# Optimal Dynamic Multi-Server Allocation to Parallel Queues With Independent Random Connectivity

Hussein Al-Zubaidy
Systems and Computer Engineering
Carleton University

February 2009

**Abstract**

We investigate an optimal scheduling problem in a discrete-time system of $L$ parallel queues that are served by $K$ identical, randomly connected servers. This model has been widely used in studies of emerging 3G/4G wireless systems. We introduce the class of Most Balancing (MB) policies and provide their mathematical characterization. We prove that MB policies are optimal; we define optimality as minimization, in stochastic ordering sense, of a range of cost functions of the queue lengths, including the process of total number of packets in the system. We use stochastic coupling arguments for our proof. We propose an implementation algorithm for an MB policy. We also introduce the Least Connected Server First/Longest Connected Queue (LCSF/LCQ) policy as an easy-to-implement approximation of MB policies. The server-queue (channel) connectivities, during each time slot, are modeled by independent Bernoulli random variables. The exogenous arrivals to individual queues are assumed to be symmetrical and independent across the queues in the system and independent of the connectivities. We conduct a simulation study to compare the performance of several policies. The simulation results show that: (a) in all cases, LCSF/LCQ approximations to the MB policies outperform the other policies, (b) randomized policies perform fairly close to the optimal one, and, (c) the performance advantage of the optimal policy over the other simulated policies increases as the channel connectivity probability decreases.

# 1 Introduction, Model Description and Prior Research

Emerging 3G/4G wireless networks can be categorized as high speed IP-based packet access networks. They utilize the channel variability, using data rate adaptation, and user diversity to increase their channel capacity. These systems usually use a mixture of Time and Code Division Multiple Access (TDMA/CDMA). Time is divided to an equal size slots, each of which can be allocated to one or more users. To optimize the use of the enhanced data rate, these systems allow several users to share the wireless channel simultaneously using CDMA. This will minimize the wasted capacity resulted from the allocation of the whole channel capacity to one user at a time even when that user is unable to utilize all of that capacity. Another reason for sharing system capacity between several users, at the same time slot, is that some of the user equipments at the receiving side might have design limitation on the amount of data they can receive and process at a given time.

The connectivity of users to the base station in any wireless system is varying with time and can be best modeled as a random process. The application of stochastic modeling and queuing theory to model wireless systems is well vetted in the literature. Modeling wireless systems using parallel queues with random queue/server connectivity was used by Tassiulas and Ephremides [3], Ganti, Modiano and Tsitsiklis [6] and many others to study scheduler optimization in wireless systems. In the following subsection, we provide a more formal model description and motivation for the problem at hand.

## 1.1 Model Description

In this work, we assume that time is slotted into equal length deterministic intervals. We model the wireless system under investigation as a set of $L$ parallel queues with infinite capacity (see Figure 1); the queues correspond to the different users in the system. We define $X_i(n)$ to represent the number of packets in the $i^{th}$ queue at the beginning of time slot $n$. The queues share a set of $K$ identical servers, each server representing transmission channels (or any other network resource, e.g., power, CDMA codes, etc.). We make no assumption regarding the number of servers relative to the number of queues, i.e., $K$ can be less, equal or greater than $L$. The packets in this

system are assumed to have constant length, and require one time slot to complete service. A server can serve one packet only at any given time slot. A server can only serve connected, non-empty queues. Therefore, the system can serve up to $K$ packets during each time slot. Those packets may belong to one or several queues.

The connectivity between a user and a channel is random. The state of the channel connecting the $i^{th}$ queue to the $j^{th}$ server during the $n^{th}$ time slot is denoted by $G_{i,j}(n)$ and can be either connected ($G_{i,j}(n) = 1$) or not connected ($G_{i,j}(n) = 0$). Hence, in a real system $G_{i,j}(n)$ will determine if a transmission channel $j$ can be used by user $i$ or not. We assume that, for all $i = 1, 2, \ldots, L$, $j = 1, 2, \ldots, K$ and $n$, $G_{i,j}(n)$ are independent Bernoulli random variables with parameter $p$.

The number of arrivals to the $i^{th}$ queue during time slot $n$ is denoted by $Z_i(n)$. We make no assumption about the distribution of $Z_i(n)$, other than $P[Z_i(n) < \infty] = 1$; the random variables $Z_i(n)$ and $Z_i(n')$ may be dependent, for $n \neq n'$. However, we require that the distribution be the same for all $i$. We require that arrival processes to different queues be independent of each other; we also require that the random processes $\{Z_i(n)\}$ be independent of the processes $\{G_{i,j}(n)\}$ for $i = 1, 2, \ldots, L$, $j = 1, 2, \ldots, K$. These assumptions are necessary for the coupling arguments we use in our optimality proofs.

A scheduler (or server allocation or scheduling policy) decides, at the beginning of each time slot, what servers will be assigned to which queue during that time slot. The objective of this work is to identify and analyze the optimal scheduling policy that minimizes, in a stochastic ordering sense, a range of cost functions of the system queue sizes, including the total number of queued packets, in the aforementioned system. The choice of the class of cost functions and the minimization process are discussed in detail in Section 5.

