

# Stable and Utility-Maximizing Scheduling for Stochastic Processing Networks

Libin Jiang and Jean Walrand  
 EECS Department, University of California at Berkeley  
 {ljiang,wlr}@eecs.berkeley.edu

**Abstract**—Stochastic Processing Networks (SPNs) model manufacturing, communication, and service systems. In such a network, service activities require parts and resources to produce other parts. Because service activities compete for resources, a scheduling problem arises. This paper proposes a *deficit maximum weight* (DMW) algorithm to achieve throughput optimality and maximize the net utility of the production. It overcomes the instability problem of Maximum-Weight Scheduling in SPNs.

## I. INTRODUCTION

Stochastic Processing Networks (SPNs) are models of service, processing, communication, or manufacturing systems [1]. In such a network, service activities require parts and resources to produce new parts. Thus, parts flow through a network of buffers served by activities that consume parts and produce new ones. Typically, service activities compete for resources, which yields a scheduling problem. The goal of the scheduling is to maximize some measure of performance of the network, such as the net utility of parts being produced.

As SPNs are more general than queuing networks, one may expect the scheduling that minimizes an average cost such as total waiting time to be complex. Indeed, the optimal scheduling of queuing networks is known only for simple cases, such as serving the longest queue or the Klimov network [2]. For SPNs, one approach has been to consider these networks under heavy-traffic regime [3]. In such a regime, a suitable scheduling may collapse the state space. For instance, when serving the longest queue, under heavy traffic the queue lengths become equal. It is then sometimes possible to analyze the SPN under heavy traffic as in [4]. Using this approach, in [5], the authors prove the asymptotic optimality under heavy traffic of maximum pressure policies for a class of SPNs. It may also happen that the control of the heavy traffic diffusion model is tractable while the original problem is not [7].

Another line of investigation explores a less ambitious formulation of the problem. Instead of considering the Markov decision problem of minimizing an average cost, this approach searches for controls that stabilize the network or that maximize the utility of its flows. This approach has been followed successfully for communication networks.

Tassiulas and Ephremides [8] proposed a maximum weight scheduling algorithm (MWS) that schedules the independent set (non-conflicting nodes) with the maximum sum of queue lengths. The authors prove that this algorithm achieves the maximum possible throughput. The central idea of considering the maximization of the sum of the user utilities is due to [9]. See also [10], [11]. Combining this objective with the scheduling appears in [12], [13]. In these works, it is shown that maximum backpressure algorithms combined with congestion control maximize the total utility of the flows in the network.

This paper follows a similar approach. The objective is to achieve throughput optimality and maximize the total net utility of flows of parts that the network produces. However, the scheme proposed in the paper differs from previous work. For instance, simple examples show that MWS is not stable for some SPNs and that a new approach is needed. The basic difficulty is that the maximum weight and related algorithms are too greedy and may lead some service activities to starve other service activities. Dai and Lin [6] show that MWS is stable in SPNs *if* the network structure satisfies a certain assumption (for example, in a limited class of SPNs where each service activity consumes parts from a single queue). We propose a *deficit maximum weight* (DMW) algorithm that automatically makes certain service activities wait instead of always grabbing the parts they can use, therefore achieving throughput optimality without the assumption in [6].

The paper is organized as follows. Section II illustrates through examples the basic difficulties of scheduling SPNs and the operations of the DMW scheduling algorithm. Section III defines the basic model. Section IV describes the DMW algorithm formally and proves that it stabilizes the network. Section V explains that the algorithm, combined with the control of the input activities, maximizes the sum of the utilities of the network. Section VII provides a number of simulation results to confirm the results of the paper.

## II. EXAMPLES

This section illustrates critical aspects of the scheduling of SPNs on simple examples. Figure 1 shows a SPN with one *input activity* (IA) represented by the shaded circle and four *service activities* (SAs) represented by white circles. SA2 needs one part from queue 2 and produces one part that leaves

the network, similarly for SA4. SA3 needs one part from each of the queues 2, 3 and 4 and produces one part that leaves the network. SA1 needs one part from queue 1 and produces one part which is added to queue 4. Each SA takes one unit of time. There is a dashed line between two SAs if they cannot be performed simultaneously. These conflicts may be due to common resources that the SAs require. The parts arrive at the queues as follows: at even times, IA1 generates one part for each of the queues 1, 2 and 3; at odd times, no part arrives.

One simple scheduling algorithm for this network is as follows. At time 0, buffer the parts that arrive at queues 1, 2 and 3. At time 1, perform SA1 which removes one part from queue 1 and adds one part to queue 4. At time 2, use the three parts in queue 2, 3, 4 to perform SA3 and buffer the new arrivals. Repeat this schedule forever, i.e., perform SA1 and SA3 alternately. This schedule makes the system stable.

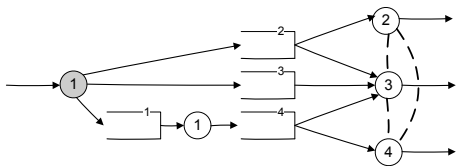


Fig. 1: A network unstable under MWS

Interestingly, the maximum weight algorithm makes this system unstable (in a similar way to a counter example in [6]). By definition, at each time, this algorithm schedules the SAs that maximize the sum of the backpressures. Accordingly, at time 1, one part has arrived in queue 1, 2 and 3 (at time 0). Since queue 4 is empty, SA3 and SA4 cannot be scheduled, so this algorithm schedules SA1 and SA2, after which one part remains in queue 3 and queue 4. At time 2, the algorithm schedules SA4, and buffers new arrivals, after which two parts remain in queue 3, and one part in queue 1 and queue 2. Continuing in this way, the number of parts in queue 3 increases without bound since the algorithm never schedules SA3 and never serves queue 3. (In fact, any work-conserving algorithm leads to the same result in this example.) The deficit maximum weight algorithm that we propose in this paper addresses this instability.

Fig. 2 provides another example of instability, this time due to randomness. There, SA1 processes each part in queue 1 and then produces one part for queue 2 or queue 3, each with probability 0.5. Each activation of SA2 assembles one part from queue 2 and one part from queue 3. Each SA takes one unit of time. If the parts arrive at queue 1 at rate  $\lambda_1 < 1$ , then one would expect the SPN to be able to process these parts. However, the difference between the number of parts that enter the queues 2 and 3 is null recurrent. Thus, no scheduling algorithm can keep the backlogs in the queues 2 and 3 bounded at the same time. In this paper, we are only interested in those networks which are feasible to stabilize.

Figure 3 shows another SPN. IA1 produces one part for queue 1. IA2 produces one part for queue 2 and one part for queue 3. The synchronized arrivals generated by

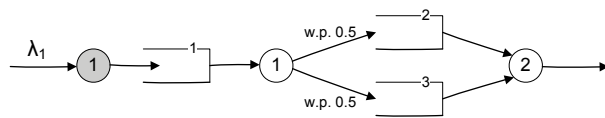


Fig. 2: An infeasible example

IA2 correspond to the ordering of a pair of parts, as one knows that such a pair is needed for SA2. This mechanism eliminates the difficulty encountered in the example of Figure 2. In Figure 3, we say that each IA is “source” of a “flow” of parts (as a generalization of a “flow” in data networks). SA1 and SA2 in this network conflict, as indicated by the dashed line between the SAs. Similarly, SA2 and SA3 conflict. One may consider the problem of scheduling both the IAs (ordering parts) and the SAs to maximize some measure of performance. Our model assumes the appropriate ordering of sets of parts to match the requirements of the SAs.

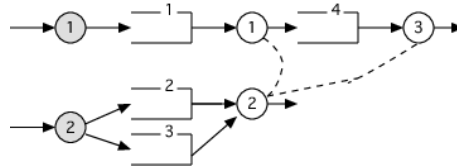


Fig. 3: Arrivals and conflicting service activities

We explain the *deficit maximum weight* (DMW) scheduling algorithm on the example of Figure 1. In that example, we saw that MWS is unstable because it starves SA3. Specifically, MWS schedules SA2 and SA4 before the three queues can accumulate parts for SA3. The idea of DMW is to pretend that certain queues are empty even when they have parts, so that the parts can wait for the activation of SA3. The algorithm is similar to MWS but the weight of each SA is computed from the “virtual queue lengths”  $q_k = Q_k - D_k, \forall k$ . Here,  $Q_k$  is the actual length of queue  $k$  and  $D_k \geq 0$  is called “deficit”.

