On Scheduling Optical Packet Switches With Reconfiguration Delay

Xin Li, *Student Member, IEEE*, and Mounir Hamdi, *Member, IEEE*

***Abstract*—Using optical technology for the design of packet switches/routers offers several advantages such as scalability, high bandwidth, power consumption, and cost. However, reconfiguring the optical fabric of these switches requires significant time under current technology (microelectromechanical system mirrors, tunable elements, bubble switches, etc.). As a result, conventional slot-by-slot scheduling may severely cripple the performance of these optical switches due to the frequent fabric reconfiguration that may entail. A more appropriate way is to use a time slot assignment (TSA) scheduling approach to slow down the scheduling rate. The switch gathers the incoming packets periodically and schedules them in batches, holding each fabric configuration for a period of time. The goal is to minimize the total transmission time, which includes the actual traffic-sending process and the reconfiguration overhead. This optical switch scheduling problem is defined in this paper and proved to be NP-complete. In particular, earlier TSA algorithms normally assume the reconfiguration delay to be either zero or infinity for simplicity. To this end, we propose a practical algorithm ADJUST that breaks this limitation and self-adjusts with different reconfiguration delay values. The algorithm runs at $O(\lambda N^2 \log N)$ time complexity and guarantees 100% throughput and bounded worst-case delay. In addition, it outperforms existing TSA algorithms across a large spectrum of reconfiguration values.**

***Index Terms*—Optical packet switch, reconfiguration delay, scheduling, time slot assignment (TSA).**

I. Introduction

THE EXPLOSION of Internet traffic has brought about an acute need for broadband communication networks. This strengthens the demand for high-performance switches/routers than ever before. Although transmission line rates have increased rapidly over the past years [OC-3 (155 Mb/s) to OC-192 (10 Gb/s)], the use of dense wavelength division multiplexing (DWDM) technology makes the aggregate switch throughput grow mainly along with the increase of port number rather than port speed. However, due to the considerations in physical implementation, such as port failure avoidance, port density does not change much. To further comply with the strict network equipment building system (NEBS) physical-packaging requirements on the number of cards in a single rack, it is necessary to distribute the whole switch/router over several racks.

Manuscript received July 30, 2002; revised March 28, 2003. This work was supported in part by the Hong Kong Research Grant Council under Grant RGC HKUST6202-99E. This paper was presented in part at the IEEE International Conference on Communications (ICC), Anchorage, AK, May 2003.

The authors are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: lixin@cs.ust.hk; hamdi@cs.ust.hk).

Digital Object Identifier 10.1109/JSAC.2003.815843

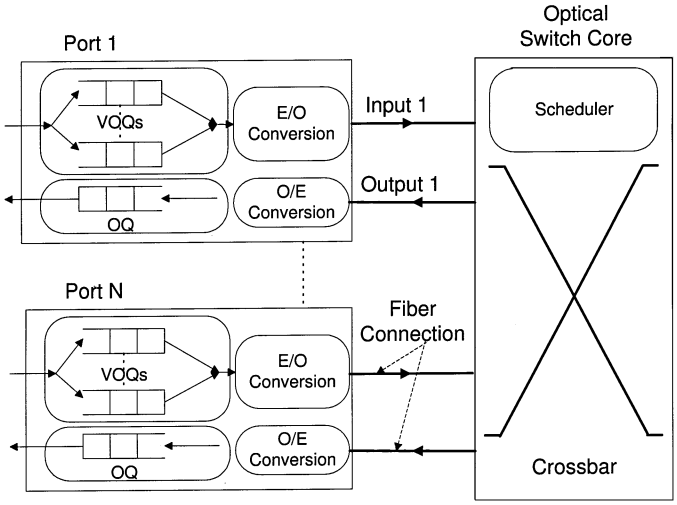

Fig. 1. System architecture of multirack hybrid packet switch.

In multirack systems, cables replace backplanes. It is possible to transport electrical signals between different racks, using coax, twisted pair, etc., but considering there maybe potentially thousands of signals traveling between different racks, this quickly leads to a cabling nightmare. One technology increasingly used to solve the problem is optical fiber interconnections. Since buffering in optical domain, such as using fiber delay lines (FDLs), is still immature, most packets are still stored in electronic buffers. If traditional electronic switching fabric is used, a series of electro-opitcal/opto-electronic conversions are needed at the buffer/interconnection and interconnection/fabric edge. This translates to significant cost and power consumptions.

An obvious solution is to use optical switching fabric to reduce the number of opto-electronic conversions. Optical fabrics based on two–dimensional (2-D)/three-dimensional (3-D) microelectromechanical system (MEMS) mirror [1], thermal bubble [2], waveguide [3], and similar technologies have been developed. They further provide potential benefits including scalability, high bit rate, and low power consumption on economical bases. The architecture of the switch is shown in Fig. 1. It is a hybrid structure with electronic buffers, optical switching fabric, and optical interconnections.

However, reconfiguring the fabric connections for these switches is more difficult than that of their electronic counterparts. For example, the MEMS-based optical switch needs to adjust the angles of the fabric mirrors to set up new connections. This introduces mechanical settling, synchronization, and other time-consuming operations. Normally, the reconfiguration

overheads range from milliseconds to microseconds. This is around 10^{-1} to 10^{5} slot time for a system with slotted time equals to 50 ns (64 bytes at 10 Gb/s).

It is obvious that traditional slot-by-slot scheduling method may severely cripple the performance of optical switches because of frequent fabric reconfiguration. The scheduling rate has to be slowed down and each schedule holds for some time. Time slot assignment (TSA) method [5]–[11] is a common approach to achieve this. The system periodically accumulates incoming traffic and determines a set of schedules to deliver them. Schedules are held for some certain length to achieve the shortest total transmission time of all accumulated traffic.

Unfortunately, the existence of reconfiguration delay complicates the scheduling problem. Transmission time now includes not only actual traffic-sending time but reconfiguration time as well . The corresponding transmission-time optimization problem [named as optical switch scheduling (OSS) problem] is proven to be NP-complete in this paper. Although extensive studies about TSA algorithms have been conducted, to the best of our knowledge, almost all of them assume the reconfiguration delay is zero [5]–[8] or infinity [9], [10] for simplicity. The assumption leads to a biased attention on minimizing either the actual traffic-sending time or reconfiguration overhead, thus preventing these approaches perform well under medium reconfiguration overhead. A recently proposed DOUBLE algorithm [11] tries to use a near minimum number (2N, N is the number of switch ports) of configurations to achieve some balance. However, it ignores the effect of reconfiguration delay when schedules the traffic. For two systems with the same traffic arrival pattern but different reconfiguration overhead, DOUBLE always enforces the same scheduling strategy.