## 1.2 Previous Work and Our Contributions

In the literature, there is substantial research effort focusing on the optimal server allocation in wireless networks. Tassiulas and Ephremides [3] for example, tackled a similar, simpler problem where a single server (i.e., $K = 1$) can only be allocated to one user and can only serve one packet at each time slot. They proved, using stochastic coupling argument, that LCQ (Longest Connected Queue) is optimal. In our work we show that LCQ is not always optimal in a multi-server system since servers can be assigned to one or

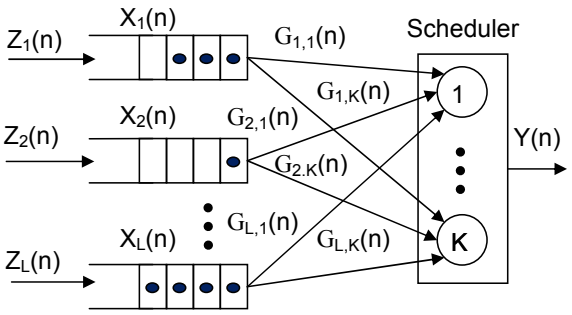



Figure 1: Abstraction of downlink scheduler in a 3G wireless network.

more queues simultaneously. Bambos and Michailidis [4] worked on a similar model (a continuous time version of [3] with finite buffer capacity) and found that under stationary ergodic input job flow and modulation processes, both MCW (Maximum Connected Workload) and LCQ dynamic allocation policies maximize the stability region for this system. Furthermore, they proved that C-FES, a policy that allocates the server to the connected queue with the fewest empty spaces, stochastically minimizes the loss flow and maximizes the throughput [5].

Another relevant result is that reported by Ganti, Modiano and Tsitsiklis [6]. They presented a model for a satellite node that has $K$ transmitters. The system was modeled by a set of parallel queues with symmetrical statistics competing for $K$ identical servers. At each time slot, no more than one server is allocated to each scheduled queue. They proved, using stochastic coupling arguments, that LCQ, a policy that allocates the $K$ servers to the $K$ longest connected queues at each time slot, is optimal. This model is similar to the one we consider in this work, except that in our model one or more servers can be allocated to each queue in the system. A further, stronger difference between the two models is that we consider the case where each queue has *independent connectivities* to different servers. We make these assumptions for a more suitable representation of the 3G wireless systems described earlier. These differences make it substantially harder to identify (and even describe) the optimal policy (see Section 3). A more recent result that has relevance to our work is the one reported by Kittipiyakul and Javidi in [7]. They proved, using dynamic programming, that a maximum-throughput and load-balancing (MTLB) policy minimizes the expected average cost for a two-queue multi-server system. In our research work we proved the optimality of the most balancing policies in the more general problem of a multi-queue

(more than two queues) and multi-server system with random channel connectivity. A stronger distinction of our work is that we proved the optimality in a stochastic ordering sense which is a stronger notion of optimality compared to the expected average cost criterion that was used in [7]. Lott and Teneketzis [8] investigated a multi-class system of $N$ weighted cost parallel queues and $M$ servers. They also used the same restriction of one server per queue used in [6]. They showed that an index rule is optimal and provided conditions sufficient, but not necessary, to guarantee its optimality.

Koole et al [9] studied a model similar to that of [3] and [5]. They found that the Best User (BU) policy maximizes the expected discounted number of successful transmissions. Liu et al [10], [11] studied the optimality of opportunistic schedulers (e.g., Proportional Fair (PF) scheduler). They presented the characteristics and optimality conditions for such schedulers. However, Andrews [13] showed that there are six different implementation algorithms of a PF scheduler, none of which is stable. For more information on resource allocation and optimization in wireless networks the reader may consult [12], [14], [15], [16], [17], and [18].

In summary, the main contributions of our work are the following:

- We introduce the class of Most Balancing (MB) policies for server allocation in the model of Figure 1 and prove their optimality for minimizing, in stochastic ordering sense, a set of functions of the queue lengths (e.g., total system occupancy).

- An MB policy attempts to balance all queue sizes at every time slot, so that the total sum of queue size differences will be minimized. Such a policy exist and may be determined through a finite search of all possible server allocations. In our work, we present a method that reduces this search by searching for policies that assign servers to the "longest connected queue" (LCQ allocation) in an ordered manner.

- We provide a heuristic approximation for an MB policy. At any time slot, such policies allocate the "least connected servers first" to their "longest connected queues" (LCSF/LCQ). These policies require minimum complexity $O(L \times K)$ for their implementation. We further show, using simulation, that their performance (on average) is identical to the one achieved by MB policies.

The rest of the paper is organized as follows. In section II, we introduce notation and define the server allocation policies. In section III, we intro-

duce and provide a detailed description of the MB server allocation policies. We also present an implementation algorithm for such policies. In section IV, we introduce and characterize the balancing interchange. In section V, we present the main result, i.e., the optimality of MB policies. In section VI, we present the Least Balancing (LB) policies, and show that these policies perform the worst among all work conserving policies. MB and LB policies provide upper and lower performance bounds. In section VII, we introduce a practical low-overhead approximations for such policies, namely the LCSF/LCQ policy and the MCSF/SCQ policy, with their implementation algorithms. In section VIII, we present simulation results for different scheduling policies. We present proofs for some of our results in the Appendix.

