

# Optimal Allocation in a Queuing System with Shared Resources

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

**Abstract**—In this paper, we study the problem of dynamic allocation of heterogeneous processors to parallel heterogeneous job traffic flows. Each traffic flow is a stationary ergodic random marked point process, with a parameter (traffic intensity rate) that depends on the job class. The service rates of the various job flows depend on both the job class and the processor class. This model captures the essential features of several practical systems, including flexible manufacturing ones, packet switches, distribution systems, etc. We first specify precisely the necessary and sufficient condition for stability of the system. We then identify a *family* of policies that achieves maximum throughput; i.e. they stabilize the system under the maximum possible input rates.

The approach taken introduces the concept of *virtual queuing*, which proves powerful in establishing strong probabilistic results (convergence in distribution to a finite stationary regime) for queuing systems with complex dynamics due to sharing resources, under a very general stationary ergodic probabilistic structure.

## I. INTRODUCTION AND NOTATION

Consider a queuing system consisting of  $Q$  processors in parallel and  $Q$  infinite capacity first-come-first-served (FCFS) queues, each queue corresponding to a different class of job traffic. The input to queue  $q \in \mathbf{Q} = \{1, \dots, Q\}$  is according to a random marked point process  $\mathcal{S}_q = \{(t_j^q, \sigma_j^q), j \in \mathbb{Z}_+\}$ , where  $t_j^q$  is the arrival time of the  $j^{\text{th}}$  job into queue  $q$  (assume a canonical numbering  $0 \leq t_1^q \leq t_2^q \leq \dots \leq t_j^q \leq \dots$ ) and  $\sigma_j^q$  is its service requirement. The latter implies that if the  $j^{\text{th}}$  job were to be served at *constant* rate  $r$ , its service time would be  $\sigma_j^q/r$  time units. The stochastic processes  $\{\mathcal{S}_q; q \in \mathbf{Q}\}$  are defined on some common probability space  $(\Omega, \mathcal{F}, P)$  and are assumed to be stationary and ergodic with respect to time shifts  $\theta_z \mathcal{S}_q = \{(t_j^q - \theta_z, \sigma_j^q), j \in \mathbb{Z}_+\}$ . The *traffic intensity* (average workload per unit of time) entering queue  $q$  is given by

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

There are also  $Q$  classes of processors (servers), with a class  $k$ ,  $k = 1, \dots, Q$  server processing class  $q$  jobs at a rate  $\mu_{kq} > 0$ . A schematic representation of the simplest possible version of the model under study involving two job classes and two servers is shown in Figure 1. It is worth noting that if there is only a single processor, then the model corresponds

to a Generalized Processor Sharing system [6].<sup>1</sup>

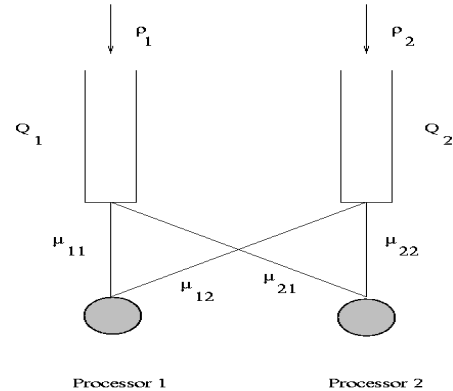


Fig. 1. A system consisting of two job classes and two servers of varying processing power.

A service policy  $\pi$  is a rule that determines to which job traffic class every processor should be allocated at time  $t$ . Let  $\Pi$  be the set of all work-conserving possible service policies. Define  $W_\pi^q(t)$  to be the workload in queue  $q$  at time  $t$ , when the system operates under policy  $\pi \in \Pi$ . This is easily seen to evolve according to the equation:

$$W_\pi^q(t) = W_\pi^q(s) + V^q(s, t) - \sum_{k=1}^c \mu_{kq} \int_s^t z_{kq}(u) \mathbf{1}_{\{W_\pi^q(u) > 0\}} du, \quad (2)$$

where  $s < t$ ,  $V^q(s, t) = \sum_{j \in \mathbb{Z}_+} \sigma_j^q \mathbf{1}_{\{t_j^q \in (s, t]\}}$ , and  $z_{kq}(u) \in \{0, 1\}$  denotes that processor  $k$  is allocated to queue  $q$  at time  $u$ . The workload state of the system is then defined by  $\vec{W}_\pi(t) = \{W_\pi^1(t), \dots, W_\pi^Q(t)\}$  and the initial workload is  $\vec{W}(0)$  (the same under all policies). The workload state at time  $t$  depends on the history of the input processes  $\mathcal{S}_q$ ,  $q \in \mathbf{Q}$  up to time  $t$  and the followed service policy  $\pi \in \Pi$ .

