

Improving Quality of Service for Switched Processing Systems

Ying-Chao Hung
Graduate Institute of Statistics
National Central University
Jhongli 32049, Taiwan
hungy@stat.ncu.edu.tw

George Michailidis
Department of Statistics
The University of Michigan
Ann Arbor, MI 48109-1107
gmichail@umich.edu

Abstract- Switched Processing Systems (SPS) capture the essence of a fundamental resource allocation problem in many modern communication, computer and manufacturing systems involving heterogeneous processors and multiple classes of job traffic flows. Recently, increased attention has been paid to the issue of improving quality of service (QoS) performance in terms of delays and backlogs of the associated scheduling policies, rather than simply maximizing the system's throughput. In this study, we investigate a class of throughput maximizing scheduling policies called MaxProduct policies. The objective is that through a use of dynamically changing "optimal" queue weights, the corresponding QoS performance measure -e.g. the average system delay- can be significantly improved. The proposed approach involves a statistical smoothing technique for tracking the system's workload and utilizes the result of how the MaxProduct policies drain out an initially placed workload in the shortest possible time. It is further shown that the proposed modification of the MaxProduct policy, achieves maximum throughput without requiring knowledge of the incoming traffic's statistics. The scheduling policy is illustrated on a small SPS subject to different types of input traffic.

I. INTRODUCTION

Switched Processing Systems (SPS) arise in diverse application areas such as computing, manufacturing, packet-switching, wireless networking and call centers. An important feature of such systems is that servers are flexible, interdependent, and have randomly varying service capabilities. A basic objective in the study of such systems is to dynamically select the service configurations, so that the throughput of the system is maximized. Over the years maximum throughput issues have been extensively studied for many scheduling policies [1, 2, 5, 7, 14, 15]. More recently, increased attention has been paid to the quality of service (QoS) that these such scheduling policies can achieve.

Analytical studies for the QoS performance of scheduling policies go back to the work of Cruz [4] and Chang et al. [3],

where deterministic bounds for QoS measures of single queues were derived under tight traffic conditions. Stolyar [13] introduced the largest weighted delay first (LWDF) discipline and showed that it is an optimal solution of delay tails in a multi-class network using large deviations techniques. In subsequent work, he showed that the equivalent workload of a generalized switch model is minimized under heavy traffic assumptions by extending the MWM algorithm to the MaxWeight policy [12]. Ross and Bambos [9, 10] introduced a randomized algorithm and compared its QoS performance with a class of projective cone scheduling policies. It is worth noting that most of these scheduling policies can be shown to belong to the class of Max-Product policies introduced by Armony and Bambos [1]. This class of policies are also known as *Cone* policies since they partition the workload space into adjacent decision cones.

In the present paper, we further explore the class of Max-Product policies. Our goal is to associate an "optimal" weight with each queue in a multi-class SPS, so that a QoS measure (e.g. average system delay) can be significantly improved. It should be noted that by placing an arbitrary (but meaningful) weight on each queue, a new cone partition (see Section II) of the workload space is formed, which does not affect the throughput of the system [1]. However, a bad cone partition of the workload space might not be able to manage well the QoS tradeoffs. In order to construct an optimal cone partition of the workload space, we propose a two-stage approach. In the first stage, a flexible model for estimating the traffic intensity of each input traffic flow is developed by tracking the amount of work coming into the system. This flexible model allows us to capture well the fluctuations of traffic intensity for each input flow. In the second stage, the "optimal" weights are assigned on each queue at certain decision points in time, so that the MaxProduct policy performs well. These optimal weights are obtained by utilizing the flexible model derived in the first stage and the result of how the MaxProduct policies drain out an initially placed workload in the shortest possible time. The proposed approach exhibits the following characteristics: it is

(i) throughput maximizing (ii) adaptive to traffic fluctuations and does not require explicit knowledge about the shape of the input traffic.

The remainder of the paper is organized as follows. In Section II, a general switched processing system and the MaxProduct scheduling policies are introduced. In Section III, the algorithm of obtaining the optimal weight for each queue together with some theoretical results are presented. In Section IV, a statistical smoothing technique for estimating the traffic intensity is introduced, while in section V a performance assessment of the proposed policy is undertaken through a simulation study. Finally, some concluding remarks are drawn in Section VI.

II. SWITCHED PROCESSING SYSTEMS

Consider a multiclass queueing system comprised of Q parallel, infinite capacity, first-in-first-out (FIFO) queues, with each queue corresponding to a different class of job traffic. The class q jobs arrive according to a general process \mathcal{A}_q , $q = 1, \dots, Q$. Suppose the j^{th} job of class q arrives to the system at time $t_j^q \in \mathbb{R}$ and carries the service requirement σ_j^q . We model these random quantities as elements of a random marked point process $\mathcal{I}_q = (t_j^q, \sigma_j^q)$, $j \in \mathbb{Z}$, defined on some probability space (Ω, \mathcal{F}, P) . Define

$$\rho_q = \lim_{t \rightarrow \infty} \left[\frac{1}{t} \sum_{j \in \mathbb{Z}} \sigma_j^q \mathbf{1}_{\{t_j^q \in (0, t]\}} \right] \quad (1)$$

to be the long-term *traffic intensity* of jobs of class q . At any point in time, the system can be in one of M service modes. When it is in service mode m , the jobs of the queue q receives service at rate μ_{mq} . Therefore, mode m is associated with the service rate vector $\vec{\mu}_m = (\mu_{m1}, \mu_{m2}, \dots, \mu_{mQ})$.