Obviously, this can be further improved. Taking system parameters such as reconfiguration delay δ, length of accumulating period T and number of switch ports N into consideration, we propose a new scheduling algorithm ADJUST. The algorithm uses a regulating factor $\lambda = \sqrt{T/\delta N}$ to self-adjust with different systems. According to mathematical deduction and simulation, ADJUST outperforms DOUBLE and the previous extreme approaches across a large range of reconfiguration values. It runs with $O(\lambda N^2 \log N)$ time complexity, guarantees to send out traffic within 3T time slots, and is stable for all admissible traffic patterns.

The remainder of this paper defines the scheduling problem and then explores the design of the ADJUST algorithm. Section II introduces the notations and switch model used throughout the paper. OSS problem is formally defined in Section III. Section IV is a review of previous research which only consider zero or infinite reconfiguration overhead values. The ADJUST algorithm is described and discussed in detail in Section V. Finally, the conclusion and future work is presented in Section VI. NP-completeness proof of the OSS problem is included in the Appendix.

II. Preliminaries

Below is a list of notations used throughout the paper:

N number of switch ports;
δ reconfiguration overhead in slot times;
S internal speedup of switch;
T traffic accumulation interval in slot times;
D cumulative traffic matrix in T time slots;
s number of schedules for traffic matrix D;
P fabric configuration/schedule (permutation matrix);
ϕ holding length of schedule (weight);
λ regulating factor of ADJUST.

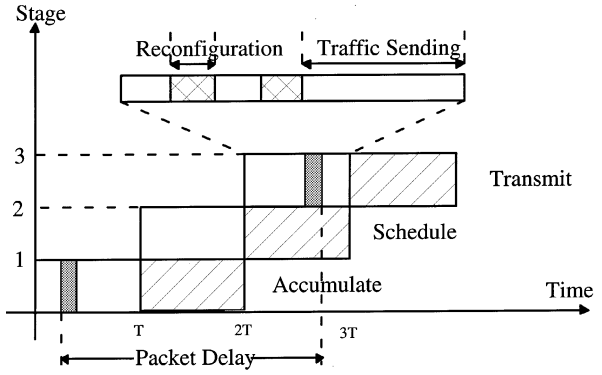

Fig. 2. Three-stage pipeline diagram.

This paper is based on a nonblocking switch model running in slotted manner. Arbitrary fabric configuration P (one-to-one input–output mapping) can always be set up. P is 0–1 matrix with at most one nonzero element for each row and column. $p_{ij} = 1$ results in connecting input i to output j. A fixed, positive reconfiguration overhead δ is entailed whenever the switch fabric resets. Speedup S is needed to compensate this switching overhead. It is the ratio of internal line rate to the input line rate.

The switch works in an accumulate-schedule-transmit cycle. The length of the accumulating stage is set to be a predefined system constant T. Incoming traffic in these T time slots is stored in traffic matrix D. $D = (d_{ij})$ is a non-negative integer $N \times N$ matrix. d_{ij} represents the number of packets received in input i whose destination is output j during the accumulating stage. Scheduling stage then finds out a set of schedules for the accumulated traffic. Holding lengths of the schedules are determined to achieve shortest transmission time. Transmission stage follows the scheduling decision, alternatively changing the fabric between actual traffic-sending state and reconfiguration state. The operations can be further pipelined as shown in Fig. 2.

The transmission stage is composed of a set of traffic-sending substages and reconfiguration substages. One particular traffic-sending substage may contains several input–output connections, which all hold connected until none of them have traffic to send. Since some input ports may have transmitted out all their packets while others still have some left, connections may be wasted (see Fig. 3). These idle connections are called empty time slots.

The pipeline diagram indicates that a packet will go through the switch within $3T$ slot times (here, we assume scheduling period equals to T for simplicity). Traffic from at most three different batches may appear in a particular input buffer at the same time. Assume B is the number of bits sent to one input per time slot. A buffer of size $3TB$ is enough for each input and $2TB$ for each output. If all ports are considered, the switch needs at most $5TBN$ bits buffer size. Since the traffic will always be sent out

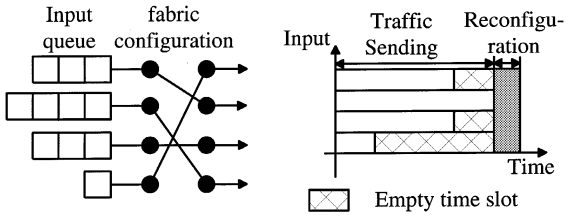

Fig. 3. Illustration of empty time slots.

in bounded time, the scheduling algorithm is stable under any admissible traffic patterns. Furthermore, bounded worst delay $(3T)$ makes it easy to provide quality-of-service (QoS) guarantees.

III. OSS Problem

This section presents the TSA scheduling task in mathematical forms. The OSS problem is a NP-complete optimization problem.

Definition 1 (Admissible Traffic): If traffic matrix D stands for the packets that are accumulated during T time slots, a traffic pattern is admissible only when the line sum of D is no larger than T. That is, $\sum_{i=1}^{N} d_{ij} \leq T$ and $\sum_{j=1}^{N} d_{ij} \leq T$. This paper assumes traffic to the switch is admissible.

Definition 2 (Matrix Covering): A traffic matrix D is covered by a set of fabric configurations $P^{(1)}, \ldots, P^{(s)}$ and corresponding weights $\phi_1, \ldots, \phi_s$, if $\sum_{k=1}^{s} \phi_k p_{ij}^{(k)} \geq d_{ij}, \forall i, j \in 1, \ldots, N$. Here, $p_{ij}^{(k)}$ is the (i, j) element of configuration matrix $P^{(k)}$. In the case of equality for all i and j, the switch configurations *exactly cover* D.

Definition 3 (Covering Cost): Given a traffic matrix D and reconfiguration delay δ. If a particular set of switch configurations $P^{(1)}, \ldots, P^{(s)}$ with weights $\phi_1, \ldots, \phi_s$ covers the traffic matrix, the covering cost is defined as $cost = \sum_{k=1}^{s} \phi_k + s\delta$.

Definition 4 (OSS Problem): Given a traffic matrix D, reconfiguration delay δ and accumulation time T, find a set of switch configurations and their respective weights, which covers the traffic matrix and minimizes the covering cost.