We focus on the practical problem of finding efficient processing policies in  $\Pi$  which maximize the sustainable throughput of the system, maintaining its stability (quantified below) under the most workload intensive input rates.

This basic queuing model captures the essence of a fundamental resource allocation problem encountered in many

<sup>1</sup>A straightforward generalization of the model assumes that there are  $c_k \geq 1$  processors of class  $k$ . All the results established in the paper carry over; however, we restrict attention to the  $c_k = 1$  case in order to avoid complicating the notation.

modern communication, computer, and manufacturing systems involving heterogeneous processors and multiple classes of job traffic flows. For example, consider the dynamic routing problem in a communication network implementing circuit-switching or wormhole routing [9]. Messages arrive to the router of a source node, distinguished by their destinations, and must be routed along one of a number of non-equivalent paths. For each source-destination pair, there are multiple paths through the network. The average transmission delay of a message depends on its destination *and* the path along which it is routed. Once the transmission of a message begins on a path, the path is not available to transmit other messages until this message has been completely received at the destination. The router faces the problem of allocating paths to messages.

Problems of dynamic allocation of processing resources for throughput maximization of complex processing systems have attracted considerable attention in various contexts during the past several years. Adaptive policies have been considered in [1], [4], [12], just to name a few, under different stochastic assumptions of the input and service processes (Markov, renewal, stationary ergodic).

The objectives and structure of the paper are as follows. At the *methodological* level, it develops the concept of virtual queueing that allows to decompose queueing systems with complex interactions into conceptually simpler subsystems, without altering their operational characteristics. This decomposition expands significantly the space of allocation policies to be considered, while at the same time provides a deeper insight into the dynamics of the system under study. Virtual queueing simplifies the study of system stability, which is the most fundamental issue of their behavior [1]. At the *technical* level, the paper establishes the existence of a *stationary regime* for a large family of maximum throughput, dynamically adaptive policies under a general stationary ergodic stochastic framework. The concept of virtual queueing allows us to use a Loynes [7] type construction for proving the results, appropriately modified and extended so as to fit the complex framework of the model under study. It is worth noting that for a similar model studied under ergodic input processes in [1], the significantly weaker result of *rate stability* (job departure rates are equal to job arrival rates) [2] is established for various adaptive scheduling policies.

The paper is organized as follows. In section 2, the stability region of the system is characterized. In section 3, the concept of *virtual* queueing is introduced, along with a family of policies that are later shown to maximize the sustainable system throughput. In section 4, the properties of the proposed adaptive policies are established, while in section 5 some extensions are discussed and some concluding remarks drawn.

## II. SYSTEM STABILITY ISSUES

In this section, we address the issue of system stability. For a given service rate matrix  $\mathcal{M} = \{\mu_{kq}\}$ ,  $k, q \in \mathbf{Q}$  and traffic intensity vector  $\vec{\rho} = \{\rho_q\}_{q=1}^Q$ , the system is said to be *stable* under  $\pi \in \Pi$ , if  $W_\pi^q(t) < \infty$  at all time points  $t \in \mathbb{R}_+$ . The set of traffic intensity vectors for which there exists some policy  $\pi \in \Pi$  that keeps the backlogs finite defines the *stability region* of the system.

Let  $\mathcal{F}$  be the set of all  $Q \times Q$  matrices  $f = \{f_{qk}\}$  such that  $f_{qk} \in [0, 1]$  for  $k \in \mathbf{Q}$  and  $\sum_{k=1}^Q f_{qk} = 1$ . Define the set

$$\mathbf{S} = \left\{ \rho \in \mathbb{R}_+^Q : \exists f \in \mathcal{F} \text{ for which } \sum_{q=1}^Q \frac{\rho_q f_{qk}}{\mu_{kq}} < 1, \forall k \in \mathbf{Q} \right\}. \quad (3)$$

A schematic representation of the stability region  $\mathbf{S}$  for a system with two queues and two interacting servers is given in Figure 2. It can be shown in general that  $\mathbf{S}$  for the model under study is a polyhedron.

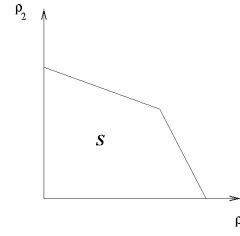


Fig. 2. The stability region  $\mathbf{S}$  for a system comprised of two job classes and two interacting servers.

**Proposition 1:** (The Case of Instability). For any stationary and ergodic input  $\{\mathcal{J}_q, q \in \mathbf{Q}\}$  and *any* allocation policy  $\pi \in \Pi$ , if

$$\vec{\rho} \notin \mathbf{S} \Rightarrow \limsup_{t \rightarrow \infty} \frac{W_\pi^q(t)}{t} > 0, \text{ for some queue } q \in \mathbf{Q}, \quad (4)$$

that is the workload of at least one queue in the system blows up to infinity.