Assume the service requirements are mutually independent and independent of the arrival processes. The *workload* process $\vec{W}(t) = (W_1(t), \dots, W_Q(t))$ is then defined as

$$W_q(t) = \left[\sum_{j \in \mathbb{Z}} \sigma_j^q \mathbf{1}_{\{t_j^q \in (0, t]\}} - \sum_{m=1}^M \mu_{mq} \int_0^t \mathbf{1}_{\{W_q(s) > 0 \text{ and } \vec{\mu}_m \text{ is used at time } s\}} ds \right]^+ \quad (2)$$

for every $q = 1, 2, \dots, Q$. This basic queueing model can be viewed as a generalization of switch models under complete resource pooling.

A. Scheduling Policies

At certain points in time the system needs to decide which service vector $\vec{\mu}_i \in \{\vec{\mu}_1, \vec{\mu}_2, \dots, \vec{\mu}_M\}$ should be used, or which service mode to switch into. Furthermore, it is assumed that the allocation of resources is preemptive; that is, the service of a job under a service mode can be interrupted and be resumed

later on. In a recent study [1], a family of scheduling policies, called *Cone* policies, were constructed as follows: *Cone* policies partition the workload space \mathcal{W} into geometric decision cones. When the workload $\vec{W}(t) = (w_1(t), \dots, w_Q(t))$ is in a certain cone, the system is switched into any service mode associated with that cone. To illustrate the nature of such scheduling policies, let us consider an arbitrary weight vector $\vec{\alpha}$ with each component $\alpha_q > 0$ corresponding to each job class q , $q = 1, \dots, Q$. The weighted inner product of every two vectors $\vec{W}(t)$ and $\vec{\mu}_m$ is then defined as

$$\langle \vec{W}(t), \vec{\mu}_m \rangle_{\vec{\alpha}} = \sum_{q=1}^Q \alpha_q w_q(t) \mu_{mq}. \quad (3)$$

The MaxProduct policy, denoted by $\pi_{\vec{\alpha}}$, structures its cones by choosing the service vectors that have maximal inner product with the workload vector. The decision cone corresponding to service mode m in the workload space is then defined as

$$C_m = \{ \vec{W}(t) : m = \arg \max_{m' \in \{1, \dots, M\}} \langle \vec{W}(t), \vec{\mu}_{m'} \rangle_{\vec{\alpha}} \}. \quad (4)$$

Note that these decision cones form a partition of the workload space, which is presented by $\mathcal{W} = \cup_m C_m$. One example of such decision cones for a 2-queue system is shown in Fig. 1.

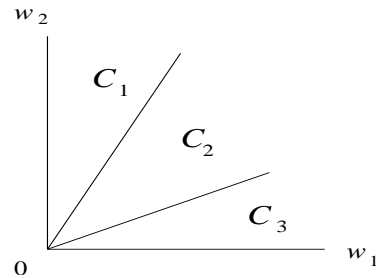


Figure 1: The decision cones of $\pi_{\vec{\alpha}}$ in the workload space for a 2-queue system with service vectors $\vec{\mu}_1 = (0, 4)$, $\vec{\mu}_2 = (2, 3)$, $\vec{\mu}_3 = (3, 0)$ and $\vec{\mu}_4 = (2.5, 1)$.

It is noted that in Fig. 1, $\vec{\mu}_4$ is never selected by the MaxProduct policy, since it is “dominated” by a convex combination of $\vec{\mu}_2$ and $\vec{\mu}_3$, and therefore never maximizes the above defined inner product at any point in time. In the special case, where $\vec{W}(t)$ is on the boundary segment of two or more decision cones, the policy arbitrarily chooses any service vector from those having maximal inner product with $\vec{W}(t)$.

Remark: In many practical situations, the MaxProduct policies can be modified by using the queue length or delay rather than the workload. If the queue length is considered, it is called the queue-length driven MaxProduct policy [1] or MaxWeight discipline [12].

B. System Stability

It is shown in [1] that policy $\pi_{\vec{\alpha}}$ maximizes the throughput and maintains *rate stability* of the queueing system. Specifi-

cally, the *stability region* of the system under consideration is defined as

$$S = \left\{ \vec{\rho} \in \mathbb{R}_+^Q : \rho_q < \sum_{m=1}^M \omega_m \mu_{mq} \text{ for all } q \in \{1, \dots, Q\} \right\} \quad (5)$$

where $\omega_m \in [0, 1]$ for all $m = 1, \dots, M$ and $\sum_{m=1}^M \omega_m = 1$. It can further be shown that the stability region is the *convex hull* generated by all service vectors $\vec{\mu}_m$. To illustrate, the stability region for a 2-queue system with 4 service vectors $\vec{\mu}_1 = (3, 0)$, $\vec{\mu}_2 = (2, 3)$, $\vec{\mu}_3 = (0, 4)$ and $\vec{\mu}_4 = (2.5, 1)$ is shown in Fig. 2.

