# Design and Analysis of Scheduling Algorithms for Switches with Reconfiguration Overhead

Xin Li and Mounir Hamdi
Department of Computer Science
Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, HongKong
Email: hamdi@cs.ust.hk

***Abstract*— Hybrid architectures with electronic buffering /processing and optical switching fabric are receiving a lot of attention as potential candidates for the design of scalable high-performance switches/routers. However, the reconfiguration overhead of optical fabrics introduces new challenges for the traffic scheduling across these switches. Existing algorithms proposed for this architecture can be classified as either *batch-scheduling* or *single-scheduling*. This paper is the first which analyzes and compares the performance of these two scheduling classes. The comparison is based on speedup requirement, average and worst-case delay and algorithm efficiency. Moreover, analytical models for the stability and average delay of single-scheduling algorithms are introduced in this paper. We will demonstrate that these results are helpful in determining the right type of algorithm and choosing the appropriate algorithm parameters for a given switch system.**

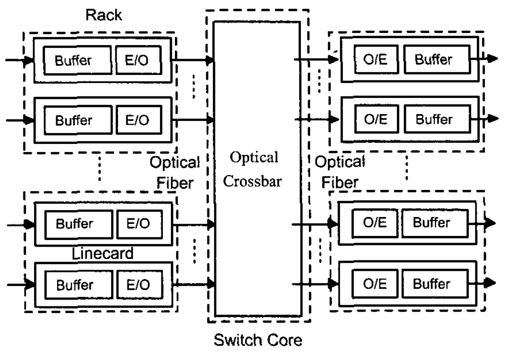

Fig. 1. Schematic view of a multi-rack hybrid packet switch.

## I. Introduction

Recent advances in communication link technology, such as Dense Wavelength Division Multiplexing (DWDM), have dramatically increased the transmission capacity of optical fibers. As a result, switches/routers are replacing the transmission link as the bottleneck of network. This has lead to a high demand for high-speed switches/routers with large I/O ports than ever before. For example, switches of size 256 to 1024, running at OC192 (10Gb/s) or even higher speed are becoming (will soon become) a necessity for most IP core networks.

Most high-performance switches/routers today use an all-electronic solution, with Virtual Output Queuing (VOQ) and a crossbar-based switching fabric. Current state-of-the-art electrical switching fabrics support only up to 2.5Gb/s [1]. A recent break-through demonstrated that a $256\times256$ 2Gb/sec per port non-cascading crossbar chip can be practically designed [2]. The power consumption of this crossbar chip and the scheduler chip is around 1kW. However if we want to further upgrade the design in size (e.g., $1024\times1024$) and/or data rate (e.g., OC768, 40Gb/s), and at the same time incorporate redundancy for fault tolerance, then the power consumption and number of chips needed will be an order of magnitude higher. Given the fact that the default power consumption per piece of equipment (e.g., rack) is typically 6kW in addition to compliance with NEBS physical-packaging requirements (i.e., rack size), then this necessitates the distributed design of the whole switch/router over several racks where one rack would contain the switch fabric and the scheduler, and the other racks would contain the linecards.

This work was supported in part by a grant from Hong Kong Research Grant Council (Grant Number: RGC HKUST6202-99E).

It is possible to transport electrical signals between different racks, using coax, twisted-pair, etc. But consider there maybe potentially thousands of signals travelling between different racks, and this quickly leads to a cabling nightmare. One technology increasingly used to solve the problem is optical fiber interconnections [1]. When such a switch architecture is deployed, packets that arrive from the external interfaces are processed and buffered in the electronic domain. Data are then transmitted over a fiber link to the switch fabric rack, where they are processed again in electronics. Once they are switched, a different fiber connection is used to transmit the packets to the egress linecards, where they are again processed and buffered in electronics, before finally being transmitted in optics. Hence series of opto-electronic conversions are needed, that translate into significant cost and power requirements. In other words, if the whole switch/router design cannot fit within a single rack, then it becomes extremely difficult to come up with a cost-effective solution using an electronic switch fabric.

For exactly these reasons, a hybrid switch architecture with electronic buffers and optical switching fabric, as shown in Fig. 1, is proposed for designing high-capacity scalable switches. Moreover, optical fabrics have many advantages over their electrical counterparts in terms of higher capacity, lower power consumption and less cost.

