switch is assumed to receive packets of same size. This might be modified in the future. The switch uses input smoothing where a frame of T packets is examined at each input before being launched into the switching fabric [8]. The packets are scheduled according to their priority. The performance is compared for uniform packet distribution. The proposed switch is based on λ-scheduler [6], which in turn is based on the scheduling switch [5]. Below, we describe briefly these two switch architectures.

A. The scheduling switch [5]

The scheduling switch, a block diagram of which is given in Fig.1, comprises of the scheduler and the N × N space switch. The purpose of the scheduler is to rearrange the incoming packets so that packets appearing during the same slot at the outputs of the Scheduler require different outgoing links of the crossbar switch. If the scheduler satisfies this property, then the crossbar switch will be able

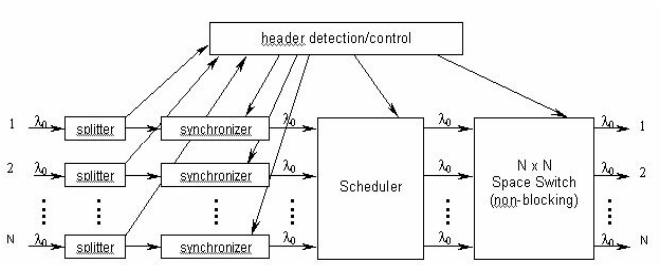

Fig. 1. Scheduling Switch, operates with a single wavelength λ_0.

to route each packet to its desired outgoing link without any collisions. The scheduler is composed of N parallel branches, one for each input, where each branch delays the packets arriving over that incoming link, until their timeslots are available. This is equivalent to a time-slot interchanger and is implemented using 2logT-1 elementary switches, where T is assumed as a power of 2 (described later in this paper). So if m=logT, then we need 2m-1 delay blocks. Also it is been shown that to avoid collisions in the scheduling switch, the outgoing frame must start at least (3T)/2 – 2 packet slots after the incoming frame begins.

B. The Lambda-scheduler [6]

The lambda-scheduler uses the advantage of WDM to fold the switch architecture. This is done by either collapsing or compressing the delay branches. Collapsing several branches to a single physical branch is achieved by wavelength multiplexing packets from multiple inputs onto a single fiber where each input uses a different wavelength. Compressing each branch of the delay stages reduces the number of delay blocks per branch by using the internal wavelengths to realize different groups of delays. According to [6], folding the scheduling switch architecture in the space and time domains can reduce the number of components used in the switch. Though the switch reduces the number of components used, it does not offer any services like reducing time delay.

In this paper, the switch we propose is based on the λ-scheduler. The switch uses a priority-based scheduler instead of the normal scheduler, which combines priority scheduling along with FCFS scheduling. We present simulation results about the improved services it offers over the λ-scheduler.

The proposed Priority-based λ-scheduling switch, shown in Fig. 2 is composed of a priority-based scheduler, which schedules packets according to their priority, followed by N × N non-blocking Space Switch, where N is the number of inputs and outputs. At each of the N inputs, a splitter is used for header detection, and packets are synchronized to a local clock so that scheduling and switching operate synchronously. This is done with the help of the Header Detection/Control block. The priority is read from the header. The priority-based scheduler, shown in Fig. 3, rearranges the incoming packets according to their priority

so that the packets appearing during the same slot at the

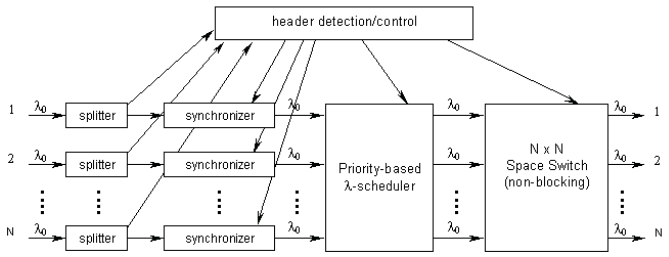

Fig. 2. Priority-based λ-scheduling switch.

outputs of the scheduler require different outgoing links of the Space Switch and packets of high priority get the earliest possible slots so that they suffer less time delay. As it is based on λ-scheduler, it uses k wavelengths to collapse or compress the N parallel branches of delay blocks to reduce the switch components' count. Fig. 3 is based on the collapsing architecture of λ-scheduler, which is explained in detail in [6]. In the following section, we will see the design of simulation experiments, designed to analyze the performance of the architecture.