Input: An $N \times N$ non-negative integer matrix D, positive integer δ, and T, D satisfies

$$\sum_{i=1}^{N} d_{ij} \leq T, \quad \sum_{j=1}^{N} d_{ij} \leq T, \quad \forall i, j \in 1, \ldots, N. \tag{1}$$

Output: A set of configuration matrices $P^{(1)}, \ldots, P^{(s)}$ and the corresponding non-negative integer weights $\phi_1, \ldots, \phi_s$ satisfy

$$\sum_{k=1}^{s} \phi_k p_{ij}^{(k)} \geq d_{ij}, \quad \forall i, j \in 1, \ldots, N \tag{2}$$

$$\sum_{k=1}^{s} \phi_k + s\delta \text{ is minimized.} \tag{3}$$

Admissible traffic ensures none of the switch output is overloaded. If the configuration matrices cover the traffic matrix, all of the accumulated traffic can be sent out. OSS tries to find out those matrices which minimize the transmission time.

Theorem 1: Optimization problem OSS is NP-complete when $\delta > 0$.

Proof: The proof involves a reduction from the timetable design problem [12]. Please refer to the Appendix for details. The authors in [13] offers some basic explanation of NP-completeness and related concepts.

The above theorem forces us to abandon the research for efficient polynomial time algorithm that achieves optimal solutions. We can only achieve suboptimal solutions using heuristic approach.

IV. Related Works on Simplified OSS Problem

This section introduces some previous works. Most of them assume reconfiguration delays to be zero or infinity to simplify the OSS problem. Comprehension of their execution strategy helps to understand the property of OSS problem and motivate the proposal of ADJUST algorithm.

A. Zero Reconfiguration Delay

Please note OSS problem with zero configuration overhead is not NP-complete and its optimal solution is achieved by many algorithms [5], [8], [10]. Equation (3) now changes to minimize $\sum_{k=1}^{s} \phi_k$. The lower bound of $\sum_{k=1}^{s} \phi_k$ equals $\max(\sum_{i=1}^{N} d_{ij}, \sum_{j=1}^{N} d_{ij})$, $\forall i, j \in 1, \ldots, N$, which is the maximum line sum. As a representative, *TSA-1* algorithm [5] achieves this lower bound using a system of distinct representatives (SDR) method. A large number of configurations may be used and is bounded in $N^2 - 2N + 2$. It has a $O(N^5)$ time complexity.

B. Infinite Reconfiguration Delay

Normally, reconfiguration delay is considered to be infinite if it is comparable large to T. Now, $s\delta$ occupies the major portion of covering cost $\sum_{k=1}^{s} \phi_k + s\delta$. Equation (3) is satisfied only when s is first minimized. The lower bound of s is $\max(\max_i(r_i), \max_j(c_j))$, $\forall i, j \in 1, \ldots, N$, where $r_i(c_j)$ is the number of nonzero entries on row i (column j). The OSS problem with infinite configuration delay is still NP-complete, and only a suboptimal solution can be achieved. The representative is the K-transponders (KT) algorithm proposed in [9]. KT uses the least possible number of configurations and thus minimizes the $s\delta$ part. Furthermore, it tries to group the matrix entries which are roughly equal in the same switching matrix. This helps to avoid empty time slots and, hence, reduces $\sum_{k=1}^{s} \phi_k$. KT has an $O(N^4)$ time complexity.

V. Adjust Algorithm

Although assuming the reconfiguration delay to be extreme values may greatly simplify the problem, scheduling algorithms need to deal with medium reconfiguration values in most cases. An example in Fig. 4 shows two different methods for covering a traffic matrix. The strategy in Fig. 4(a) covers the matrix exactly with four configurations, using $12 + 4\delta$ time slots. Its alternative

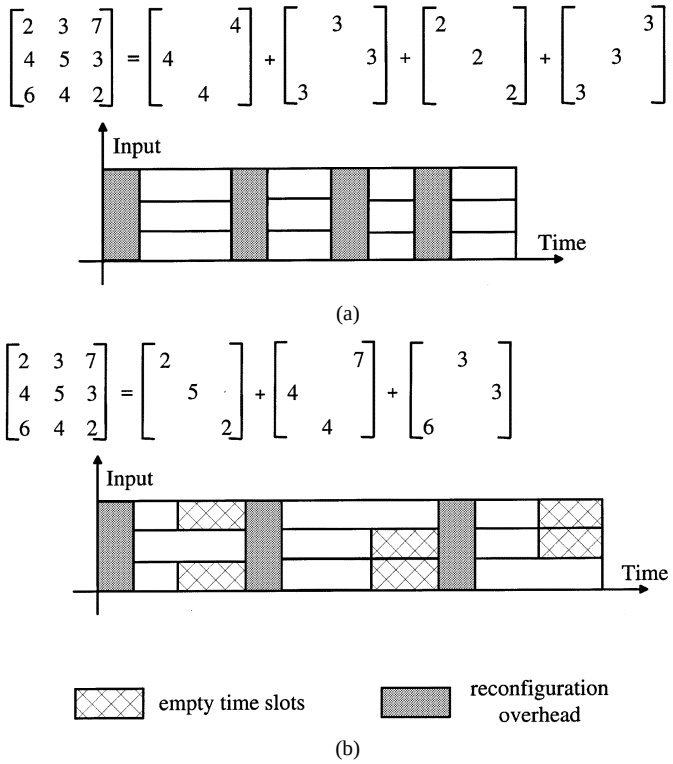

Fig. 4. Tradeoff between empty time slots and number of configuration matrices.

in Fig. 4(b) uses only three configurations with cost $18+3\delta$. The better strategy is determined by different δ values.

A. Tradeoff Between Empty Time Slots and Reconfiguration Overheads

Extreme approaches, such as TSA-1 and KT, perform well at their targeting cases, but they are not scalable to medium reconfiguration delay values. The covering cost of TSA-1 is bounded by $T + (N^2 + 2N - 2)\delta$, which grows at least with the square of the switch port number. This is unacceptable for a system with large port number or reconfiguration delay. KT algorithm avoids the above problem by extending the holding length of the configurations, taking the risk of introducing more empty slots. Although KT groups similar elements together to reduce the empty time slots, it cannot fully avoid wastage. The authors in [11] show the traffic-sending time may be as large as $\Omega(\log N)$ times of the actual time needed.

The problem of TSA-1 and KT is that they only concentrate on minimizing one of the two influencing factors: number of configurations or empty time slots, rather than finding a balance between them. To keep the summation of traffic-sending time and reconfiguration overhead minimal, the algorithms should carefully take care both of them.