However, a large-scale hybrid packet switch/router presents unique challenges: the reconfiguration time of an optical switch fabric is *much* longer than that of an electronic fabric. With typical cell sizes on the order of 50ns (64bytes at 10Gb/s), the fabric takes from 0.2 to 20,000 cell times to reconfigure [3]. Traffic may still arrive during the reconfiguration periods, but are blocked before being switched. Hence the fabric should run at higher switching speed than the line rate during the transfer periods. In other words, the switch should have fabric speedup to compensate the blockage at the reconfiguration period.

It is obvious that traditional slot-by-slot scheduling algorithms may severely cripple the switch performance by having too much reconfiguration overhead. Instead, schedules should be held for some time to avoid frequent fabric reconfigurations. There exists a tradeoff between outdated schedule and reconfiguration overhead. As a result, how to determine the schedule and its holding time efficiently is extremely important, and is our key motivation in this paper. Recent research scheduling algorithms for optical fabrics can be roughly divided into two categories, namely *batch-scheduling* and *single-scheduling*, based on the number of schedules they make at each run.

The reminder of this paper introduces the two classes of algorithms, emphasizing their performance and comparing them to each other. In addition, we give analytical models for the stability and average delay of single-scheduling class of algorithms. Section II is about the performance and analysis of these scheduling algorithms. Analytical models are presented in Section III. Finally, conclusions are drawn in Section IV.

## II. Algorithms for Switches with Reconfiguration Delay

We assume the algorithm runs on a VOQ non-blocking $N \times N$ switch with reconfiguration delay $z_a$. The switch operates on fixed-size data units (say, cell) in a slotted manner and hence has integer speedup $S$. There is at most 1 arrival for each input at each time slot. Denote $\lambda_{ij}$ to be the arrival rate of traffic from input $i$ to output $j$. Only *admissible* traffic is considered in this paper. That is, $\forall j \in 1 \cdots N, \sum_{i=1}^{N} \lambda_{ij} < 1$ and $\forall i \in 1 \cdots N, \sum_{j=1}^{N} \lambda_{ij} < 1$.

### A. Batch Scheduling: a TSA-Based scheudling

The idea of batch-scheduling comes from Time Slot Assignment (TSA) algorithms for Satellite Switching Time Division Multiple Access (SS/TDMA) system, in which reconfiguration delay also exists. The algorithm runs in accumulate-schedule-transfer cycles as shown in Fig.2. Incoming traffic are accumulated periodically and the accumulation time is a predefined integer T. A particular cycle first accumulates traffic in a $N \times N$ traffic matrix D. Batch scheduling algorithm then decomposes D into a set of permutation matrices $P_i$, each has weight $l_i$. These matrices correspond to a batch of schedules and the lengths these schedules hold. Assume the number of permutation matrices is $K$, the decomposition should satisfy: 1) D is covered, that is, $\sum_{i=1}^{K} l_i P_i \geq D$; 2) The total transmission time of D (including reconfiguration delay), which is $\sum_{i=1}^{K} l_i + K z_a$, is minimized. Traffic in D are sent out according to the schedules and the transmission period should finishes in T time slots. Normally speedup is needed to achieve this. Traffic accumulation of the next cycle starts simultaneously with the transmission period.

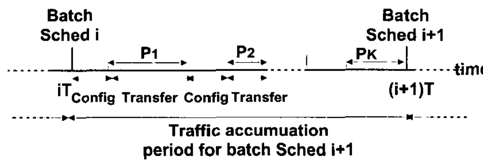

Fig. 2. Execution timeline of batch scheduling algorithms.

*1) A Representative Batch-Scheduling Algorithm:* Find out the schedules that minimize the transmission time is a NP-complete problem. Most previous TSA algorithms, as in [4][5], normally assume the reconfiguration delay to be either zero or infinity for simplification. A recent enhancement, DOUBLE algorithm in [3], breaks this limitation and claims better performance over large range of reconfiguration delay values. At each batch scheduling point, DOUBLE executes the following procedure:

- Divide the traffic matrix $D$ into *coarse* matrix $A$ and *fine* matrix $B$. Assume $d_{ij}$, $a_{ij}$ and $b_{ij}$ $(1 \leq i, j \leq N)$ are the elements in the corresponding matrix. Matrix $A$ and $B$ are calculated by,

$$a_{ij} = \lfloor \frac{d_{ij}}{T/N} \rfloor \quad b_{ij} = max(0, d_{ij} - \lceil T/N \rceil a_{ij})$$

- Under admissible traffic, [3] proves that
  - For a coarse matrix $A$, the column sum and row sum are less than $N$. $A$ can be edge-colored using $N$ colors.
  - For a fine matrix $B$, all elements are less than $\lceil T/N \rceil$

  Using the above properties, DOUBLE generates $K = 2N$ schedules, each with holding length $\lceil T/N \rceil$. $N$ of them are produced by edge-coloring coarse matrix $A$; the other $N$ schedules are just $N$ non-overlapping permutation matrices. [3] shows that the traffic matrix D will always be covered by such schedule method.

