Multicast Scheduling in WDM Switching Networks

Zhenghao Zhang and Yuanyuan Yang
Dept. of Electrical & Computer Engineering, State University of New York, Stony Brook, NY 11794, USA

***Abstract*—Optical WDM networks are attracting more and more attentions because of its huge bandwidth to meet the ever increasing demand of modern networking applications. In this paper we study supporting multicast in WDM switching networks. Multicast is the operation to send information from one source to multiple destinations. In WDM switching networks, contention occurs when one output fiber is the destinations of more than k inputs, where k is the number of wavelengths on each fiber. In this paper, we study scheduling algorithms which can select a group of multicast connection requests that are contention-free. We first prove that the problem of scheduling the maximum number of such connection requests through the network simultaneously is NP-hard. On the other hand, lack of optical buffers in WDM switching networks requires a very fast scheduling algorithm. We then turn to develop approximation scheduling algorithms that can provide sub-optimal solutions. We present four polynomial approximation scheduling algorithms and study their performance through simulations. We also discuss their performance ratio to the optimal algorithm. Our results demonstrate that one of the simple algorithms (the fastest one) yields throughput close to other three more complex algorithms, thus could be a good candidate for multicast scheduling in WDM switching networks.**

I. Introduction

Many networking applications, such as video conferencing, video on demand, E-commerce and image distributing, require very high network bandwidth often far beyond that today's high-speed networks can offer. Optical networking is a promising solution to this problem because of the huge bandwidth of optics: a single fiber has a bandwidth of nearly 50 THz [1]. To fully utilize the bandwidth, a fiber is divided into a number of independent channels, with each channel having a different wavelength and carrying its own data. This is called *wavelength-division-multiplexing (WDM)*. It has been reported that on a single fiber there can be up to 1,022 channels [16]. It is believed that optical WDM networks will serve as the backbone networks in the near future.

Multicast is the operation that sends information from one source to multiple destinations. The applications mentioned earlier typically require multicast operations for efficiency purpose. In this paper we study supporting multicast in WDM switching networks (also called WDM switches in some literature). The network we consider is an $m \times m$ network, i.e., there are m fibers on the network input side and m fibers on the network output side. On each fiber there are k wavelengths that carry independent data. Thus, there are a total of mk inputs in the network. Each wavelength on each input fiber may be connected to any wavelength on any output fiber. The change in wavelength can be achieved by using *wavelength converters*. Moreover, we assume that the network is multicast-capable, which means that an input wavelength may be connected to more than one output wavelengths at the same time, with the restriction that an input wavelength cannot be connected to more than one output wavelengths on the same output fiber. [1] showed how to construct such a switching network in a nonblocking manner.

The research work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085 and CCR-0207999.

Future WDM switching networks will most likely be operated in a synchronous mode [15], in which connection requests or optical packets arrive at a switching network at the beginning of time slots. We consider the case where each connection holds for the same time length and is of the same priority. In a k-wavelength WDM switching network, *output contention* occurs if an output fiber is the destination of more than k connection requests arrived during a time slot. Thus, a general scheme for resolving such contentions needs to be studied. Figure 1 illustrates an example of multicast connections with contentions in a 3×3 WDM switching network with 4 wavelengths on each fiber. The nodes on the left side represent input wavelengths of the network and the circles on the right side represent output fibers. Input wavelengths are also associated with a number according to their fiber and wavelength indexes, as shown in the figure. We draw a link between input x and output y if y is one of the destinations of the connection originated from x. Since there are only four wavelengths on each fiber in this example, when more than four connections share a destination fiber, contention occurs. We can see that in Figure 1, contention occurs on output fibers 1 and 3. In this case, we need to select some of the connection requests to transmit while reject or buffer others. As optical buffers are currently made of fiber delay lines and are still very expensive [4], we consider unbuffered WDM switching networks in this paper. Connection requests that cannot be realized are dropped. Lack of optical buffers also means that we cannot split a multicast connection and realize only a portion of it each time. In other words, we consider the situation that a multicast connection from the same source is either fully realized in the network or blocked. This is referred to as "one-shot" in the literature [8]. In Figure 1, we can drop the connection originated from λ_3 on input fiber 1 (input 3) and realize the others. Alternatively, we can drop the connections originated from inputs 9 and 12 on input fiber 3 and realize the others. However, fewer connection requests are realized in the latter case.