# 2 Policies for Server Allocation

Recall that $L$ and $K$ denote the number of queues and servers respectively in the model introduced in Figure 1. We will use **bold face**, UPPER CASE and lower case letters to represent vector/matrix quantities, random variables and sample values respectively. In order to represent the policy action that corresponds to "idling" a server, we introduce a special, "dummy" queue which is denoted as queue 0. Allocating a server to this queue is equivalent to idling that server. By default, queue 0 is permanently connected to all servers, contains only "dummy" packets. Let $\mathbb{1}_{\{A\}}$ denotes the indicator function for condition $A$. Throughout this paper, we will use the following notation:

- $\mathbf{G}(n)$ is an $(L+1) \times K$ matrix, where $G_{i,j}(n)$ for $i > 0$ is the channel connectivity random variable as defined in Section 1. We assume that $G_{0,j}(n) = 1$ for all $j, n$.
- $\mathbf{X}(n) = (X_0(n), X_1(n), X_2(n), \ldots, X_L(n))^T$ is the vector of queue lengths at the beginning of time slot $n$, measured in number of packets. We assume $X_0(1) = 0$.
- $\mathbf{Q}(n) = (Q_1(n), \ldots, Q_K(n))^T$ is the server allocation control vector. $Q_j(n) \in \{0, 1, \ldots, L\}$ denotes the index of the queue that is selected (according to some rule) to be served by server $j$ during time slot $n$. Note that serving the "dummy" queue, i.e., setting $Q_j(n) = 0$ means that server $j$ is idling during time slot $n$.

- $\mathbf{V}(n)$ is a $(L+1) \times K$ matrix such that $V_{i,j}(n) = \mathbb{1}_{\{i=Q_j(n)\}} \cdot G_{i,j}(n), i = 0, \ldots, L$ and $j = 1, \ldots, K$. Hence $V_{i,j}(n)$ will be equal to 1 iff server $j$ is both connected to queue $i$ and assigned to serve it.

- $\mathbf{Y}(n) = (Y_0(n), Y_1(n), Y_2(n), \ldots, Y_L(n))^T$ is the vector of the number of packets withdrawn from the system during time slot $n$. For any $i$, $Y_i(n) \in \{0, 1, \ldots, K\}$ denotes the number of packets withdrawn from queue $i$ (and assigned to servers) during time slot $n$.

- $\mathbf{Z}(n) = (Z_0(n), Z_1(n), Z_2(n), \ldots, Z_L(n))^T$ is the (column) vector of the number of exogenous arrivals during time slot $n = 1, 2, \ldots$. Arrivals to queue $i \neq 0$ are as defined in Section 1. We let $Z_0(n) = Y_0(n)$.

- The tuple $(\mathbf{X}(n), \mathbf{G}(n))$ denotes the "state" of the system at the beginning of time slot $n$.

For future reference, we will call $\mathbf{Q}(n)$ the *scheduling (or server allocation) control* and $\mathbf{Y}(n)$ the *withdrawal control*. The matrix $\mathbf{V}(n)$ will be useful in describing feasibility constraints on such controls (see Equations (2) and (3)).

## 2.1 Feasible Scheduling and Withdrawal Controls

Using the previous notation and given a scheduling control vector $\mathbf{Q}(n)$ we can compute the withdrawal control vector as:

$$Y_i(n) = \sum_{j=1}^{K} \mathbb{1}_{\{i=Q_j(n)\}}, \quad i = 0, 1, 2, \ldots, L. \tag{1}$$

We assume that the controller has complete knowledge of the system state information at the beginning of each time slot. Then we say that a given vector $\mathbf{Q}(n) \in \{0, 1, \ldots, L\}^K$ is a *feasible* scheduling control (during time slot $n$) if: (a) a server is allocated to one connected queue, and, (b) the number of servers allocated to a queue (dummy queue excluded) cannot exceed the size of the queue at time $n$. Mathematically, these conditions are captured by the following (necessary and sufficient) constraints:

$$\mathbf{V}^T(n) \cdot \mathbf{I}_{L+1} = \mathbf{I}_K \tag{2}$$

$$\mathbf{V}^*(n) \cdot \mathbf{I}_K \leq \mathbf{X}(n) \tag{3}$$

where $\mathbf{I}_l$ is a column vector of size $l$, with all entries equal to one, and

$$V^*_{i,j}(n) = \begin{cases} 0, & i = 0; \\ V_{i,j}(n), & \text{otherwise.} \end{cases}$$

The $K$ constraints in Equation (2) capture condition (a) above; indeed, equality in Equation (2) is not possible if a server $j$ is allocated to a non-connected queue, since $V_{i,j}(n) = 0$ for all $i$ in this case. The point-wise inequality in Inequality (3) captures condition (b); with the choice of $\mathbf{V}^*(n)$ we guarantee that Inequality (3) is satisfied for the dummy queue. Note that more than one server may be allocated to any queue.