*2) Speedup and Accumulation Time Requirement:* The $2N$ schedules has a total transmission time $2N \times \lceil T/N \rceil$. As indicated in Corollary 3 in [3], a speedup of $\frac{2T}{T - 2z_a N}$ is needed to transmit these schedules in $T$ time slots. Fig.3 shows the speedup required by DOUBLE for a $16 \times 16$ switch under different reconfiguration delay settings. Because the speedup is an integer in slotted switching system, DOUBLE can only run on switches with speedup larger or equals to 3.

For an $N \times N$ switch with reconfiguration delay $z_a$, the accumulation time $T$ of DOUBLE should be no less than $T_{min} = 2z_a N$. Accumulation time may be relatively large for switches with large number of ports and/or long reconfiguration delay, e.g. a $128 \times 128$ switch with $z_a = 50$ may require an accumulation time larger than $10^5$ time slots.

*3) Bound of Average Delay and Worst Delay:* Consider a cell arriving at some time in the accumulation period of batch

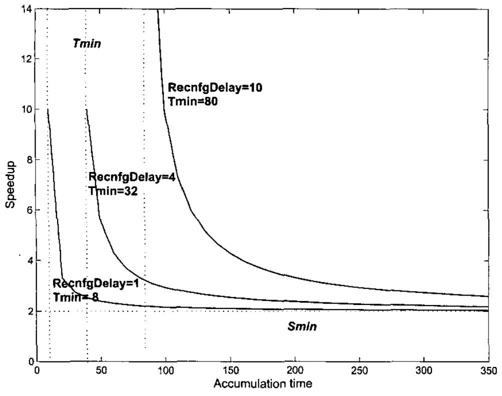

Fig. 3. Relationship between traffic accumulation time and required speedup (use N=16 as example).

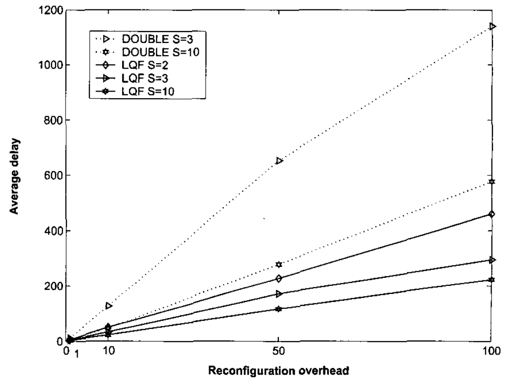

Fig. 4. Minimum achievable average delay by DOUBLE and LQF+Hold (for 4 × 4 switch under uniform traffic with load 0.99)

schedule $i$. The queuing delay of that cell includes: 1) waiting time before batch schedule $i$ is executed, which takes value of $[0, T]$ and 2) waiting time before its schedule comes. This may take a value in the range of $[z_a, T]$. Under an i.i.d. arrival, the expected value of waiting time in 1) is $T/2$. Thus the average cell delay for a TSA-based algorithm is at least larger than half of its accumulation time. For the worst-case delay bound, the execution of TSA ensures cell delay is less than $2T$.

*4) Algorithm Efficiency of DOUBLE under Uniform Traffic:* A $N \times N$ switch is able to set up $N$ non-contending connections between inputs and outputs. We say this switch has a capacity of $N$. If a particular algorithm can manage to set up to $K (0 \leq K \leq N)$ connections on average for the switch, we say the algorithm efficiency is $K/N$.

Let us look at the coarse matrix A generated by DOUBLE. Under admissible uniform traffic, the line sum and row sum of traffic matrix D is less than its accumulation time $T$, and $E[d_{ij}] = T/N$. That indicates for each row/column, the expected number of elements which are larger than $T/N$ is less than $N/2$. Since $a_{ij} = \lfloor \frac{d_{ij}}{T/N} \rfloor$, the expected number of non-zero entries in A is less than $N^2/2$. This half-empty coarse matrix A is scheduled to be covered by $N$ schedules, which may have $N^2$ potential transfer capacity. It is clear that around 50% of the transfer capacity is wasted.