B. Algorithm Description

The idea of ADJUST is motivated from using a moderate number of configurations to achieve balance. The algorithm should self-adjust to different system parameters and has low time complexity.

ADJUST works by separating traffic D into a *quotient* matrix and a *residue* matrix and assigning configurations to each.

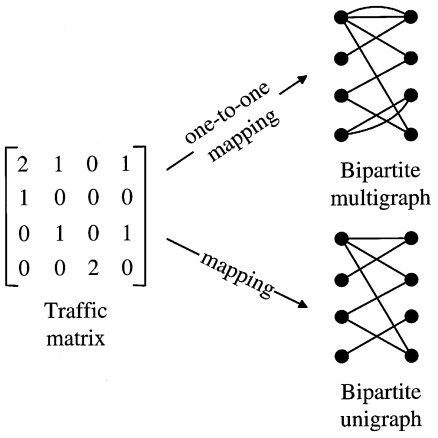

Fig. 5. One-to-one mapping between traffic matrix and bipartite multigraph.

The algorithm generates quotient matrix A by dividing the elements in D by $T/\lambda N$ and taking the floor. That is, $a_{ij} = \lfloor (d_{ij})/(T/\lambda N) \rfloor$, $1 \le i, j \le N$. Here, λ is the regulating factor. Choosing an appropriate value for λ is crucial to the performance of the algorithm.

Lemma 1: The line sum of the quotient matrix is bounded by λN.

Proof: The admissible traffic ensures, for $\forall i, j \in 1, 2, \ldots, N$, $\sum_{i=1}^{N} d_{ij} \le T$, $\sum_{j=1}^{N} d_{ij} \le T$. Consider a row in quotient matrix A

$$\sum_{j=1}^{N} a_{ij} = \sum_{j=1}^{N} \left\lfloor \frac{d_{ij}}{\frac{T}{\lambda N}} \right\rfloor \le \left\lfloor \frac{\sum_{j=1}^{N} d_{ij}}{\frac{T}{\lambda N}} \right\rfloor \le \left\lfloor \frac{T}{\frac{T}{\lambda N}} \right\rfloor \le \lambda N.$$

Proof for columns in A follows the same way.

Lemma 2: An $N \times N$ traffic matrix D with maximum line sum L_s can be covered using L_s configuration matrices.

Proof: There exists a one-to-one mapping between this matrix-covering problem and edge-coloring problem for bipartite multigraph. The upper part of Fig. 5 shows an example of mapping between traffic matrix and bipartite multigraph. Input and output ports are mapped to be vertex sets. If $d_{ij} = \omega$ in traffic matrix D, there are ω edges between the corresponding vertices. Bipartite multigraph with maximum degree L_s is proven to be L_s colorable. Vertices and edges belonging to a particular color can be mapped back to a switch configuration. For detailed proof of the edge coloring [14, Ch. 5].

Corollary 1: Quotient matrix A can be covered using λN configuration matrices.

Proof: This comes directly from Lemma 1 and Lemma 2.

The residue matrix B is defined as $b_{ij} = \max(0, d_{ij} - \lceil T/\lambda N \rceil \times a_{ij})$. From its construction method, we can see $0 \le b_{ij} < \lceil T/\lambda N \rceil$. The simplest way to cover it is to find N configuration matrices which collectively represent an all $1s$ matrix, each with weight $\lceil T/\lambda N \rceil$. KT algorithm may provide better performance with higher time complexity $(O(N^4))$. Edge-coloring algorithm is another candidate and its input now is a bipartite "unigraph" (lower part of Fig. 5) constructed from residue matrix B. All three covering methods use at most N configurations. Furthermore, the latter two use the minimum possible number of configurations.

By now, we can see

$$D = \left\lceil \frac{T}{\lambda N} \right\rceil \times A + B \leq \left\lceil \frac{T}{\lambda N} \right\rceil \times \sum_{k=1}^{\lambda N} P^{(k)} + \sum_{k=\lambda N+1}^{\lambda N+N} \left\lceil \frac{T}{\lambda N} \right\rceil \times P^{(k)}. \quad (4)$$

The covering cost of quotient and residue matrix is

$$\begin{aligned} \text{cost} &= \sum_{k=1}^{s} \phi_k + \delta s \\ &= \left(\sum_{k=1}^{\lambda N} \left\lceil \frac{T}{\lambda N} \right\rceil + \delta \times \lambda N \right) + \left(\sum_{k=\lambda N+1}^{\lambda N+N} \left\lceil \frac{T}{\lambda N} \right\rceil + \delta N \right) \\ &= (T + \delta N) + \left(\delta \lambda N + \frac{T}{\lambda} \right). \end{aligned} \quad (5)$$

The first part of (5) is a constant determined by system settings. In order to minimize (5), $\delta\lambda N + T/\lambda$ needs to be minimized.

Corollary 2: Covering cost of ADJUST is minimized when $\lambda = \sqrt{T/\delta N}$.

Proof: Since $\delta\lambda N \times T/\lambda = \delta NT = \text{constant}$, the covering cost of ADJUST is minimized when $\delta\lambda N = T/\lambda$. That is, to set the regulation factor λ equivalent to $\sqrt{T/\delta N}$.

DOUBLE algorithm previously proposed in [11] can be viewed as a special case of ADJUST, which always sets $\lambda = 1$. Traffic matrix D is divided by T/N and generates two matrices: *coarse* matrix A $a_{ij} = \lfloor (d_{ij})/(T/N) \rfloor$ and *fine* matrix B $b_{ij} = max(0, d_{ij} - \lceil T/N \rceil \times a_{ij})$. Both matrices can be covered using N configurations with weight $\lceil T/N \rceil$.

An example of the ADJUST execution is shown in Fig. 7(c). For $N = 3$, $T = 48$, and $\delta = 1$, the regulating factor $\lambda = \sqrt{T/\delta N} = 4$. Quotient matrix A is generated by first dividing each element of traffic matrix D by $T/\lambda N = 4$ and then taking the floor. For example, a_{11} is calculated by $a_{11} = \lfloor d_{11}/4 \rfloor$. The edge coloring algorithm is used to find out the configurations to cover A. Configurations for residue matrix B can be found using any algorithm listed above.

The two main operations of ADJUST determine its time complexity. Dividing the traffic matrix D into quotient and residue part takes $O(N^2)$ time. The edge-coloring algorithm in Step 2 has a complexity of $O(E \log V)$ [15]. E is the number of edges and V is the number of vertices in the bipartite multigraph. Because the bipartite multigraph corresponds to quotient matrix A, $V = O(N)$, and $E = O(\lambda N^2)$. As a whole, ADJUST has a time complexity of $O(\lambda N^2 \log N)$.