DMW automatically finds the suitable values of the deficits  $D_k$ . To do this, DMW uses the maximum-weighted schedules without considering whether there are enough input parts available. When the algorithm activates a SA which does not have enough input parts in queue  $k$ , the SA produces fictitious parts, decreases  $q_k$  (which is allowed to be negative) and increases the deficit of queue  $k$ . This algorithm produces the results in Table I, where each column gives the values of  $\mathbf{q}$  and  $\mathbf{D}$  after the activities in a slot. For deficits, only  $D_4$  is shown since the deficits of all other queues are 0. In the table, SA0 means that no SA is scheduled because all the weights of the activities are non-positive. Note that when SA3 is activated for the first time, queue 4 is empty:  $Q_4 = 0$ . Therefore  $q_4$  is decreased to -1,  $D_4$  is increased to 1 and a fictitious part is produced. But since SA1 is activated simultaneously,  $q_4$  becomes 0 after this slot. After that, the sequence (SA0+IA1, SA3+SA1) repeats forever and no more fictitious parts are produced. The key observation is that, although the virtual queue lengths are allowed to become

Activity→	SA0+IA1	SA3+SA1	SA0+IA1	SA3+SA1	...
$q_1$	0	1	0	1	0
$q_2$	0	1	0	1	0
$q_3$	0	1	0	1	0
$q_4$	0	0	0	0	0
$D_4$	0	0	1	1	1

TABLE I: Deficit Maximum Weight scheduling

negative, they remain bounded in this example. Consequently, with proper  $\mathbf{D}$ , the actual queue lengths  $\mathbf{Q} = \mathbf{q} + \mathbf{D}$  are always non-negative, and thus avoid the starvation problem.

### III. BASIC MODEL

For simplicity, assume a time-slotted system. In each slot, a set of input activities (IAs) and service activities (SAs) are activated. Assume that each activity lasts one slot for the ease of exposition. (In section VI, we will discuss the case where different activities have different durations.) There are  $M$  IAs,  $N$  SAs, and  $K$  queues in the network. Each IA, when activated, produces a deterministic number of parts for each of a fixed set of queues. Each SA, when activated, consumes parts from a set of queues and produces parts for another set of queues, and/or some “products” that leave the network.

The set of IAs, SAs and queues are defined to be consistent with the following. (i) Each IA is the “source” of a “flow” of parts, like in Figure 3. In other words, the parts generated by IA  $m$  can be exactly served by activating some SAs and eventually produce a number of products that leave the network. (This will be made more formal later.) Otherwise, it is impossible to stabilize the network. There are  $M$  IAs and  $M$  flows. (ii) Parts in different flows are buffered in separate queues. (iii) A SA  $n$  is associated with a set of input queues  $\mathcal{I}_n$  and a set of output queues  $\mathcal{O}_n$ . Due to the way we define the queues in (ii), different flows are served by *disjoint* sets of SAs. (Even if two SAs in different flows essentially perform the same task, we still label them differently.) Also, a SA is defined only if it is used by some flow.

Each activation of IA  $m$  adds  $a_{k,m}$  parts to queue  $k$ . Define the “input matrix”  $A \in \mathcal{R}^{K \times M}$  where  $A_{k,m} = -a_{k,m}, \forall m, k$ . Each activation of SA  $n$  consumes  $b_{k,n}$  parts from each queue  $k \in \mathcal{I}_n$  (the “input set” of SA  $n$ ), and produces  $b'_{k,n}$  parts that are added to each queue  $k \in \mathcal{O}_n$  (the “output set” of SA  $n$ ), and (possible) a number of final products that leave the network. Assume that  $\mathcal{I}_n \cap \mathcal{O}_n = \emptyset$ . Accordingly, define the “service matrix”  $B \in \mathcal{R}^{K \times N}$ , where  $B_{k,n} = b_{k,n}$  if  $k \in \mathcal{I}_n$ ,  $B_{k,n} = -b'_{k,n}$  if  $k \in \mathcal{O}_n$ , and  $B_{k,n} = 0$  otherwise. Assume that all elements of  $A$  and  $B$  are integers. Also assume that the directed graph that represents the network has no cycle (see, for example, Fig. 1 and Fig. 3).

Let  $\mathbf{a}(t) \in \{0, 1\}^M, t = 0, 1, \dots$  be the “arrival vector” in slot  $t$ , where  $a_m(t) = 1$  if IA  $m$  is activated and  $a_m(t) = 0$  otherwise. Let  $\lambda \in \mathcal{R}^M$  be the vector of (average) arrival rates. Let  $\mathbf{x}(t) \in \{0, 1\}^N$  be the “service vector” in slot  $t$ , where  $x_n(t) = 1$  if SA  $n$  is activated and  $x_n(t) = 0$  otherwise. Let  $\mathbf{s} \in \mathcal{R}^N$  be a vector of (average) service rates.

Point (i) above means that, for any activation rate  $\lambda_m > 0$

of flow  $m$ , there exists  $\mathbf{s}_m \in \mathcal{R}^N$  such that

$$A_m \cdot \lambda_m + B \cdot \mathbf{s}_m = \mathbf{0} \quad (1)$$

where  $A_m$  is the  $m$ 'th column of  $A$ . The vector  $\mathbf{s}_m$  is the service rate vector for flow  $m$  that can serve  $\lambda_m$ . We also make the reasonable assumption that  $\mathbf{s}_m$  is unique given  $\lambda_m$ , i.e., there is only one way to serve the arrivals. Summing up (1) over  $m$  gives

$$A \cdot \lambda + B \cdot \mathbf{s} = \mathbf{0}. \quad (2)$$

where  $\mathbf{s} = \sum_m \mathbf{s}_m \succ \mathbf{0}$ . (Note that since each flow is associated with a separate set of queues and SAs, equation (2) also implies (1) for all  $m$ .) By assumption,  $\mathbf{s}$  is unique given  $\lambda$ , so we also write  $\mathbf{s}$  in (2) as  $\mathbf{s}(\lambda)$ .

Due to resource sharing constraints among the SAs, not all SAs can be performed simultaneously at a given time. Assuming that all queues have enough parts such that any SA can be performed, let  $\tilde{\mathbf{x}} \in \{0, 1\}^N$  be a feasible service vector, and  $\mathcal{X}$  be the set of such  $\tilde{\mathbf{x}}$ 's. (We also call  $\tilde{\mathbf{x}}$  an independent set since the active SAs in  $\tilde{\mathbf{x}}$  can be performed without conflicts.) Denote by  $\Lambda$  be the convex hull of  $\mathcal{X}$ , i.e.,

$$\Lambda := \{\mathbf{s} | \exists \mathbf{p} \succ \mathbf{0} : \sum_{\tilde{\mathbf{x}} \in \mathcal{X}} p_{\tilde{\mathbf{x}}} = 1, \mathbf{s} = \sum_{\tilde{\mathbf{x}} \in \mathcal{X}} (p_{\tilde{\mathbf{x}}} \cdot \tilde{\mathbf{x}})\}$$

and let  $\Lambda^\circ$  be the interior of  $\Lambda$ . (That is, for any  $\mathbf{s} \in \Lambda^\circ$ , there is a ball  $\tilde{B}$  centered at  $\mathbf{s}$  with radius  $r > 0$  such that  $\tilde{B} \subset \Lambda$ .)

We say that  $\lambda$  is *feasible* iff  $\mathbf{s}(\lambda) \in \Lambda$ ;  $\lambda$  is *strictly feasible* iff  $\mathbf{s}(\lambda) \in \Lambda^\circ$ .

In a more general setting, the output parts of a certain SA can split and go to more than one output sets. The split can be random or deterministic. For example, in a hospital, after a patient is diagnosed, he goes to a certain room based on the result. A probabilistic model for this is that the patients go to different rooms with certain probabilities after the SA (i.e., the diagnosis). The split can also be deterministic. For example, in manufacturing, the output parts of a SA may be put into two different queues alternately.

In both cases, we can define the element  $B_{k,n}$  in the matrix  $B$  to be the average rate that SA  $n$  consumes (or adds) parts from (to) queue  $k$ . However, note that in the random case, it may not be feasible to stabilize all queues by any algorithm, even if there exist average rates satisfying (1). Fig. 2 described earlier is such an example. For simplicity, here we mainly consider networks without splitting.

### IV. DMW SCHEDULING

In this section we consider the scheduling problem with strictly feasible arrival rates  $\lambda$ . We first describe the DMW algorithm and then show its throughput optimality.

Let the actual queue lengths at time  $t$  be  $Q_k(t), k = 1, 2, \dots, K$ . Define a “deficit”  $D_k(t) \geq 0$ . DMW uses the “virtual queue length”  $q_k(t) = Q_k(t) - D_k(t)$  to compute the schedule in each slot.

#### DMW (Deficit Maximum Weight) Scheduling

Initially (at time 0), set  $\mathbf{q}(0) = \mathbf{Q}(0) = \mathbf{D}(0) = \mathbf{0}$ . Clearly,  $\mathbf{Q}(0) = \mathbf{q}(0) + \mathbf{D}(0)$ .

(i) Update of virtual queues  $\mathbf{q}(t)$ : In each time slot  $t = 0, 1, 2, \dots$ , the set of SAs with the maximal backpressure is scheduled:

$$\mathbf{x}^*(t) \in \arg \max_{\mathbf{x} \in \mathcal{X}} \mathbf{d}^T(t) \cdot \mathbf{x}. \quad (3)$$

where  $\mathbf{d}(t) \in \mathcal{R}^N$  is the vector of backpressure, defined as

$$\mathbf{d}(t) = B^T \mathbf{q}(t), \quad (4)$$

and  $\mathcal{X}$  is the set of independent sets including non-maximal ones. Also, for any SA  $n$ , we require that  $x_n^*(t) = 0$  if  $d_n(t) \leq 0$ .