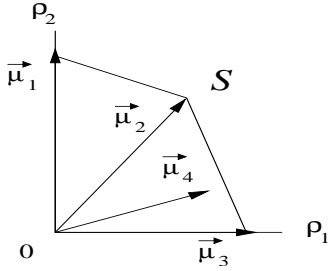


Figure 2: The stability region S for a 2-queue system with $\vec{\mu}_1 = (3, 0)$, $\vec{\mu}_2 = (2, 3)$, $\vec{\mu}_3 = (0, 4)$ and $\vec{\mu}_4 = (2.5, 1)$.

Remark: The rate stability of the system is defined as follows. For all input loads $\vec{\rho} \in S$, with probability 1, $\lim_{t \rightarrow \infty} \frac{\vec{W}(t)}{t} = \vec{0}$.

Notice that by choosing different weight vectors $\vec{\alpha}$, a rich family of MaxProduct policies $\pi_{\vec{\alpha}}$ can be generated. Although these weight vectors do not affect the system throughput [1], they nevertheless play an important role on improving the QoS performance. It is easy to see that the higher the individual queue weight α_q is, the more system resources, policy $\pi_{\vec{\alpha}}$ will devote to queue q . The latter is achieved by selecting service vectors $\vec{\mu}_m$ having large service rate μ_{mq} . Therefore, increasing the weight of a queue will result in lowering its backlog and delay.

III. OPTIMAL SELECTION OF WEIGHTS

We begin our discussion of selecting the optimal queue weights by considering the following hypothetical setup. Suppose that an initial workload \vec{W}_0 is placed in the system and that all subsequent job arrivals are blocked; then, an interesting question is how policies $\pi_{\vec{\alpha}}$ empty all queues in an optimal fashion. Obviously, the specific choice of the weight vector $\vec{\alpha}$ would influence the result. The proposed solution is to select $\vec{\alpha}$ so as to minimize the emptying time for \vec{W}_0 . We proceed by describing the algorithm that constructs the optimal weight vector $\vec{\alpha}$ for any given initial workload \vec{W}_0 . The properties of scheduling policy $\pi_{\vec{\alpha}}$ are discussed later on.

A. Algorithm for Constructing Optimal Weights

For any initial workload \vec{W}_0 , let us denote the desired optimal solution of $\vec{\alpha}$ by $\vec{\alpha}(\vec{W})$. In order to better illustrate the construction of $\vec{\alpha}(\vec{W})$, we focus our attention on 2-queue systems. The proposed algorithm works in sequential fashion and the steps are summarized below.

Step 1. Construct simultaneously all the cones C_m generated by the admissible (non-dominated) service vectors $\vec{\mu}_m$ in the workload space by choosing $\vec{\alpha} = \vec{1}$, $m \in \{1, \dots, M\}$. Further, note that every two adjacent (and admissible) service vectors $\vec{\mu}_i$ and $\vec{\mu}_j$ also generate a cone S_{ij} which can be treated as a subset of the workload space \mathcal{W} (see the left panel in Fig. 3). Without loss of generality we assume the workload space can then be represented by $\mathcal{W} = \cup C_m = \cup S_{ij}$. Therefore, as shown in the left panel of Fig. 3, for any given initial workload $\vec{W}_0 = (W_1, W_2) \in \mathcal{W}$ we have that

$$\vec{W}_0 \in S_{ij} \text{ for some } i, j \in \{1, \dots, M\}, i \neq j. \quad (6)$$

Step 2. Identify the two cones C_i and C_j corresponding to S_{ij} derived in Step 1. Denote the boundary segment of C_i and C_j by the line L_{ij} and let $\vec{l}_{ij} = (l_1, l_2)$ be an arbitrary vector on it. Then \vec{l}_{ij} is a solution of \vec{w} which satisfies

$$\langle \vec{\mu}_i, \vec{w} \rangle_{\vec{1}} < \langle \vec{\mu}_j, \vec{w} \rangle_{\vec{1}}. \quad (7)$$

Step 3. Solve $\vec{\alpha} = (\alpha_1, \alpha_2)$ from the following equation

$$\vec{l}_{ij} = (l_1, l_2) = (\alpha_1 W_1, \alpha_2 W_2). \quad (8)$$

The solution found in this step is denoted by $\vec{\alpha}(\vec{W})$, and

$$\vec{\alpha}(\vec{W}) = \left(\frac{l_1}{W_1}, \frac{l_2}{W_2} \right). \quad (9)$$

Remark: For general Q -queue systems, the solution of $\vec{\alpha}(\vec{W})$ can be found in an analogous fashion. However, one must pay attention to the computational issues arising for identifying the geometric cone to which the initial workload \vec{W} belongs. This issue is further discussed in Section VI.