C. Discussion

Table I shows a detailed comparison between the four algorithms we mentioned in this paper: TSA-1, KT, DOUBLE, and ADJUST. They are compared with each other on time complexity, covering cost bound and the number of configurations used to cover the traffic matrix. ADJUST and DOUBLE are generally better in terms of cost bound and time complexity.

TABLE I
COMPARISON BETWEEN TSA-I, KT, DOUBLE, AND ADJUST

	Configuration Number	Covering Cost	Time Complexity
TSA-1	$N^2 - 2N + 2$	$T + (N^2 - 2N + 2)\delta$	$O(N^5)$
KT	N	$T \log N + N\delta$	$O(N^4)$
DOUBLE	$2N$	$2T + 2N\delta$	$O(N^2 \log N)$
ADJUST	$(\sqrt{T/\delta N} + 1)N$	$T + \delta N + 2\sqrt{\delta TN}$	$O(\lambda N^2 \log N)$

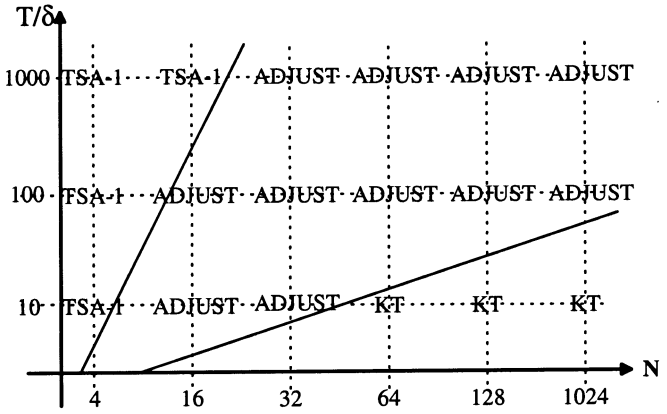

Fig. 6. Minimum speedup. The algorithm which requires minimum speedup at the crosspoints is labeled.

The pipelined scheme mentioned in Section II requires traffic accumulated in T time slots to be sent out in T time slots. Speedup S is needed to compensate the reconfiguration overhead suffered in transmission stage. S equals total transmission time over the accumulation time. For example, minimum speedup for DOUBLE and ADJUST are $2 + 2N\delta/T$ and $1 + N\delta/T + 2\sqrt{N\delta/T}$. Under the same system parameter (N, δ, T), speedup required by ADJUST is always less than that of DOUBLE. Fig. 6 plots which algorithm gives the minimum speedup S for some commonly used values of N and T/δ. For small port number ($N = 4$) or relatively small reconfiguration delay ($T/\delta = 1000$), the TSA-1 algorithm is better because it minimizes the empty time slots and the induced reconfiguration delay is small. If the reconfiguration delay is relatively large ($T/\delta = 10$), KT algorithm which targets in reducing the number of schedules takes the advantage. However, for a large portion of test region, ADJUST is better.

Algorithm 1 ADJUST algorithm

Input:
$N \times N$ non-negative integer matrix D, positive integer δ and T

Output:
a set of configuration matrices $P^{(1)}, \ldots, P^{(s)}$ and the corresponding non-negative integer weights $\phi_1, \ldots, \phi_s$

Description:
Set the regulating factor $\lambda = \sqrt{T/\delta N}$. Split traffic matrix D into quotient

matrix A and residue matrix B, such that

$$a_{ij} = \left\lfloor \frac{d_{ij}}{\frac{T}{\lambda N}} \right\rfloor$$
$$b_{ij} = \max\left(0, d_{ij} - \left\lceil \frac{T}{\lambda N} \right\rceil \times a_{ij}\right)$$
$$1 \le i, j \le N$$

Schedule quotient matrix A

i) Construct an $N \times N$ bipartite multigraph G_A from A. Vertices in G_A stand for switch ports. The number of edges between vertices is equal to the value of the corresponding entry in A.
ii) Find a minimal edge coloring of G_A. For detailed procedures, please refer to [15].
iii) Map the edge-coloring results back to switch configurations. Edges with a specific color in G_A correspond to a switch configuration $P^{(i)}$. Repeat this step until no color is left. Set the holding length of configurations to be $\lceil T/\lambda N \rceil$.

Schedule residue matrix B

i) Method a. Find any N nonoverlapping switch schedules whose summation is an all 1's matrix, and set ϕ_i to be $\lceil T/\lambda N \rceil$, or
ii) Method b. Use KT algorithm
iii) Method c. Use edge-coloring algorithm. Construct an $N \times N$ bipartite unigraph G'_B from B. Vertices in G'_B stand for switch ports. There is one edge between vertex i and j if $b_{ij} \neq 0$. The following operation is similar to what is done to G_A.

An example is given in Fig. 7 to further clarify the execution of KT, DOUBLE, and ADJUST. In this particular example, δ is set to be relative small $(\delta = 1)$, compared with accumulating length $(T = 48)$. It is not surprising that KT has the largest covering cost due to huge number of wasted time slots. DOUBLE gives a schedule with a cost of 61. It is not so cost saving because the algorithm selects a partition factor $T/N = 16$, which incurs a large fine matrix with high cost (see [11] for execution detail). In other words, the choice of 16 as a partition factor makes the two matrices unbalanced. On the other hand, ADJUST finds an appropriate number of reconfiguration matrices, and also maintains a small deviation for elements in the same configuration. That is why it achieves the best result of all the three algorithms presented.

$$D = \begin{bmatrix} 28 & 8 & 2 \\ 4 & 20 & 16 \\ 2 & 20 & 20 \end{bmatrix} \quad N=3,\ T=48,\ Delay=1$$

a) KT algorithm

$$\begin{bmatrix} 0 & 0 & 2 \\ 0 & 20 & 0 \\ 2 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 8 & 0 \\ 4 & 0 & 0 \\ 0 & 0 & 20 \end{bmatrix} + \begin{bmatrix} 28 & 0 & 0 \\ 0 & 0 & 16 \\ 0 & 20 & 0 \end{bmatrix}$$

s=3, Traffic-sending=20+20+28=68, cost=71

b) DOUBLE algorithm

$$16\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 12 & 8 & 2 \\ 4 & 4 & 0 \\ 2 & 4 & 4 \end{bmatrix}$$
$$= 16\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right) + \left(\begin{bmatrix} 12 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 8 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 2 \\ 4 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix}\right)$$

s=2+3, Traffic-sending=16x2+12+8+4=56, cost=61

c) ADJUST algorithm

$$4\begin{bmatrix} 7 & 2 & 0 \\ 1 & 5 & 4 \\ 0 & 5 & 5 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$
$$= 4\left(\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 4 \\ 0 & 5 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) + \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

s=3+1, Traffic-sending=4x(5+5+2)+2=50, cost=54

Fig. 7. Example of execution of KT, DOUBLE, and ADJUST.