Recall that an independent set is a set of SAs that can be performed simultaneously *assuming that all input queues have enough parts*. So, it is possible that SA  $n$  is scheduled (i.e.,  $x_n^*(t) = 1$ ) even if there are not enough parts in some input queues of SA  $n$ . In this case, SA  $n$  is activated as a “null activity” (to be further explained in (ii)). Then, update  $\mathbf{q}$  as

$$\mathbf{q}(t+1) = \mathbf{q}(t) - A \cdot \mathbf{a}(t) - B \cdot \mathbf{x}^*(t) \quad (5)$$

where (recall that)  $\mathbf{a}(t)$  is the vector of actual arrivals in slot  $t$  (where the  $m$ 'th element  $a_m(t)$  corresponds to IA  $m$ ). In this paper,  $\mathbf{x}^*(t)$  and  $\mathbf{x}^*(\mathbf{q}(t))$  are interchangeable.

(5) can also be written as

$$q_k(t+1) = q_k(t) - \mu_{out,k}(t) + \mu_{in,k}(t), \forall k$$

where  $\mu_{out,k}(t)$  and  $\mu_{in,k}(t)$  are the number of parts coming out of or into virtual queue  $k$  in slot  $t$ , expressed below. (We use  $v^+$  and  $v^-$  to denote the positive and negative part of  $v$ . That is,  $v^+ = \max\{0, v\}$  and  $v^- = \max\{0, -v\}$ . Clearly,  $v = v^+ - v^-$ .)

$$\begin{aligned} \mu_{out,k}(t) &= \sum_{n=1}^N [B_{k,n}^+ x_n^*(t)] \\ \mu_{in,k}(t) &= \sum_{m=1}^M [A_{k,m}^- a_m(t)] + \sum_{n=1}^N [B_{k,n}^- x_n^*(t)]. \end{aligned}$$

(ii) Update of actual queues  $\mathbf{Q}(t)$  and deficits  $\mathbf{D}(t)$ : If SA  $n$  is scheduled in slot  $t$  but there are not enough parts in some of its input queues (or some input parts are fictitious, further explained below), SA  $n$  is activated as a null activity. Although the null activity  $n$  does not actually consume or produce parts, parts are removed from the input queues and fictitious parts are added to the output queues as if SA  $n$  was activated normally. So the actual queue length

$$Q_k(t+1) = [Q_k(t) - \mu_{out,k}(t)]_+ + \mu_{in,k}(t). \quad (6)$$

Then the deficit is computed as

$$D_k(t+1) = Q_k(t+1) - q_k(t+1). \quad (7)$$

*Remark:* The key idea of DMW is to augment the “state” by including the virtual queues  $\mathbf{q}(t)$ , and then show the relationship between the virtual queues and actual queues. In comparison, reference [14] addresses another class of instability problems by perturbing the Lyapunov functions.

The following is a useful property of  $D_k(t)$ .

*Lemma 1:*  $D_k(t)$  is non-decreasing with  $t$ , and satisfies

$$D_k(t+1) = D_k(t) + [\mu_{out,k}(t) - Q_k(t)]_+.$$

*Proof:* By (7), (5) and (6), we have

$$\begin{aligned} D_k(t+1) &= Q_k(t+1) - q_k(t+1) \\ &= [Q_k(t) - \mu_{out,k}(t)]_+ - [q_k(t) - \mu_{out,k}(t)] \\ &= Q_k(t) - \mu_{out,k}(t) + [\mu_{out,k}(t) - Q_k(t)]_+ \\ &\quad - [q_k(t) - \mu_{out,k}(t)] \\ &= D_k(t) + [\mu_{out,k}(t) - Q_k(t)]_+, \end{aligned} \quad (8)$$

which also implies that  $D_k(t)$  is non-decreasing with  $t$ . ■

*Proposition 1:* If  $\|\mathbf{q}(t)\|^2 \leq G$  at all time  $t$  for some constant  $G > 0$ , then

(i)  $\mathbf{D}(t)$  is bounded. Also, only a finite number of null activities occur. So in the long term the null activities do not affect the average throughput.

(ii)  $\mathbf{Q}(t)$  is bounded.

*Proof:* Since  $\|\mathbf{q}(t)\|^2 \leq G$ , we have  $-G' \leq q_k(t) \leq G'$ ,  $\forall k, t$  where  $G' := \lceil \sqrt{G} \rceil$ . We claim that  $D_k(t) \leq G' + \mu_{out}$ ,  $\forall k, t$  where  $\mu_{out}$  is the maximum number of parts that could leave a queue in one slot. By the definition of the DMW algorithm,  $D_k(t)$  is non-decreasing with  $t$  and initially  $D_k(t) = 0$ . Suppose to the contrary that  $D_k(t)$  is above  $G' + \mu_{out}$  for some  $k$  and  $t$ . Then there exists  $t'$  which is the first time that  $D_k(t')$  is above  $G' + \mu_{out}$ . In other words,  $D_k(t') > G' + \mu_{out}$  and  $D_k(t' - 1) \leq G' + \mu_{out}$ .

By (8) and (7), we have

$$\begin{aligned} D_k(t+1) &= D_k(t) + [\mu_{out,k}(t) - Q_k(t)]_+ \\ &= D_k(t) + \max\{0, \mu_{out,k}(t) - Q_k(t)\} \\ &= \max\{D_k(t), D_k(t) + \mu_{out,k}(t) - Q_k(t)\} \\ &= \max\{D_k(t), -q_k(t) + \mu_{out,k}(t)\} \end{aligned}$$

So  $D_k(t') = \max\{D_k(t' - 1), -q_k(t' - 1) + \mu_{out,k}(t' - 1)\}$ . Since  $q_k(t' - 1) \geq -G'$ ,  $\mu_{out,k}(t) \leq \mu_{out}$ , we have  $D_k(t') \leq G' + \mu_{out}$ . This leads to a contradiction. Therefore,  $D_k(t) \leq G' + \mu_{out}$ ,  $\forall t, k$ .

Note that when a queue underflow (i.e., when  $\mu_{out,k}(t) > Q_k(t)$  for some  $k, t$ ) occurs,  $D_k$  is increased. Also, the increase of  $D_k$  is a positive integer. Since  $\mathbf{D}(0) = \mathbf{0}$ ,  $\mathbf{D}(t)$  is non-decreasing and remains bounded for all  $t$ , the number of queue underflows must be finite. Since we have assumed that the directed graph which represents the network has no cycle, it is clear that each underflow only “pollutes” a finite number of final outputs (i.e., the products). Therefore, in the long term the queue underflows (and the resulting null activities) do not affect the average throughput.

Part (ii):  $Q_k(t) = q_k(t) + D_k(t) \leq 2G' + \mu_{out}$ ,  $\forall k, t$ . ■

In section IV-A we will show that  $\mathbf{q}(t)$  is bounded under certain conditions on the arrivals. By Proposition 1,  $\mathbf{Q}(t)$  is bounded and the maximal throughput is achieved. But before that, we need to identify some useful properties of the system. Our analysis differs from existing analysis of MWS-like algorithms, e.g., in [8], [12], since we allow  $q_k(t)$  to be negative, and an activity generally involves multiple input and output queues.

*Lemma 2:* In DMW, if  $\mathbf{d}(t) = \mathbf{0}$  at some time  $t$ , then  $\mathbf{q}(t) = \mathbf{0}$ .

*Proof:* Let  $z_m(t)$  and  $w_n(t)$  be, respectively, the number of times that IA  $m$  and SA  $n$  have been performed until time  $t$ . Write  $\mathbf{z}(t)$  and  $\mathbf{w}(t)$  as the corresponding vectors. Using  $\mathbf{q}(0) = \mathbf{0}$  and equation (5), we have

$$\mathbf{q}(t) = -A \cdot \mathbf{z}(t) - B \cdot \mathbf{w}(t). \quad (9)$$

For  $\lambda_m = 1$ , there exists  $\mathbf{s}_m = \mathbf{s}'_m$  such that (1) holds. So  $A_m = -B \cdot \mathbf{s}'_m$ . Using this and (9),

$$\begin{aligned} \mathbf{q}(t) &= \sum_m [B \cdot \mathbf{s}'_m z_m(t)] - B \cdot \mathbf{w}(t) \\ &= B \cdot \mathbf{v} \end{aligned}$$

where  $\mathbf{v} := \sum_m [\mathbf{s}'_m z_m(t)] - \mathbf{w}(t)$ . By assumption,

$$\mathbf{d}(t) = B^T \mathbf{q}(t) = B^T B \cdot \mathbf{v} = \mathbf{0}.$$

Thus,  $\mathbf{v}^T B^T B \cdot \mathbf{v} = \|B \cdot \mathbf{v}\|^2 = 0$ . So,  $B \cdot \mathbf{v} = \mathbf{q}(t) = \mathbf{0}$ . ■

*Remark:* We have used the fact that for any  $t$ ,  $\mathbf{q}(t)$  is always in the subspace  $\mathcal{B} := \{\mathbf{u} | \mathbf{u} = B \cdot \mathbf{v} \text{ for some } \mathbf{v}\}$ .

