SASA: A Synthesis Scheduling Algorithm with Prediction and Sorting Features

P.G.Sarigiannidis, G.I.Papadimitriou, Senior Member, IEEE, and A.S.Pomportsis
Department of Informatics, Aristotle University, Box 888, 54124 Thessaloniki, Greece
sarpan@csd.auth.gr

Abstract

We present a new scheduling algorithm, which is a synthesis of all the remarkable features of pretransmission coordination-based protocols for broadcast and select star networks. The new algorithm follows a novel policy for switching, in order to improve the efficiency of the network. It is based on all notable elements of a specific family of protocols and tries to accommodate the different ways of scheduling. The protocol manages to bring some improvement, in terms of channel utilization and network throughput, which is proven by simulation results.

1. Introduction

Seeing the ever-growing need for more bandwidth a lot of scientific research focuses to the improvement of utilization in the way of the access of the transmission media. If we ask about the kind of the transmission media, the response is easy, if we consider that the optical networking is able to satisfy the user's growing demands for bandwidth and also supports protocol transparency, more reliability than the traditional copper networks and in some cases easy and simple protocol functionality [1, 2]. In order to utilize in the best way the optical technology we need an appropriate multiplexing technique. Wavelength division multiplexing (WDM) seems to control the area of the multiplexing, since WDM offers an excellent way of exploiting the huge bandwidth of optical fibers, by introducing concurrency among multiple users transmitting at feasible rates [3, 4]. Such a network consists of one optical N×N passive star coupler and N nodes. Each node is connected via two way fibers to the optical star coupler and it can transmit with the aim of W channels. This paper focuses on the Broadcast-and-Select Star local area networks with one tunable transmitter and one fixed receiver (TT-FR) per node (Fig. 1).

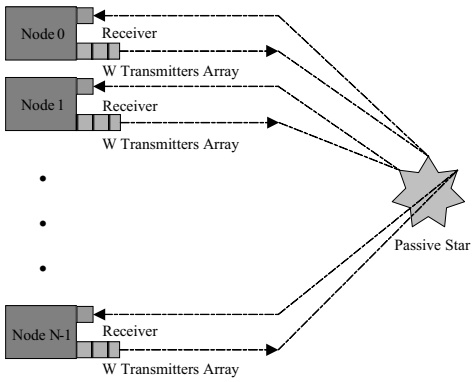

Figure 1. Broadcast and Select star network with tunable transmitter and fixed receiver per node.

So, each node can transmit in every channel in the network. Also, each channel has one dedicated channel, in order to accept data, named home channel [5].

It is very important for a network to adopt a common policy for the coordination between the nodes and the channels for some issues: a) which node will transmit, b) on which channel will transmit, and c) during which time the transmission will be executed. These responsibilities are dedicated to media access protocol (MAC). The subject of how nodes should access the various channels is a crucial issue for the efficient the network, since a MAC protocol usually aims to prevent collisions from occurring [6]. If we want to classify the MAC protocols for optical networks we can say that there are generally categorized as either pre-transmission co-ordination based or pre-allocation based. In pre-allocation MAC protocols channel allocations to transmitting nodes are fixed and scheduled in advance. So, the available wavelengths are used only as data channels and no wavelength serves as a shared control channel for coordination. Conversely, pre-transmission

Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)
0-7695-2588-1/06 $20.00 © 2006 IEEE

coordination based protocols perform schedule computation at the beginning of each time slot. These schemes differ from pre-allocation ones in that they designate at least one wavelength to be used as a control channel. The coordination process entails the following steps: the algorithm accepts initially the requirements of all nodes and organizes them in a transmission frame, called demand (or traffic) matrix, $D = [d_{i,j}]$. The matrix has N rows, as N is the number of the nodes and W columns, as W is the number of the channels. Hence, the cell in the i ($i \in N$) row, j ($.j \in$ W) column contains the amount of time (usually in timeslots), which i node requests to transmit on the j channel. Time is divided in timeslots. Usually, transmission is organized as frames where each frame is composed of a reservation phase followed by a data phase. The frame then stores for every node the number of timeslots required for transmission to a specific channel. Then the nodes transmit the requested data during the current frame at different moments. Lastly, a scheduling algorithm is referred to as offline, when it needs the entire demand matrix, i.e., the whole requests of all nodes to begin the construction of the schedule, while an online scheduling algorithm begins schedule computation just after reading the requests of the first node. In this work we study the performances of online algorithms.