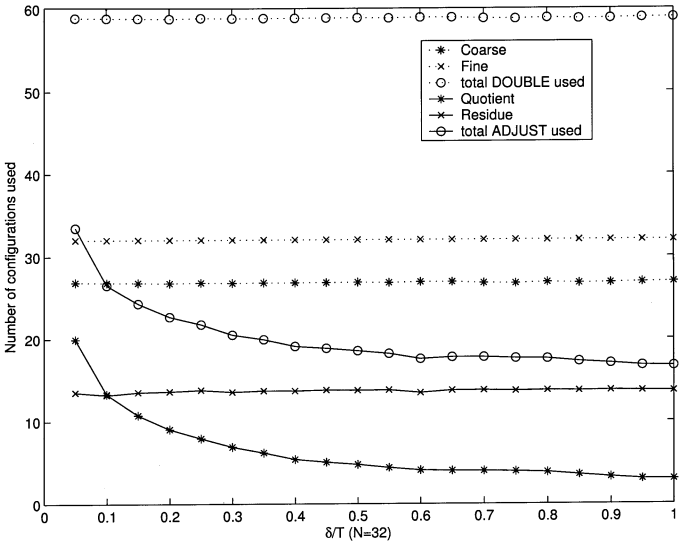

Fig. 8. Number of configurations DOUBLE and ADJUST used to cover matrices for 32 × 32 switch.

Fig. 8 shows the number of configurations DOUBLE and ADJUST algorithm will use for switch with port number 32. The three almost horizontal dotted lines stand for the number of configurations DOUBLE needs to cover *coarse* matrix, *fine* matrix, and the whole traffic matrix (from bottom to up, respectively). Even if δ changes from relatively small value $(0.05T)$ to large value (T), DOUBLE fixes its decision. In contrast, the

total number of configurations that ADJUST used decreases as δ get larger. This automatically defeats the inverse effects of increased reconfiguration overhead. Simulation shows ADJUST can keep this good property regardless the value of N. Because of its property of self-adjusting to different system parameters, ADJUST may save 20% covering cost over DOUBLE on average for small switch port number. The saving effect increases for larger switches, i.e., 50% for 32×32 switches and longer reconfiguration time.

Let us further consider the performance of ADJUST under the extreme overhead values as $\delta \to 0$ and $\delta \to \infty$. When $\delta \to 0$, $\lambda = \sqrt{T/\delta N} \to \infty$, $T/\lambda N \to 0$, which implies traffic matrix D is divided as $D = D + 0$. Now, ADJUST equals using only Step 2 (edge coloring) to find out a covering for the traffic matrix. It can achieve the lower bound of $\sum_{k=1}^{s} \lambda_k$ (which is the same result as *TSA-1*) but may use slightly more configurations. The result is not bad compare to the reduction of time complexity from $O(N^4)$ to $O(\lambda N^2 \log N)$. If $\delta \to \infty$, ADJUST will put traffic matrix solely into residue matrix. Step 3 determines its performance. Choices can be made from the three candidates based on complexity and performance requirements.

VI. Conclusion

Along with the fast development of Internet, optical switching technologies are becoming attractive for its huge capacity and scalability. However, the existence of reconfiguration delay makes the OSS problem to be NP-complete. A good heuristic algorithm should find out a balance between the number of configurations and empty time slots over all possible delay values. The ADJUST algorithm proposed in this paper successfully achieves this by covering the traffic matrix using quotient and residue matrices. The number of the configurations and their length dynamically suits the system parameters (δ, T, and N) to get the best result. Mathematical deduction and simulation show that ADJUST outperforms performance of previous algorithm over a large range of configuration overhead. In addition, the algorithm is stable and provides bounded delay guarantee with small time complexity.

Scheduling of the optical switches with reconfiguration delay raises many interesting questions for future studies. Our solution is based on a time slot assignment approach. It is also possible to use some existing scheduling algorithm of electronic switch [16]–[18] plus the idea of burst scheduling [4]. How does this different approach compare with TSA based algorithm? Is it stable under all traffic pattern? Also, if the cross points of optical switch is allowed to reset asynchronously, how should the algorithm changes to accommodate this more complicated environment? Further research is needed for all the above questions.

Appendix
Proof of the NP-Completeness of OSS Problem

A. Optimization Problem: OSS

Given a non-negative integer $N \times N$ traffic matrix D, positive integer constant δ, and T, find a set of configuration matrices $P^{(1)}, \ldots, P^{(s)}$ and corresponding coefficients $\phi^{(1)}, \ldots, \phi^{(s)}$, which satisfies

$$\sum_{i=1}^{N} d_{ij} \le T, \quad \sum_{j=1}^{N} d_{ij} \le T, \quad \forall i, j \in 1, 2, \ldots, N$$

$$\sum_{k=1}^{s} \phi_k p_{ij}^{(k)} \ge d_{ij}, \quad \forall i, j \in 1, \ldots, N$$

$$\sum_{k=1}^{s} \phi_k + s\delta \text{ is minimized.}$$

B. Corresponding Decision Problem: Decision Optical Switch Scheduling (Dec-OSS)

Given a non-negative integer $N \times N$ traffic matrix D, positive integer constant δ, T, and C. Does there exist a set of configuration matrices $P^{(1)}, \ldots, P^{(s)}$ and corresponding coefficients $\phi^{(1)}, \ldots, \phi^{(s)}$, which satisfies

$$\sum_{k=1}^{s} \phi_k p_{ij}^{(k)} \ge d_{ij}, \quad \forall i, j \in 1, \ldots, N$$

$$\sum_{k=1}^{s} \phi_k + s\delta \le C$$

.

C. Existing NP-Complete Problem: Restricted Timetable Design (R-TTD) Problem

The TTD problem is as follows:

Given:

1) a set $H = \{h_1, h_2, \ldots, h_l\}$ of l hours;
2) a set $T = \{T_1, T_2, \ldots, T_n\}$ of n teachers;
3) a set $C = \{C_1, C_2, \ldots, C_m\}$ of m classes;
4) a relation A: $T \to H$ which represents the availability of the teachers, teacher T_i being available in time h_k if and only if $h_k \in A(T_i)$;
5) a relation B: $C \to H$ which similarly represents the availability of the classes;
6) a requirement matrix R of dimension $N \times M$, where entry r_{ij} represents the requirement that teacher T_i must teach class C_j for r_{ij} hours.

