Two-Layer Parallel Switching: A Practical and Survivable Design for Performance Guaranteed Optical Packet Switches

Bin Wu, Kwan L. Yeung and Victor O. K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong
E-mail: {binwu, kyeung, vli}@eee.hku.hk

***Abstract*—An optical packet switch (OPS) is called performance guaranteed if it can achieve 100% throughput with bounded packet delay. Presently, high speedup requirement and large packet delay are two main disadvantages in designing performance guaranteed OPS. Survivability is another important issue that must be considered for real OPS implementations. In this paper, we propose a two-layer parallel OPS architecture together with an efficient scheduling scheme to address all the above issues. The tradeoff between speedup and packet delay under this new parallel architecture is also formulated to provide more design flexibility. Compared to the single-layer OPS, our proposed solution can simultaneously reduce both speedup and packet delay. For example, a delay of $4\delta N$ slots can be achieved with a speedup of 2 in our solution (where N is the switch size and δ is the switch reconfiguration overhead), whereas the single-layer OPS needs a speedup of 6 for a delay of $7\delta N$ slots. We show that this significant improvement benefits from a careful overall design rather than simply adding an extra switching layer.**

Keywords-Optical packet switch (OPS); parallel switching; performance guaranteed scheduling; reconfiguration overhead.

I. Introduction

Optical packet switch (OPS) is receiving significant attention in recent years. This is mainly due to two reasons. First, the explosion of Internet traffic calls for high line rate, huge capacity and scalable switches, and OPS meets these requirements very well. Second, the recent progress of optical transmission and switching technologies provides a solid foundation for OPS implementation [1-4].

A typical electronic-buffered OPS switch architecture is shown in Fig. 1. Assume time on each input/output line is slotted such that each time slot can accommodate a single packet. When the incoming packets arrive, they are first buffered in the VOQs (virtual output queues [5]) at each input port. For every pre-defined T time slots, the buffered packets form an $N\times N$ traffic matrix $\boldsymbol{C(T)}$ where N is the switch size. Each entry c_{ij} of $\boldsymbol{C(T)}$ denotes the number of packets received

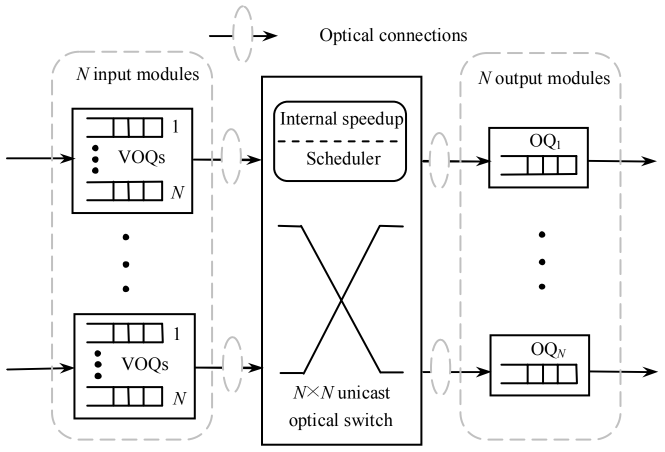

Fig. 1. A scalable high speed optical packet switch.

at input i and destined to output j. $\boldsymbol{C(T)}$ is called *admissible* if each of its lines (rows or columns) sum to at most T. In this paper, we only consider admissible traffic patterns. Packets in $\boldsymbol{C(T)}$ are then scheduled for transmitting across the switch. The associated scheduling algorithm follows the batch-based TSA (time slot assignment) approach. A scheduling algorithm is called *performance guaranteed* if it can achieve 100% throughput with bounded packet delay for any admissible traffic matrix.

Unlike electronic switches, most OPS need relatively long time to change their cross-connection states, during which no packets can be sent across the switch. This *reconfiguration overhead* time can range from nanoseconds to milliseconds depending on the particular optical switching technology adopted [1-4]. To accommodate the reconfiguration overhead, the switch fabric must operate at a speed higher than the individual input/output line rates in order to be non-blocking. This *speedup* (the ratio of the internal transmission rate to the external line rate) is to compensate for both the idle time introduced by the reconfiguration overhead and the possible scheduling inefficiency introduced by the scheduling algorithm [6-9] (also refer to Section II). In general, a higher speedup produces a lower packet delay, at the expense of higher bandwidth cost.