*Proof:* Given any arbitrary fixed policy  $\pi$  and arguing by contradiction, assume that  $\lim_{t \rightarrow \infty} W_\pi^q(t)/t = 0$  for every queue  $q \in \mathbf{Q}$ , for some arbitrarily fixed  $\vec{\rho} \notin \mathbf{S}$ . From (2) we get

$$W_\pi^q(t) = W_\pi^q(0) + V^q(0, t) - \sum_{k=1}^Q \mu_{kq} \int_0^t z_{kq}(u) \mathbf{1}_{\{W_\pi^q(u) > 0\}} du, \quad (5)$$

for each queue  $q \in \mathbf{Q}$ . Dividing both sides of (5) by  $t$ , letting  $t \rightarrow \infty$  and using (1) the ergodicity of the input processes together with the contradiction assumption, and rearranging terms, we get

$$\sum_{k=1}^Q \mu_{kq} \lim_{m \rightarrow \infty} \left\{ \frac{1}{t_m} \int_0^{t_m} z_{kq}(u) du \right\} = \rho_q, \quad (6)$$

for some increasing unbounded time sequence  $\{t_m\}_{m=1}^\infty$ . Such a subsequence can be directly constructed through successive thinnings of convergent subsequences of the limit terms for consecutive  $m$  indices. In view of the above, setting

$$x_{kq} = \lim_{m \rightarrow \infty} \left\{ \frac{1}{t_m} \int_0^{t_m} z_{kq}(u) \mathbf{1}_{\{W_x^q(u)\}} du \right\} \geq 0, \text{ we get } \rho_q = \sum_{k=1}^Q x_{kq} \mu_{kq}.$$

Define  $f_{qk} = x_{kq} \mu_{kq} / \rho_q$ . Some algebra shows that  $\sum_{k=1}^Q f_{qk} = \sum_{k=1}^Q x_{kq} \mu_{kq} / \rho_q = 1$  in view of (6), for  $q \in \mathbf{Q}$  and therefore  $f \in \mathcal{F}$ . Moreover,  $\sum_{q=1}^Q \rho_q f_{qk} / \mu_{kq} = \sum_{k=1}^Q x_{kq} \leq 1$ , since each processor can serve at most one job class at a time. The latter contradicts the hypothesis in (4) and hence the result is established. ■

Proposition (1) implies that there is *no* service policy that would maintain finite backlogs in case  $\vec{\rho} \notin \mathbf{S}$ . The issue then becomes whether there are adaptive, backlog responsive policies that require no *a priori knowledge* of the traffic intensity vector  $\vec{\rho}$  that would ensure finite backlogs when  $\vec{\rho} \in \mathbf{S}$ . Indeed, as long as  $\vec{\rho} \in \mathbf{S}$ , such policies will keep the system stable, without knowing the specific value of  $\vec{\rho}$ . The latter implies that even if  $\vec{\rho}$  changes over a time scale that is much longer than that for the time-averaged incoming workload to converge to its traffic intensity rate, an adaptive policy would automatically respond and keep the backlogs finite [1]; thus, such policies are robust to long term changes of the input rates.

### III. THE CONCEPT OF VIRTUAL QUEUEING

In this section we study a family of adaptive (backlog-responsive) policies, which are based on the notion of *virtual queueing*. The notion of *virtual queueing* can be described as follows: every job flow maintains a separate queue for each possible server configuration of the system. In the present setting, job flow  $q \in \mathbf{Q}$  should maintain  $Q$  virtual separate queues, one for each server. Let  $V_k^q$ ,  $k, q \in \mathbf{Q}$  denote the virtual queue of job flow queue that can *only* receive service from processor  $k$ . Define  $\mathcal{D}_q = \{V_k^q : q' = q\}$  to be the set of virtual queues where type  $q$  traffic flows can be *routed* to. Similarly, define  $\mathcal{S}_k = \{V_k^q : k' = k\}$  to be the set of virtual queues that can receive service from processor  $k$ . It can be seen that both  $|\mathcal{D}_q| = Q$  and  $|\mathcal{S}_k| = Q$ . An example of a simple system comprised of two job flows and two processors along with their virtual queues is shown in Figure 3. It can be seen that for the new system involving virtual queues a *joint routing/scheduling* policy would be required. However, a close inspection of the new system shows that on the scheduling side *any* work conserving policy would work and therefore the goal becomes to specify routing policies; i.e. given an arriving job belonging to flow  $q$  to which of the  $Q$  virtual queues of the destination set  $\mathcal{D}_q$  it should be routed to. Notice that the introduction of the virtual queue systems  $\mathcal{S}_k$  *decouples* the scheduling dynamics of the processors, since in the new setting every processor has to serve *its own* set of virtual queues  $\mathcal{S}_k$ . Actually every queuing system  $\mathcal{S}_k$  has all the characteristics of a Generalized Processor System (GPS)