In general, we model an $m \times m$ WDM multicast switching network with k wavelengths per fiber equipped with wavelength converters as a network with mk inputs and m outputs, where each output can be involved in up to k connections simultaneously. For any $1 \leq i \leq m$ and $1 \leq j \leq k$, the input corresponding to λ_j on input fiber i is represented by number $(i-1) \times k + j$. At a certain time slot, up to mk multicast requests may arrive at the network inputs. If the connection re-

0-7803-7802-4/03/$17.00 © 2003 IEEE

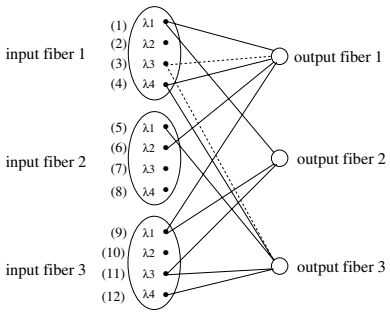

Fig. 1. *Multicast connections in a* 3×3 *WDM switching network with 4 wavelengths per fiber.*

quests do not have any contention, they can be realized simultaneously. If they do have contention, a scheduling algorithm can be used to select some of them to transmit and drop other unselected connection requests, as in the example discussed above. To maximize the network throughput, we need to find the largest group of connection requests that are contention (or blocking) free. In the example in Figure 1, the largest group of connection requests that can be transmitted simultaneously is $\{1, 4, 5, 6, 9, 11, 12\}$.

Extensive research has been conducted on scheduling algorithms for electronic switching networks in the literature. For example, [10] and [11] considered scheduling algorithms in input-buffered electronic switching networks for unicast traffic, and [12] and [9] proposed scheduling algorithms for the same type of networks for multicast traffic, in which request-splitting was allowed to achieve higher throughput. However, due to the reasons mentioned earlier, in WDM switching networks, we need to consider one-shot scheduling algorithms in which request-splitting is not allowed. Scheduling algorithms have also been proposed for multicast traffic in WDM single-hop networks, where nodes are connected in a star topology by a passive coupler, see, for example, [13]-[14], for a survey. In this type of network, an input node chooses one wavelength (channel) and broadcast its information to all output nodes by the passive coupler. If an output node is its destination it should "tune in" this wavelength. At any time only one wavelength is carrying data, either on the fiber of the input side or the output side in a time-division manner. On the other hand, a WDM switching network allows each wavelength on each fiber to be utilized simultaneously in a space-division manner. To the best of our knowledge, no research has been reported on how to schedule multicast requests to resolve output contentions in WDM switching networks. In Section 2, we will formalize the problem of finding the largest group of contention-free multicast requests as *connection request packing problem (CRP)*. We will show that this problem is not only NP-hard but also hard to approximate. Another issue is that a good scheduling algorithm for a WDM switching network should be very fast, since in a time slotted scheme, a cell or an optical packet must be transmitted in a single time slot. Thus, the decision on whether a cell can be transmitted or not must be made real time within a time slot, which is in the order of μs if ATM cells are transmitted [4]. Therefore, we are facing a very challenging problem: approximating a NP-hard problem within a very limited amount of time.

II. Preliminaries

In this section we give the definitions and terms we are going to use in this paper.

Definition 1: The multicast connection requests arrived at an $m \times m$ WDM switching network with k wavelengths during a time slot can be represented by an $m \times mk$ binary matrix, called *request matrix* and denoted as R, in which each row represents an output fiber and each column represents a multicast connection request from an input. Entry $r_{i,j}$ is 1 if and only if output fiber i is one of the destinations of connection request from input j.

For example, the request matrix for the example in Figure 1 is:

$$R = \begin{bmatrix} 101101001000 \\ 100000001010 \\ 001110000011 \end{bmatrix}$$

In the following discussions, we will also call the connection requests *columns*.

Definition 2: We define the *row degree* of a row in the request matrix as the number of 1's in the row, and the *column degree* of a column as the number of 1's in the column. Furthermore, for a subset of the columns in the request matrix, written as P, if a row has t '1's in P, we say this row has degree t in P.

Definition 3: Connection request packing problem: Given an $m \times mk$ binary matrix, find the largest group of columns such that no any row has more than k 1's in these columns. In this paper we will also refer to it as the *CRP* problem.

Definition 4: We give a measure for the performance of approximation algorithms. For a request matrix R, let the number of columns selected by the optimal algorithm be N_{opt}. Let N_{app} be that of an approximation algorithm. The *performance ratio* of this approximation algorithm is defined as the largest value of N_{opt}/N_{app} among all possible request matrices for a given network size. Thus, the ratio indicates how much the optimal algorithm outperforms the approximation algorithm in the worst case. The smaller the ratio, the better the approximation algorithm. We also say that an algorithm approximates the *CRP* problem within β if it has a performance ratio β.