Similarly, we say that a vector $\mathbf{Y}(n) \in \{0, 1, \ldots, K\}^{L+1}$ is a *feasible* withdrawal control (during time slot $n$) if there is a matrix $\mathbf{V}(n)$ that satisfies the feasibility constraints (2) and (3) such that

$$\mathbf{Y}(n) = \mathbf{V}(n) \cdot \mathbf{I}_K \tag{4}$$

From constraints (2), it is clear that a server can be allocated to one and only one connected queue; summing the constraints in (2), we can see that the controller can only withdraw a total of up to $K$ packets from the connected nonempty queues in the system. For any feasible $\mathbf{Y}(n)$, from the definition of $V_{i,j}(n)$ and Inequality (3) it follows that, at any time slot, the number of packets withdrawn from any queue cannot be larger than the size of the queue or larger than the total number of servers connected to the queue. Therefore, a feasible withdrawal control $\mathbf{Y}(n)$ satisfies the (necessary) conditions

$$0 \leq Y_i(n) \leq \min\left(X_i(n), \sum_{j=1}^{K} G_{i,j}(n)\right), \forall n, i \neq 0, \tag{5}$$

$$\sum_{i=0}^{L} Y_i(n) = K, \quad \forall n. \tag{6}$$

It is clear that conditions (5) and (6) are not sufficient for the feasibility of the withdrawal vector $\mathbf{Y}(n)$.

For future reference, we denote the set of all feasible withdrawal controls while in state $(\mathbf{x}, \mathbf{g})$ by $\mathcal{Y}(\mathbf{x}, \mathbf{g})$.

Note from Equation (1) that, given a feasible scheduling control $\mathbf{Q}(n)$, the withdrawal control $\mathbf{Y}(n)$ is determined uniquely and is feasible (Equations

(4) and (1) are the same, for a feasible scheduling control). However, for a given system state Equation (4) may have more than one solution, i.e., $\mathbf{V}(n)$ (and hence $\mathbf{Q}(n)$), that satisfy Equations (2) and (3). The feasibilities of withdrawal control and scheduling control are entwined and by definition imply each other. Nevertheless, Deriving $\mathbf{Q}(n)$ from a feasible withdrawal control is not straightforward. One way to do that is to devise an algorithm that searches through all possible scheduling vectors to find one that satisfies Equation (4). Building such algorithm is out of the scope of this paper.

For the rest of this paper, we will refer to $\mathbf{q}(n)$ as an *implementation* of the given feasible control $\mathbf{y}(n)$.

## 2.2 Definition of Policies for Server Allocation

For any feasible control ($\mathbf{Y}(n)$), the system described previously evolves according to

$$\mathbf{X}(n+1) = \mathbf{X}(n) - \mathbf{Y}(n) + \mathbf{Z}(n), \qquad n = 1, 2, \ldots \tag{7}$$

We assume that arrivals during time slot $n$ are added after removing served packets. Therefore, packets that arrive during time slot $n$ have no effect on the controller decision at that time slot and may only be withdrawn during $t = n + 1$ or later. In order to ensure that $X_0(n) = 0$ for all $n$, we define $Z_0(n) = Y_0(n)$. We define controller policies more formally next.

A *server allocation policy* $\pi$ (or policy $\pi$ for simplicity) is a rule that determines feasible withdrawal vectors $\mathbf{Y}(n)$ for all $n$, as a function of the past history and current state of the system $\mathbf{H}(n)$. The state history is given by the sequence of random variables

$$\begin{aligned} \mathbf{H}(1) &= (\mathbf{X}(1)), \quad \text{and} \\ \mathbf{H}(n) &= (\mathbf{X}(1), \mathbf{G}(1), \mathbf{Z}(1), \ldots, \mathbf{G}(n-1), \mathbf{Z}(n-1), \mathbf{G}(n)), \\ &\qquad n = 2, 3, \ldots \end{aligned} \tag{8}$$

Let $\mathcal{H}_n$ be the set of all histories up to time slot $n$. Then a policy $\pi$ can be formally defined as the sequence of measurable functions

$$\begin{aligned} & u_n : \mathcal{H}_n \longmapsto \mathcal{Z}_+^{L+1}, \\ \text{s.t.} \quad & u_n(\mathbf{H}(n)) \in \mathcal{Y}(\mathbf{X}(n), \mathbf{G}(n)), \quad n = 1, 2, \ldots \end{aligned} \tag{9}$$

where $\mathcal{Z}_+$ is the set of non-negative integers and $\mathcal{Z}_+^{L+1} = \mathcal{Z}_+ \times \ldots \times \mathcal{Z}_+$, where the Cartesian product is taken $L + 1$ times.

At each time slot, the following sequence of events happens: first, the connectivities $\mathbf{G}(n)$ and the queue lengths $\mathbf{X}(n)$ are observed. Second, the packet withdrawal vector $\mathbf{Y}(n)$ is determined according to a given policy. Finally, the new arrivals $\mathbf{Z}(n)$ are added to determine the next queue length vector $\mathbf{X}(n+1)$.

We denote by $\Pi$ the set of all policies described by Equation (9). The goal of this work, as it will be shown in Section 5, is to prove that policies that belong to the class of *Most Balancing* (MB) policies, that we introduce next, are optimal: they minimize (in the stochastic ordering sense) a range of cost functions including the total number of packets in the system.

# 3 The Class of MB Policies

In this section, we provide a description and mathematical characterization of the class of MB policies.

Intuitively, the MB policies "attempt to minimize the queue length differences in the system at every time slot $n$".