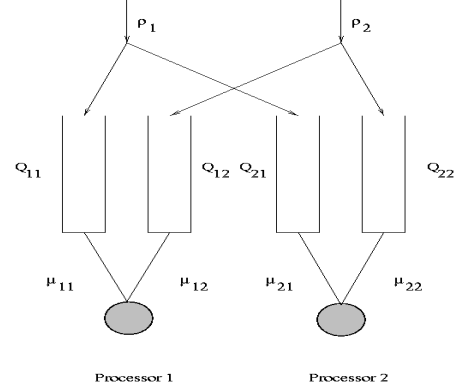


Fig. 3. A system with 2 input flows and two processors with the corresponding virtual queues

[6]. It can then be easily seen that the stability condition corresponds to the one for a GPS adjusted for the proportion of input flows directed to it.

For the transformed system that involves virtual queues, the definition of the stability region  $\mathbf{S}$  remains the same and is given by

$$\mathbf{S} = \left\{ \rho \in \mathbb{R}_+^Q : \exists f \in \mathcal{F} \text{ for which } \sum_{q \in \mathcal{S}_k} \frac{\rho_q f_{qk}}{\mu_{kq}} < 1, \forall k \in \mathbf{Q} \right\}. \quad (7)$$

In virtual queueing the interpretation of the set of  $f_{qk}$  becomes transparent; they represent the proportion of the overall job flow of class  $q$  that will be routed to system  $\mathcal{S}_k$  and thus eventually processed by server  $k \in \mathbf{Q}$ .

Let  $\mathcal{W}_k^q(t)$  denote the workload of virtual queue  $V_k^q$  at time  $t \geq 0$ . The evolution of the workload is according to:

$$\mathcal{W}_k^q(t) = \mathcal{W}_k^q(0) + V_k^q(0, t) - \mu_{kq} \int_0^t z_{kq}(u) \mathbf{1}_{\{\mathcal{W}_k^q(u) > 0\}} du, \quad (8)$$

where  $V_k^q(0, t) = \sum_{j \in \mathbb{Z}_+} \sigma_j^q \mathbf{1}_{\{t_j^q \in (0, t]\}} \mathbf{1}_{\{c_j^q \rightarrow V_k^q\}}$ . It can be seen that the quantity  $V_k^q(0, t)$  depends on the policy  $\pi$  through the indicator function  $\mathbf{1}_{\{c_j^q \rightarrow V_k^q\}}$  that directs the  $j^{\text{th}}$  customer of class  $q$  to virtual queue  $V_k^q$ , but we have suppressed this dependence to simplify the notation.

We focus next on a particular routing policy, which is shown to exhibit optimal behavior. The proposed policy is called Shortest Virtual Queue (SVQ) and denoted by  $\mathcal{R}_{\text{SVQ}}$ , and routes an incoming job to the virtual queue with the smallest number of jobs. Actually, the SVQ policy defines a huge family of policies, since they can be combined with *any* work conserving (non-idling) scheduling policy.

Since we have focused on workloads rather than job lengths we will examine a variation of the SVQ policy, called Shortest Virtual Workload (SVW) and denoted by  $\mathcal{R}_{\text{SVW}}$ , which routes the service requirement of an incoming job to the virtual queue with the smallest amount of workload. In

case the workloads of two or more virtual queues in the destination set  $\mathcal{D}_q$  become equal, the SVW policy distributes the incoming workload equally among those virtual queues.

*Remark 3.1:* Notice that both the SVQ and SVW policies try to balance the virtual queue sizes/workloads; of course this balancing act depends on the exact form of the scheduling part of the joint policy. However, the choice of the scheduling policy would affect only performance measures such as delays and not the overall throughput of the system.

#### IV. STABILITY PROPERTIES AND OPTIMALITY ASPECTS OF THE SMVW POLICY

Throughout this section we consider the virtual queueing system operating under the  $\mathcal{R}_{\text{SVW}}$  policy. However, in order to satisfy the technical requirements of a Loynes [7] type construction for a finite stationary regime for the system we need to consider a scheduling policy for each server  $k \in \mathbf{Q}$ , which combined with the  $\mathcal{R}_{\text{SVW}}$  routing policy preserve the monotonicity of the workloads of all the virtual queues  $Q_k^q$ . In the remainder we consider the policy that allocates server  $k \in \mathbf{Q}$  to the Maximum Virtual Workload queue  $\mathcal{A}_{\text{MVW}}$  among those in  $\mathcal{S}_k$ . Thus, the joint routing-scheduling policy to be considered is  $\pi^* \equiv \mathcal{R}_{\text{SVW}} - \mathcal{A}_{\text{MVW}}$ .