2. Previous work

Online interval scheduling (OIS) [7] is a very simple online scheduling algorithm. It shares the advantage of online algorithms in that it does not require that the entire demand matrix becomes available before it begins schedule computation. Instead, OIS begins constructing the transmission schedule as soon as the requests of the first node are available. OIS functions as follows: assume that node n_1 requests t_1 timeslots for transmission in channel w_1 and that OIS determines that there is a time period in the next frame during which the requested channel is available i.e. there exists a value t such that the channel is available from timeslot t to timeslot $t + (t_1 - 1)$. As we mentioned the protocol must ensure the absence of collisions. In this context, OIS examines the possibility of contentions, i.e., the possibility of the concurrent transmission of two or more nodes on the same channel. If the algorithm concludes that the scheduled transmission does not result in any collisions, it includes it in the scheduling matrix that is being constructed. As a result, at any given timeslot the request table (scheduling matrix) of OIS comprises of the nodes that are scheduled to transmit and the wavelengths they will transmit in.

The basic problem with OIS is the large amount of time, of the schedule computation period of each frame. In order to decrease the delay that a ready node experiences while waiting for OIS to compute the schedule, predictive online scheduling algorithm (POSA) [8] attempts to eliminate the duration of the schedule computation process by predicting the nodes' requests for the next frame. In this direction POSA makes use of a hidden Markov chain and attempts to predict the requests of the nodes for the subsequent frame based on their requests for the previous frame. Because the algorithm does not wait for the nodes to send their requests in order to compute the schedule but starts working based on the predictions, a significant amount of time is saved. The predictor uses two different algorithms, the learning algorithm and the prediction algorithm. During each frame of data, the predictor first runs the learning algorithm and then the prediction algorithm. The first algorithm is responsible for informing and updating the data of the history queue, while the second one is responsible for predicting the demand matrix as accurately as possible. More details about the predictor can be found in [8].

Check and sort-predictive online scheduling algorithm (CS-POSA) [9] is an extension of POSA. Its aim is to extend POSA, while maintaining the pipelining of the schedule computation and the full operation of the predictor. The extension of CS-POSA is based on shifting of the schedule computation of the nodes or in other words, on guiding the order of checking and programming of the nodes. It examines the cumulative workload, i.e., the sum of the requests of each node to all destinations and based on it, it processes them in a declining order. Shifting is based on the workload of each node, which means that the CS-POSA comprehends better not only the general traffic of the network but also the specific workload in each node. Before CS-POSA constructs the schedule matrix, it takes the two following steps. In the first step CS-POSA adds each row of the traffic matrix D in a new vector S that will register the total amount of requests by each node. So, vector S consists of the total amount of the requests of the whole nodes for the whole transmission channels. In the second step CS-POSA grades vector S in a declining order. In case those two nodes are found with the same total number of requests, then the selection is random.

Wait for fullness (WFF) [10], is a protocol that is based on the two previous algorithms that were discussed namely OIS [7] and POSA [8]. WFF introduces a new schedule computation mechanism called the cleanup mechanism. It is actually a

procedure during which the timeslots that contain at least one idle channel are located and logically erased so that the total number of idle timeslots is minimized and the channel utilization is increased. When we refer to an idle timeslot we mean that during this timeslot no transmission is carried away. In other words during this timeslot channels remain idle. After constructing the scheduling matrix according to OIS, the lines of the matrix (corresponding to channels) are scanned one by one for all columns (i.e. for all timeslots). When a slot containing at least one idle channel is located it is logically erased which means that the transmissions it contains will be performed in one of the following frames. This means that the requests that were rescheduled will be added to the new requests that the nodes will send for the following frame and the actual data will continue to be stored in queues while their transmission is being scheduled.

The cleaning mechanism includes a process called refresh function during which the contents of the waiting queues are emptied. This action leads to a forced scheduling of all waiting packets, in the manner of OIS. This refresh function is performed at regular frame intervals.

3. The new synthesis protocol

The new proposed protocol is a synthesis of all previous scheduling algorithms. Synthesis accommodation scheduling algorithm (SASA) maintains all the positive and remarkable aspects of the OIS, POSA, CS-POSA, and WFF algorithms, with a co-operative way. Its goal is to maximize the efficiency of the network, in terms of channel utilization and network throughput. At the same time, tries to exploit all the advantages of the aforementioned algorithms, in order to combine a high network throughput with a low packet delay. The performance of the new protocol is examined via simulation tests. Also, we study the performance of the specified network supported by all the previous scheduling algorithms.