For a more formal definition of MB policies, we first define the following:

Given a state $(\mathbf{x}(n), \mathbf{g}(n))$ and a policy $\pi$ that chooses the feasible control $\mathbf{y}(n)$ at time slot $n$, define the "updated queue size" $\hat{x}_i(n) = x_i(n) - y_i(n)$ as the size of queue $i, i = 0, 1, \ldots, L$, after applying the control $y_i(n)$ and just before adding the arrivals during time slot $n$. Note that because we let $z_0(n) = y_0(n)$, we have $\hat{x}_0(n) \in \mathcal{Z}$, i.e., we allow $\hat{x}_0(n)$ to be negative. Furthermore, we define the *"imbalance index"*, $\kappa_n(\pi)$, as the total sum of differences of the $L+1$-dimensional vector $\hat{x}(n)$ under the policy $\pi$ at time slot $n$ (where $\pi$ takes the control $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})$ at time slot $n$), i.e.,

$$\kappa_n(\pi) = \sum_{i=1}^{L+1} \sum_{j=i+1}^{L+1} (\hat{x}_{[i]}(n) - \hat{x}_{[j]}(n)) \tag{10}$$

where $[k]$ denotes the index of the $k^{th}$ longest queue after applying the control $\mathbf{y}(n)$ and before adding the arrivals at time slot $n$. By convention, queue '0' (the "dummy queue") will always have order $L+1$ (i.e., the queue with the minimum length). It follows from Equation (10) that the minimum possible imbalance index is $L \cdot \hat{x}_{[L]}$ (i.e., all $L$ queues have the same length which is equal to the shortest queue length) which is indicative of a fully

balanced system. Let $\Pi^{MB}$ denotes the set of all MB policies, then we define the elements of this set as follows[1]:

**Definition:** A *Most Balancing* (MB) policy is a policy $\pi \in \Pi^{MB}$ that, at $n = 1, 2, \ldots$, chooses feasible withdrawal vectors $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})$ such that the imbalance index is minimized at every $n$, i.e.,

$$\Pi^{MB} = \left\{\pi : \underset{\mathbf{y}(n)\in\mathcal{Y}(\mathbf{x},\mathbf{g})}{\operatorname{argmin}} \ \kappa_n(\pi), \quad \forall n\right\} \tag{11}$$

The set $\Pi^{MB}$ in Equation (11) is well-defined and non-empty, since the minimization is over a finite set.

The set of MB policies may have more than one element. This could happen, for example, when at a given time slot $n$, a server $k$ is connected to two or more queues of equal size, which happen to be the longest queues connected to this server. Then, serving either one of them will satisfy Equation (11) even if these allocations result in different $\mathbf{Y}(n)$ (i.e., different policies). The resulting queue length vectors ($\hat{x}(n)$) under any of these policies will be permutations of each other.

**Remark 1.** *Note that the LCQ policy in [3] is a most balancing (MB) policy for $K = 1$ (i.e., the one server system presented in [3]). Extension of LCQ to $K > 1$ (i.e., allocating all the servers to the longest queue) may not result in a MB policy.* □

A construction of an MB policy given $\mathbf{X}(t)$ and $\mathbf{G}(t)$ can be done using a direct search over all possible server allocations. This can be a challenging computational task for larger $L$ and $K$. In Section 7, we provide a low-complexity heuristic algorithm (LCSF/LCQ) to approximate MB policies. Our simulation study showed that the LCSF/LCQ performance is statistically indistinguishable from that of the MB policy.

## 3.1 Possible Implementation of MB Policies

A determination of an MB policy given $\mathbf{X}(t)$ and $\mathbf{G}(t)$ can be done using a direct search over all possible server allocations. This can be a challenging computational task. In what follows we present a more efficient approach for the construction of an MB policy.

[1]The max-min formulation will not work for our model because of the independent random server/queue connectivity. A queue maybe connected to a subset of the servers and not connected to the others at any given time slot.

We consider a given permutation (ordering) of the $K$ servers in the system. For this permutation we define a "sequential LCQ server allocation" a process of allocating the servers to queues in $K$ steps as follows: Starting from the first server, we assign it to its longest connected queue and we update (i.e., reduce by one) its queue size. We continue with the second server following the same principle until we exhaust all servers in $K$ steps. There are $K!$ server orderings that we have to consider. We will show that at least one "sequential LCQ server allocation" corresponding to an ordering among the $K!$ server permutations will result in an MB policy.