Note that by choosing $\vec{\alpha} = \vec{\alpha}(\vec{W})$ in Step 3, the original decision cones C_1, \dots, C_M (or decision boundaries L_{ij}) are tilted so that a new partition $\{C'_1, \dots, C'_M\}$ of the workload space (or new decision boundaries L'_{ij}) is formed (see the right panel of Fig. 3). The following fact explains the benefits obtained from the proposed policy $\pi_{\vec{\alpha}(\vec{W})}$.

B. Properties of the $\pi_{\vec{\alpha}}$ Policy

Fact 1: For any initial workload \vec{W} , $\pi_{\vec{\alpha}}$ empties the system in the shortest path by choosing $\vec{\alpha} = \vec{\alpha}(\vec{W})$.

Proof: We establish the result for a 2-queue system; the proof for the general case follows along similar lines. It is noted that in Step 3 of the algorithm, a new cone partition of the workload space \mathcal{W} is formed by choosing $\vec{\alpha} = \vec{\alpha}(\vec{W})$ in $\pi_{\vec{\alpha}}$. The idea behind is that, by tilting the decision cones accordingly, $\vec{\mu}_i$ and $\vec{\mu}_j$ will have two maximal projections on \vec{W}_0 among

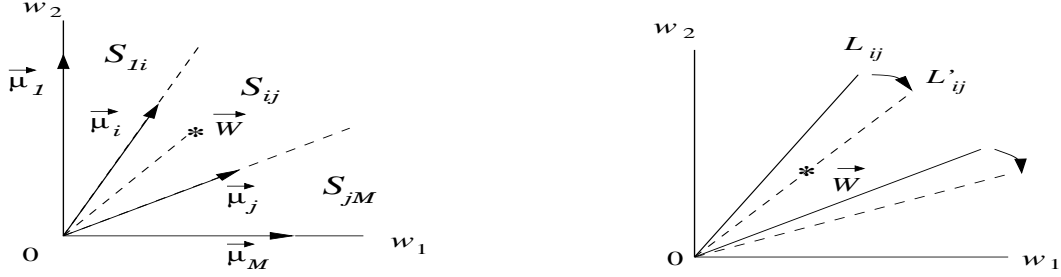


Figure 3: (Left panel): The particular S_{ij} cone shown in the workload space such that $\vec{W} \in S_{ij}$. (Right panel): A new decision cone partition (the dash line) of the workload space by choosing the weight vector $\vec{\alpha} = \vec{\alpha}(\vec{W})$ in $\pi_{\vec{\alpha}}$.

all service vectors. Therefore, $\pi_{\vec{\alpha}(\vec{W})}$ will use only $\vec{\mu}_i$ and $\vec{\mu}_j$ to drain out the initial workload \vec{W}_0 . Furthermore, we assume that it is allowed to split service mode i and service mode j with proportions r_i and r_j , respectively, $r_i + r_j = 1$. The initial workload \vec{W}_0 can then be viewed as being drained out by an average service vector $r_i\vec{\mu}_i + r_j\vec{\mu}_j$. Denote $\vec{W}_0 = (W_1, W_2)$, $\vec{\mu}_i = (\mu_{i1}, \mu_{i2})$ and $\vec{\mu}_j = (\mu_{j1}, \mu_{j2})$, and without loss of generality assume that $\mu_{i1} < \mu_{j1}$ and $\mu_{i2} > \mu_{j2}$, it is clear that $r_i\vec{\mu}_i + r_j\vec{\mu}_j$ can empty the system in the shortest path (i.e. $r_i\vec{\mu}_i + r_j\vec{\mu}_j$ is exactly located on the vector \vec{W}_0) by choosing

$$r_i = \frac{\mu_{j1}W_2 - \mu_{j2}W_1}{(\mu_{i2} - \mu_{j2})W_1 + (\mu_{j1} - \mu_{i1})W_2} \quad \text{and} \quad (10)$$

$$r_j = \frac{\mu_{i2}W_1 - \mu_{i1}W_2}{(\mu_{i2} - \mu_{j2})W_1 + (\mu_{j1} - \mu_{i1})W_2}. \quad (11)$$

The latter fact can be easily established by considering the ratio (slope of $r_i\vec{\mu}_i + r_j\vec{\mu}_j$):

$$\frac{r_i\mu_{i2} + r_j\mu_{j2}}{r_i\mu_{i1} + r_j\mu_{j1}} = \frac{W_2}{W_1}. \quad (12)$$

The result of Fact 1 shows that, by choosing $\vec{\alpha} = \vec{\alpha}(\vec{W})$ in $\pi_{\vec{\alpha}}$, the initial workload \vec{W} is decomposed along $\vec{\mu}_i$ and $\vec{\mu}_j$, say, $\vec{W} = k(r_i\vec{\mu}_i + r_j\vec{\mu}_j)$ for some $k > 0$. Moreover, the result also agrees with the solution to the following optimization problem. Suppose $\vec{\mu}_i = (\mu_{i1}, \mu_{i2})$ is used for time t_i and $\vec{\mu}_j = (\mu_{j1}, \mu_{j2})$ is used for time t_j to drain out an initial workload $\vec{W}_0 = (W_1, W_2)$. We consider the optimization problem:

$$\begin{aligned} & \underset{t_i, t_j}{\text{minimize}} && (t_i + t_j) \\ & \text{subject to} && W_1 \leq \mu_{i1}t_i + \mu_{j1}t_j, \\ & && W_2 \leq \mu_{i2}t_i + \mu_{j2}t_j. \end{aligned}$$