**Proposition 2:** (Workload Monotonicity under  $\pi^*$ ). Under the  $\pi^*$  policy, for any fixed  $s, t \in \mathbb{R}$ ,  $s < t$  and for initial workloads  $\vec{w}_1, \vec{w}_2 \in \mathbb{R}_+^{Q \times Q}$ , we have that

$$\vec{w}_1 \leq \vec{w}_2 \Rightarrow \vec{\mathcal{W}}(s, t; \vec{w}_1) \leq \vec{\mathcal{W}}(s, t; \vec{w}_2) \quad (9)$$

almost surely. That is, the workload is an increasing function of its initial value.

*Proof:* (sketch) We outline next the main idea of the proof. On any fixed sample path of  $\{\mathcal{S}_q\}_{q=1}^Q$  we observe the evolution in  $(s, t]$  of two copies of the system,  $\mathcal{S}^1$  with initial state  $\vec{w}_1$ , and  $\mathcal{S}^2$  with initial state  $\vec{w}_2$ . Due to the fact that  $\{\mathcal{S}_q\}$  are RMPP and the nature of the  $\mathcal{R}_{\text{SVW}}$  policy, it can be easily seen that we can partition  $(s, t]$  into a union of disjoint intervals  $(T_m, T_{m+1}]$ , with  $s = T_0 < T_1 < \dots < T_m < T_{m+1} \dots < T_{M-1} < T_M = t$ , each of them having the following properties:

- 1) There is no job arrival in any virtual queue in  $(T_m, T_{m+1})$ .
- 2) The set  $Q_m^1$  of virtual queues receiving service in  $\mathcal{S}^1$  remains invariant throughout  $(T_m, T_{m+1})$ . The same holds true for the set  $Q_m^2$ , analogously defined for  $\mathcal{S}^2$ . Note that due to the technical assumption made above we have  $Q_m^1 = Q_m^2$ .

The epochs  $T_m$  correspond to occurrences (possibly simultaneous) of 1) job arrivals, and 2) changes in the set of queues receiving service under the scheduling part of the SVW policy in  $\mathcal{S}^1$  and in  $\mathcal{S}^2$ .

Due to the structure of the system, for every intermediate epoch  $z \in (s, t]$  and every initial state  $\vec{w}$  we have

$$\vec{\mathcal{W}}(s, t; \vec{w}) = \vec{\mathcal{W}}(z, t; \vec{\mathcal{W}}(s, z; \vec{w})). \quad (10)$$

Therefore, in order to show Proposition 2 for  $(s, t]$ , it suffices to prove it holds for any arbitrarily chosen interval  $(T_m, T_{m+1}]$ ; we can then use induction on consecutive intervals. ■

*Remark 4.1:* Some additional work along the lines presented in the proof of Proposition 2 shows that the following scheduling policies, Maximum Service Rate (MSR) (where every server  $k \in \mathbf{Q}$  is allocated to the virtual queue in  $\mathcal{S}_k$  with the maximum service rate efficiency  $\mu_{kq}$ ) and Weighted Maximum Workload (WMVW) (where every server  $k \in \mathbf{Q}$  is allocated to the virtual queue in  $\mathcal{S}_k$  with the maximum product  $\mu_{kq} \mathcal{W}_k^q$ ) satisfy the monotonicity property. Therefore, one can use any of the combinations SVW-MSR, SVW-MSR, SVW-WMVW as the joint routing-scheduling policy, or their queue length counterparts, to establish their throughput maximizing properties following similar arguments as those presented in Proposition 3.

Based on Proposition 2 we can use a Loynes type procedure to construct a stationary operational regime of the system. Observe that for every  $s' < s$ , we have  $\mathcal{W}_k^q(s, t; \vec{0}) \leq \mathcal{W}_k^q(\vec{W}(s', s; \vec{0})) = \mathcal{W}_k^q(s', t; \vec{0})$ , where the inequality follows from Proposition 2 and the equality by arguing as in (10). Therefore, since  $\mathcal{W}_k^q(s, t; \vec{0})$  is increasing as  $s \rightarrow -\infty$ , we can pathwise define the processes

$$\tilde{W}_k^q(t) = \lim_{s \rightarrow -\infty} \mathcal{W}_k^q(s, t; \vec{0}), \quad (11)$$

for every  $k, q \in \mathbf{Q}$ , which are shown below to provide a proper (finite) stationary operational regime for the system. It can be immediately seen that  $\{\tilde{W}_k^q(t), t \in \mathbb{R}\}$  is time stationary and ergodic.

In the proof of the following Proposition we will make use of the following assumption:

*Working Hypothesis:* For some  $q \in \mathbf{Q}$  there exists an index  $k_0 \in \{1, 2, \dots, Q-1\}$ , such that  $\rho_q > \sum_{k=1}^{k_0} \max_{k \in \mathbf{Q}} \{\mu_{kq}\}$ .