**Algorithm 1** (MB Policy Implementation)**.**

1. $for \quad t = 1, 2, \ldots \quad do \quad \Big\{$
2. *Input:* $\mathbf{X}(t), \mathbf{G}(t)$. *Calculate:* $\mathbb{Q}_{[k]}$, $k = 1, \ldots, K$.
3. *Let:* $\kappa_t^{min} = L \cdot \max\limits_{l} X_l$ *; maximum possible* $\kappa_t$
4. $forall \quad \theta \in \Theta \quad do \quad \Big\{$ *; loop* $|\Theta| = K!$ *times*
5. $\mathbf{X}' \longleftarrow \mathbf{X}(t)$, $\mathbf{Y}' \longleftarrow \mathbf{0}$, $\mathbf{Q}' \longleftarrow \mathbf{0}$
6. $for \quad j = 1 \quad to \quad K \Big\{$ *; allocate servers sequentially*
7. $Q'_{[j]^\theta} = \min \left( k : k \in \left\{ \underset{l: l \in \mathbb{Q}_{[j]^\theta}}{\operatorname{argmax}} (X'_l | X'_l > 0) \right\} \right)$
8. *Let:* $i = Q'_{[j]^\theta}$
9. $Y'_i = Y'_i + 1$
10. $X'_i = X'_i(t) - 1 \quad \Big\}$
11. *Compute:* $\kappa_t^\theta \quad$ *from Equation(10)*
12. $if \quad (\kappa_t^\theta < \kappa_t^{min}) \quad \Big\{$
13. $\kappa_t^{min} = \kappa_t^\theta$
14. $\mathbf{y}(t) \longleftarrow \mathbf{Y}'$, $\mathbf{q}(t) \longleftarrow \mathbf{Q}'$, $\theta(t) \longleftarrow \theta \Big\}$
15. $\Big\} \quad \Big\} \quad$ *; End of Algorithm 1.*

Algorithm 1 present the pseudo-code for the approach we described previously. We introduce the following notation:

We define the set $\mathcal{M}_i^t$ as the set of servers connected to queue $i$ during time slot $t$. Let $M_i(t) \triangleq |\mathcal{M}_i^t|$ be the number of servers that are connected to queue $i$ during time slot $t$, that is

$$M_i(t) = \sum_{j=1}^{K} G_{i,j}(t) \tag{12}$$

Let $\mathbb{Q}_k \triangleq \{i : k \in \mathcal{M}_i^t\}$ denote the set of queues that are connected to server $k$ during time slot $t$; we omit the dependence on $t$ to simplify notation. Let $\Theta$ denote the set of all possible permutations of the set $\{1, \ldots, K\}$. We define a server ordering at time $n$ as a permutation $\theta(n) \in \Theta$. There are $|\Theta| = K!$ possible server orderings. We use the subscript $[j]^\theta$ to denote the $j^{th}$ server to be allocated under the ordering rule $\theta(n)$.

The algorithm choses a servers' order $\theta \in \Theta$, then allocates the $K$ servers sequentially (according to the selected order) to their Longest Connected Queue (LCQ allocation). The resulted server allocation vector $\mathbf{q}(t)$ depends on the order $\theta$. Therefore, different servers' orderings may result in different vectors $\mathbf{q}(t)$. The algorithm computes $\kappa_t(\pi)$ for every server allocation vector $\mathbf{q}(t)$. It searches through all the $K!$ different permutations of the servers' order. Then it selects the ordering rule out of the set of all possible orders (i.e., the set $\Theta$) that results in the minimum $\kappa_t(\pi)$, and report the outputs $\theta(t)$, $\mathbf{q}(t)$ and $\mathbf{y}(t)$.

**Theorem 1.** *The server allocation policy obtained by applying Algorithm 1 is an MB policy.*

*Proof.* A policy $\pi$ is an MB policy if it has the MB property at every time slot $n = 1, 2, \ldots$, i.e., it minimizes the total differences between the queues' lengths in the system at every $n$. To prove Theorem 1 we have to show that Algorithm 1 produces a policy that has the MB property at all time slots.

For the proof of Theorem 1 we first introduce the following properties of a server in a sequential server allocation during time slot $n$ (where the queue lengths are updated after each server allocation) such as that used in Algorithm 1. Given a server allocation policy[2] $\pi^\theta$ and following a server

[2]By $\pi^\theta$ we denote a policy that is implemented using a sequential server allocation following the order $\theta \in \Theta$

ordering $\theta$, we define the allocation of a server to its longest connected queue (according to $\pi^\theta$) as "LCQ allocation" (equivalently we may say that the server has the LCQ property). Otherwise, we refer to this allocation as "NLCQ allocation" or we say that the server has NLCQ property. We should note that the LCQ or NLCQ properties of the servers depend on the selected server allocation order $\theta$.

Proceeding with the proof for Theorem 1, we consider a policy $\pi^{\theta_1}$, that is implemented using sequential server allocation and a server order $\theta_1$ during time slot $n$, such that $\pi^{\theta_1}$ has the MB property at time slot $n$. We assume that at least one server has the NLCQ property (i.e., allocated to a queue that is not its longest connected queue) according to this implementation, since otherwise the theorem is trivially true. We will show next that we can construct a server allocation ordering $\theta_2$ under which $\pi^{\theta_2}$ has the MB property at time slot $n$ and all servers have the LCQ property under $\theta_2$. Toward this end, we need the following lemmas that are proved in Appendix A.2:

**Lemma 1.** *Given a server allocation ordering $\theta \in \Theta$ at time slot $n$ and a policy $\pi^\theta$ that has the MB property at time slot $n$, if $s_{[K]^\theta}$ (i.e., the last allocated server under $\theta$ at time slot $n$) has the NLCQ property then a policy $\pi^*$ can be constructed in which: (i) $s^*_{[K]^\theta}$ (i.e., the last server to be allocated under $\pi^*$) has the LCQ property at time slot $n$ and (ii) $s^*_{[k]^\theta}$ has the same allocation as $s_{[k]^\theta}, \forall k : 1 \leq k < K$, such that $\pi^*$ has the MB property at time slot $n$.*

**Lemma 2.** *Given a policy $\pi^\theta$ that has the MB property during time slot $n$, swapping the order of two consecutively allocated servers $s_1, s_2$ under $\pi^\theta$, the second of which ($s_2$) has the LCQ property, will not change the LCQ property of that server under the new ordering.*

The construction of the new ordering $\theta_2$ during time slot $n$ (as described earlier) is summarized in the following steps:

(1) We identify the last NLCQ allocated server ($s_i$) under $\pi^{\theta_1}$. Denote the order (in $\theta_1$) of this server by $i^*$ (i.e., $s_i = s_{[i^*]^{\theta_1}}$). Now if this is the last server to be allocated (i.e., $i^* = K$), then go to step (4).