This work is supported in part by Hong Kong Research Grants Council Earmarked Grant HKU 7032/01E, and in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99).

Recently, several ingenious algorithms are proposed to schedule OPS traffic with guaranteed performance [6-9]. MIN [6] and α^i-SCALE [9] can minimize the packet delay (by using only the minimum number of N configurations to schedule the traffic), but they require a very high speedup. DOUBLE [6] and ADAPTIVE [8] reduce the speedup requirement by slightly sacrificing the delay performance (i.e. requiring more configurations for scheduling). The tradeoff relationship between speedup and delay is mathematically formulated in [8]. It gives additional insights in designing new scheduling schemes.

All the scheduling algorithms mentioned above are based on the switch architecture shown in Fig. 1, which consists of a single switching layer. Since the OPS switch fabric is responsible for huge volume of data transmission, survivability is of paramount importance. To avoid the possible failure of the single layer switching, we construct a two-layer switch architecture by adding another switching layer as shown in Fig. 3. Against the conventional wisdom of dedicating one layer as backup, we propose a new scheduling scheme to utilize the additional switching layer for packet transmission too.

In particular, the two layers work alternatively in switching and reconfiguration phases. In contrast to its single-layer counterpart, the two-layer OPS switch can simultaneously reduce speedup and packet delay (refer to the example in Part B of Section III). We show that all the aforementioned advantages are not simply due to adding an extra layer of switching hardware, but rather a careful overall design on the scheduling scheme.

In case of single-layer failure, the proposed two-layer switch architecture degenerates into a single-layer switch. So it can still achieve 100% throughput but with an increased bounded delay.

II. Single-Layer OPS Switch Architecture

The scheduling procedure in the single-layer OPS can be divided into four stages [6-9], as shown in Fig. 2. In Stage 1, incoming packets are accumulated in the input VOQ buffers over T time slots to construct the traffic matrix $\boldsymbol{C(T)}$. Assume that $\boldsymbol{C(T)}$ is admissible. The scheduling algorithm (such as ADAPTIVE [8]) takes H time slots in Stage 2 to generate N_S configurations $\boldsymbol{P}_1$, …, $\boldsymbol{P}_{N_S}$ to cover [1] $\boldsymbol{C(T)}$. Configuration $\boldsymbol{P}_n=\{p^{(n)}_{ij}\}$ is an $N\times N$ matrix with at most a single "1" in each line. $p^{(n)}_{ij}=1$ indicates that a packet can be sent from input i to output j in one slot; $p^{(n)}_{ij}=0$ otherwise. In Stage 3, the switch fabric is reconfigured according to these N_S configurations. An internal speedup S_1 is applied to ensure that this stage occupies only T (*regular*) slots. After the speedup is applied, the switch holds each $\boldsymbol{P}_n$ for ϕ_n *compressed* slots (each of which has a shorter duration than a regular slot) for packet transmission. Throughout this paper, "compressed slots" refers to the slots in the transmission phase in Stage 3, whereas "slots" refers to regular slots. Finally in Stage 4 packets are sent onto the output lines from the output buffers (in T slots).

[1] $\boldsymbol{C(T)}$ is covered by a set of configurations $\boldsymbol{P}_1$, …, $\boldsymbol{P}_{N_S}$, each weighted by a non-negative integer ϕ_1, …, ϕ_{N_S}, if and only if $\sum^{N_S}_{n=1}\phi_n\, p^{(n)}_{ij}\geq c_{ij}$ for any $i,j\in\{1,\ldots,N\}$. Note that $\boldsymbol{P}_n=\{p^{(n)}_{ij}\}$.

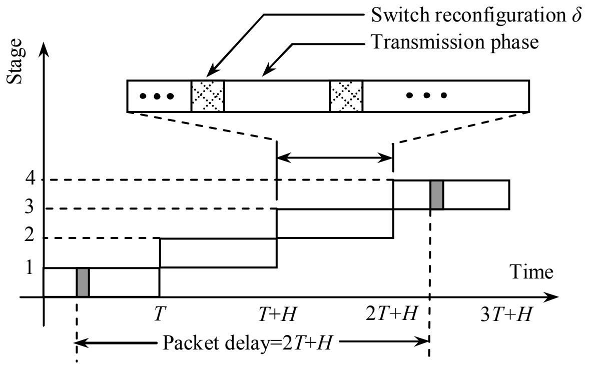

Fig. 2. Optical packet switch scheduling stages.