The schedule of fine matrix B is not optimistic either. Elements in B are uniformly distributed in the range $[0, \lfloor T/N \rfloor]$. However each of the N schedules holds $\lceil T/N \rceil$ time slots. Many connections are idle for a large portion of the schedule holding time. Our simulation shows that the schedule efficiency for coarse matrix is 40% to 50%, and 30% to 40% for fine matrix under uniform traffic.

## B. Single Scheduling

Single-scheduling algorithms can be viewed as a slowed-down version of traditional electronic switch scheduling algorithms. It generates one schedule each time and hold this schedule for several time slots. Similar ideas can be found in burst-mode switching in [6] and packet-mode scheduling in [7]. The system works as schedule-reconfiguration-transfer cycle. The holding length of a particular schedule can either be constant or vary with the switch state. In this paper, we assume the holding length to be constant.

*1) A Representative Single-Scheduling Algorithm:* The queueing states change along with the transfer period and the schedule becomes outdated. Some VOQs may have already sent out all its cells before the transfer period ends but the connections for them cannot be released. Intuitively, a schedule favoring longer queues helps to reduce the resource wastage mentioned above.*Longest Queue First* (LQF) algorithm should be a good choice. LQF+Hold is used in this paper to refer to a single-scheduling which uses LQF algorithm to find matchings.

*2) Speedup Requirement:* As proved in Section III-A, a speedup of 2 is sufficient for a LQF+Hold scheduling algorithm to be stable.

*3) Average and Worst Delay:* An estimation model of average delay of the LQF+Hold scheduling algorithm under uniform traffic is given in Section III-B. The average delay is a function of switch size, reconfiguration delay, holding time and traffic arrival rate. No worst delay bound is guaranteed.

*4) Algorithm Efficiency:* Simulation shows that under heavy load, the LQF+Hold scheduling algorithm achieves an efficiency of over 90%. This is a direct consequence of the good properties of LQF.

## C. Comparison Between Batch and Single Scheduling

DOUBLE and LQF+Hold are simulated under same system setting. Fig. 4 shows the minimum average delay they achieve under different reconfiguration overheads and speedup settings for a switch of size 4. As can be seen, DOUBLE is sensitive to the speedup, and increase of speedup may significantly reduce the average delay. As the reconfiguration delay increases, both algorithms ensure the average delay will not increase a lot. For small switch size or short reconfiguration delay, DOUBLE and LQF+Hold provide an average delay of the same magnitude. Choice between them depends on what is the focus: small average delay or a guarantee of worst-case delay bound. For switches with large sizes and/or large reconfiguration delays, since DOUBLE may require a rather long accumulation length and hence increases the delay time, LQF+Hold scheduling may serve as a better choice.

## III. Analytical Models of Single Scheduling Algorithm

This section presents the analytical models for the stability and average delay of the LQF+Hold scheduling algorithm.

### A. Stability Analysis

*1) Definitions and Preliminary Results:* The behavior of an $N \times N$ switch can be approximated using $M$ ($M = N^2$) discrete-time queues of infinite capacities. Let $Q_t$ be the row vector of queue lengths at time $t$, i.e., $Q_t = \{q_t^1, q_t^2, \dots, q_t^M\}$, where $q_t^i$ is the number of cells in queue $i$ at time $t$. Denote the arrivals at time $t$ by $A_t = \{a_t^1, a_t^2, \dots, a_t^M\}$ and the departures by $D_t = \{d_t^1, d_t^2, \dots, d_t^M\}$. We assume that $a_t^i$ of vector $A_t$ are independent and identically distributed (i.i.d.) for variable $t$ with fixed $i$. Since at each time slot, for each input, there is at most one arrival, each element $a_t^i$ can only take values 0 or 1 for all $i$ and $t$. If the system has speedup $S$ ($S$ is assumed to be integer in this paper), then each element $d_t^i$ can take values $0, 1, \dots, S$. The system evolves according to $Q_{t+1} = Q_t + A_t - D_t$.

Denote $||X||$ by the Euclidean norm of vector $X = \{x^1, x^2, \dots, x^K\}$; $||X|| = \sqrt{\sum_{i=1}^{K}(x^i)^2}$. In addition, $V(X_t)$ is denoted to a special Lyapunov function: $V(X_t) = X_t X_t^T$.

The evolution of most practical discrete-time queuing systems can be described by an irreducible discrete-time Markov chain (DTMC), whose state vector at time $t$ is $Q_t$, $Q_t \in IN^M$. The following theorem is a paraphrase of Corollary 1 in [7]. It provides a criterion for the strong stability of DTMC queuing systems.