III. NP-Hardness of the Connection Request Packing Problem

In this section, we study some useful properties of the CRP problem defined in the last section. We will prove that the CRP problem is not only NP-hard but also hard to approximate. We first show that the *CRP* problem is NP-hard when $k = 1$. We then prove the general case by induction.

Lemma 1: Unless NP = ZPP, i.e., NP-hard problems can be solved in expected polynomial time by a probabilistic algorithm that never makes mistakes [5], the connection request packing problem cannot be approximated within $m^{1/2-\epsilon}$ in polynomial time for any $\epsilon > 0$ when $k = 1$.

Proof. We can use an undirected graph G with v vertices to describe the *CRP* problem as follows. Let $m = v$, and let U be the vertices of G numbered from 1 to m, i.e., $U = \{1, 2, \ldots, m\}$. Suppose M is the adjacency matrix of G. Thus, M is an $m \times m$ binary matrix. M can be used as the request matrix for an $m \times m$ k-wavelength WDM switching network when $k = 1$. It is not difficult to see that the solution to the *CRP* problem is also the solution to the *maximum independent set* problem for G. It is well known that the latter is NP-hard. By using the results in [5], no efficient $m^{1/2-\epsilon}$ approximation algorithm exists for the problem unless NP = ZPP. Thus, the lemma follows. ∎

The result of Lemma 1 can be applied to a k-wavelength WDM switching network with no wavelength conversion. This is because that an $m \times m$ k-wavelength WDM switching network with no wavelength conversion can be viewed as the aggregation of k independent $m \times m$ 1-wavelength networks. We now give the proof for a WDM switching network with full wavelength conversion.

Theorem 1: Unless NP = ZPP, the connection request packing problem cannot be approximated within $m^{1/2-\epsilon}$ in polynomial time for any $\epsilon > 0$ and for any positive integer k.

Proof. We prove the theorem by induction. By Lemma 1, when $k = 1$, the theorem is true. Now suppose the theorem holds for $k = l$. For any given instance for $k = l$, we append an $m \times m$ identity matrix I to the right of the request matrix. For example, we can append a 3×3 identity matrix to the request matrix of the example in Figure 1 as follows:

$$R = \begin{bmatrix} 101101001000|100 \\ 100000001010|010 \\ 001110000011|001 \end{bmatrix}$$

We then have an instance for $k = l + 1$. Now if there exists an efficient $m^{1/2-\epsilon}$ algorithm for $k = l + 1$, we can apply this algorithm to this instance. Suppose S is the solution, where S is a group of columns in the request matrix. The first thing to notice is that if S contains the identity matrix I we appended, then after removing the identity matrix, the remaining columns, written as T, are the solution to the original instance of $k = l$. This is true because by definition, if S is the solution to the *CRP* problem, then S is the maximum number of columns such that no row has more than $l + 1$ 1's in S. If T is not the solution to the original instance, there exists another group of columns T' which also satisfies the constraint that no row has more than l 1's, but with more columns. Now no row has more than $l+1$ 1's in T' and I. But there are more columns in T' and I than in S, which contradicts the assumption that S is the solution to the instance of $l + 1$. We conclude that if S contains I, we can obtain the solution to the original instance easily.

Now suppose I is not within S completely. Without loss of generality, suppose columns $e_1, e_2, \ldots, e_p$ are not in S, where e_j is the j_{th} column in I. We can apply a swapping procedure to transform S into S' which is also a solution to the *CRP* problem but with I in S'. Starting with e_1, we can find a column in S that has a 1 entry in row 1. This is true because if we cannot find such a column in S, we can simply add column e_1 to S which results in more columns than S while not violating the constraint. We then swap this column with e_1. The same procedure can be applied to $e_2, e_3, \ldots, e_p$. In each step, we can always find a column in S with 1 entry in the desired row. Notice that this procedure can be done in polynomial time. Thus, there is a $m^{1/2-\epsilon}$ polynomial time algorithm for $k = l+1$, and there is also a $m^{1/2-\epsilon}$ polynomial time algorithm for $k = l$. This contradicts with the assumption. Hence, the theorem follows. ∎

Algorithm 1
Step 0: $P \leftarrow \emptyset$;
let $S = \{S_1, S_2, \ldots, S_n\}$ be the columns in matrix A;
Step 1: Select the column in S with the smallest index, say, S_j; remove S_j from S,
$P \leftarrow P \cup \{S_j\}$;
Step 2: Remove from S the columns that have at least one 1 entry in at least one of the rows that have degree k in P.
if $S = \emptyset$ stop and output P else goto Step 1.