III. Design Of Experiments

This section discusses the variables that are being considered for the experiments. They are defined below.

A. Dependent Variables

1) Average Delay

The experiment calculates the average time delay for high-priority and low-priority packets.

$$average\ Time\ Delay_n = \frac{Delay\ faced\ by\ packets}{Total\ no.\ of\ packets}$$

2) Packet Loss Rate

This gives an idea about packet loss, when smoothness property is not satisfied.

B. Independent Variables

1) Number of Inputs/Outputs (N)

This value denotes the number of incoming and outgoing links for a switch.

2) Slot Size

The slot size is represented by some number to denote the size of the slot.

3) Frame Size (T)

We view the time axis on a link as being divided into frames of length equal to T slots. The frame size T is an important parameter and can be viewed as a measure of the traffic burstiness that is allowed.

4) Priority Ratio (δ)

This variable is the ratio of low priority packets to the high priority packets.

5) Traffic Load

This is denoted by the ratio of total number of packets

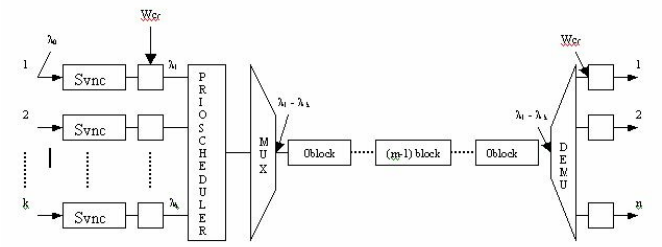

Fig. 3. Priority-based λ-scheduler uses k wavelengths to collapse k parallel branches.

generated and total number of slots.

6) Mean Interarrival Time

This is the average arrival time between the packets. This is used as a parameter, when smoothness property is not satisfied.

C. Smoothness Priority

A session is said to have the {*n, T*}-smoothness property at a node if at most n packets ($n \in \{1,2, ..., T\}$) of the session arrive at that node during a frame of size *T*. By shaping traffic at the source, and by ensuring frame integrity at the intermediate nodes, a session can be made to have the {*n, T*}-smoothness property throughout the network.

IV. Simulation

C++SIM, documented in [9], an object-oriented discrete event simulation library was used in the simulation. The active components of the network are represented by active entities called threads. The advantage of using threads is that they resemble the real-time situation more accurately.

A. Simulation Entities

1. *Packet:* This is the piece of data transferred between the nodes for communication
2. *Frame:* This is an entity that carries a fixed number of time slots. Each slot carries one packet. So a frame can carry a fixed number of packets.
3. *Switch:* This represents priority-based λ-scheduler, which schedules the packets according to the priority and routes packets according to the destination.
4. *Queue:* This is the structure of linked list, built to sort the packets according to their priority.
5. *Controller:* This active entity controls the whole of the simulation. This is responsible for reading the input files and carrying out the simulation.
6. *Bucket:* This entity stabilizes the flowing traffic. This acts like a leaky bucket which has inbuilt priority scheduling in it.

B. Parameters

The different parameters listed in the parameters file are switch size (N), frame size (T), slot size, simulation time, probability of the packets being high priority, number of wavelengths, probability of the slot being not empty. For

Poisson traffic distribution, mean is also taken as a parameter.

C. *Operation*

The main program initializes the thread package and creates an instance of the controller. After activating the controller, the main program, which is a thread itself, sleeps. The controller takes over. It reads the input files and instantiates the switch by activating the thread corresponding to it. The switch works until the simulation time is over. Then the switch thread is terminated and the results are obtained. Then the controller thread suspends itself and the main thread is resumed.

V. RESULTS

We began the experiments to test the effectiveness of priority scheduler in an environment that satisfies the smoothness property. All the experiments were performed using slot size as ten, unless otherwise mentioned. As you can see from Fig. 4, the high priority packets (priority =1) have reduced average time delay. So as the traffic increases, the average time delay is reduced for the high priority packets. But this is not true in the case of low priority packets.

The graph in Fig. 5 shows as the traffic rate increases the packet time delays of different priorities differ. For low priority packets, the time delay increases and reaches a high, when traffic rate=1. The high priority packets' time delay gets reduced gradually and after that it maintains a constant curve (not much change in time delay).