*Lemma 3:* Assume that  $\lambda$  is a strictly feasible, i.e.,  $\exists \mathbf{y} \in \Lambda^\circ$  such that

$$A \cdot \lambda + B \cdot \mathbf{y} = \mathbf{0}. \quad (10)$$

Then there exists  $\delta > 0$  such that for any  $\mathbf{q}$  satisfying  $\mathbf{q} \in \mathcal{B}$ ,

$$q^T B \cdot [\mathbf{x}^*(\mathbf{q}) - \mathbf{y}] \geq \delta \|\mathbf{q}\|. \quad (11)$$

where

$$\mathbf{x}^*(\mathbf{q}) \in \arg \max_{\mathbf{x} \in \mathcal{X}} q^T B \cdot \mathbf{x}. \quad (12)$$

*Proof:* Since  $\mathbf{y} \in \Lambda^\circ$ ,  $\exists \sigma' > 0$  such that  $\mathbf{y}' \in \Lambda$  for any  $\mathbf{y}'$  satisfying  $\|\mathbf{y}' - \mathbf{y}\| \leq \sigma'$ .

For any  $\hat{\mathbf{q}}$  satisfying  $\|\hat{\mathbf{q}}\| = 1, \hat{\mathbf{q}} \in \mathcal{B}$ , by Lemma 2, we have  $\hat{\mathbf{d}} := B^T \hat{\mathbf{q}} \neq \mathbf{0}$ . Also,  $\|B^T \hat{\mathbf{q}}\| \geq \hat{\sigma} := \min_{\|\mathbf{q}'\|=1, \mathbf{q}' \in \mathcal{B}} \|B^T \mathbf{q}'\| > 0$ . Choose  $\hat{\epsilon} > 0$  (which may depend on  $\hat{\mathbf{q}}$ ) so that  $\|\hat{\epsilon} \cdot B^T \hat{\mathbf{q}}\| = \sigma'$ . Then,  $\mathbf{y} + \hat{\epsilon} \cdot B^T \hat{\mathbf{q}} \in \Lambda$ . Also, (12) implies that  $\mathbf{x}^*(\mathbf{q}) \in \arg \max_{\mathbf{x} \in \mathcal{X}} q^T B \cdot \mathbf{x}$ . So,  $\hat{\mathbf{q}}^T B \cdot \mathbf{x}^*(\hat{\mathbf{q}}) \geq \hat{\mathbf{q}}^T B \cdot [\mathbf{y} + \hat{\epsilon} \cdot B^T \hat{\mathbf{q}}] = \hat{\mathbf{q}}^T B \cdot \mathbf{y} + \hat{\epsilon} \cdot \|B^T \hat{\mathbf{q}}\|^2 \geq \hat{\mathbf{q}}^T B \cdot \mathbf{y} + \hat{\sigma} \cdot \sigma'$ . Let  $\delta := \hat{\sigma} \cdot \sigma'$ . Then

$$\min_{\|\hat{\mathbf{q}}\|=1, \hat{\mathbf{q}} \in \mathcal{B}} \hat{\mathbf{q}}^T B \cdot [\mathbf{x}^*(\hat{\mathbf{q}}) - \mathbf{y}] \geq \delta. \quad (13)$$

Consider any  $\mathbf{q} \neq \mathbf{0}$ . Let  $\hat{\mathbf{q}} := \mathbf{q}/\|\mathbf{q}\|$ , then  $\|\hat{\mathbf{q}}\| = 1$ . Note that if  $\mathbf{x}^*(\hat{\mathbf{q}}) \in \arg \max_{\mathbf{x} \in \mathcal{X}} \hat{\mathbf{q}}^T B \cdot \mathbf{x}$ , then  $\mathbf{x}^*(\hat{\mathbf{q}}) \in \arg \max_{\mathbf{x} \in \mathcal{X}} q^T B \cdot \mathbf{x}$  by linearity, so  $q^T B \cdot \mathbf{x}^*(\hat{\mathbf{q}}) = q^T B \cdot \mathbf{x}^*(\mathbf{q})$ . Therefore  $q^T B \cdot [\mathbf{x}^*(\mathbf{q}) - \mathbf{y}] = q^T B \cdot [\mathbf{x}^*(\hat{\mathbf{q}}) - \mathbf{y}] = \|\mathbf{q}\| \cdot \hat{\mathbf{q}}^T B \cdot [\mathbf{x}^*(\hat{\mathbf{q}}) - \mathbf{y}] \geq \delta \|\mathbf{q}\|$ , proving (11). If  $\mathbf{q} = \mathbf{0}$ , then (11) holds trivially. ■

#### A. Arrivals that are smooth enough

Recall that  $\lambda$  is strictly feasible and (10) is satisfied. First consider a simple case when the arrival rates are “almost constant” at  $\lambda$ . Specifically, assume that  $a_m(t) = \lfloor \lambda_m \cdot (t+1) \rfloor - \lfloor \lambda_m \cdot t \rfloor, \forall m, t$ . Then  $\sum_{\tau=0}^{t-1} a_m(\tau) = \lfloor \lambda_m \cdot t \rfloor \approx \lambda_m \cdot t, \forall t$ , so that the arrival rates are almost constant. Later, we all show that  $\mathbf{q}(t)$  is bounded under such arrivals. By Proposition 1,  $\mathbf{Q}(t)$  is bounded and the maximal throughput is achieved.

However, since the “almost constant” assumption is quite strong in practice, it is useful to relax it and consider more

general arrival processes. In particular, consider the following (mild) smoothness condition.

**Condition 1:** There exists  $\sigma > 0$  and a positive integer  $T$  such that for all  $l = 0, 1, 2, \dots, \tilde{\mathbf{a}}_l + \sigma \cdot \mathbf{1}$  and  $\tilde{\mathbf{a}}_l - \sigma \cdot \mathbf{1}$  are feasible vectors of arrival rates, where

$$\tilde{\mathbf{a}}_l := \sum_{\tau=l \cdot T}^{(l+1) \cdot T - 1} \mathbf{a}(\tau) / T \quad (14)$$

is the vector of average arrival rates in the  $l$ 'th time window of length  $T$ . In other words, there exists a large enough time window  $T$  such as the  $\tilde{\mathbf{a}}_l$  is “uniformly” strictly feasible.

*Remark:* Note that  $\tilde{\mathbf{a}}_l$  can be very different for different  $l$ 's. That is,  $\tilde{\mathbf{a}}_l, l = 0, 1, \dots$  do not need to be all close to a certain strictly feasible  $\lambda$ .

*Theorem 1:* Under Condition 1,  $\mathbf{q}(t)$  is bounded for all  $t$ . Therefore (i) and (ii) in Prop. 1 hold.

The proof is in Appendix A.

*Corollary 1:* With the “almost constant” arrivals,  $\mathbf{q}(t)$  is bounded for all  $t$ .

*Proof:* Since  $\sum_{\tau=0}^{t-1} a_m(\tau) = \lfloor \lambda_m \cdot t \rfloor$ , we have  $|\sum_{\tau=0}^{t-1} a_m(\tau) - \lambda_m \cdot t| \leq 1, \forall t$ . So  $|\sum_{\tau=l \cdot T}^{(l+1) \cdot T - 1} a_m(\tau) / T - \lambda_m| = (1/T) \cdot |\sum_{\tau=0}^{(l+1) \cdot T - 1} a_m(\tau) - \lambda_m \cdot (l+1)T| - |\sum_{\tau=0}^{l \cdot T - 1} a_m(\tau) - \lambda_m lT| \leq 2/T$ . Since  $\lambda$  is strictly feasible, there exists a large enough  $T$  to satisfy Condition 1. ■

#### B. More random arrivals

Assume that  $a_m(t) \in \mathcal{Z}_+$  is a random variable with bounded support, and satisfies

$$E(a_m(t)) = \lambda_m, \forall t. \quad (15)$$

For simplicity, also assume that the random variables  $\{a_m(t), m = 1, 2, \dots, M, t = 0, 1, 2, \dots\}$  are independent. (This assumption, however, can be easily relaxed.) Suppose that the vector  $\lambda$  is strictly feasible.

In general, this arrival process does not satisfy the smoothness condition (although when  $T$  is large,  $\sum_{\tau=t}^{t+T-1} \mathbf{a}(\tau) / T$  is close to  $\lambda$  with high probability). With such arrivals, it is not difficult to show that  $\mathbf{q}(t)$  is stable, but may not be bounded. As a result, the deficits  $\mathbf{D}(t)$  may increase without bound. In this case, we show that the system is still “rate stable”, in the sense that in the long term, the average output rates of the final products converge to the optimum output rates (with probability 1). The intuitive reason is that as  $\mathbf{D}(t)$  becomes very large, the probability of generating fictitious parts approaches 0.

*Theorem 2:* With the arrival process defined above, the system is “rate stable”.

The formal proof is given in Appendix B.