Algorithm 2
Step 0: $P \leftarrow \emptyset$;
let $S = \{S_1, S_2, \ldots, S_n\}$ be the columns in matrix A;
sort the columns according to their degrees in an ascending order;
Step 1: Select the column in S with the smallest index, say, S_j; remove S_j from S,
$P \leftarrow P \cup \{S_j\}$;
Step 2: Remove from S the columns that have at least one 1 entry in at least one of the rows that have degree k in P.
if $S = \emptyset$ stop and output P else goto Step 1.

The results of Theorem 1 indicate that: (1) the *CRP* problem is NP-hard; (2) it is hard to find an approximation algorithm that guarantees a good performance ratio. Since $m^{1/2}$ grows very fast with m, for a relatively large network, we cannot expect the solution of heuristic to be close to the optimal algorithm in the worst case.

IV. Scheduling Algorithms

In this section we study scheduling algorithms that give suboptimal solutions. For presentational convenience, for a given request matrix at a time slot, we remove the columns with all 0 entries because these columns represent those idle inputs and need not to be considered in scheduling. Let the simplified matrix be A. Suppose there are n columns left. Thus, A is an $m \times n$ binary matrix. Entry $a_{i,j}$ is 1 when output i is one of the destinations of input j, otherwise $a_{i,j}$ is 0. In the following, the input to a scheduling algorithm is the simplified request matrix A. The output is P, the set of the columns selected.

We will start with a very simple algorithm shown in Table Algorithm 1. What this algorithm does can be viewed as a sequential search. If the connection requests arrive at random,

Algorithm 3

Step 0: $P \leftarrow \emptyset$; $u \leftarrow 0$;
set the threshold $\alpha \leftarrow m/2$;
divide the columns of the request matrix into two classes, one with degrees less than α denoted by T_1 and the other denoted by T_2; columns within one class are in an arbitrary order.

Step 1: $u \leftarrow u + 1$;
if $u > 2$, stop and output P else goto Step 2;

Step 2: Select the column in T_u with the smallest index, say, S_j, remove S_j from T_u;
$P \leftarrow P \cup \{S_j\}$;

Step 3: Remove the columns in T_i, $u \leq i \leq 2$, that have at least one 1 entry in at least one of the rows that have degree k in P.
if $T_u = \emptyset$ goto Step 1 else goto Step 2.

Algorithm 4

Step 0: $P \leftarrow \{1, 2, \ldots, n\}$;

Step 1: If no row in the request matrix has more than k 1's stop and output P else goto Step 2;

Step 2: Find the row with the maximum degree;
among the columns that have a 1 entry in this row, find the column with the maximum degree; say, l_{th};
$P \leftarrow P - \{l\}$;
remove this column from the request matrix;
goto Step 1.

it is equivalent to a random picking algorithm. Since Steps 1 and 2 may be executed at most n times, this algorithm has a complexity of $O(n)$. The actual time could be much less, as in Step 2 we may remove several columns in each iteration. Notice that we do not require any sort of ordering in the columns in this algorithm. Thus, we can simply let them in the same order as the inputs. The solution yielded by Algorithm 1 contains at least k columns. The optimal solution, on the other hand, can have at most mk columns. Hence, we have the following result.

Theorem 2: The number of columns in the optimal solution is no more than m times that in Algorithm 1, or the performance ratio of Algorithm 1 is m.

Note that in this algorithm lower-indexed connection requests are more likely to be selected because they are tested first. To ensure the fairness to each input, we can implement Algorithm 1 in a round-robin manner. Suppose at some time slot Algorithm 1 starts from input i. Then in the next time slot, it will move to input $(i + k) \mod mk$, (the first k columns are guaranteed in P by the algorithm). Due to its simplicity, Algorithm 1 can be easily implemented in hardware.

Algorithm 2 is shown in Table Algorithm 2. The only difference between Algorithm 2 and Algorithm 1 is that Algorithm 2 sorts the columns according to their degrees so that columns with fewer number of 1's are picked first. Intuitively, it will yield better solutions than Algorithm 1. However, the penalty is the additional time required to sort the columns or perform equivalent operations. Generally, a sorting algorithm has a complexity of $O(n \log n)$, so does Algorithm 2.