If (t_i^*, t_j^*) is the one solving the above optimization problem, it is easy to show that

$$r_i = \frac{t_i^*}{t_i^* + t_j^*} \quad \text{and} \quad r_j = \frac{t_j^*}{t_i^* + t_j^*}. \quad (13)$$

This implies that, by choosing (r_i, r_j) according to (13), policy $\pi_{\vec{\alpha}(\vec{W})}$ not only empties the system in the shortest path but also minimizes the total emptying time. The minimal emptying time is then given by

$$\frac{|\vec{W}_0|}{|r_i\vec{\mu}_i + r_j\vec{\mu}_j|},$$

where the norm of \vec{W}_0 is defined as $\sqrt{W_1^2 + W_2^2}$ and that defined similarly for $r_i\vec{\mu}_i + r_j\vec{\mu}_j$.

Remark: The preceding minimal emptying time based on the MaxProduct policy is the same as that derived by the FastEmpty policy introduced in [1]. However, the difference is that the FastEmpty policy might not drain out the initial workload in the shortest path (i.e. in a straight line).

IV. ESTIMATION OF TRAFFIC INTENSITY

In this Section, we proceed to make the proposed policy $\pi_{\vec{\alpha}}$ operational. Suppose the system starts empty at time 0 and no service is allowed before time t (where t is considered to be fairly large); then we have that $W_i(t) \approx \rho_i t$ for any input traffic flows where ρ_i does not change over time. This implies that the result of Fact 1 can directly be applied by replacing the initial workload \vec{W}_0 with the traffic intensity $\vec{\rho}$ in the proposed algorithm. In this case, we denote the optimal weight solution by $\vec{\alpha}(\vec{\rho})$. However, it may be hard to implement the policy associated with this optimal weight solution since in many applications the traffic intensities are unknown or changing over time (i.e. for non-stationary input processes). Therefore, building a flexible model that predicts well the fluctuations of the traffic intensity over time is a necessary component for implementation purposes. We describe next a statistical smoothing technique called *running medians* [16], whose objective is to provide a time series smoother for the traffic intensity of each input by tracking the amount of work coming into the system.

Consider a sequence of equally spaced time intervals $(0, \Delta]$, $(\Delta, 2\Delta]$, \dots , where queue q has $N_q(n)$ arriving jobs in the n th time interval and the j th job of queue q carries the service requirement σ_j^q , $j = 1, \dots, N_q(n)$. The traffic intensity of queue

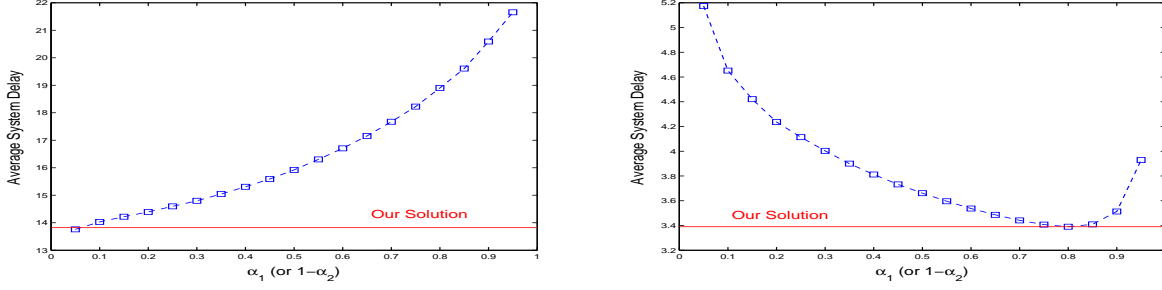


Figure 5: (Left panel): The resulting average system delay of our approach and $\pi_{\vec{\alpha}}$ with possible choices of $\vec{\alpha}$. The input processes are compound Poisson with $\mathcal{S}_1 = \mathcal{S}_2 = 1$, and $\mathcal{T}_1 = 0.345$, $\mathcal{T}_2 = 10$. (Right panel): The resulting average system delay of our approach and $\pi_{\vec{\alpha}}$ with possible choices of $\vec{\alpha}$. The input processes are compound Poisson with $\mathcal{S}_1 = \mathcal{S}_2 = 1$, and $\mathcal{T}_1 = 2$, $\mathcal{T}_2 = 0.286$.