Question: Does there exist an assignment function f: $T \times C \times H \to 0, 1$, such that we have the following:

1) $f(T_i, C_j, h_k) = 1$ implies $h_k \in A(T_i) \cap B(C_j)$, i.e., the teacher and class must be available.
2) $\sum_{T_i \in T}(T_i, C_j, h_k) \le 1$ for all $C_j \in C$, $h_i \in H$, i.e., no class is taught by more than one teacher in any one hour.
3) $\sum_{C_j \in C}(T_i, C_j, h_k) \le 1$ for all $T_i \in T$, $h_k \in H$, i.e., no teacher teaches in more than one class in any given hour.
4) $\sum_{h_k \in H}(T_i, C_j, h_k) = r_{ij}$ for all $T_i \in T$, $C_j \in C$, i.e., the requirements are satisfied.

Restricted timetable design has only 3 hours ($l = 3$), in which the classes are always available ($B(C_i) = H$) for all $C_i \in C$, and the requirement matrix R is the 0–1 matrix. It is known to be NP-complete [12].

Claim: The Dec-OSS Is NP-Complete:

Proof: The Dec-OSS problem is proven to be NP-complete by reducing from the R-TTD problem.

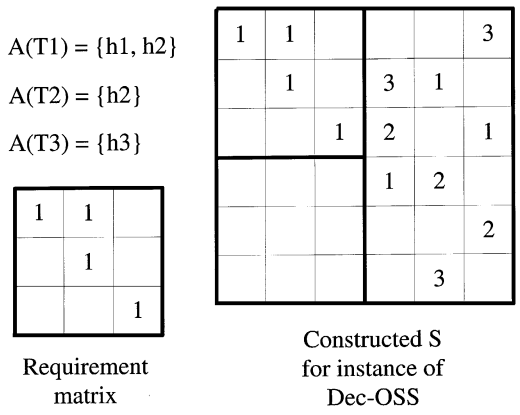

Fig. 9. Example of mapping from R-TTD to Dec-OSS.

Step 1: R-TTD $\propto$Dec-OSS: Given an instance of R-TTD, we create a $2N \times (M+N)$ matrix E for instance of Dec-OSS. The procedure is as follows:

1) Upper left-hand $N \times M$ submatrix of E: Simply copy exactly the 0–1 requirement matrix R from the R-TTD instance.
2) Lower left-hand $N \times M$ submatrix of E is filled entirely with zero entries.
3) The right-hand $2N \times N$ submatrix is constructed from the teacher availability function A in the following manner.
   - Initialize all entries as zero.
   - For each h_k, $k = 1, 2, 3$, and for each $T_i, i = 1, 2, \ldots, N$, set entry e_{pq} equals k. The values of p and q are given by

$$p = \begin{cases} i, & h_k \notin a(T_i) \\ i+N, & \text{otherwise} \end{cases}$$

$$q = \begin{cases} M+i+k-1, & i+k-1 \le N \\ M+i+k-N-1, & i+k-1 > N \end{cases}.$$

Fig. 9 shows an example of the above mapping from requirement matrix of R-TTD to an instance of Dec-OSS problem.

Now, define an instance of problem Dec-OSS: Whether E can be covered with the linear combination of a set of permutation matrices and the cost is no more than $6 + 3\delta$.

Below are s some observations of matrix E.

1) *The minimum number of permutation matrices needed to cover E is three*. Denote $r_i(c_j)$ as the number of nonzero entries in row i(column j). For the columns of E, clearly, since $|H| = 3$, R must have no more than three 1*s* in each column, and therefore, $c_j \le 3, j = 1, 2, \ldots, M$. Similarly, by the construction of E, $c_j = 3$ for $j = M+1, \ldots, M+N$. For the rows of E, the construction again ensures that $r_i \le 3$ for $i = N+1, \ldots, 2N$. For $i = 1, \ldots, N$, r_i represents the number of hours teacher T_i must teach, plus the number of hours he is not available, which again can be no greater than the total number of hours: three. Thus, for each row or column of E, there are at most three nonzero elements. By Theorem 2, for such matrix, the minimum achievable number of permutation matrices is three.
2) *The column summation in right-hand submatrix equals six*. That is, $\sum_{i=1}^{2N} e_{ij} = 6$, $M+1 \le j \le M+N$. It is obvious from the construction of E.

Step 2: Dec-OSS is Satisfiable if the R-TTD Problem Has a Solution:

Proof:

1) "*If*" *part*: Given a true instance of R-TTD, we can find a scheduling of the traffic matrix E in the following way to make Dec-OSS satisfied. First, consider the upper left-hand of E and pick out the entities corresponding to those classes taught in hour h_1. It is clear that there is at most one "h_1-entry" in each line according to requirement 2 and 3 of the definition of TTD. Then, we pick up those entities valued 1 in $2N \times N$ right-hand submatrix of E. From the traffic matrix construction procedure, no two such 1-entries occupy the same line. These "h_1-entries" will generate a switching matrix. Otherwise, the only possibility is that one "h_1-entry" and one 1-entry occupy the same row. This means the teacher teaching the corresponding class is unavailable in hour h_1 and, thus, is a contradiction. This switching matrix has duration of one. Similarly, picking up "h_2-entries" and the 2-entries forms a switching matrix with duration 2 and "h_3-entries," and the 3-entries forms a switching matrix with duration 3. These three switching matrices together form a cover for E, and the cost is $1 + 2 + 3 + 3\delta = 6 + 3\delta$. That implies Dec-OSS is satisfiable.
2) "*Only if*" *part*: From the first observation of E, at least three permutation matrices are needed to cover the matrix E. The second observation shows $\sum_{i=1}^{s} \phi_i \ge 6$. These imply, for any constructed Dec-OSS instance, $\sum_{i=1}^{s} \phi_i + s\delta \ge 6 + 3\delta$. So, if such Dec-OSS instance has a "yes" solution, we can conclude there exists a scheduling such that the number of permutations is three and that the corresponding $\sum_{i=1}^{s} \phi_i$ equals six. Just consider the right-hand $2N \times N$ submatrix of E. To make $\sum_{i=1}^{s} \phi_i$ equals six, the only possible way is to have three switching matrices, containing all N 3-entries, all N 2-entries, and all N 1-entries, respectively. Consider the switching matrix containing all N 3-entries. Clearly, there is at most one nonzero entry in each for the upper left-hand $N \times N$ submatrix. We can assign all corresponding classes in hour h_1. Classes in hours h_2 and h_3 can be similarly arranged without any confliction.