*Theorem 1:* Given a system of queues with state vector $Q_t$, if there exists $\epsilon \in \mathcal{R}^+$ and $B \in \mathcal{R}^+$ such that $\forall Q_t : ||Q_t|| > B$, $E[V(Q_{t+1}) - V(Q_t)|Q_t] < -\epsilon||Q_t||$, then the system of queues is strongly stable, and all the polynomial moments of the queue length distributions are finite.

*2) Stability of LQF+Hold Scheduling:* Let us consider the stability of the LQF+Hold scheduling. The system works in configuration-transfer cycles as shown in Fig. 5. New scheduling decisions are made at a sequence of time instants $t_n \in IN^+$, based on current queuing state $Q_{t_n}$. The connection is set up in the switching fabric after $z_a$ reconfiguration time slots and is held for a predetermined $z_b$ slots. Since the evolution of the system following $t_n$ is conditionally independent of the evolution of the system before $t_n$, the sequence of $t_n$ is a *non-defective* sequence of regeneration instants.

*Lemma 1:* Given a switch with reconfiguration delay $z_a$, LQF+Hold scheduling with a predefined holding time $z_b$ is stable under any admissible i.i.d. input traffic pattern $A_n$ as long as the switch has a speedup $S \geq \lceil \frac{z_a + z_b}{z_b} \rceil$.

*proof:* The evolution of the system is represented by a DTMC whose state is a vector of queue lengths $Q_{t_n}$. Between

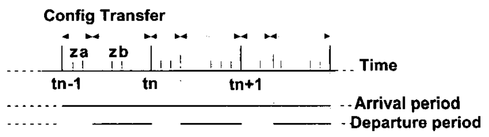

Fig. 5. Execution time line of single schedule+hold algorithm.

neighboring regeneration time instants, the system evolutes according to:

$$Q_{t_{n+1}} = Q_{t_n} + \left( \sum_{i=0}^{z_a+z_b-1} A_{t_n+i} - \sum_{j=z_a}^{z_a+Sz_b-1} D_{t_n+j} \right)$$

Please note that in the above equation the elements in $D_{t_n+j}$ are assumed to be either 0 or 1 (different from $D_t$ defined previously) to simplify the presentation. In addition, $\sum_{i=0}^{z_a+z_b-1} A_{t_n+i}$ and $\sum_{j=z_a}^{z_a+Sz_b-1} D_{t_n+j}$ will be represented by $\sum A$ and $\sum D$ in the following proof.

We start as

$$E[V(Q_{t_{n+1}}) - V(Q_{t_n})|Q_{t_n}] = E[2(\sum A - \sum D)Q_{t_n}^T + (\sum A - \sum D)(\sum A - \sum D)^T|Q_{t_n}]$$

Under the constraint of 0/1 arrival and [0,S] departures at every time slot, with a finite holding time $z_b$, $E[(\sum A - \sum D)(\sum A - \sum D)^T]$ is also finite. Thus,

$$\lim_{||Q_{t_n}|| \to \infty} \frac{E[V(Q_{t_{n+1}}) - V(Q_{t_n})|Q_{t_n}]}{||Q_{t_n}||} = \lim_{||Q_{t_n}|| \to \infty} \frac{2E[(\sum A - \sum D)Q_{t_n}^T|Q_{t_n}]}{||Q_{t_n}||}$$

Please note although all $D_{t_n+i}$ refer to the same matching, they are not necessarily equal. Since some queues assigned to transfer may have been empty before the next scheduling instant.

Define $D_\delta = St_b D_{t_n+z_a} - \sum_{j=z_a}^{z_a+Sz_b-1} D_{t_n+j}$; it is the difference between the assigned transfer capacity and the actual transmission number in one cycle. Thus,

$$\begin{aligned} E[(\sum A - \sum D)Q_{t_n}^T|Q_{t_n}] &= E[(\sum A - Sz_b D_{t_n+z_a} + D_\delta)Q_{t_n}^T|Q_{t_n}] \\ &= (z_a + z_b)E[A_n]Q_{t_n}^T - Sz_b D_{t_n+z_a}Q_{t_n}^T + E[D_\delta]Q_{tn}^T \end{aligned}$$