parameter H , ρ_q is the traffic intensity, and σ_q is the variance of traffic in a time unit. For simplicity purposes, we assume that the job interarrival times of each queue are deterministic, say, equal to one unit. Further, assume the system starts at time 0, and that the n th job of queue q carries the service requirement

$$A_q(n) - A_q(n-1) = \rho_q + \sigma_q[Z_q(n) - Z_q(n-1)], \quad (17)$$

$q = 1, 2$. We first consider the system with $(\rho_1, \rho_2) = (2.9, 0.1)$ (similar to the first example in part A), and assume $\sigma_1 = 100$ and $\sigma_2 = 10$. Note that such choices for ρ_q and σ_q can guarantee that $A_q(t)$ will not go negative, almost surely. In addition, the Hurst parameter is chosen to be $H = 0.7$ for $Z_q(t)$, so that the two input processes can be characterized as *long-range dependent*. Once again, computer simulations were performed to derive the average system delay under the proposed adaptive strategy and the MaxProduct policies based on fixed choices for the weights $\vec{\alpha}$. The resulting average system delays are shown in the left panel of Fig. 6. We next consider the system with $(\rho_1, \rho_2) = (0.5, 3.5)$ (similar to the second example in part A), and assume $\sigma_1 = 10$, $\sigma_2 = 100$. The resulting system delays are shown in the right panel of Fig. 6. As can be seen from the results shown in Fig. 6, the adaptive policy outperforms the entire class of the fixed MaxProduct policies (i.e. outperforms $\pi_{\vec{\alpha}}$ for almost all possible choices of $\vec{\alpha}$). For the cases where the fixed MaxProduct policy performs better (right side in the right panel of Fig. 6) the numerical differences are practically negligible.

C. A 2-queue System with Non-stationary Input Processes

We next consider the same fractional Brownian motion input processes but with changing input rates (i.e. non-stationary input processes) for the 2-queue system. Assume the system starts at time 0 with the initial traffic intensities $(\rho_1, \rho_2) = (0.8, 3.0)$ and (ρ_1, ρ_2) can change periodically at certain points in time, say, $t = \{2500, 5000, 7500, \dots\}$. Specifically, define the traffic intensities at time t by

$$(\rho_1(t), \rho_2(t)) = (0.8, 4.0) \quad (18)$$

for $t \in [2500i, 2500(i+1))$, $i = 0, 2, 4, \dots$, and

$$(\rho_1(t), \rho_2(t)) = (3.0, 1.8) \quad (19)$$

for $t \in [2500i, 2500(i+1))$, $i = 1, 3, 5, \dots$. Note that the long-term traffic intensities for both queues are then $(\rho_1, \rho_2) = (1.9, 2.9)$, which clearly belong to the stability region. Analogous to (21), the service requirement of the n th job of queue q which arrives at time $t \in [2500i, 2500(i+1))$, is denoted by

$$\rho_q(t) + \sigma_q(t)[Z_q(i_n) - Z_q(i_{n-1})] \quad (20)$$

where i_n and i_{n-1} are the arrival times of the n th and $(n-1)$ th job of queue q in the time interval $[2500i, 2500(i+1))$, $i = 0, 1, 2, \dots$, and $\sigma_q(t)$ is the corresponding variance of traffic. We also assume that the variance functions for both queues are $\sigma_1(t) = 10$, $t \in [2500i, 2500(i+1))$, $i = 0, 2, 4, \dots$, and $\sigma_2(t) = 50$, $t \in [2500i, 2500(i+1))$, $i = 1, 3, 5, \dots$, respectively. The resulting average system delays and another performance measure—the 95th percentile of job delays, are shown in Fig. 8. Note that our policy chooses a smaller window size $K = 11$ for the running median smoother in this case. The goal is to be able to detect early the change of traffic intensities over time, so that the MaxProduct policy can quickly respond by choosing the optimal queue weights accordingly. As can be seen from Fig. 8, our policy significantly outperforms the MaxProduct policies with fixed weight vectors both in terms of the average system delay and in terms of the 95th percentile of job delays.

D. A 3-queue System with Compound Poisson Input Processes

Let's now consider a 3-queue system with 6 admissible service vectors $\vec{\mu}_1 = (0, 0, 6)$, $\vec{\mu}_2 = (4, 0, 3)$, $\vec{\mu}_3 = (6, 0, 0)$, $\vec{\mu}_4 = (4, 5, 0)$, $\vec{\mu}_5 = (0, 8, 0)$, and $\vec{\mu}_6 = (0, 5, 3)$. Analogous to the setup in the 2-queue system, assume the input processes are Poisson with mean interarrival times $\mathcal{T}_1 = 2$, $\mathcal{T}_2 = 2$, $\mathcal{T}_3 = 0.19$ and mean service requirements $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = 1$. The long-term traffic intensities are then $(\rho_1, \rho_2, \rho_3) =$