Although the system is throughput optimum, with  $\mathbf{D}(t)$  unbounded, the actual queue lengths  $\mathbf{Q}(t) = \mathbf{q}(t) + \mathbf{D}(t)$  may become large when  $\mathbf{D}(t)$  is large. An alternative to avoid large  $\mathbf{Q}(t)$  is to set an upper bound of  $D_k(t)$ , denoted by  $\bar{D}$ . In this alternative, we do not increase  $D_k(t)$  once it hits  $\bar{D}$ . But  $\mathbf{q}(t)$  still evolves according to part (i) of the DMW algorithm. Let the actual queue length be  $Q_k(t) = [q_k(t) + D_k(t)]_+$ . Fictitious parts are generated in slot  $t$  as long as

$q_k(t) - \mu_{out,k}(t) < -D_k(t)$  (or,  $Q_k(t) - \mu_{out,k}(t) < 0$ ). Given a  $\bar{D}$ , one expects that the output rates are lower than the optimum in general, since fictitious parts are generated with a certain probability after  $D_k(t)$  first hits  $\bar{D}$ . But one can make the probability arbitrarily close to 0 by choose a large enough  $\bar{D}$ . The proof is similar to that of Theorem 2 and is not included here.

## V. UTILITY MAXIMIZATION

Assume that for each IA  $m$ , there is a ‘‘reward function’’  $v_m(f_m)$  (where  $f_m$  is the activation rate), and a cost function  $c_m f_m$ , where  $c_m$  is the cost of the input materials of IA  $m$  per unit rate. Define the utility function as  $u_m(f_m) := v_m(f_m) - c_m f_m$ . Let  $\mathbf{f} \in \mathcal{R}^M$  be the vector of input activation rates. Assume that  $u_m(\cdot)$  is increasing and concave. The joint scheduling and congestion control algorithm (or ‘‘utility maximization algorithm’’) works as follows.

### Utility Maximization Algorithm

Initially let  $\mathbf{q}(0) = \mathbf{Q}(0) = \mathbf{0}$ . In each time slot  $t = 0, 1, 2, \dots$ , besides DMW Scheduling (3), i.e.,

$$\mathbf{x}^*(t) \in \arg \max_{\mathbf{x} \in \mathcal{X}} d^T(t) \cdot \mathbf{x},$$

IA  $m$  chooses the input rate

$$f_m(\mathbf{q}(t)) := \arg \max_{0 \leq f \leq 1} \{V \cdot u_m(f) + \mathbf{q}(t)^T A_m f\} \quad (16)$$

where  $V > 0$  is a constant, and  $A_m$  is the  $m$ 'th column of  $A$ . Then, update the virtual queues as

$$\mathbf{q}(t+1) = \mathbf{q}(t) - A \cdot \mathbf{f}(\mathbf{q}(t)) - B \cdot \mathbf{x}^*(t) \quad (17)$$

Since  $f_m(\mathbf{q}(t))$  in general is not integer, we let  $a_m(t) = \lfloor F_m(t+1) \rfloor - \lfloor F_m(t) \rfloor$ , where  $F_m(t) := \sum_{\tau=0}^{t-1} f_m(\mathbf{q}(\tau))$ . And update the actual queues in the same way as (6).

*Theorem 3:* With the above algorithm,  $\mathbf{q}(t)$  and  $\mathbf{Q}(t)$  are bounded. Also, there are at most a finite number of null activities which do not affect the long term throughput.

The proof is in Appendix C.

The following is a performance bound of the utility maximization algorithm.

*Theorem 4:* We have

$$\sum_m u_m(\tilde{f}_m) \geq U^* - c/V \quad (18)$$

where  $\tilde{f}_m := \liminf_{T \rightarrow \infty} \sum_{t=0}^{T-1} f_m(\mathbf{q}(t))$ ,  $U^*$  is the optimal total utility, and  $c > 0$  is a constant defined in (20). That is, a larger  $V$  leads to better a lower bound of the achieved utility (at the cost of larger queue lengths).

The proof is similar to that in [12], and is given in Appendix D.

## VI. EXTENSIONS

In the above, we have assumed that each activity lasts one slot for the ease of exposition. Our algorithms can be extended to the case where different activities have different durations under a particular assumption. The assumption is

that each activity can be suspended in the middle and resumed later. If so, we can still use the above algorithm which re-computes the maximum weight schedule in each time slot. The only difference is that the activities performed in one time slot may not be completed at the end of the slot, but are suspended and to be continued in later slots. (The above assumption was also made in the ‘‘preempted’’ networks in [6]. There, whenever a new schedule is computed, the ongoing activities are suspended, or ‘‘preempted’’.)

In this case, the algorithms are adapted in the following way. The basic idea is the same as before. That is, we run the system according to the virtual queues  $\mathbf{q}(t)$ . Let the elements in matrices  $A$  and  $B$  be the average rates of consuming (or producing) parts per slot from (or to) different queues. Even if an activity is not completed in one slot, we still update the virtual queues  $\mathbf{q}(t)$  according to the above average rates. That is, we view the parts in different queues as fluid and  $\mathbf{q}(t)$  reflects the amount of fluid at each queue. However, only when an activity is completed, the actual parts are removed from or added to the output queues. Note that when an activity is suspended, all parts involved in the activity are frozen and are not available to other activities. When there are not enough parts in the queues to perform a scheduled activity, fictitious parts are used instead (and the corresponding deficits are increased).

On the other hand, if each activity cannot be suspended in the middle once it is started, then one possible scheme is to use long time slots in our algorithms. In slot  $t$ , each SA  $n$  with  $x_n^*(t) = 1$  is activated as many times as possible. When each slot is very long, the wasted time during the slot becomes negligible, so the algorithm approximates the maximal throughput (with the cost of longer delay). Without using long slots, the non-preemptive version of the maximal-pressure algorithm proposed in [6] is not throughput-optimum in general, but it IS under a certain resource-sharing constraints [6].

## VII. SIMULATIONS

### A. DMW Scheduling

We simulate a network similar to Fig. 1, but with a different input matrix  $A$  and service matrix  $B$  below.

$$A = \begin{bmatrix} -3 \\ -2 \\ -1 \\ 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 2 & 1 \end{bmatrix}$$

It is easy to check that if  $\lambda_1 = 1/3$ , we have  $A \cdot \lambda_1 + B \cdot \mathbf{s} = \mathbf{0}$  where  $\mathbf{s} := [1, 1/3, 1/3, 1/3]^T \in \Lambda$  (and  $\mathbf{s}$  is unique). So, any  $\lambda_1 \in (0, 1/3)$  is strictly feasible.

In the simulation, IA1 is activated in slot  $5k, k = 0, 1, 2, \dots$ . So the input rate  $\lambda_1 = 1/5$  which is strictly feasible. Since SA 3 requires enough parts from several queues to perform, it is not difficult to see that normal MWS fails to stabilize queue 3. Fig. 4 shows that DMW stabilizes all queues and have bounded deficits.

Now we make a change to the arrival process. In time slot  $4k, k = 0, 1, 2, \dots$ , IA 1 is independently activated with

probability 0.8. As a result, the expected arrival rate is strictly feasible and also satisfies the smoothness condition (Condition 2) with  $T = 4$ . Fig. 5 shows that our algorithm stabilizes all queues. As expected,  $D_k(t)$  stops increasing after some time since  $q(t)$  is bounded.

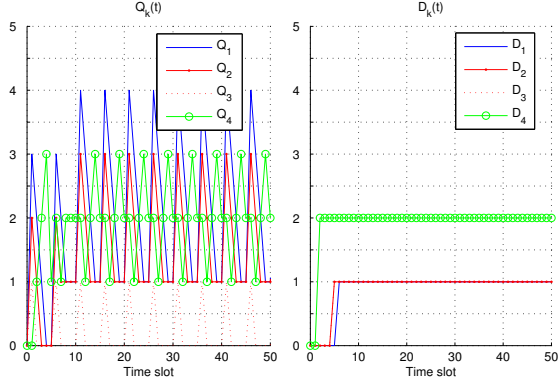


Fig. 4: DMW scheduling (with deterministic arrivals): actual queue lengths  $Q(t)$  and deficits  $D(t)$

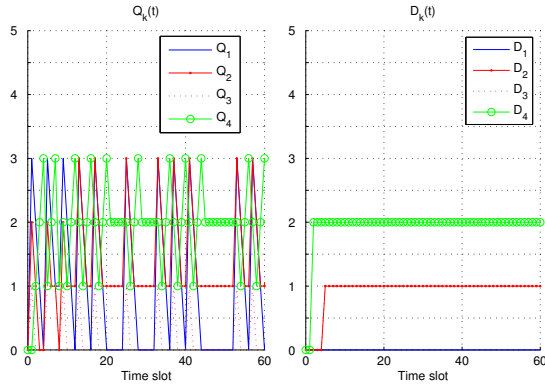


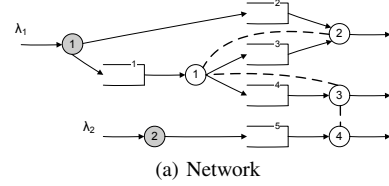
Fig. 5: DMW Scheduling (with random arrivals): actual queue lengths  $Q(t)$  and deficits  $D(t)$

### B. Utility Maximization

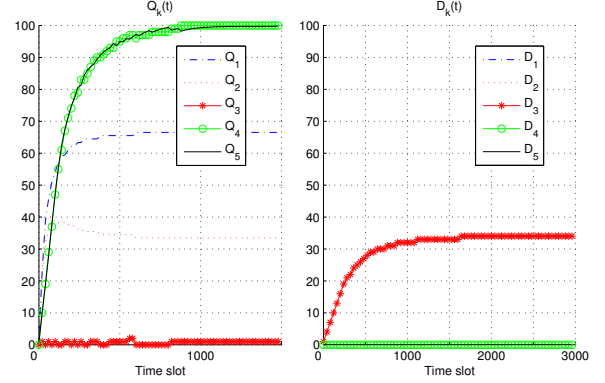
Consider the network in Fig. 6 (a). The utility functions of both flows are  $u_m(\cdot) = \log(\cdot)$ . We simulate the network with the utility maximization algorithm with  $V = 50$ . Fig. 6 (b) shows  $Q_k(t)$  and  $D_k(t)$ . As expected,  $D_k(t)$  stops increasing after some time due to the boundedness of  $q_k(t)$ . The average throughput of flow 1 and 2 is 0.4998 and 0.4998, which are very close to the theoretical optimal throughput computed by solving the utility maximization problem numerically.