From the tagged packet shown in Fig. 2, we can see that the bounded delay of any packet is $2T+H$ slots. Assume each switch reconfiguration consumes δ slots. Since δN_S slots must be used to reconfigure the switch for N_S times, $T>\delta N_S$ must be satisfied. Besides, since only $T-\delta N_S$ slots are left for transmitting $\boldsymbol{C(T)}$ in Stage 3, a speedup factor denoted by $S_{\text{reconfigure}}=T/(T-\delta N_S)$ is necessary to compensate solely for the idle time caused by the reconfiguration. At the same time, the scheduling algorithm may underutilize the bandwidth provided by the configurations, i.e. producing many empty slots [6, 8]. As a result, another speedup factor, $S_{\text{schedule}}=(1/T)\sum^{N_S}_{n=1}\phi_n$, is required to compensate for the inefficient scheduling. The overall internal speedup S_1 is then given by

$$S_1 = S_{\text{reconfigure}}\times S_{\text{schedule}} = \frac{T}{T-\delta N_S}S_{\text{schedule}}\,. \qquad (1)$$

Because the packet delay is bounded by $2T+H$, a large batch size T results in a large packet delay. Note that $T>\delta N_S$. Reducing N_S can reduce this delay (since T can be smaller). But it usually leads to a large S_{schedule} [6, 8] (also refer to Appendix A). Besides, the overall speedup S_1 in (1) involves another factor $S_{\text{reconfigure}}$, which can make the value of S_1 even larger.

III. Two-Layer Parallel Switch Architecture

The two-layer parallel OPS switch architecture shown in Fig. 3 can be used to reduce speedup. Intuitively, following the scheduling procedure in Fig. 2, the original $\boldsymbol{C(T)}$ can be equally divided into two sub-matrices (each with entries $c_{ij}'=c_{ij}/2$) before scheduling is carried out. Then each sub-matrix is fed into a dedicated switching layer for *simultaneous* packet transmission. Compared with the single-layer architecture, the required speedup *might* be halved because the packets to be transmitted in Stage 3 are halved in each switching layer. However, since the batch-size T is *not* changed and $S_{\text{reconfigure}}$ in (1) still needs to be considered, packet delay (i.e. $2T+H$) cannot be reduced and the resulting overall speedup is still high.

We propose to use the two switching layers in Fig. 3 *alternatively* for packet transmission instead of *simultaneously*. That is, in Stage 3 of Fig. 2, one layer is transmitting packets (in *switching phase*) while the other layer is being reconfigured (in *reconfiguration phase*). The amount of time spent in each phase is the same and equal to δ slots, which is the time overhead for one reconfiguration. Fig. 4 shows the timing for

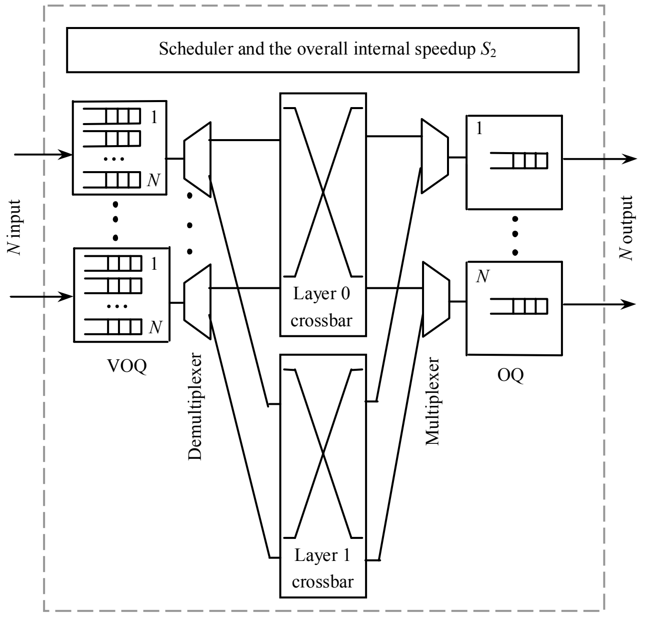

Fig. 3. Two-layer parallel OPS switch architecture.