The basic idea of SASA is to combine the four specific elements of all the previous algorithms, with the best way. In other words we develop a dynamic scheme, which adopts 4 features from OIS, POSA, CS-POSA, and WFF. Let us see the source algorithm of each different element:

a) Simple online scheduling algorithm. SASA borrows from OIS the simplicity of its algorithm and the absence of any complex procedure or manufacture, keeping the complexity in low levels. Also, SASA adopts OIS's online feature and that means that SASA starts to build the schedule matrix, when the first set of requests is known.

b) Prediction mechanism. POSA is a very powerful scheme, which is based on predicting the traffic or demand matrix that is input to OIS. If we suppose that the network is comprised of N Nodes and W channels then the overall predictor is the output of a NXW matrix. Each cell of this matrix represents the expected next number of requested slots for the next frame for each node in each channel. This is the first feature, which SASA adopts from POSA. Apart from this, POSA operates in three phases: the learning phase, the switching phase and the prediction phase. In the first phase POSA operates in simple way, like OIS, without any predictions and supports with history information the queues of the predictors. During the second phase we suppose that the predictor is ready to predict, since it has learned the state of the network. Lastly, during the last phase POSA starts the prediction, hence the demand matrix is informed by the predictor and computes the schedule matrix with the predictor's input. The set of the three phases is adopted by SASA.

c) Shifting of channel servicing. The extension of CS-POSA is based on shifting of the schedule computation of the nodes. SASA adopts the shifting feature from CS-POSA with an additional characteristic. In case two or more nodes are found with the same total number of requests, then the algorithm chooses the node with the highest value, i.e., the node with the maximum value in its row. So, SASA gets from the predictor the constructed scheduling matrix for the following data frame and simultaneously shifts the order of control and service of the nodes, starting from the one with the additively most requests and finishing with the one with the least.

d) Cleanup and refresh mechanism. WFF introduces a new schedule computation mechanism called the cleanup mechanism, combined with a reset process, named refresh function. The function of the cleaning mechanism can be divided into the following four steps:

Locate the timeslots that contain at least one idle channel (referred to as idle timeslots).

Logically erase these timeslots and construct the scheduling matrix without these timeslots.

Reschedule the requests that were contained in the deleted timeslots.

At regular predetermined intervals perform the refresh function and schedule all stored (in queues) packets so as to put an upper bound on the incurred service delay. For the frame that the refresh function is performed, WFF functions as OIS or POSA, and the specific frame is named as refresh frame. This pair of processes is adopted by SASA.

4. SASA

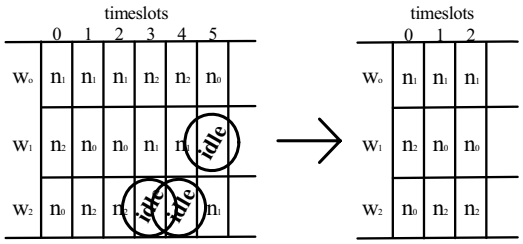

Figure 2. The function of clean up (frame f).

SASA acts in three phases: the learning phase, the switching phase and the prediction phase. During the first phase SASA processes like OIS. In the second one SASA changes and starts to construct the schedule matrix with the predictor's inputs. During the last phase SASA predicts the requests of each node for next frame and uses them to fill the demand matrix of the next frame. After constructing the scheduling matrix SASA examines the cumulative workload, i.e., the sum of the requests of each node to all destinations and based on it, it processes them in a declining order. At the end SASA functions the cleanup mechanism on the scheduling matrix. Firstly, SASA finds the idle timeslots, secondly SASA isolates the idle timeslots and reconstructs the new schedule algorithm, and thirdly reschedules the isolated requests for the next frame. As we mentioned before SASA performs at regular frame intervals the refresh function, in order to clean the queues of each node. In order to understand better the whole process of SASA, a specific example is examined. Let us suppose that the following demand matrix D_f has been constructed by nine individual predictors, for the frame f, for a network with three nodes (n_0, n_1, n_2) and three channels (w_0, w_1, w_2).