## VIII. CONCLUSION

We have proposed a ‘‘Deficit Maximum Weight’’ (DMW) scheduling algorithm for SPNs. It has been shown that the algorithm overcomes the instability issue of the Maximum Weight algorithm that arises in general SPNs. Under different assumptions of the arrival processes, DMW achieves the maximum throughput in different senses. We have also



(a) Network



(b) Actual queue lengths  $Q(t)$  and deficits  $D(t)$

Fig. 6: Utility Maximization

combined DMW scheduling with input rate control in order to approach the maximum utility of the SPN.

The performance of DMW could be further improved. For example, with null activities, a queue may receive fictitious parts in some slots and drop normal parts in other slots. So, one could store the dropped parts and replace the fictitious parts later when they arrive. The performance improvement with such schemes is interesting for future research.

The current algorithm achieves bounded queues when the arrival processes are smooth enough, and guarantees rate stability otherwise. Rate stability is a weaker form of stability than positive recurrence of the queues. We are also interested to enhance the current algorithm or design new algorithms to achieve positive recurrence under such arrival processes.

## APPENDIX

### A. Proof of Theorem 1

To analyze the queue dynamics, consider the Lyapunov function  $L(\mathbf{q}(t)) = \|\mathbf{q}(t)\|^2$ . We have

$$\begin{aligned} \Delta(\mathbf{q}(t)) &:= L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) \\ &= \|\mathbf{q}(t) - A \cdot \mathbf{a}(t) - B \cdot \mathbf{x}^*(\mathbf{q}(t))\|^2 - \|\mathbf{q}(t)\|^2 \\ &= -\mathbf{q}(t)^T A \cdot \mathbf{a}(t) - \mathbf{q}(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) \\ &\quad + \|A \cdot \mathbf{a}(t) + B \cdot \mathbf{x}^*(\mathbf{q}(t))\|^2 \\ &\leq -\mathbf{q}(t)^T A \cdot \mathbf{a}(t) - \mathbf{q}(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) + c \end{aligned} \quad (19)$$

where  $c > 0$  is a constant, defined as

$$c := \sum_k (\mu_{k,in}^2 + \mu_{k,out}^2) \quad (20)$$

where  $\mu_{k,in}$ ,  $\mu_{k,out}$  are, respectively, the maximum amount of parts that can enter or leave queue  $k$  in one time slot.

*Lemma 4:* Assume that  $\mathbf{q}(0) = \mathbf{0}$ . If for any  $t$ ,

$$L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) \leq -\delta\|\mathbf{q}(t)\| + c \quad (21)$$

where  $\delta > 0$  is a constant, then  $\mathbf{q}(t)$  is always bounded. In particular,  $L(\mathbf{q}(t)) \leq c^2/\delta^2 + c$ .

*Proof:* We prove this through the principle of mathematical induction. First,  $L(\mathbf{q}(0)) = 0 \leq c^2/\delta^2 + c$ .

Next, as the induction hypothesis, assume that  $L(\mathbf{q}(t)) \leq c^2/\delta^2 + c$ . Consider two cases. (i) If  $L(\mathbf{q}(t)) \in [c^2/\delta^2, c^2/\delta^2 + c]$ , then  $\|\mathbf{q}(t)\| \geq c/\delta$ . By (21), we have  $L(\mathbf{q}(t+1)) \leq L(\mathbf{q}(t)) \leq c^2/\delta^2 + c$ . (ii) If  $L(\mathbf{q}(t)) < c^2/\delta^2$ , since  $L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) \leq -\delta\|\mathbf{q}(t)\| + c \leq c$ , we also have  $L(\mathbf{q}(t+1)) \leq c^2/\delta^2 + c$ . This completes the proof. ■

*Lemma 5:* Assume that condition 1 holds. Let  $\mathbf{y}(l \cdot T)$  be the (unique) vector that satisfies

$$A \cdot \tilde{\mathbf{a}}_l + B \cdot \mathbf{y}(l \cdot T) = \mathbf{0} \quad (22)$$

where  $\tilde{\mathbf{a}}_l$  is defined in (14). Then there exists  $\bar{\delta} > 0$  such that

$$\mathbf{q}^T B \cdot [\mathbf{x}^*(\mathbf{q}) - \mathbf{y}(l \cdot T)] \geq \bar{\delta}\|\mathbf{q}\|, \forall l, \forall \mathbf{q} \in \mathcal{B} \quad (23)$$

where  $\mathbf{x}^*(\mathbf{q})$  is defined in (12).

*Proof:* By Condition 1,  $\exists \sigma > 0$  such that for all  $l$ ,  $\tilde{\mathbf{a}}_l + \sigma \cdot \mathbf{1}$  and  $\tilde{\mathbf{a}}_l - \sigma \cdot \mathbf{1}$  are feasible. Therefore,  $\mathbf{y}(l \cdot T) + s(\sigma \cdot \mathbf{1}_M) \in \Lambda$  and  $\mathbf{y}(l \cdot T) - s(\sigma \cdot \mathbf{1}_M) \in \Lambda$ . Define  $\sigma' > 0$  to be the minimum element of  $s(\sigma \cdot \mathbf{1}_M) \succ \mathbf{0}$ , then  $\mathbf{y}' \in \Lambda$  for any  $\mathbf{y}'$  satisfying  $\|\mathbf{y}' - \mathbf{y}(l \cdot T)\| \leq \sigma'$ . (This is because the set  $\Lambda$  is ‘‘comprehensive’’: if  $\mathbf{s} \in \Lambda$ , then  $\mathbf{s}' \in \Lambda$  for any  $\mathbf{0} \preceq \mathbf{s}' \preceq \mathbf{s}$ .) Then, following the proof of Lemma 3, letting  $\bar{\delta} := \hat{\sigma} \cdot \sigma'$  (which do not depend on  $l$  or  $\mathbf{q}$ ) completes the proof. ■

*Lemma 6:* Assume that the maximum change of any queue in one time slot is bounded by  $\alpha$ . And the absolute value of every element of  $A$  and  $B$  is bounded by  $\bar{b}$ . Then

$$\begin{aligned} & L(\mathbf{q}((l+1)T)) - L(\mathbf{q}(l \cdot T)) \\ & \leq -T \cdot \bar{\delta}\|\mathbf{q}(l \cdot T)\| + c_2 \end{aligned}$$

where  $c_2 > 0$  is a constant, defined as

$$c_2 := T \cdot c + KT^2\alpha \cdot (M + K)\bar{b}.$$

*Proof:* From (19), we have

$$\begin{aligned} & L(\mathbf{q}((l+1)T)) - L(\mathbf{q}(l \cdot T)) \\ & \leq -\sum_{\tau=l \cdot T}^{(l+1)T-1} \mathbf{q}(\tau)^T A \cdot \mathbf{a}(\tau) \\ & \quad - \sum_{\tau=l \cdot T}^{(l+1)T-1} \mathbf{q}(\tau)^T B \cdot \mathbf{x}^*(\mathbf{q}(\tau)) + T \cdot c. \end{aligned}$$

For any  $\tau \in \{l \cdot T, \dots, (l+1)T - 1\}$ ,

$$\begin{aligned} & \mathbf{q}(\tau)^T B \cdot \mathbf{x}^*(\mathbf{q}(\tau)) \\ & \geq \mathbf{q}(\tau)^T B \cdot \mathbf{x}^*(\mathbf{q}(l \cdot T)) \\ & = \mathbf{q}(l \cdot T)^T B \cdot \mathbf{x}^*(\mathbf{q}(l \cdot T)) + \\ & \quad [\mathbf{q}(\tau) - \mathbf{q}(l \cdot T)]^T B \cdot \mathbf{x}^*(\mathbf{q}(l \cdot T)). \end{aligned}$$