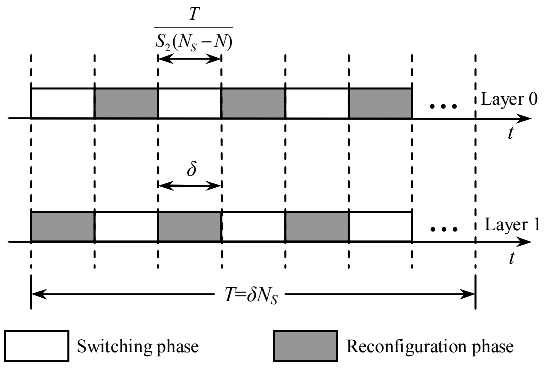

Fig. 4. Two-layer parallel OPS switch architecture timing procedure (with an overall speedup of S_2).

the two layers operating in parallel. Let N_S denote the total number of configurations required in this two-layer OPS architecture. For simplicity, assume N_S is an even number such that it can be evenly divided between the two switching layers. From Fig. 4, we can see that the superimposed schedule of the two layers contains no idle periods for switch reconfiguration (compared to Fig. 2). In other words, the proposed two-layer OPS mimics the performance of a single-layer OPS without reconfiguration overhead, i.e. δ=0. Therefore, the overall internal speedup should be solely determined by S_{schedule}.

A. Overall Internal Speedup

Based on the work in [8], we show in Appendix A that any admissible traffic matrix $\boldsymbol{C(T)}$ can be covered by N_S configurations, each weighted by $\phi_n=T/(N_S-N)$. The total number of *compressed* slots required for transmitting $\boldsymbol{C(T)}$ in Stage 3 of Fig. 2 is then equal to $\sum_{n=1}^{N_S}\phi_n=N_ST/(N_S-N)$.

In our proposed parallel architecture, we first decompose $\boldsymbol{C(T)}$ into N_S configurations in the same way as mentioned above. Define the traffic accumulation time T (or batch size) as follows:

$$T=\delta N_S \ \text{ or } \ \delta=\frac{T}{N_S}. \tag{2}$$

Compared to the single-layer architecture, the constraint of $T>\delta N_S$ is removed and the batch size T is reduced to δN_S. By allocating the N_S configurations alternatively to the two switching layers, each layer is responsible for $N_S/2$ configurations. Note that the internal speedup can shorten the packet transmission time but not the time required for reconfiguration. From Fig. 4, each switching phase lasts for δ slots, which must be able to accommodate $\phi_n=T/(N_S-N)$ compressed slots. Therefore according to (2), the overall internal speedup S_2 for the two-layer architecture is given by

$$S_2=\frac{T/(N_S-N)}{\delta}=\frac{\delta N_S/(N_S-N)}{\delta}=1+\frac{N}{N_S-N}. \tag{3}$$

According to (8) in Appendix A, S_2 in (3) is indeed the same as S_{schedule}. Comparing with (1), it is clear that the factor $S_{\text{reconfigure}}=T/(T-\delta N_S)$ is removed from the overall internal speedup expression. This results in smaller speedup and shorter delay than the single-layer architecture.

B. Packet Delay Bound

From Fig. 2, packet delay is bounded by $2T+H$ slots. Assume that the algorithm's execution time H is the same for both single-layer and two-layer architectures. Then the delay is determined only by T. We can further reduce the packet delay bound by slightly modifying the scheduling procedure in Fig. 2 to allow pipelined algorithm execution (Stage 2) and packet transmission (Stage 3), as discussed below.

Among the N_S configurations constructed according to Appendix A, the N configurations for scheduling the residue matrix $\boldsymbol{B}$ can be *any* N configurations as long as they can cover every entry of an $N\times N$ matrix. Therefore, these N configurations can be *pre-defined offline*. It implies that Stage 2 (algorithm execution) and Stage 3 (packet transmission) in Fig. 2 can be *partially overlapped* (or pipelined) in time. As a result, even if the scheduling algorithm is still running (for calculating the rest N_S-N configurations), packet transmission based on the N pre-defined configurations can take place in parallel, as shown in Fig. 5.