Intuitively, this working hypothesis implies that the amount of traffic originating from job class  $q \in \mathbf{Q}$  can not be handled by a single processor. To see this, suppose that  $\rho_{q'} = 0$  for every  $q' \neq q$  and suppose that the most specialized (fastest) processor for job class  $q$  was continuously allocated to it. In this case we are basically dealing with an *unstable* G/G/1 queue, which implies that help from some additional processor(s) is required to keep the queue backlog finite. An illustration of the working assumption is given in Figure 4. It can be seen that  $\mu_{22} > \mu_{21}$  and that  $\mu_{11} > \mu_{12}$ . Nevertheless, because  $\rho_2 > \mu_{22}$ , in the long run the processing power of the second server would not suffice to keep job class 2 finite and additional processing power from the first server would be required to achieve this objective. On the other hand, if  $\vec{\rho}$  belongs to the box defined by the two axes and the lines drawn at  $\mu_{11}$  and  $\mu_{22}$ , the the processing power of the two servers is sufficient for handling the traffic of the two job classes in the long run individually, without the need for any interaction and cooperation between the available resources.

*Remark 4.2: System Stressing: If the traffic intensity vector  $\vec{\rho}$  does not satisfy the working hypothesis, we can always inflate it to  $\vec{\rho}'$  so that the new inflated vector satisfies it. This stressing effect can be achieved by an artificial inflation by a factor  $\gamma \in \mathbb{R}_+$  of the service times of all jobs arriving to an appropriately chosen subset of job classes  $q$ , so that from  $\sigma_j^q$  they become  $(1 + \gamma)\sigma_j^q$ ,  $j \in \mathbb{Z}$ , while the rest of the system parameters remain unchanged. This stressing idea builds on a similar concept used in a different context in [3].*

The pathwise evolution of the system under virtual queueing depends exclusively on the sample paths of the arrival processes  $\{\mathcal{S}_q; q \in \mathbf{Q}\}$ . Therefore, the system stressing induces a well defined transformation of every quantity associated with the system (e.g. the workloads of the virtual queues). In [8] it is shown that due to Proposition 2 the workloads of a stressed system pathwise dominate the workloads of a non-stressed one. Therefore, if a stressed version of the system under consideration is stable, its original version is also stable. Hence, in an appropriately stressed version of the system, all the processors need to cooperate, thus allowing us to directly establish the finiteness of the workloads in one dominating case, as opposed to the alternative approach of braking the proof into numerous subcases.

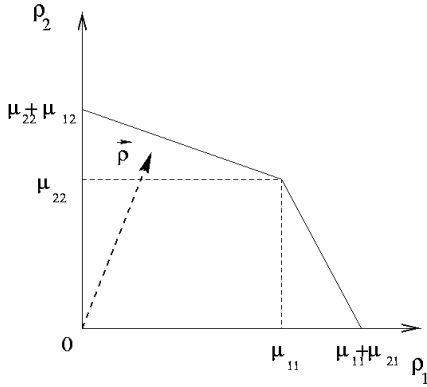


Fig. 4. An illustration of the working assumption in a system of two job classes and two servers. The working assumption implies that the traffic intensity vector  $\vec{\rho}$  should belong to the two triangles formed within the stability region  $\mathbf{S}$ .

**Proposition 3:** (Finiteness of the Stationary Workloads under policy  $\pi^*$ ). For any stationary and ergodic input processes  $\{\mathcal{S}_q, q \in \mathbf{Q}\}$ , if

$$\vec{\rho} \in \mathbf{S} \quad (12)$$

then

$$\tilde{W}_k^q(t) = \lim_{s \rightarrow -\infty} \mathcal{W}_k^q(s, t; \vec{0}) < \infty, \quad \forall t \in \mathbb{R}, \quad \text{for every } q, k \in \mathbf{Q}, \quad (13)$$

almost surely. Under this condition the processes  $\{\tilde{W}_k^q(t), t \in \mathbb{R}\}$ ,  $q, k \in \mathbf{Q}$  form a finite stationary operational regime of the system.

*Proof:* (sketch) In what follows we work on an arbitrarily fixed sample path as well as time  $t \in \mathbb{R}$ . Starting the system empty at time  $s < t$ , we define the random time

$$T_s = \inf\{z \in (s, t] : \text{no processor in } \mathbf{Q} \text{ idles in } [z, t]\}, \quad (14)$$

which is the last time (before  $t$ ) that all queues in  $\mathcal{S}_k$  for at least one  $k \in \mathbf{Q}$  are empty, due to the work conserving nature of the  $\pi^*$  policy. If none of the processors ever idles in  $(s, t]$ , we naturally set  $T_s = s$ , while if at least one processor is idling at time  $t$ , we set  $T_s = t$ . Definition (14) implies that at time  $T_s^-$  there exists at least one set of queues  $\mathcal{S}_k$  which are all empty. Due to Proposition 2,  $T_s$  decreases as  $s \rightarrow -\infty$ ; therefore,  $T_* = \lim_{s \rightarrow -\infty} T_s$  exists (but may be  $-\infty$ ). Arguing by contradiction it can be established that  $T_*$  is finite (details given in [8]).