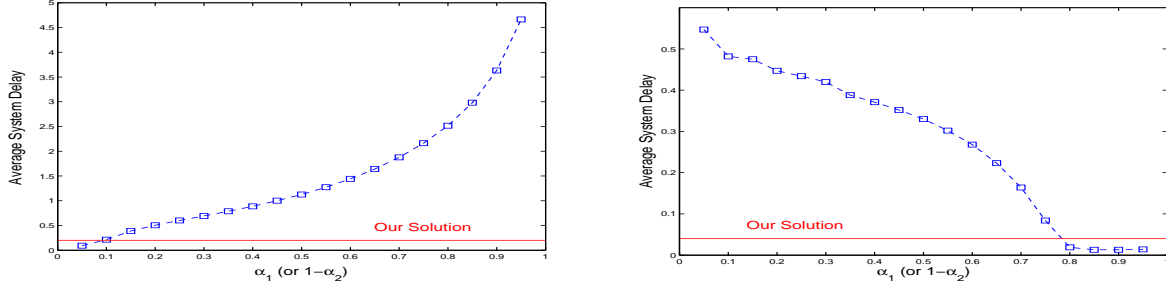


Figure 6: (Left panel): The resulting average system delay of our approach and $\pi_{\vec{\alpha}}$ with possible choices of $\vec{\alpha}$. The input processes are based on the FBM with $H = 0.7$ and the traffic intensities are $(\rho_1, \rho_2) = (2.9, 0.1)$. (Right panel): The resulting average system delay of our approach and $\pi_{\vec{\alpha}}$ with possible choices of $\vec{\alpha}$. The input processes are based on the FBM with $H = 0.7$ and the traffic intensities are $(\rho_1, \rho_2) = (0.5, 3.5)$.

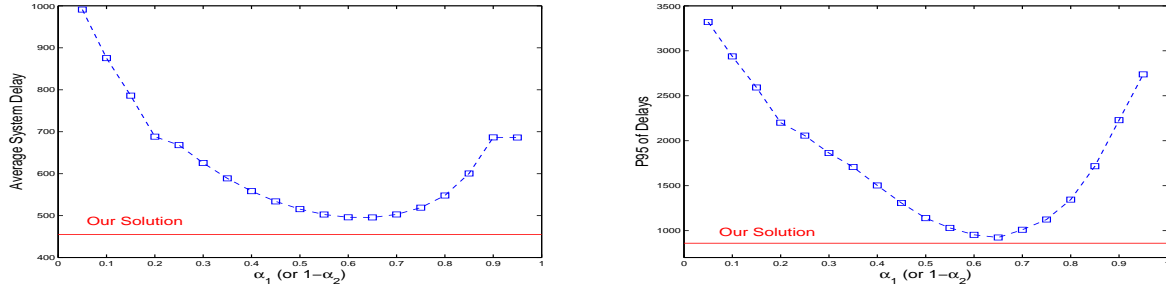


Figure 7: (Left panel): The resulting average system delay of our approach and $\pi_{\vec{\alpha}}$ with all possible choices of $\vec{\alpha}$. The input processes are based on the FBM with $H = 0.7$ with changing rates ρ_q^i described in (22) and (23). (Right panel): The resulting 95th percentile of job delays of our approach and $\pi_{\vec{\alpha}}$ with all possible choices of $\vec{\alpha}$. The input processes are based on the FBM with $H = 0.7$ with changing rates ρ_q^i described in (22) and (23).

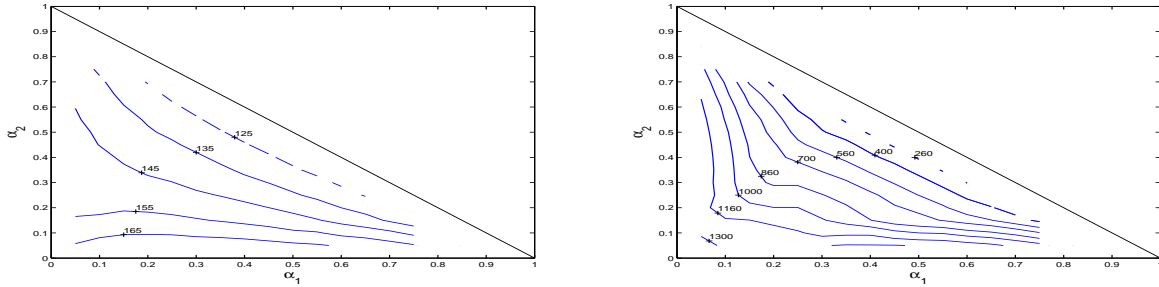


Figure 8: (Left panel): The resulting contour plot of the average system delay for $\pi_{\vec{\alpha}}$ with all possible choices of $\vec{\alpha}$. (Right panel): The resulting contour plot of the 95th percentile of job delays for $\pi_{\vec{\alpha}}$ with all possible choices of $\vec{\alpha}$.