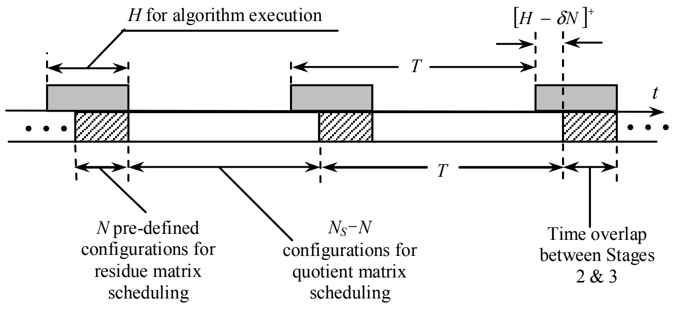
H for algorithm execution
$[H-\delta N]^+$
T
t
T
N pre-defined configurations for residue matrix scheduling
N_S-N configurations for quotient matrix scheduling
Time overlap between Stages 2 & 3

Fig. 5. Time overlap between Stages 2 & 3 for $H>\delta N$.

The same pipelining mechanism can also be applied to the two-layer OPS. We adopt the similar scheduling procedure as that in Fig. 2, but the two switching layers are synchronized in Stage 3 (packet transmission) as shown in Fig. 4. Let D and T_O denote the total packet delay and the overlapped time between Stages 2 and 3, respectively. We have

$$\begin{aligned} D &= T(\textit{traffic accumulation}) + H(\textit{algorithm execution}) \\ &\quad + T(\textit{traffic sending}) - T_O(\textit{overlap between Stages 2 \& 3}) \\ &= 2T + [H-\delta N]^+ = 2\delta N_S + [H-\delta N]^+ \qquad (4) \end{aligned}$$

where $[H-\delta N]^+=\max\{0, H-\delta N\}$. Equation (4) indicates that if the scheduler runs fast enough such that $H \le \delta N$, then the packet delay can be as small as $D=2T=2\delta N_S$ time slots. To consider the worst case, we assume $H>\delta N$. Therefore, (4) can be simplified as

$$D = 2\delta N_S + H - \delta N\,. \qquad (5)$$

Rearranging (3), we get

$$N_S = \left(\frac{S_2}{S_2-1}\right)N\,. \qquad (6)$$

Combining (5) and (6), we have

$$D = 2\delta N\left(\frac{S_2}{S_2-1}\right) + H - \delta N = \left(\frac{S_2+1}{S_2-1}\right)\delta N + H\,. \qquad (7)$$

Equations (3)-(7) govern the interplay among parameters N_S, S_2 and D in the proposed two-layer architecture. For example, let $N_S=2N$ and $H=\delta N$. From (3) the overall internal speedup is $S_2=2$, and from (7) the packet delay is $D=3\delta N+H=4\delta N$. In comparison, for the original single-layer architecture, if $N_S=2N$ and ADAPTIVE [8] or DOUBLE [6] algorithm is used, a delay of $7\delta N$ can be achieved with an overall speedup of $S_1=6$. Both speedup and delay are much lower in our proposed parallel scheduling scheme.

Particularly, equation (7) formulates the tradeoff between speedup and packet delay for the two-layer OPS. It provides additional design flexibility for OPS switches.

C. *Buffer Size*

Refer to the two-layer OPS shown in Fig. 3. Let L be the number of bits arrived at a single input port during a slot time. Since the packet delay is $2\delta N_S+[H-\delta N]^+$ slots, $(2\delta N_S+[H-\delta N]^+)L$ bits of data buffers are required for each VOQ. As a result, $(2\delta N_S+[H-\delta N]^+)LN^2$ bits are required for all the VOQ buffers at the input ports.

At the output ports, the OQ buffers (output queues [5]) need to store packets arrived over $2\delta N_S$ slots. Thus $2\delta N_S L$ bits are enough for each output port. In total, $2\delta N_S LN$ bits are required for all the OQ buffers.

D. *Flow Order*

We define the flow order as the packet sequence in the same flow, where a flow refers to the packets coming from the same input port i and destined to the same output port j. Our goal is to keep the flow order unchanged before and after switching. However, two different flows may interleave with each other.

As shown in Fig. 4, the two switching layers work in switching and reconfiguration phases alternatively, and at any time instant, there is only one layer in switching phase. In other words, the two layers transmit packets in a Round Robin manner. Assume that the demultiplexers/multiplexers in Fig. 3 cooperate with the crossbars in the same Round Robin manner (i.e. they always connect the corresponding buffers to the layer working in the switching phase). Obviously, the flow order will not be changed. This eliminates the need of reordering packets at the output ports.