Algorithm 3 shown in Table Algorithm 3 is a simplified version of Algorithm 2. The columns are divided into only 2 classes (instead of m classes in Algorithm 2) based on their degrees. Columns in the small degree class are tested first. This algorithm can also be easily implemented in hardware. Steps 2 and 3 may be repeated up to n times. Thus, it is also an $O(n)$ algorithm. However, it may provide a higher throughput than Algorithm 1. Threshold α should be adjusted according to the fanout distribution of multicast connections. We can also divide the columns into more than 2 classes but this will increase the hardware cost.

Algorithm 4 is described in Table Algorithm 4. In Algorithm 4, we first find the maximum degree row, then delete the maximum degree column that has a 1 entry in this row. By doing this we guarantee that while maximum row degree is reduced by one each time, we reduce the degrees of maximum number of columns. The algorithm keeps doing this until no row has a degree more than k. No sorting is needed in this algorithm, but we need to search for the maximum degree row and column in each iteration. Thus, the complexity of this algorithm is $O(mn)$.

V. Simulation Results and Discussions

In this section we study network throughput of the algorithms presented in the previous section through simulations. In our simulations we consider a time slotted $m \times m$ WDM multicast switching network with k wavelengths per fiber. We assume optical packet switching in the network. The general assumptions are: (1). The length of an optical packet is 1 time slot. (2). For a packet, the probability that an output fiber is one of its destinations is θ and independent of other output fibers. Thus the fanout of a packet may range from 0 to m and follows Binomial distribution (m, θ). We define network throughput as the ratio of realized multicast connection requests over arrived multicast connection requests. We conducted simulations under two types of traffic, the Bernoulli traffic and the bursty traffic. In Bernoulli traffic, at each time slot, for each input, the probability that there is an arriving packet is b and independent of other inputs. The arrival probability for each time slot is also independent of other time slots. In bursty traffic, we assume that each input independently alternates between the "active" state and the "idle" state. In an active state, packets arrive continuously and go to the same destinations. In an idle state, no packet arrives. The duration of active and idle states follows geometric distribution with parameter p and q, respectively. The expected duration of the active and idle state is $1/p$ and $1/q$, respectively.

We conducted simulations on networks of different sizes and wavelengths. Each simulation runs for 2000 instances. The results are shown in Figures 2(a) - 2(b). In Figure 2(a), we plot the throughput under Bernoulli traffic for different arrival probability b, where θ which controls multicast fanout is set to be 0.5 so that the average fanout is $m/2$. Two network sizes are considered, $m = k = 16$ and $m = k = 32$. In Figure 2(b), we plot the throughput under bursty traffic for different average

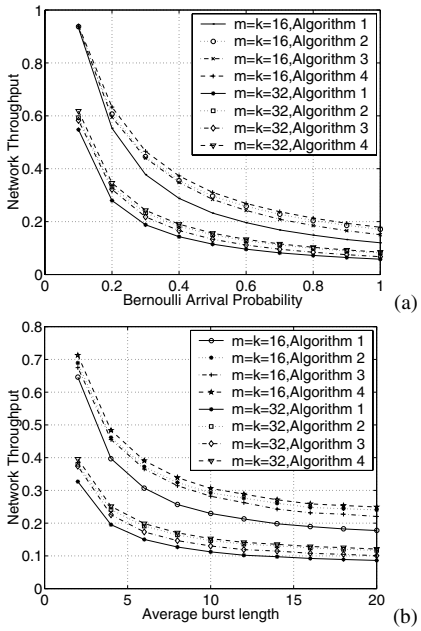

Fig. 2. *Network throughput. (a) Bernoulli traffic, average multicast fanout* $= m/2$ $(\theta = 0.5)$. *(b) Bursty traffic, average multicast fanout* $= m/2$ $(\theta = 0.5)$*, average idle state duration* $= 10$ $(q = 0.1)$.

burst lengths, where the expected idle period is set to be 10 $(q = 0.1)$ for convenience. Again $\theta = 0.5$, and the networks considered are of size $m = k = 16$ and $m = k = 32$. From these two figures we can observe that: (1) The throughput of Algorithm 4 is the highest, then Algorithm 2, Algorithm 3, and finally Algorithm 1; (2) The performance of these algorithms, measured by network throughput, is very close. For example, in Figure 2(a), the simulation results show that under Bernoulli traffic, for $m = k = 32$ the throughput of Algorithm 1 is no less than 75% of Algorithm 4. The same fact can also be observed in Figure 2(b) for bursty traffic. For $m = k = 16$, the difference is slightly larger. The worst case is in Figure 2(a) when $b = 1.0$, Algorithm 1 is about 66% of Algorithm 4.