We know from [8] that LQF has the property of $(E[A_n] - D_{t_n})Q_{t_n}^T < 0$. If the speedup $S$ satisfies $S \geq \lceil \frac{z_a+z_b}{z_b} \rceil$. Note $E[D_\delta]Q_{t_n}^T \leq ME[z_b^2] < \infty$. Now we have,

$$\begin{aligned} &\lim_{||Q_{t_n}|| \to \infty} \frac{E[V(Q_{t_{n+1}}) - V(Q_{t_n})|Q_{t_n}]}{||Q_{t_n}||} \\ &\leq \lim_{||Q_{t_n}|| \to \infty} \frac{2(z_a + z_b)(E[A] - D_{t_n})Q_{t_n}^T}{||Q_{t_n}||} \\ &< -2(z_a + z_b)\xi \end{aligned}$$

According to Theorem 1, the system is stable. □

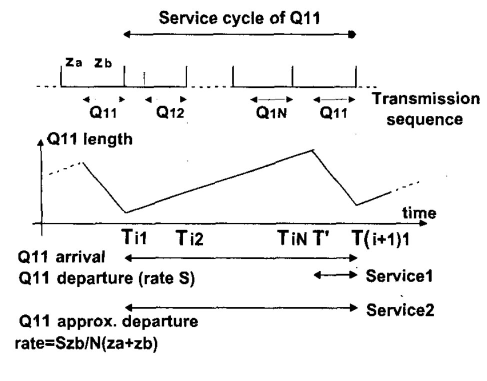

Fig. 6. Service cycle for a particular VOQ (use $Q^{11}$ as example).

The conclusion from the above analysis suggests that if the holding time is chosen to be larger than the reconfiguration delay, a speedup of 2 is sufficient for the LQF+Holding scheduling algorithm to be stable.

## B. Approximate Average Delay Estimation

Besides the property of stability, average cell delay is also a common criterion to evaluate the performance of scheduling algorithms. Given a switching system (N×N ports, reconfiguration delay $z_a$ and speedup $S$), if the LQF+Hold scheduling is used, then the cell delay is directly determined by the algorithm holding length $z_b$. In order to provide some clarification of the relationship between average delay and holding time, we construct an approximate queuing model. The model works under uniform traffic, but can be extended to non-uniform cases. It may also serve as a guidance on how to choose an appropriate holding time. In this paper, we only consider the case where the LQF+Hold scheduling algorithm have a constant holding time.

*1) Notation and Assumption:* Below is a list of notations used throughout the delay estimation:

| | |
|---|---|
| $p$ | traffic arrival rate for a single VOQ |
| $Q^{ij}$ | VOQ in input i holding cells to output j |
| $Q^{i*}$ | all VOQs in input i |
| $Q^{ij}_t$ | number of cells in $Q^{ij}$ at time $t$ |
| $N_s$ | expected number of cells in a VOQ at the end of service cycle |

Under uniform traffic, the traffic arrival to VOQs can be approximated using $N^2$ i.i.d. possion processes with arrival rate $p$. Traffic is admissible; thus $p < 1/N$. In other words, $p$ is the expected number of arrivals for a VOQ in one time slot.

Under uniform traffic, LQF can be approximated by a time division multiplexing (TDM) scheme. Since the VOQs receive similar arrival and service, let us focus on VOQs in one input port, say input port 1. Input port 1 connects to output 1, output 2, $\cdots$, output $N$ in a cyclical manner. $Q^{11}, Q^{12}, \cdots, Q^{1N}$ are served in sequence and have similar behavior. The service cycle for each single queue is of length $N(z_a + z_b)$ and is illustrated by Fig. 6.

*2) Expected Number of Cells in a Single VOQ at the End of Service Cycle:* Given some holding time $z_b$ that ensures the stability of the system, take $Q^{11}$ as a representative of VOQs. Assume time interval $[T_{i1}, T_{(i+1)1}]$ is a service cycle for $Q^{11}$ when the switch is in stable state. As illustrated in Fig. 6, $Q^{11}$ may have cell arrivals during $[T_{i1}, T_{(i+1)1}]$ at rate $p$, while the departure may only happen during $[T', T_{(i+1)1}]$ at rate $S$. We approximate the rate $S$ service (denoted by Service1) in $[T', T_{(i+1)1}]$ by a rate $\frac{Sz_b}{N(z_a+z_b)}$ service (denoted by Service2) in $[T_{i1}, T_{(i+1)1}]$, shown in Fig.6. Under such approximation, $Q^{11}$ behaves as an M/D/1 system with $\lambda = p$ and $\mu = \frac{Sz_b}{N(z_a+z_b)}$. Denote $\rho = \lambda/\mu$, according to the P-K formula, the expected number of cells in $Q^{11}$ at time $T_{i1}(T_{(i+1)1})$ is given by:

$$E[Q^{11}_{T_{i1}}] = E[Q^{11}_{T_{(i+1)1}}] = \frac{\rho^2}{2(1-\rho)}$$

This can be extended to $Q^{1k}, k \in [1, N]$ as follows:

$$N_s = E[Q^{1k}_{T_{ik}}] = E[Q^{1k}_{T_{(i+1)k}}] = \frac{\rho^2}{2(1-\rho)} \qquad (1)$$