The charts in Figs. 6 and 7 show the set of simulations for different durations for switch size=64 and frame size=512. These two graphs differ in the ratio of number of low priority packets to high priority packets. As the number of packets increases, the average delay steadily decreases for the high priority packets, until it reaches the threshold. Even if the ratio changes, the time delay of high priority packets does not degrade.

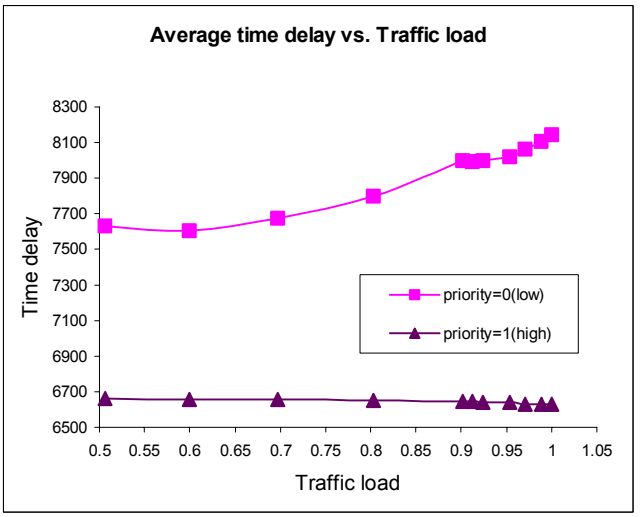

Fig. 5. Average time delay vs. Traffic load for switch size=64 and frame size =512, priority ratio=1.

Additionally, the experiments were conducted for the same packets, when no priority scheduling was performed. From the graph, we observe that there is a noticeable improvement in (see middle curve in Fig. 6) the time delay for high priority packets. We can also conclude from the graph that there is not much deterioration in the performance of low priority packets.

In Fig. 8, the experiments were performed, not assuming the smoothness property, unlike the previous experiments. The traffic distribution is Poisson and the packet loss rate decreases as the mean time between the arrivals of the

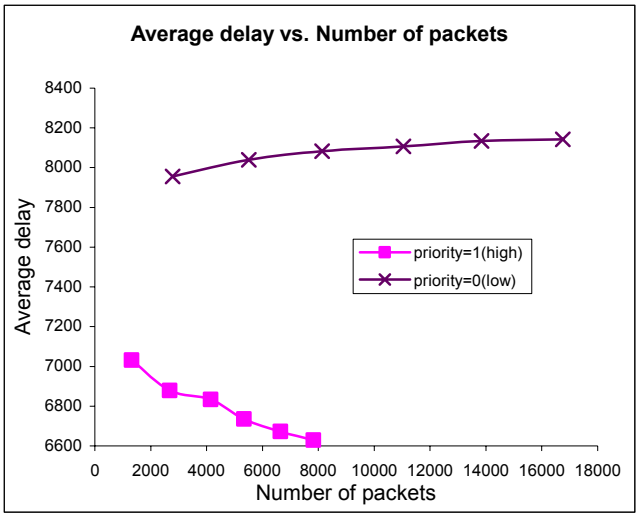

Fig. 4. Average delay vs. Number of packets for switch size =8, frame size =512, priority ratio =1.

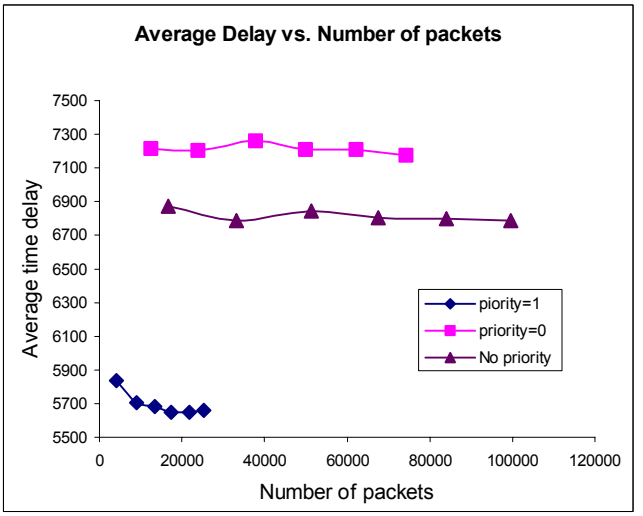

Fig. 6. Average delay vs. Number of packets for switch size=64, frame size =512 and priority ratio=3.