From the above discussion, we know OSS problem is NP-complete.

ACKNOWLEDGMENT

The authors would like to thank Prof. X.-R. Cao and W.-X. Shang for many helpful discussions.

REFERENCES

[1] A. Neukermans and R. Ramaswami, "MEMS technology for optical networking applications," *IEEE Commun. Mag.*, vol. 39, pp. 62–69, Jan. 2001.

[2] J. E. Fouquet, S. Venkatesh, M. Troll, D. Chen, H. F. Wong, and P. W. Barth, "A compact, scalable cross-connect switch using total internal reflection due to thermally-generated bubbles," in *Proc. Lasers and Electro-Optics Society Annual Meeting, 1998 (LEOS'98)*, Orlando, FL, Dec. 1998, pp. 169–170.
[3] O. B. Spahn, C. Sullivan, J. Burkhart, C. Tigges, and E. Garcia, "GaAs-based microelectromechanical waveguide switch," in *Proc. 2000 IEEE/LEOS Int. Conf. Optical MEMS*, Honolulu, HI, Aug. 2000, pp. 41–42.
[4] G. Nong and M. Hamdi, "Burst-based scheduling algorithms for nonblocking atm switches with multiple input queues," *IEEE Commun. Lett.*, vol. 4, pp. 202–204, June 2000.
[5] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun.*, vol. 27, pp. 1449–1455, Oct. 1979.
[6] G. Bongiovanni, D. Coppersmith, and C. K. Wong, "An optimum time slot assignment algorithm for an SS/TDMA system with variable number of transponders," *IEEE Trans. Commun.*, vol. 29, pp. 721–726, May 1981.
[7] E. M. Varvarigos, "The "packing" and the "scheduling packet" switch architectures for almost all-optical lossless networks," *J. Lightwave Technol.*, vol. 16, pp. 1757–1767, Oct. 1998.
[8] K. L. Yeung, "Efficient time slot assignment algorithms for TDM hierarchical and nonhierarchical switching systems," *IEEE Trans. Commun.*, vol. 49, pp. 351–359, Feb. 2001.
[9] I. S. Gopal and C. K. Wong, "Minimizing the number of switchings in an SS/TDMA system," *IEEE Trans. Commun.*, vol. 33, pp. 497–501, June 1985.
[10] M. Chen and T. S. Yum, "A conflict-free protocol for optical WDMA networks," in *Proc. IEEE Globe Communications Conf. (GLOBECOM'91)*, Phoenix, AZ, Dec. 1991, pp. 1276–1281.
[11] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," in *Proc. 21st Annual Joint Conf. IEEE Computer and Communications Societies (INFOCOM'02)*, New York, June 2002, pp. 342–351.
[12] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comput.*, vol. 5, pp. 691–703, Dec. 1976.
[13] V. V. Vazirani, *Approximation Algorithms*. Berlin, Germany: Springer-Verlag, 2001.
[14] R. Diestel, *Graph Theory*, 2nd ed. New York: Springer-Verlag, 2000.
[15] R. Cole and J. Hopcroft, "On edge coloring bipartite graphs," *SIAM J. Comput.*, vol. 11, pp. 540–546, Aug. 1982.
[16] C. S. Chang, W. J. Chen, and H. Y. Huang, "Birkhoff-von Neumann input-buffered crossbar switches for guaranteed-rate services," *IEEE Trans. Commun.*, vol. 49, pp. 1145–1147, July 2001.
[17] N. McKewon, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, pp. 20–28, Apr. 1999.
[18] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 47, pp. 1260–1267, Aug. 1999.

Xin Li (S'03) received the B.S. degree in computer science (with distinction) from Xian Jiao Tong University, Xian, China, in 2000, and is currently working toward the Ph.D. degree in computer science from Hong Kong University of Science and Technology, Clear Water Bay, Kowloon.

Her research focus on scheduling of high-performance scalable switches, switch architecture, and high-speed switching in optical networks.

Mounir Hamdi (S'89–M'90) received the B.S. degree in computer engineering (with distinction) from the University of Louisiana, Lafayette, in 1985 and the M.S. and Ph.D. degrees in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, in 1987 and 1991, respectively.

He has been a Faculty Member in the Department of Computer Science, Hong Kong University of Science and Technology, since 1991, where he is now Associate Professor of computer science and the Director of the Computer Engineering Program that has some 350 undergraduate students. From 1999 to 2000, he held Visiting Professor positions at Stanford University, Stanford, CA, and the Swiss Federal Institute of Technology, Zürich, Switzerland. His general areas of research are in high-speed packet switches/routers and all-optical networks, in which he has published more than 180 research publications, received numerous research grants, supervised some 20 postgraduate students, and for which he has served as Consultant to various international companies. Currently, he is working on high-speed networks including the design, analysis, scheduling, and management of high-speed switches/routers, wavelength division multiplexing (WDM) networks/switches, and wireless networks. He is currently leading a team that is designing one of the highest capacity chip sets for Terabit switches/routers. This chip set is targeted toward a 256×256 OC-192 switch and includes a crossbar fabric chip, a scheduler/arbiter chip, and traffic management chip.

Dr. Hamdi was on the Editorial Boards of the IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE C*ommunication Magazine*, *Computer Networks*, *Wireless Communications and Mobile Computing*, and *Parallel Computing* and has been on the program committees of more than 50 international conferences and workshops. He was a Guest Editor of IEEE *Communications Magazine*, the IEEE JOURNAL ON SELECTED AREAS OF COMMUNICATIONS, and *Optical Networks Magazine* and has Chaired more than five international conferences and workshops, including the IEEE GLOBECOM/ICC Optical Networking Workshop, the IEEE ICC High-Speed Access Workshop, and the IEEE IPPS HiNets Workshop. He is the Chair of IEEE Communications Society Technical Committee on transmissions, access, and optical systems and Vice Chair of the Optical Networking Technical Committee, as well as ComSoc Technical Activities Council. He received the Best Paper Award out of 152 papers at the International Conference on Information and Networking in 1998. In addition to his commitment to research and professional service, he is also a dedicated teacher. He received the Best 10 Lecturers Award (through university-wide students voting for all university faculty held once a year) and the Distinguished Teaching Award from the Hong Kong University of Science and Technology. He is a Member of ACM.