(2) Using Lemma 2 we can swap server $s_i$ with the server next in order, i.e., $s_{[i^*+1]^{\theta_1}}$, (which has the LCQ property according to step (1)) and create a new server ordering $\theta'_1$, in which the swapped server has order $[i^*]^{\theta'_1}$ and retain it's LCQ property.

(3) Repeat step (2) until $s_i$ is the last server to be allocated under $\theta_1'$.

(4) Allocate server $s_i$ to its longest connected queue. According to Lemma 1 the resulting policy will have the MB property at time slot $n$ (with the last server has LCQ property)

(5) Repeat steps (1) to (4) until all servers have the LCQ property. This will result in a new ordering $\theta_2$ and a new policy $\pi^{\theta_2}$ that has the MB property at time slot $n$ with all servers having the LCQ property under $\theta_2$ at the corresponding time slot. □

# 4 Balancing Interchanges

In this section, we introduce the notion of "balancing interchanges". Intuitively, an interchange $I(f,t)$ between two queues, $f$ and $t$, describes the action of withdrawing a packet from queue $f$ instead of queue $t$ (see Equations (15) and (16)). Such interchanges are used to relate the imbalance indices of various policies (see Equation 28); balancing interchanges improve (i.e, lower) the imbalance indices and thus provide a means to describe how a policy can be enhanced. Interchanges can be implemented via server reallocation . Since there are $K$ servers, it is intuitive that at most $K$ balancing interchanges should suffice to convert any arbitrary policy to an MB policy; this is the crux of Lemma 5, the main result of this section.

## 4.1 Two-queue packet interchanges

Let $f \in \{0,1,\ldots,L\},\ t \in \{0,1,\ldots,L\}$ represent the indices of two queues that we refer to as the 'from' and 'to' queues. Define the $(L+1) \times 1$-dimensional vector $I(f,t)$, whose $j$-th element is given by[3]:

$$I_j(f,t) \;=\; \begin{cases} 0, & t = f; \\ +1, & j = f,\ f \neq t; \\ -1, & j = t,\ t \neq f; \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Fix an initial state $\mathbf{x}(n)$ at time slot $n$; consider a policy $\pi$ with a (feasible) withdrawal vector $\mathbf{y}(n)$. Let

$$\mathbf{y}^*(n) = \mathbf{y}(n) + I(f,t), \quad f \neq t, \tag{14}$$

[3]In other words, $I(f,t)$ represents an operation of removing a packet 'from' queue $f$ and adding it 'to' queue $t$.

be another withdrawal vector. The two vectors $\mathbf{y}(n), \mathbf{y}^*(n)$ differ only in the two components $t, f$; under the withdrawal vector $\mathbf{y}^*(n)$, an additional packet is removed from queue $f$, while one packet less is removed from queue $t$. Note that either $t$ or $f$ can be the dummy queue. In other words,

$$\begin{aligned} y_f^*(n) &= y_f(n) + 1 && (15)\\ y_t^*(n) &= y_t(n) - 1 && (16)\\ y_i^*(n) &= y_i(n), \quad \forall i \neq f, t. && (17) \end{aligned}$$

In the sequel, we will call $I(f,t)$ an *interchange* between queues $f$ and $t$. We will call $I(f,t)$ a *feasible interchange* if it results in a feasible withdrawal vector $\mathbf{y}^*(n)$. From Equations (7) and (15) – (17), it is clear that the $I(f,t)$ interchange will result in a new vector $\hat{\mathbf{x}}^*(n)$ such that:

$$\begin{aligned} \hat{x}_f^*(n) &= \hat{x}_f(n) - 1, \quad f \in \{0, 1, \ldots L\} && (18)\\ \hat{x}_t^*(n) &= \hat{x}_t(n) + 1, \quad t \in \{0, 1, \ldots L\} && (19)\\ \hat{x}_i^*(n) &= \hat{x}_i(n), \quad \forall i \neq f, t; \quad i \in \{0, 1, \ldots L\} && (20) \end{aligned}$$

or, in vector notation,

$$\hat{\mathbf{x}}^*(n) = \hat{\mathbf{x}}(n) - I(f,t), \quad f \neq t. \tag{21}$$

## 4.2 Feasible Server Reallocation

Given the state $(\mathbf{x}, \mathbf{g})$, let $\mathbf{y}(n)$ be any feasible withdrawal vector at time slot $n$. We define a "feasible server reallocation" as the reallocation of a server $k$ to a connected, non-empty queue $i$ (i.e., $g_{i,k}(n) = 1$ and $\hat{x}_i(n) > 0$). Any feasible server reallocation results into a feasible interchange. However, the reverse may not be true, e.g., a feasible packet interchange may result from a sequence of feasible server reallocations among several queues $(r_1, r_2, \ldots, r_{m+1})$ as demonstrated in Figure 2. This will be detailed in the following section.