It is clear that the predictor $p_{0,0}$ predicted one timeslot for node n_0 with channel w_0, the predictor $p_{0,1}$ predicted two timeslots for node n_0 with channel w_1, and so on. The first job of SASA is to add each row of the traffic matrix D_f in a new vector S_f that will register the total amount of requests by each node. So, S_f= [1+2+1, 3+2+1, 2+1+2] = [4, 6, 5].

$$D_f \begin{bmatrix} 1..2..1 \\ 3..2..1 \\ 2..1..2 \end{bmatrix}$$

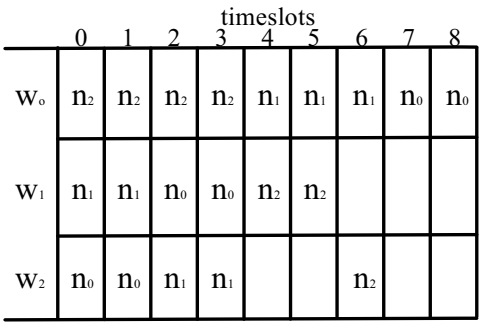

Figure 3. The schedule matrix of SASA after the sorting (frame f+1).

The second job of SASA is to grade vector S_f in a declining order. If two or more rows have the same value then the algorithm examines the maximum value of each row. In this way, vector S_f changes in the ordered vector S'_f = [6, 5, 4]. The new ordered vector S'_f denotes that the service order of the nodes. First of all requests of n_1 will be served, then the requests of n_2 and lastly the requests of n_0. The third job of SASA is to construct the schedule matrix, with the same manner like OIS and POSA (the left part of Fig. 3). At this point SASA functions the cleanup mechanism. The timeslots that include at least one idle channel are identified. These timeslots are 3, 4, and 5. Hence, idle timeslots are located and are isolated and a new schedule matrix is reconstructed (right part of Fig. 3.). It must be pointed out that the requests included in the isolated timeslots are not overlooked. SASA reschedules these requests in the frame that follows (together with the nodes' actual requests for the next frame). Therefore one timeslots for node n_0 and two timeslots for node n_2 in channel w_0, two timeslots for node n_1 in channel w_1, and one timeslot for node n_1 in channel w_2 will be rescheduled. The new demand matrix that contains the cumulative nodes' requests for frame f+1, denoted as $D`_f$. Let us suppose that the following demand matrix D_{f+1} has been constructed by nine individual predictors, for the frame f+1:

$$D'_f \begin{bmatrix} 1..0..0 \\ 0..2..1 \\ 2..0..0 \end{bmatrix} . \; D_{f+1} \begin{bmatrix} 1..2..2 \\ 3..0..1 \\ 2..2..1 \end{bmatrix} .$$

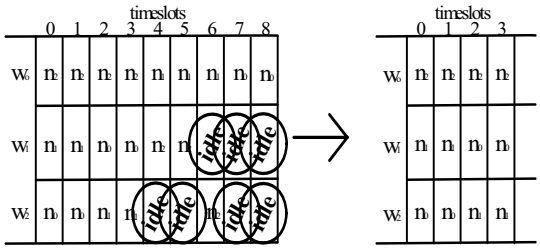

Figure 4. The function of clean up (frame f+1).

According to the cleanup mechanism the demand matrix for frame f+1 would be the result of the addition of tables $D`_f$+D_{f+1}. Therefore the new demand matrix for frame f+1 would be equal to:

$$D`_f+D_{f+1} = \begin{bmatrix} 1..0..0 \\ 0..2..1 \\ 2..0..0 \end{bmatrix} + \begin{bmatrix} 1..2..2 \\ 3..0..1 \\ 2..2..1 \end{bmatrix} = \begin{bmatrix} 2..2..2 \\ 3..2..2 \\ 4..2..1 \end{bmatrix}.$$