$(0.5, 0.5, 5.3)$. In addition, the length of each time interval chosen to estimate the traffic intensities is $\Delta = 40$, while the window size for constructing the smoothed estimate of the traffic intensities is chosen to be $K = 21$. Computer simulations are then performed to derive the average system delay and the 95th percentile of job delays for the proposed policy and the Max-Product policy $\pi_{\vec{\alpha}}$ with different choices of weights $\vec{\alpha}$. Note that it suffices to look at the weights $(\alpha_1, \alpha_2, \alpha_3)$ satisfying

that $\alpha_1 + \alpha_2 + \alpha_3 = 1$ (which include all possible directions in \mathbb{R}^3). The result for the MaxProduct policy with different choices of weights (α_1, α_2) (and therefore $\alpha_3 = 1 - \alpha_1 - \alpha_2$) is shown in Fig. 8. Numerical results reveal that the minimal average system delay and the 95th percentile of delays are both given around $\vec{\alpha}=(0.35,0.55,0.1)$. The minimal values are 121.97 and 238.1, respectively. In addition, the simulation result shows that our policy has the average system delay

119.38 and the 95th percentile of delays 206.12, which are significantly smaller than the minimal values of that derived from the MaxProduct policy with fixed weights.

VI. CONCLUDING REMARKS AND FURTHER RESEARCH

In this paper, an adaptive strategy was proposed for improving the QoS performance for a class of MaxProduct policies in general switched processing systems. The proposed approach employs a statistical smoothing technique, which allows us to capture well the changes of traffic intensity over time and is based on the nature of the MaxProduct policy for draining out an initial workload in the shortest path. As noted before, this dynamic resource allocation policy is throughput maximizing and does not require a priori knowledge of the input traffic statistics. The simulation results show that it achieves significant improvement in terms of average delay or the 95th percentile of delays for various types of traffic.

We are currently investigating the following topics: (i) the performance of the policy for general non-stationary input processes; and (ii) the computational complexity of obtaining the optimal weight solution in a SPS comprised of a very large number of queues and service configurations. The issues in (ii) are best illustrated by the well-known example—the $N \times N$ input-buffered crossbar switch. Intuitively, the optimal weight solution can also be found by applying the Birkhoff-von Neumann decomposition, that is, to decompose the initial workload (or the estimate of traffic intensity) on the right service vectors. However, finding such a decomposition for large N is an NP-hard problem. One way to reduce the computational complexity is considering a policy which searches “locally” over a small neighborhood of service vectors on each decision epoch and then chooses a better one. However, as shown in [11], such a search procedure will cause higher delay. Our further research will focus on how to utilize all these ideas to improve our proposed approach by considering the trade-off between complexity and delay.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants CCR-0325571 and DMS-0505535 (GM).

References

- [1] M. Armony and N. Bambos, “Queueing Dynamics and Maximal Throughput Scheduling in Switched Processing Systems”, *Queueing Systems: Theory and Applications*, 44(3), pp. 209-252, 2003.
- [2] N. Bambos and G. Michailidis, “Queueing Networks in Random Environments”, *Advances in Applied Probability*, 36, pp. 293-337, 2004.
- [3] C.S. Chang, W. Cheng, and H. Huang, “On Service Guarantees for Input Buffered Crossbar Switches: A Capacity Decomposition Approach by Birkhoff and von Neumann”, *IEEE IWQoS’99*, pp. 79-86, 1999.
- [4] R.L. Cruz, “A Calculus for network delay, Part I: Network elements in isolation”, *IEEE Transactions on Information Theory*, vol. 37, pp. 114-131, 1991.
- [5] Y.C. Hung, “Modeling and Analysis of Stochastic Networks with Shared Resources”, *Ph.D. thesis*, Department of Statistics, The University of Michigan, 2002.
- [6] W.E. Leland, M.S. Taquq, W. Willinger, and D.V. Wilson, “On the self-similar nature of Ethernet traffic (extended version)”, *IEEE/ACM Trans. Networking*, 2(1), pp. 1-15, 1994.
- [7] G. Michailidis, “Optimal Resource Allocation in a Queueing System with Shared Resources”, *Proceedings of the 42nd Conference on Decision and Control*, 2003.
- [8] I. Norros, “On the use of fractional Brownian motion in the theory of connectionless networks”, *IEEE J. Selected Areas Commun.*, 13(6), pp. 953-962, 1995.
- [9] K. Ross and N. Bambos, “Dynamic Quality of Service Control in Packet Switch Scheduling”, *Proceedings of IEEE International Conference on Communications*, 2005.
- [10] K. Ross and N. Bambos, “Optimizing Quality of Service in Packet Switch Scheduling”, *Proceedings of IEEE International Conference on Communications*, 2004.
- [11] K. Ross and N. Bambos, “Local Search Scheduling Algorithms for Maximal Throughput in Packet Switches”, *Proceedings of IEEE INFOCOM*, 2004.
- [12] A.L. Stolyar, “MaxWeight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic”, *Annals of Applied Probability*, 14(1), pp. 1-53, 2004.
- [13] A.L. Stolyar, “Control of End-to-End Delay Tails in a Multiclass Network: LWDF Discipline Optimality”, *Annals of Applied Probability*, 13(3), pp. 1151-1206, 2003.
- [14] L. Tassiulas and P.P. Bhattacharya, “Allocation of Interdependent Resources for Maximal Throughput”, *Stochastic Models*, 16(1), pp. 27-48, 1999.
- [15] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks”, *IEEE Transactions on Automatic Control*, 37(12), pp. 1936-1949, 1992.
- [16] J.W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977.