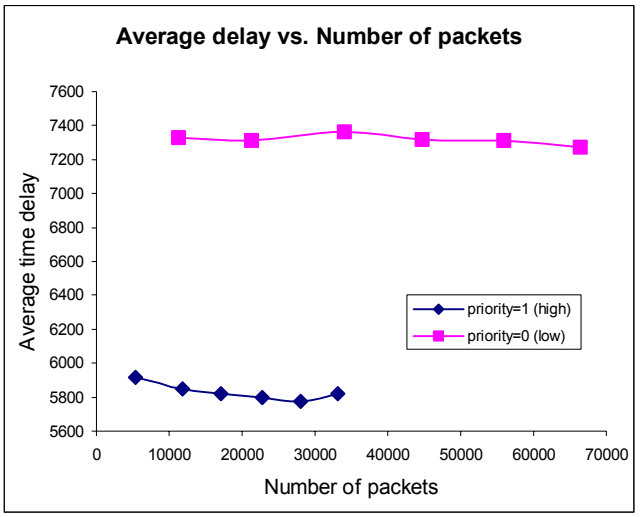

Fig. 7. Average delay vs. Number of packets for switch size =64, frame size =512 and priority ratio =2.

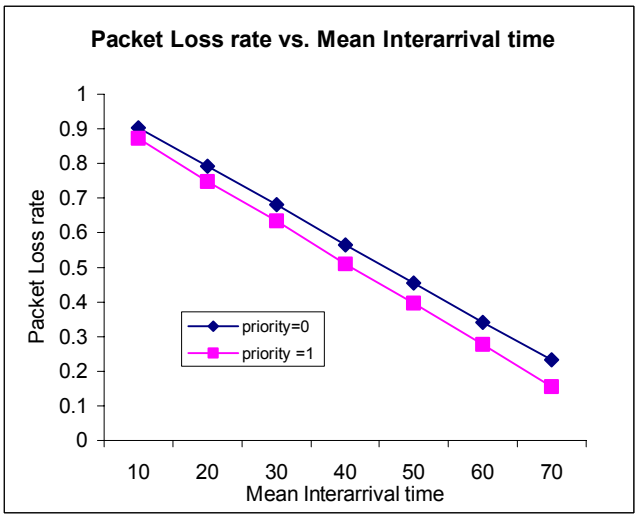

Fig. 8. Packet loss rate vs. Mean interarrival time, for switch size =8, frame size =512, slot size =35 and priority ratio =2.

packets increases. For high priority packets, packet loss rate is less than that for low priority packets, irrespective of the mean interarrival time.

VI. Conclusion

Differentiated services are provided and simulated in the proposed priority-based λ-scheduler architecture. The case when smoothness property is not satisfied is also simulated. The results indicate that our improved architecture gives better time delay for high-priority packets without degrading much the performance of low priority packets. The architecture also has the advantages of λ-scheduler, thereby having a very low component cost. This architecture will be very useful as the need for differentiated services arises because of future demands. This architecture can be extended to support variable length packets.

References

[1] B. Mukherjee, Optical Communication Networks, McGraw Hill, 1999.

[2] T. E. Stern, K. Bala, Multiwavelength Optical Networks: A Layered Approach, Prentice Hall, 1999.

[3] A. Huang, S. Knauer, "Starlight: A wideband digital switch," Proc. IEEE Global Telecommun. Conf. (GLOBECOM'84) Atlanta, GA, November 1984.

[4] Z. Hass, "The staggering switch: an electronically controlled optical packet switch," IEEE/OSA Journal of Lightwave Technology, vol. 11, pp. 925-936, 1993.

[5] E. A. Varvarigos, "The packing and the scheduling packet switch architectures for almost all-optical lossless Networks," IEEE/OSA Journal of Lightwave Technology, October 1998.

[6] J. P. Lang, E. A. Varvarigos, D. J. Blumenthal, "The λ-Scheduler: A multiwavelength scheduling switch," Proceedings Thirty-Seventh Annual Allerton Conference on Communication, Control and Computing, September 1999.

[7] A. Eckberg, D. Luan, D. Lucantoni, "An approach to controlling congestion in ATM networks," Proc. Int. J. Dig. Analog Communi. Syst., April-June 1990.

[8] M. G. Hluchyi, M. J. Karol, "Queueing in high-performance packet switching," J. Select Areas Commun., December 1988.

[9] C++SIM webpage, available at http://cxxsim.ncl.ac.uk/