Again SASA adds each row of the demand matrix in a new vector S_{f+1} that will register the total amount of requests by each node. S_{f+1} = [2+2+2, 3+2+2, 4+2+1] = [6, 7, 7]. Now SASA grades vector S_{f+1} in a declining order. The value of the sum of n_1 and n_2 is the same, but the row of n_2 has maximum value equal to four, when the row of n_1 has maximum value equal to three. In this context the algorithm chooses to serve n_2. In this way, vector S_f changes in the ordered vector S'_f = [7, 7, 6]. First of all requests of n_2 will be served (sum is equal to seven and the maximum rate is equal to four), then the requests of n1 (sum is equal to seven and the maximum rate is equal to three) and lastly the requests of n_0 (sum is equal to six). So, SASA will construct the following scheduling matrix (Fig. 4.). At this point SASA functions the cleanup mechanism. The timeslots that include at least one idle channel are identified. These timeslots are 4, 5, 6, 7, and 8. Hence, idle timeslots are located and are isolated and a new schedule matrix is reconstructed (Fig 5.). SASA reschedules the isolated requests in the frame that follows. Therefore two timeslots for node n_0 and three timeslots for node n_1 in channel w_0, two timeslots for node n_2 in channel w_1, and one timeslot for node n_2 in channel w_2 will be rescheduled. The new demand matrix that contains the cumulative nodes' requests for frame f+2, denoted as $D`_{f+1}$ is shown below. Let us examine one last frame with the following demand matrix D_{f+2}:

$$D'_{f+1} = \begin{bmatrix} 2..0..0 \\ 3..0..0 \\ 0..2..1 \end{bmatrix}. \; D_{f+2} = \begin{bmatrix} 1..1..1 \\ 0..2..2 \\ 2..1..2 \end{bmatrix}.$$

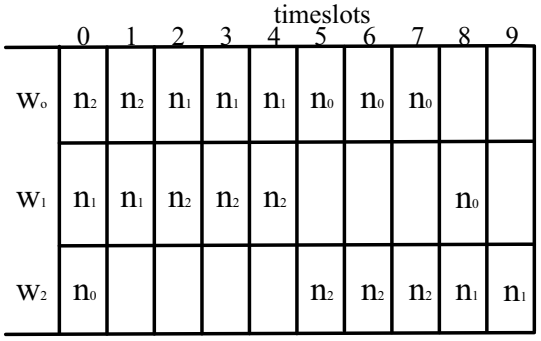

Figure 5. The schedule matrix of SASA after the function of refresh mechanism (frame f+2).

The whole demand matrix for frame f+2 would be the result of the addition of tables $D`_{f+1}$+D_{f+2}. Therefore the new demand matrix for frame f+2 would be equal to:

$$D`f+1+Df+2 = \begin{bmatrix} 2..0..0 \\ 3..0..0 \\ 0..2..1 \end{bmatrix} + \begin{bmatrix} 1..1..1 \\ 0..2..2 \\ 2..1..2 \end{bmatrix} = \begin{bmatrix} 3..1..1 \\ 3..2..2 \\ 2..3..3 \end{bmatrix}$$

Again SASA adds each row of the demand matrix in a new vector S_{f+2} that will register the total amount of requests by each node: S_{f+2} = [3+1+1, 3+2+2, 2+3+3] = [5, 7, 8]. Now SASA grades vector S_{f+2} in a declining order. In this way, vector S_{f+2} changes in the ordered vector S'_{f+2} = [8, 7, 5]. First of all requests of n_2 will be served, then the requests of n_1 and lastly the requests of n_0. If we assume that refresh function is performed during frame f+2 then the final schedule matrix for frame f+2 will be like that in Figure. 6.

5. Simulations results

In this section, we present the results of a set of performance comparison experiments between the four scheduling algorithms: POSA, CS-POSA, WFF, and SASA. The behavior of the four algorithms is presented under uniform traffic. In the results of the simulation, it is assumed that N is the number of nodes; W is the number of the channels and K is the maximum value over all entries in the demand matrix. Also, it should be mentioned that the tuning latency time is considered to be equal to zero timeslots for simplicity reasons. The values range between 0 and K and in order the goal of scalability to be achieved, the value of K is not constant in the following experiments

but each time it is equal to: Floor(NW/5).