Consider the accuracy of the above approximation. Under high traffic load, $Q^{11}$ has a high probability of being non-empty during $[T_{i1}, T_{(i+1)1}]$. Thus Service1 and Service2 may serve the same number of cells. But if there is a situation with rare cell arrival near $T$ and the queue is often empty, while a burst arrives from the middle of the duration. This arrival pattern does not influence Service1 since it accumulates the traffic and then sends; however, some of the service capacity of Service2 is wasted at first. So the number of cells at time $T_{(i+1)1}$ by Service2 is an upper bound of that by Service1. Although our approximation will be reasonably accurate only for relatively high traffic loads; still, this estimation is meaningful since normally switches are best tested and evaluated under their full capacity.

*3) Expected Number of Cells in an Input at Time $T_{ik}$:* The expected total number of cells in input 1 at time $T_{iN}$ is $E[Q^{1*}_{T_{iN}}] = \sum_{j=1}^{N} E[Q^{1j}_{T_{iN}}]$. According to the service cycle,

$$E[Q^{11}_{T_{iN}}] = N_s + (N-1)(z_a + z_b)p$$

$$\vdots$$

$$E[Q^{1j}_{T_{iN}}] = N_s + (N-j)(z_a + z_b)p$$
$$E[Q^{iN}_{T_{iN}}] = N_s$$

So we get

$$E[Q^{1*}_{T_{iN}}] = N \times N_s + \frac{N(N-1)}{2}(z_a + z_b)p \qquad (2)$$

Equation 2 can be extended to any time instant $T_{ik}, k \in [1, N]$.

*4) Expected Average Number of Cells in an Input:* Finally, let us consider the expected average number of cells (call it $E[AVG_{Q^{1*}}]$) in input 1. Under the stable assumption, $E[AVG_{Q^{1*}}]$ is the same at any time interval $[T_{ik}, T_{i(k+1)}](k \in 1, \cdots, N-1)$ and $[T_{iN}, T_{(i+1)1}]$. Without loss of generality, we just consider the time interval $[T_{iN}, T_{(i+1)1}]$. Suppose we have $Q^{1*}_{T_{iN}}$ cells at the starting time slot $T_{iN}$. A cell arrives

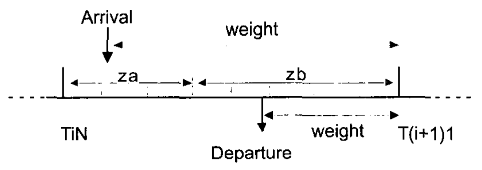

Fig. 7. Illustration of weighted arrival and departure. (Here we assume cells arrives at the beginning of a time slot, and departures at the end of a time slot.)

at that time slot and there is no other arrival or departure in $[T_{iN}, T_{(i+1)1}]$. Obviously, $AVG_{Q^{1*}} = Q^{1*}_{T_{iN}} + 1$. Consider another arrival situation: there is one cell arriving at the middle of the time interval, and no other arrival or departure. This time, $AVG_{Q^{1*}}$ equals to $Q^{1*}_{T_{iN}} + 0.5$. Here we can see arrivals contribute to the average queue length on a weighted basis, so do the departures. The weight is defined as the difference between the end of time interval and arrival/departure time instant, as shown in Fig. 7.