In view of the above, all queues in  $\mathcal{S}_{k^*}$  are empty at time  $T_*^-$ . Since  $T_*$  is finite, there is only a finite amount of workload that can arrive to the queues in  $\mathcal{S}_{k^*}$  in the finite interval  $[T_*, t]$ . Therefore,

$$\tilde{\mathcal{W}}_k^q(t) < \infty, \quad \text{for every } q \in \mathcal{S}_{k^*}. \quad (15)$$

Using Fact 2 below, (15) implies that  $\tilde{\mathcal{W}}_k^q(t) < \infty$ , for every  $q \in \mathcal{S}_k$ , all  $k \in \mathbf{Q}$ , proving (13).

The stationarity of the  $\{\tilde{W}_k^q(t), q \in \mathcal{S}_k, k \in \mathbf{Q}\}$  processes follows from (10)). Moreover, observe that, due to the nature of the  $\pi^*$  policy,  $\mathcal{W}(s, t; \vec{w})$  is a continuous function of the initial workload  $\vec{w}$ , and is a piecewise continuous function of  $t$  between successive job arrival times. Therefore, it is easy to see that the evolution equations of the workload processes  $\mathcal{W}_k^q(s, t; \vec{0})$  are also satisfied by their limiting counterparts  $\tilde{\mathcal{W}}_k^q(t)$ ; hence, the latter constitute an *operational* regime of the system.

**Fact 1 :** Given  $\vec{\rho}$  that satisfies the working hypothesis, we have that on any sample path of  $\{\mathcal{S}_q, q \in \mathbf{Q}\}$  (almost surely), if  $\lim_{s \rightarrow -\infty} T_s = T_* = -\infty$ , then the following statement is true:

Given set of queues in system  $\mathcal{S}_k$ , if there exists some decreasing sequence  $\{s_a, a \in \mathbb{Z}_+\}$  with  $\lim_{k \rightarrow \infty} s_k = -\infty$ , such that

$$\lim_{a \rightarrow \infty} \frac{\sum_{q \in \mathcal{S}_k} \mathcal{W}_k^q(s_a, T_{s_a})}{t - T_{s_a}} = 0, \quad (16)$$

then for the systems  $\mathcal{S}_{k'}$ ,  $k' \neq k$  and a subsequence  $\{s_b, b \in \mathbb{Z}\}$  with  $\lim_{b \rightarrow \infty} s_b = -\infty$ , such that

$$\lim_{b \rightarrow \infty} \frac{\sum_{q \in \mathcal{S}_{k'}} \mathcal{W}_{k'}^q(s_b, T_{s_b})}{t - T_{s_b}} = 0, \quad (17)$$

*Proof of Fact 1:* The proof of Fact 1, due to its extensive length, is omitted and can be found in [8].

**Fact 2 :** On any sample path (almost surely) of the input  $\{\mathcal{S}_q, q \in \mathbf{Q}\}$ , the following statement is true for every  $t \in \mathbb{R}$ :

If the queues in system  $\mathcal{S}_k$  satisfy under the  $\pi^*$  policy

$$\lim_{s \rightarrow -\infty} \mathcal{W}_k^q(s, t; \vec{0}) = \tilde{\mathcal{W}}_k^q(t) < \infty, \quad \text{for every } q \in \mathcal{S}_k, \quad (18)$$

then

$$\widetilde{\mathcal{W}}_k^q(t) < \infty, \text{ for every } q \in \mathcal{S}_{k'}, k' \neq k, k' \in \mathbf{Q}. \quad (19)$$

*Proof of Fact 2:* The proof of Fact 2, due to its extensive length, is also omitted and can be found in [8].

This completes the proof of the Proposition 3. ■

*Remark 4.3:* (The Use of the Working Hypothesis) *Note that the main purpose of Fact 1 is to extend the local property (16) (summation over a particular set of virtual queues  $\mathcal{S}_k$ ) to a global one (17) (valid over all  $\mathcal{S}_k, k \in \mathbf{Q}$ ). Similarly, the main purpose of Fact 2 is to extend the local property (18) to a global one given by (19). The main tool in the proofs of both results is the fact that we are employing the working hypothesis, that forces the cooperation and interaction of all the servers, thus inducing structural relationships between the virtual queues of processor  $k$  and those of processor  $k' \neq k$ . This provides a key piece of intuition behind the omitted proofs of these two Facts.*

Given the proper (finite) nature of the stationary operational regime of the system when  $\vec{\rho} \in \mathbf{S}$  we can now establish that the workload process of the virtual queues converge to this stationary regime at large times.