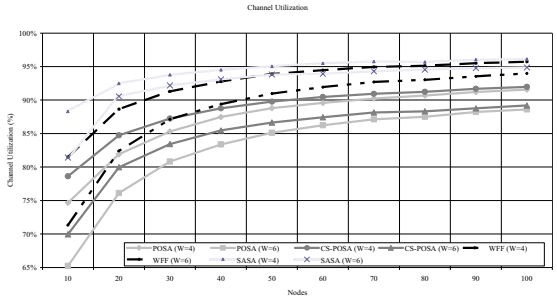

Figure 6. Channel utilization

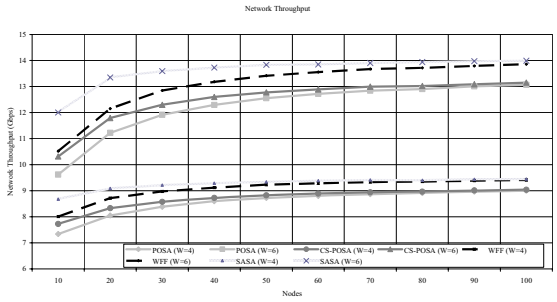

Figure 7. Network throughput

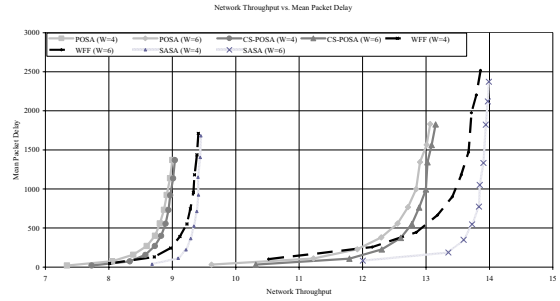

Figure 8. Throughput vs. delay

The line speed has been set in 2.4 Gbps. The metric, called refresh rate, means how often refresh function is executed and is equal to 5 for WFF and SASA. The results from the comparison between the four algorithms, in terms of channel utilization are shown in Figure 6 for four and six channels. It is clear that SASA is obviously improved from all the others, with a maximum difference with WFF of 10% in ten nodes and six channels and 16% with POSA. Figure 7 shows the comparison of the algorithms in terms of network throughput. It is obvious that SASA improves the throughput of the network with a maximum difference of 1400 Mbps with WFF and 3700 Mbps when nodes are equal to 10 and channels are 6. Lastly, Figure 8 shows the relation throughput-delay. It can be observed that for each value of K, SASA improves the network throughput, while keeps lower mean packet delay from WFF.

6. Conclusions

In this paper, we have proposed a new synthesis algorithm for scheduling data in a WDM broadcast and select star network. The proposed algorithm maintains the considerable elements of the pre-transmission coordination based protocols. It has shown that the proposed algorithm can improve the performance of the network, regardless of the amount of the nodes or the amount of the channels in the network.

7. References

[1] P. Green, “Progress in optical networking, IEEE Communications magazine”, vol. 39, no. 1, 2001, pp. 54-61.

[2] G. I. Papadimitriou, Ch. Papazoglou, and A. S. Pompotrsis, “Optical Switching : Switch Fabrics, Techniques, and Architectures”, IEEE/OSA Journal of Lightwave Technology, vol. 21, no. 2, 2003, pp. 384-405.

[3] G. I. Papadimitriou, P. A. Tsimoulas, M. S. Obaidat, and A. S. Pomportsis, Multiwavelength Optical LANs, Wiley, 2003.

[4] S. Chatterjee and S. Pawlowski, “All-optical networks,” Commun. ACM,vol. 42, no. 6, pp. 74–83, June 1999.

[5] G. I. Papadimitriou and A. S. Pomportsis, “Self-Adaptive TDMA protocols for WDM star networks: A learning-automata-based approach,” IEEE Photon. Technol. Lett., vol. 11, pp. 1322–1324, Oct. 1999.

[6] C.A. Brackett, “Dense wavelength division multiplexing network: Principles and applications”, IEEE J. Selected Areas Commun., vol. 8, 1990, pp. 948-964.

[7] K. M. Sivalingam, J. Wang, J. Wu and M. Mishra, “An interval-based scheduling algorithm for optical WDM star networks”, Photonic Network Communications, vol. 4, no. 1, 2002, pp. 73-87.

[8] E. Johnson, M. Mishra, and K. M. Sivalingam, “Scheduling in optical WDM networks using hidden Markov chain based traffic prediction, Photonic Network Communications”, vol. 3, no. 3, 2001, pp. 271-286.

[9] P. G. Sarigiannidis, G. I. Papadimitriou, and A. S. Pomportsis, “A New Prediciton and Channel Sorting Based Scheduling Algorithm for WDM Star Networks”, 12th Annual Symposium of the IEEE/CVT, Nov. 3, 2005, Enschede, the Netherlands

[10] P. G. Sarigiannidis, G. I. Papadimitriou, and A. S. Pomportsis, “WFF: A High Performance Scheduling Algorithm for WDM Star Networks that Minimizes Idle Timeslots”, 12th Annual Symposium of the IEEE/CVT, Nov. 3, 2005, Enschede, the Netherlands