From the above analysis, $E[AVG_{Q^{1*}}]$ can be expressed as follows:

$$E[AVG_{Q^{1*}}] = \mathcal{Q} + \mathcal{A} - \mathcal{D} \tag{3}$$

where $\mathcal{Q}$: expected number of cells in $Q^{1*}$ at time $T_{iN}$

$\mathcal{A}$: expected number of weighted arrival in time interval $[T_{iN}, T_{(i+1)1}]$

$\mathcal{D}$: expected number of weighted departure in time interval $[T_{iN}, T_{(i+1)1}]$

Assume there are $A^{1*}$ cells arrive input 1 during $[T_{iN}, T_{(i+1)1}]$, the $ith$ cell has a weight $w_i$. Then A can be expressed as

$$\begin{aligned}\mathcal{A} &= E[\sum_{i=1}^{A^{1*}} w_i] = E[E[\sum_{i=1}^{A^{1*}} w_i | A^{1*}]] && (4)\\ &= E[\frac{1}{2}A^{1*}] = \frac{(z_a+z_b)Np}{2} && (5)\end{aligned}$$

The transformation from (4) to (5) depends on the property of the possion process: given there are $n$ arrivals in a time interval with length $t$, then those $n$ arrivals are i.i.d. uniformly distributed along the duration.

Now let us consider the departures of input 1. During $[T_{iN}, T_{(i+1)1}]$, only $Q^{11}$ can transmit cell out in $[T', T_{(i+1)1}]$. An accurate description of when all the departures from $Q^{11}$ happen is extremely difficult. Since $Q^{11}$ may have many accumulated cells near time $T'$ and they leave at full service capacity $S$. But as time passes by, the service rate may follow a decreasing trend, with small trembling according to the burst arrival. We approximate the departures using a flat service with rate $S'$. $\mathcal{D}$ under real situation is upper-bounded by $\mathcal{D}$ calculated under the above approximation. Since we assume the switch to be stable, during the service cycle of $Q^{11}$ (which is $[T_{i1}, T_{(i+1)1}]$), the expected number of arrivals to $Q^{11}$ equals to the expected number of departures. Thus $S' = \frac{N(z_a+z_b)\times p}{z_b}$.

From the above analysis, we get

$$\mathcal{D} = \frac{N(z_a+z_b)\times p}{z_b} \times \frac{(z_b-1)+\cdots+0}{z_a+z_b} = Np \times \frac{z_b-1}{2} \tag{6}$$

We've already shown $\mathcal{Q} = E[Q^{1*}_{iN}]$ in Eq.(2). Together with Eq.(5) and Eq.(6), $E[AVG_{Q^{1*}}]$ can be calculated.

$$E[AVG_{Q^{1*}}] = \frac{N\rho^2}{2(1-\rho)} + \frac{(z_a+z_b)N^2p}{2} - \frac{(z_b-1)Np}{2} \tag{7}$$

Following Little's Formula, under a given $p$, the expected average cell waiting time $E[AVG_W]$ in input 1 is

$$E[AVG_W] = E[AVG_{Q^{1*}}]/Np$$

To minimize the average delay is equivalent to minimizing the average number of cells. So a suitable $z_b$ can be chosen according to

$$z_b^* = \arg_{z_b} min\{\frac{\rho^2}{1-\rho} + (N-1)pz_b\} \tag{8}$$

given $\rho = pN(z_a+z_b)/Sz_b$, and $z_b$ is an integer that satisfies the requirement in Lemma 1.

## IV. Conclusion

Scheduling optical-fabric switches with reconfiguration delay is a relatively new topic motivated by the recent increase of line rates, requirements on switch scalability, and the use of multi-rack switch architectures. This paper compares the performance of two classes of scheduling schemes designed for switches with reconfiguration delay. In addition, we investigated the stability requirement as well as average/worst delay. Mathematical models are formulated for the LQF+Hold scheduling algorithm.The goal of this research is to provide some guidance on the choice of the appropriate scheduling algorithms for given hybrid optical-electronic switch systems.

## References

[1] M. Zirngibl, *Optical Fiber Communications.* Kaminow/Li Academic Press, 2002.

[2] T. Wu, C. Y. Tsui, and M. Hamdi, "A 2 Gb/s 256*256 CMOS crossbar switch fabric core design using pipelined mux," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'02)*, Scottsdale, Arizona, May 2002, pp. 568–571.

[3] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with confi guration overhead," in *Proc. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, New York, June 2002, pp. 342–351.

[4] T. Inukai, "An effi cient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun.*, vol. 27, pp. 1449–1455, Oct. 1979.

[5] 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.

[6] G. Nong and M. Hamdi, "Burst-based scheduling algorithms for non-blocking atm switches with multiple input queues," *IEEE Commun. Lett.*, vol. 4, pp. 202–204, June 2000.

[7] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Pakcet-mode scheduling in input-queued cell-based switches," *IEEE/ACM Trans. Networking*, vol. 10, pp. 666–678, Oct. 2002.

[8] 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.