Since  $|q_k(\tau) - q_k(l \cdot T)| \leq T \cdot \alpha$ , and each element of  $\mathbf{x}^*(\mathbf{q}(l \cdot T))$  is bounded by 1, we have

$$|[\mathbf{q}(\tau) - \mathbf{q}(l \cdot T)]^T B \cdot \mathbf{x}^*(\mathbf{q}(l \cdot T))| \leq KN\bar{b}T\alpha.$$

Therefore,

$$\begin{aligned} & \mathbf{q}(\tau)^T B \cdot \mathbf{x}^*(\mathbf{q}(\tau)) \\ & \geq \mathbf{q}(l \cdot T)^T B \cdot \mathbf{x}^*(\mathbf{q}(l \cdot T)) - KN\bar{b}T\alpha. \end{aligned} \quad (24)$$

Also,  $\mathbf{q}(\tau)^T A \cdot \mathbf{a}(\tau) \geq \mathbf{q}(l \cdot T)^T A \cdot \mathbf{a}(\tau) - KM\bar{b}T\alpha$ . Then

$$\begin{aligned} & L(\mathbf{q}((l+1)T)) - L(\mathbf{q}(l \cdot T)) \\ & \leq T \cdot \{-\mathbf{q}(l \cdot T)^T A \cdot \tilde{\mathbf{a}}_l + KM\bar{b}T\alpha \\ & \quad - \mathbf{q}(l \cdot T)^T B \cdot \mathbf{x}^*(\mathbf{q}(l \cdot T)) + KN\bar{b}T\alpha\} + T \cdot c \\ & = -T \cdot \mathbf{q}(l \cdot T)^T B \cdot [\mathbf{x}^*(\mathbf{q}(l \cdot T)) - \mathbf{y}(l \cdot T)] + c_2 \\ & \leq -T \cdot \bar{\delta}\|\mathbf{q}(l \cdot T)\| + c_2 \end{aligned}$$

where the last two steps have used (22) and condition (23). ■

Now Theorem 1 can be proved as follows.

*Proof:* Lemma 6 and Lemma 4 imply that  $\mathbf{q}(l \cdot T)$  is bounded for all  $l$ . Because each queue has bounded increments per slot,  $\mathbf{q}(t)$  is bounded for all  $t$ . ■

## B. Proof of Theorem 2

By (19),  $L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) \leq -\mathbf{q}(t)^T A \cdot \mathbf{a}(t) - \mathbf{q}(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) + c$ . So

$$\begin{aligned} & E[L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) | \mathbf{q}(t)] \\ & \leq -\mathbf{q}(t)^T A \cdot E[\mathbf{a}(t)] - \mathbf{q}(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) + c \\ & = -\mathbf{q}(t)^T A \cdot \lambda - \mathbf{q}(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) + c \\ & = \mathbf{q}(t)^T B \cdot \mathbf{y} - \mathbf{q}(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) + c \\ & \leq -\delta\|\mathbf{q}(t)\| + c. \end{aligned} \quad (25)$$

Let  $\mathcal{E}_0 := \{\mathbf{q}(t) | \|\mathbf{q}(t)\| \leq (c+1)/\delta\}$ . Then if  $\mathbf{q}(t) \notin \mathcal{E}_0$ ,  $E[L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) | \mathbf{q}(t)] \leq -1$ ; if  $\mathbf{q}(t) \in \mathcal{E}_0$ ,  $E[L(\mathbf{q}(t+1)) - L(\mathbf{q}(t)) | \mathbf{q}(t)] < \infty$  due to the bounded change of queue lengths in each slot. Therefore, by Foster’s criteria as used in [8],  $\mathbf{q}(t)$  is stable.

Also, we claim that given a set  $\mathcal{E}$ , with probability 1, the time average  $P(\mathcal{E}) := \lim_{T \rightarrow \infty} \sum_{t=0}^{T-1} I(\mathbf{q}(t) \in \mathcal{E})/T$  exists. To see this, partition the state space of  $\mathbf{q}(t)$  into set  $\mathcal{T}, \mathcal{R}_1, \mathcal{R}_2, \dots$  where  $\mathcal{R}_j, j = 1, 2, \dots$  are closed sets of communicating states and  $\mathcal{T}$  is the set of states not in  $\cup_j \mathcal{R}_j$ . If  $\mathbf{q}(0) = \mathbf{0} \in \mathcal{R}_j$  for some  $j$ , then  $\mathbf{q}(t)$  will not leave the set and all states in  $\mathcal{R}_j$  are positive recurrent. Therefore there is a well defined stationary distribution in  $\mathcal{R}_j$ , so  $P(\mathcal{E})$  exists w. p. 1. If  $\mathbf{q}(0) = \mathbf{0} \in \mathcal{T}$ , by Foster’s criteria as used in [8] (Theorem 3.1), the negative drift implies that w. p. 1,  $\mathbf{q}(t)$  enters some  $\mathcal{R}_j$  in finite time. After that there is a well defined time average of  $I(\mathbf{q}(t) \in \mathcal{E})$  w. p. 1. Therefore, the overall time average  $P(\mathcal{E})$  exists. In both cases,

$$P(\mathcal{E}) = \pi_j(\mathcal{E}) \quad (26)$$

where  $\pi_j(\cdot)$  is the stationary distribution on the  $\mathcal{R}_j$ , and  $\mathcal{R}_j$  is the closed set of communicating states  $\mathbf{q}(t)$  eventually enters.

To show the rate stability, consider two kinds of queues. WLOG, let  $\mathcal{U}$  be the set of queues whose deficits go unbounded. According to Proposition 1, the queues outside the set only induce a finite number of null activities.

Consider queue  $k \in \mathcal{U}$ . For any  $C > 0$ , since  $D_k(t) \rightarrow \infty$ , there exists finite time  $t_k$  such that  $D_k(t) \geq C, \forall t \geq t_k$ . For  $t \geq t_k$ , queue  $k$  induces null activities at slot  $t - 1$  only when  $q_k(t) < -D_k(t) \leq -C$ . So the total number of null activities induced by queue  $k$  is not more than  $N \cdot [t_k + \sum_{t=t_k}^{\infty} I(q_k(t) < -C)] \leq N \cdot [t_k + \sum_{t=0}^{\infty} I(q_k(t) < -C)]$ , since queue  $k$  at most induces  $N$  null activities in one time slot. Therefore, the average rate the queue  $k$  induces null activities is

$$r_k \leq N \cdot \lim_{T \rightarrow \infty} \frac{1}{T} [t_k + \sum_{t=0}^{T-1} I(q_k(t) < -C)] = N \cdot Pr(q_k < -C). \quad (27)$$

where the marginal probability on the RHS is induced by the stationary distribution  $\pi_j(\cdot)$  on the set  $R_j$  which  $\mathbf{q}(t)$  eventually enters. So  $\lim_{C \rightarrow +\infty} Pr(q_k < -C) = 0$ . Since (27) holds for any  $C > 0$ , letting  $C \rightarrow +\infty$  yields  $r_k = 0$ .

Therefore, the average rate of null activities is 0 in the long term w. p. 1. Also, if we imagine that the null activities produce real parts, then the output rates of the final products would be the maximum since the virtual queues  $\mathbf{q}(t)$  are stable. Combining the two facts concludes the proof.

### C. Proof of Theorem 3

*Lemma 7:*  $\mathbf{q}(t)$  is bounded.

*Proof:* Choose any  $\mathbf{f}' \in \mathcal{R}^m$  and  $\mathbf{y}' > \mathbf{0}$  in  $\Lambda^\circ$  such that the flow conservation constraint is satisfied:  $A \cdot \mathbf{f}' + B \cdot \mathbf{y}' = \mathbf{0}$ , and  $|\sum_m u_m(f'_m)| < \infty, \forall m$ . The latter is feasible by letting  $f'_m = \epsilon > 0, \forall m$  where  $\epsilon$  is small enough.

By Lemma 3, we have for any  $\mathbf{q} \in \mathcal{B}$ ,

$$q^T B \cdot [\mathbf{x}^*(\mathbf{q}) - \mathbf{y}'] \geq \delta' \|\mathbf{q}\| \quad (28)$$

for some constant  $\delta' > 0$ .