**Proposition 4: (Stability under the  $\pi^*$  Policy)** For any stationary ergodic input processes  $\{\mathcal{S}_q, q \in \mathbf{Q}\}$ , if

$$\vec{\rho} \in \mathbf{S}$$

then the queueing state process  $\{\mathcal{W}_k^q(s,t)(\vec{0}), q \in \mathcal{S}_k, \text{ all } k \in \mathbf{Q}\}$  under the  $\pi^*$  policy converges in distribution to the proper stationary regime  $\{\widetilde{\mathcal{W}}_k^q(t), q \in \mathcal{S}_k, \text{ all } k \in \mathbf{Q}\}$  at large times. Therefore, the system can be described as **stable**.

*Proof:* Given in [8]. ■

## V. EXTENSIONS AND CONCLUDING REMARKS

A natural extension of the proposed model is to open, acyclic, multiclass networks of nodes having the structure of the system considered so far. A network of nodes, with the characteristics of our system, in tandem has been studied for packet switching applications in [5]. Proposition 4 together with Proposition 3 allows us to extend the results through an analogous but much more complicated Loynes [7] construction. Therefore, a finite stationary operational regime for a network can be established.

In this paper we have investigated the stability problem of a model comprised of  $Q$  parallel queues and an equal number of servers that can serve at different rates the queues, under general ergodic job arrival flows. By introducing the concept of virtual queueing we were able to establish convergence to a maximum throughput finite stationary regime for the workload processes operating under a joint routing/scheduling policy. Finally, the performance of the various maximum throughput adaptive policies discussed in this paper, such as SVW-MVW, SVW-MSR, SVW-WMVW, etc (see Remark 4.1), with respect to other metrics (average queue backlogs, average job delays, etc) is a topic of current investigations.

## VI. ACKNOWLEDGMENTS

This work was supported in part by NSF grant IIS-9988095.

## VII. REFERENCES

- [1] Armony, M. and Bambos, N. (1999), Queuing Networks with Interacting Service Resources, *Proceedings of the Allerton Conference*, Urbana, 42-51
- [2] Bambos, N. (1993), Scheduling and Stability Aspects of a General Class of Parallel Processing Systems, *Advances in Applied Probability*, 26, 176-202
- [3] Bambos, N. and Michailidis, G. (1995), On the Stationary Dynamics of Parallel Queues with Random Server Connectivities, *Proceedings of the 34th Conference on Decision and Control*, 3638-3643
- [4] Bell, S.L. and Williams, R.J. (2001), Dynamics Scheduling of a System with Two Parallel Servers in Heavy Traffic with Complete Resource Pooling: Asymptotic Optimality of a Continuous Review Threshold Policy, *Annals of Applied Probability*, 11, 608-649
- [5] Chang, C.S., Chen, W.J. and Huang, H.Y. (2001), Birkhoff-von Neumann Input Buffered Crossbar Switches for Guaranteed-Rate Services, *IEEE Transactions on Communications*, 49, 1145-1147
- [6] Gallager, R. and Parekh, A. (1993), A Generalized Processor Sharing Approach to Flow Control in Integrated Service Networks: The Simple Node Case, *ACM/IEEE Transactions on Networking*, 1, 344-352
- [7] Loynes, R.M. (1962), The Stability of a Queue with Non-independent Inter-arrival and Service Times, *Proceedings of the Cambridge Philosophical Society*, 58, 497-520
- [8] Michailidis, G. (2002), Queueing Systems with Shared Resources, Technical Report, Department of Statistics, The University of Michigan
- [9] Ni, L.M. and McKinley, P.K. (1993), A Survey of Wormhole Routing Techniques in Direct Networks, *Computer*, 26, 62-76
- [10] Petersen, K. (1983), *Ergodic Theory*, Cambridge University Press
- [11] Walters, P. (1982), *An Introduction to Ergodic Theory*, Springer Verlag
- [12] Wasserman, K.M. and Olsen, T.L. (2001), On Mutually Interfering Parallel Servers Subject to External Disturbances, *Operations Research*, 49, 700-709
- [13] Weiss, G. and Pinedo, M. (1980), Scheduling Tasks with Exponential Service Times on Non-identical Processors to Minimize Various Cost Functions, *Journal of Applied Probability*, 17, 187-202
- [14] Weber, R.R., Varaiya, P. and Walrand, J. (1986), Scheduling Jobs with Stochastic Processing Requirements on Parallel Machines to Minimize Makespan or Flowtime, *Journal of Applied Probability*, 23, 841-847