E. *Discussion*

More than two parallel switching layers may be considered to further reduce speedup and delay for OPS switches. Here, we would like to highlight some possible side effects: 1) the hardware cost and the size of the OPS fabric may be greatly increased, especially when N is large; 2) more than one switching layer will transmit packets simultaneously, causing packet out-of-order problem, which will further complicate the buffer design; 3) the speedup may not be reduced as cost-effectively as changing from single-layer to two-layer.

To illustrate the last point, assume that electronic buffers are used at VOQs/OQs of an OPS switch. Then speedup affects the reading/writing frequency on the buffers. Our proposed two-layer scheduling scheme can intelligently utilize the idle time originally occupied by switch reconfigurations. If more than two switching layers are used, packet delay can be reduced. But we may not be able to further cut down the operation frequency on the buffers (i.e. speedup), because our proposed scheme has already fully utilized all the idle time for switch reconfigurations.

IV. Survivable OPS Switch Design

Based on the two-layer parallel architecture shown in Fig. 3, we can design a survivable OPS switch fabric. Here, survivability means that performance guaranteed scheduling can be sustained in case of single-layer failure, possibly with a *longer* bounded packet delay. In fact, if one of the two switching layers fails, the switch fabric becomes a single-layer switch as that in Fig. 1. In this case, we can still utilize the time overlap (see Fig. 5) between Stage 2 (algorithm execution) and Stage 3 (packet transmission) to achieve a shorter delay time than that in [8].

A closer look on single-layer failure reveals that instead of using $T=\delta N_S$ defined in (2), the batch size T has to be increased ($T>\delta N_S$), and $S_{\text{reconfigure}}$ cannot be removed from equation (1). This potentially increases both speedup and delay. In addition, larger buffer size is required due to the longer packet delay.

To design a survivable OPS switch, we need to address the above issues. Generally, the increase in buffer size can be easily solved by adding extra backup memory, and this does not increase the hardware cost much. But the increase in speedup may significantly boost the bandwidth cost. So, in case of single-layer failure, we propose to delay the traffic for a longer time instead of increasing the speedup. This can be achieved using the following modified ADAPTIVE algorithm. Please refer to Appendix B or [8] for the original ADAPTIVE.

Assume that a two-layer OPS switch originally runs at a speedup of S_2, and we wish to run it at the same speedup (S_1=S_2) when a single-layer failure is detected. Further assume that the values of S_2, δ and N are given. We modify the objective of ADAPTIVE to find a suitable schedule satisfying S_1=S_2. For simplicity, we only provide the steps for modifying ADAPTIVE. Refer to Appendix B. We first substitute (10) into (11) and solve equation (11) to get a suitable value of T. This batch size T also determines the required buffer size and the packet delay. Then we substitute T into (10) to calculate N_S. After N_S and T are determined, we can construct the N_S configurations according to the steps given in Appendix A.

V. CONCLUSION

We proposed a new two-layer parallel switch architecture and the corresponding scheduling scheme to reduce speedup and packet delay in OPS switch fabrics. The two switching layers in the parallel architecture work in switching phase and reconfiguration phase alternatively, so that the switch fabric can make full use of the time for packet transmission. By carefully devising the scheduling scheme, both speedup and packet delay are greatly reduced as compared with the single-layer counterpart. The tradeoff relationship between speedup and delay was also mathematically formulated. This provides more flexibility for practical designs. In addition, we showed that the proposed scheme is survivable. It can sustain performance guaranteed scheduling in case of single-layer failure, albeit with a longer bounded packet delay.

APPENDIX A DECOMPOSING THE TRAFFIC MATRIX

$S_{\text{schedule}}=(1/T)\sum_{n=1}^{N_S}\phi_n$ discussed in Section II is used to compensate for the algorithm's scheduling inefficiency. It is determined only by the algorithm and is irrelevant to the specific switch architecture adopted. In this appendix, we discuss how to decompose $\boldsymbol{C(T)}$ into N_S configurations, and the relationship between S_{schedule} and N_S is also formulated.