From the simulation results we can see that the performance of these algorithms are quite close. We now make some further comparisons of them. First we compare the complexity. Algorithms 2 and 4 have higher complexity than the other two. The complexity of Algorithm 3 is higher than Algorithm 1. Algorithms 1 and 3 can be implemented in hardware while Algorithms 2 and 4 have to be implemented in software. Next we consider fairness. When we design scheduling algorithms for multicast traffic, we always face a dilemma: on one hand, to maximize network throughput, we should select multicast connection requests with smaller fanouts; on the other hand, if all requests are of the same priority, requests with larger fanouts should be treated equally, or have the same chance being selected. Since Algorithms 2, 3 and 4 all are greedy algorithms, they favor columns with small degrees. Thus, larger degree columns have a higher probability of being rejected. Algorithm 1 is fairer to the requests with large fanouts, because the first picked k columns will always be selected regardless of their fanouts. Overall, we believe that Algorithm 1 is more practical. Although other three algorithms provide higher throughput, they have some drawbacks such as high time complexity or hardware complexity and possible starvation to multicast connection requests with large fanouts.

VI. Conclusions

In this paper we have studied the problem of scheduling a group of contention-free connection requests in WDM multicast switching network. We have proved that finding the maximum number of such requests is NP-hard. We then presented four scheduling algorithms and discussed their performance ratio to the optimal algorithm. We also studied their performance through simulations. The simulation results show that although varying greatly in complexity, the four scheduling algorithms have similar performance under the testing conditions. Thus we recommend Algorithm 1 which is faster, easier to implement and also fairer to multicast connection requests with large fanouts.

References

[1] Y. Yang, J. Wang and C. Qiao "Nonblocking WDM multicast switching networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 12, pp. 1274-1287, 2000.

[2] M. M. Halldorsson "Approximation of weighted independent set and hereditary subset problems," *Journal of Graph Algorithms and Applications*, vol. 4, no. 11, pp. 1-16, 2000.

[3] R. Gonen and D. Lehmann "Optimal solutions for multi-unit combinatorial auctions: branch and aound heuristics," *unpublished manuscript*, May 2000.

[4] D. K. Hunter, M. C. Chia and I. Andonovic "Buffering in optical packet switches," *Journal of Lightwave Technology*, vol. 16 no. 12, pp. 2081-2094, 1998.

[5] J. Hastad "Clique is hard to approximate within $n^{1-\epsilon}$," *Acta Mathematica*, vol. 182, pp. 105-142, 1999.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, 1st Edition, W.H. Freeman and Company, 1979.

[7] M. Kovacevic and A. Acampora, "Benefits of wavelength translation in all-optical clear-channel networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 868 -880, June 1996.

[8] X. Chen and J.F. Hayes "Call scheduling in multicast packet switching," in *Proc. IEEE ICC'92*, pp. 895 - 899. 1992.

[9] B. Prabhakar, N. McKeown and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 855-866, June 1997.

[10] N. McKeown, P. Varaiya, and J. Warland, "Scheduling cells in an input-queued switch," *IEEE Eletron. Lett.*, pp. 2174-2175, Dec. 1993.

[11] N. McKeown, "The iSLIP scheduling algorithm input-queued switch," *IEEE/ACM Trans. Networking*, vol. 7, pp. 188-201, Apr. 1999.

[12] W.-T. Chen, C.-F. Huang, Y.-L. Chang and W.-Y. Hwang, "An efficient cell-scheduling algorithm for multicast ATM switching systems," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 517 -525, Aug. 2000.

[13] T. Kitamura, M. Iizuka, M. Sakuta, Y. Nishino, and I. Sasase, "A new partition scheduling algorithm by prioritizing the transmission of multicast packets with less destination address overlap in WDM single-hop networks," *IEEE GLOBECOM'01*, vol. 3, pp. 1469 - 1473, 2001.

[14] H.-C. Lin and C.-H. Wang, "A hybrid multicast scheduling algorithm for single-hop WDM networks," *IEEE INFOCOM 2001*, vol. 1, pp. 169 - 178, 2001.

[15] H. J. Chao, C. H. Lam and E. Oki, *Broadband Packet Switching Technologies*, 1st Edition, Wiley-Interscience publication, 2001.

[16] Walter J. Goralski, *Optical Networking and WDM*, 1st Edition, McGraw-Hill, 2001.