Also, since  $f'_m \in [0, 1]$ , by (16),

$$\begin{aligned} & V \cdot u_m(f'_m) + \mathbf{q}(t)^T A_m \cdot f'_m \\ & \leq V \cdot u_m(f_m(\mathbf{q}(t))) + \mathbf{q}(t)^T A_m f_m(\mathbf{q}(t)), \forall m. \end{aligned}$$

Therefore

$$\begin{aligned} & V \cdot \sum_{m=1}^M u_m(f'_m) + \mathbf{q}(t)^T A \cdot \mathbf{f}' \\ & \leq V \cdot \sum_{m=1}^M u_m(f_m(\mathbf{q}(t))) + \mathbf{q}(t)^T A \cdot \mathbf{f}(\mathbf{q}(t)). \end{aligned}$$

Since  $|\sum_m u_m(f'_m)| < \infty$ , we have  $\sum_{m=1}^M u_m(f_m(\mathbf{q}(t))) - \sum_{m=1}^M u_m(f'_m) \leq \sum_{m=1}^M u_m(1) - \sum_{m=1}^M u_m(f'_m) \leq C_1$  for some positive constant  $C_1$ . So

$$-\mathbf{q}(t)^T A \cdot \mathbf{f}(\mathbf{q}(t)) \leq -\mathbf{q}(t)^T A \cdot \mathbf{f}' + V \cdot C_1. \quad (29)$$

Similar to (19), the Lyapunov drift in the algorithm is

$$\Delta(\mathbf{q}(t)) \leq -\mathbf{q}(t)^T A \cdot \mathbf{f}(\mathbf{q}(t)) - q(t)^T B \cdot \mathbf{x}^*(\mathbf{q}(t)) + c. \quad (30)$$

Plugging (28) and (29) into (30) yields

$$\begin{aligned} & \Delta(\mathbf{q}(t)) \\ & \leq -\mathbf{q}(t)^T A \cdot \mathbf{f}' + V \cdot C_1 - q(t)^T B \cdot \mathbf{y}' - \delta' \|\mathbf{q}(t)\| + c \\ & = -\mathbf{q}(t)^T [A \cdot \mathbf{f}' + B \cdot \mathbf{y}'] - \delta' \|\mathbf{q}(t)\| + V \cdot C_1 + c \\ & = -\delta' \|\mathbf{q}(t)\| + V \cdot C_1 + c. \end{aligned}$$

Using Lemma 4, the above implies that for all  $t$ ,

$$L(\mathbf{q}(t)) \leq [(V \cdot C_1 + c)/\delta']^2 + V \cdot C_1 + c.$$

So  $\mathbf{q}(t)$  is bounded.  $\blacksquare$

Define  $\tilde{\mathbf{q}}(0) = \mathbf{0}$ , and for  $t = 0, 1, \dots$ , define

$$\tilde{\mathbf{q}}(t+1) = \tilde{\mathbf{q}}(t) - A \cdot \mathbf{a}(t) - B \cdot \mathbf{x}^*(t). \quad (31)$$

*Lemma 8:* For all  $t$ ,  $\|\tilde{\mathbf{q}}(t) - \mathbf{q}(t)\| \leq Z$  for some constant  $Z > 0$ .

*Proof:* By (17) and  $\mathbf{q}(0) = \mathbf{0}$ , we have

$$\begin{aligned} \mathbf{q}(t) & = \sum_{\tau=0}^{t-1} [-A \cdot \mathbf{f}(\mathbf{q}(\tau)) - B \cdot \mathbf{x}^*(\tau)] \\ & = -A \sum_{\tau=0}^{t-1} \mathbf{f}(\mathbf{q}(\tau)) - B \cdot \sum_{\tau=0}^{t-1} \mathbf{x}^*(\tau). \end{aligned}$$

By (31) and  $\tilde{\mathbf{q}}(0) = \mathbf{0}$ , we have

$$\begin{aligned} \tilde{\mathbf{q}}(t) & = \sum_{\tau=0}^{t-1} [-A \cdot \mathbf{a}(\tau) - B \cdot \mathbf{x}^*(\tau)] \\ & = -A [\sum_{\tau=0}^{t-1} \mathbf{f}(\mathbf{q}(\tau))] - B \cdot \sum_{\tau=0}^{t-1} \mathbf{x}^*(\tau). \end{aligned}$$

So,  $\|\tilde{\mathbf{q}}(t) - \mathbf{q}(t)\| = \|A \cdot \{\sum_{\tau=0}^{t-1} \mathbf{f}(\mathbf{q}(\tau)) - [\sum_{\tau=0}^{t-1} \mathbf{f}(\mathbf{q}(\tau))]\}\|$ . Since each element of  $\sum_{\tau=0}^{t-1} \mathbf{f}(\mathbf{q}(\tau)) - [\sum_{\tau=0}^{t-1} \mathbf{f}(\mathbf{q}(\tau))]$  is between 0 and 1, and each element of  $A$  is bounded, we conclude that  $\|\tilde{\mathbf{q}}(t) - \mathbf{q}(t)\| \leq Z$  for some constant  $Z > 0$ .  $\blacksquare$

Now we are ready to complete the proof.

Since  $\|\tilde{\mathbf{q}}(t)\| \leq \|\mathbf{q}(t)\| + \|\tilde{\mathbf{q}}(t) - \mathbf{q}(t)\|$ , combining the previous two lemmas, we know that  $\|\tilde{\mathbf{q}}(t)\| \leq G, \forall t$  for some  $G > 0$ . Define  $\mathbf{D}(t) = \mathbf{Q}(t) - \tilde{\mathbf{q}}(t)$ . Comparing the dynamics of  $\mathbf{Q}(t)$  and  $\tilde{\mathbf{q}}(t)$ , it is clear that we can apply Proposition 1 to  $\tilde{\mathbf{q}}(t), \mathbf{Q}(t)$  and  $D(t)$  to complete the proof.

### D. Proof of Theorem 4

*Proof:* Assume that  $\mathbf{f}^* \in \mathcal{R}^m$  and  $\mathbf{y}^* > \mathbf{0}$  achieves the optimal utility  $U^*$ . So  $A \cdot \mathbf{f}^* + B \cdot \mathbf{y}^* = \mathbf{0}$  and  $U^* = \sum_m u_m(f_m^*)$ .

We also have  $q^T B \cdot [\mathbf{x}^*(\mathbf{q}) - \mathbf{y}^*] \geq 0$ . This is equivalent to (28) when  $\delta' = 0$ . Then, following the proof of Theorem 3 (but without using the upper bound  $C_1$ ), we have

$$\begin{aligned} \Delta(\mathbf{q}(t)) & \leq -\mathbf{q}(t)^T [A \cdot \mathbf{f}^* + B \cdot \mathbf{y}^*] + \\ & \quad V \cdot [\sum_m u_m(f_m(\mathbf{q}(t))) - \sum_m u_m(f_m^*)] + c \\ & = V \cdot [\sum_m u_m(f_m(\mathbf{q}(t))) - U^*] + c. \end{aligned}$$

Summing over  $t$  from 0 to  $T - 1$  yields

$$L(\mathbf{q}(T)) - L(\mathbf{q}(0)) \leq V \cdot \sum_{t=0}^{T-1} \sum_m u_m(f_m(\mathbf{q}(t))) - VTU^* + T \cdot c.$$

Dividing both sides by  $T \cdot V$ , and using  $L(\mathbf{q}(T)) - L(\mathbf{q}(0)) = L(\mathbf{q}(T)) \geq 0$ , one gets

$$\sum_{t=0}^{T-1} \sum_m u_m(f_m(\mathbf{q}(t))) / T \geq U^* - c/V. \quad (32)$$

Since  $u_m(\cdot)$  is concave,  $u_m(\sum_{t=0}^{T-1} f_m(\mathbf{q}(t))) / T \geq \sum_{t=0}^{T-1} u_m(f_m(\mathbf{q}(t))) / T$ . Using this, (32) and letting  $T \rightarrow \infty$ , we have (18).  $\blacksquare$

## REFERENCES

- [1] R. J. Williams, "On Stochastic Processing Networks," Lecture Notes, 2006. <http://math.ucsd.edu/~williams/talks/belz/belznotes06.pdf>
- [2] J. Walrand. *An Introduction to Queueing Networks*. Prentice Hall, 1988.
- [3] J. M. Harrison, "Brownian Models of Open Processing Networks: Canonical Representation of Workload," *Annals of Applied Probability*, vol. 10, no. 1, pp. 75-103, 2000.
- [4] J. M. Harrison and R. J. Williams, "Workload reduction of a generalized Brownian network," *Annals of Applied Probability*, vol. 15, no. 4, pp. 2255-2295, 2005.
- [5] J. G. Dai and W. Lin, "Asymptotic optimality of maximum pressure policies in stochastic processing networks," *Annals of Applied Probability*, vol. 18, no. 6, pp. 2239-2299, 2008.
- [6] J. G. Dai and W. Lin, "Maximum Pressure Policies in Stochastic Processing Networks," *Operations Research*, vol. 53, no. 2, pp. 197-218, Mar-Apr 2005.
- [7] L. M. Wein, "Optimal control of a two-station Brownian network," *Mathematics of Operations Research*, vol. 15, no. 2, , pp. 215 - 242, May 1990.
- [8] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 36, pp. 1936-1948, Dec. 1992.
- [9] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237-252, 1998.
- [10] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861-874, Dec. 1999.
- [11] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556-567, 2000.
- [12] M.J. Neely, E. Modiano, and C-P. Li, "Fairness and Optimal Stochastic Control for Heterogeneous Networks," in *IEEE INFOCOM*, Mar. 2005.
- [13] A. Eryilmaz and R. Srikant, "Fair Resource Allocation in Wireless Networks using Queue-Length-Based Scheduling and Congestion Control," in *IEEE INFOCOM*, Mar. 2005.
- [14] S. Meyn, "Stability and asymptotic optimality of generalized MaxWeight policies," *SIAM Journal on Control and Optimization*, vol. 47, no. 6, 2009.