For a given admissible $N\times N$ $\boldsymbol{C(T)}$, we use $T/(N_S-N)$ to divide it and separate it into a quotient matrix $\boldsymbol{A}$ and a residue matrix $\boldsymbol{B}$: $\boldsymbol{C(T)}=[T/(N_S-N)]\times\boldsymbol{A}+\boldsymbol{B}$. Since each line of $\boldsymbol{C(T)}$ sum to at most T, the maximum line sum of $\boldsymbol{A}$ is at most N_S-N. As a result, $\boldsymbol{A}$ can be covered by N_S-N configurations, which can be obtained by performing edge-coloring [10] on the bipartite multigraph of $\boldsymbol{A}$ [7, 8]. At the same time, all the entries in $\boldsymbol{B}$ are not larger than $T/(N_S-N)$. So, $\boldsymbol{B}$ can be covered by *any* N non-overlapping configurations (i.e. any two of them do not cover the same entry), with each weighted by $T/(N_S-N)$. In total, $\boldsymbol{C(T)}$ can be covered by $(N_S-N)+N=N_S$ configurations, each weighted by $\phi_n=T/(N_S-N)$. Consequently, S_{schedule} can be formulated as

$$S_{\text{schedule}}=\frac{1}{T}\sum_{n=1}^{N_S}\phi_n=\frac{1}{T}\times\frac{T}{N_S-N}\times N_S=1+\frac{N}{N_S-N}. \qquad (8)$$

APPENDIX B ADAPTIVE ALGORITHM [8]

ADAPTIVE algorithm [8] is designed for the single-layer OPS given in Fig. 1, with the scheduling stages shown in Fig. 2. Given the values of T, N and δ, ADAPTIVE determines the best N_S value to minimize the required overall speedup S_1. In particular, ADAPTIVE also uses the steps in Appendix A to decompose $\boldsymbol{C(T)}$ and construct the N_S configurations.

Substituting S_{schedule} in (8) into (1), we have

$$S_1=\frac{T}{T-\delta N_S}S_{\text{schedule}}=\frac{TN_S}{(T-\delta N_S)(N_S-N)}. \qquad (9)$$

Solving the equation $\frac{\partial S_1}{\partial N_S}=0$ for N_S, we get

$$N_S=\sqrt{\frac{TN}{\delta}}=\lambda N\text{, where }\lambda=\sqrt{\frac{T}{\delta N}}. \qquad (10)$$

As a result, the minimized overall speedup S_1 for the single-layer OPS (using ADAPTIVE scheduler) is given by

$$S_1=\left.\frac{TN_S}{(T-\delta N_S)(N_S-N)}\right|_{N_S=\lambda N}=\frac{\lambda T}{(T-\delta\lambda N)(\lambda-1)}. \qquad (11)$$

REFERENCES

[1] J.E Fouquet et. al, "A compact, scalable cross-connect switch using total internal reflection due to thermally-generated bubbles," *IEEE LEOS Annual Meeting*, pp. 169-170, Dec. 1998.

[2] L. Y. Lin, "Micromachined free-space matrix switches with submilli-second switching time for large-scale optical crossconnect," *OFC'98 Tech. Digest*, pp. 147-148, Feb. 1998.

[3] O. B. Spahn, C. Sullivan, J. Burkhart, C. Tigges, and E. Garcia "GaAs-based microelectromechanical waveguide switch," *Proc. 2000 IEEE/LEOS Intl. Conf. on Optical MEMS*, pp. 41-42, Aug. 2000.

[4] A. J. Agranat, "Electroholographic wavelength selective crossconnect," *1999 Digest of the LEOS Summer Topical Meetings*, pp. 61-62, Jul. 1999.

[5] Shang-Tse Chuang, A. Goel, N. McKeown and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, issue 6, pp. 1030-1039, Jun. 1999.

[6] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 835-847, Oct. 2003.

[7] X. Li and M. Hamdi, "On scheduling optical packet switches with reconfiguration delay," *IEEE Journal on Selected Areas in Communications*, vol. 21, issue 7, pp. 1156-1164, Sept. 2003.

[8] B. Wu and K. L. Yeung, "Minimizing internal speedup for performance guaranteed optical packet switches," *IEEE GLOBECOM '04*, vol. 3, pp. 1742-1746, Dec. 2004.

[9] B. Wu and K. L. Yeung, "Scheduling optical packet switches with minimum number of configurations," *IEEE ICC '05*, vol. 3, pp. 1830-1835, May 2005.

[10] R. Cole and J. Hopcroft, "On edge coloring bipartite graphs," *SIAM Journal on Computing*, vol. 11, pp. 540-546, Aug. 1982.