## 4.3 Implementation of feasible two-queue packet interchange

The interchange $I(f,t)$ in Equation (14) can be implemented via a series of $m$, $1 \leq m \leq K$ *feasible server reallocations*. For example, suppose that

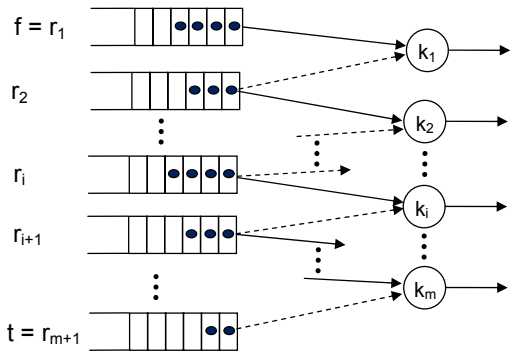



Figure 2: A sequence of $m$ server interchanges (reallocations) results in a feasible packet interchange $I(f,t)$. The dotted line denotes original server allocation. The solid line denotes server reallocation that implements $I(f,t)$.

a server $k$ is connected to both queues $t$ and $f$ (i.e., $g_{f,k}(n) \cdot g_{t,k}(n) = 1$), and that the server is allocated to queue $t$, under $\mathbf{y}(n)$ (i.e., $\mathbb{1}_{\{q_k(n)=t\}} = 1$ and $\hat{x}_f(n) \geq 1$). Then reallocating server $k$ to queue $f$ will result in the interchange $I(f,t)$ in Equation (14); this is the case $m = 1$.

Note that the condition $g_{f,k}(n) \cdot g_{t,k}(n) \cdot \mathbb{1}_{\{q_k(n)=t\}} = 1$ and $\hat{x}_f(n) \geq 1$ is sufficient but not necessary for $I(f,t)$ to be feasible. This is shown in Figure 2 where we introduce a series of $m$ server reallocations required to implement a feasible $I(f,t)$ following a suitable sequence of indices for the queues involved in that interchange. Let $\mathbf{r} \in \mathcal{Z}^{m+1}$ be such a sequence, where $r_1 = f$ and $r_{m+1} = t$. Let $k_i : k_i \in \{1, 2, \ldots, K\}$ be the server reallocated to queue $r_i$ from queue $r_{i+1}$.

For the interchange operation of Equation (14), the following are sufficient feasibility constraints:

$$\sum_{i=1}^{m} g_{r_i,k_i}(n) \cdot g_{r_{i+1},k_i}(n) \cdot \mathbb{1}_{\{q_{k_i}(n)=r_{i+1}\}} = m, \tag{22}$$

$$\hat{x}_f(n) \geq 1, \quad \text{if } f \in \{1, 2, \ldots, L\}, \tag{23}$$

$$\hat{x}_i(n) \geq 0, \quad \text{if } \forall i \neq f, i \in \{1, 2, \ldots, L\} \tag{24}$$

$$\hat{x}_0(n) \leq 0, \tag{25}$$

for some integer $m \geq 1$ and $\mathbf{r} \in \mathcal{Z}^{m+1}$.

Constraint (22) is sufficient to ensure that connectivity conditions allow for the series of $m$ server reallocations. Server $k$ reallocation to queue $j$ is

feasible only if queue $j$ is non-empty (i.e., $\hat{x}_j(n) \geq 1$) and is connected to server $k$ (i.e., $g_{j,k}(n) = 1$). Furthermore, the feasibility of $\mathbf{y}^*(n)$ implies that constraint (22) must hold for at least one $m \geq 1$ and one $\mathbf{r} \in \mathcal{Z}^{m+1}$.

Constraint (23) is necessary since a packet will be removed from queue $f$ to be added to queue $t$, therefore, queue $f$ must contain at least one packet for the interchange to be feasible. The remaining queues may be empty. The sequence of intermediate interchanges starts by removing a packet from queue $f = r_1$ and adding a packet to queue $r_2$. Therefore, constraints (22) – (25) insure that queue $r_2$ will contain at least one packet for the second intermediate server reallocation to be feasible even when $\hat{x}_{r_2}(n) = 0$. Same is true for any queue $r_i, i \in \{2, 3, \ldots m\}$. Therefore, these constraints are also sufficient for feasibility of (14).

## 4.4 "Balancing" two-queue packet interchanges

**Definition:** A feasible interchange $I(f,t)$ is "*balancing*" if

$$\hat{x}_f(n) \geq \hat{x}_t(n) + 1 \tag{26}$$

$I(f,t)$ is "*unbalancing*" if

$$\hat{x}_f(n) \leq \hat{x}_t(n) \tag{27}$$

Balancing interchanges result in policies that reduce the imbalance index, as the following lemma states.

**Lemma 3.** *Consider two policies $\pi^*$ and $\pi$, related via the balancing interchange*

$$\mathbf{y}^*(n) = \mathbf{y}(n) + I(f,t).$$

*The imbalance indices are related via*

$$\kappa_n(\pi^*) = \kappa_n(\pi) - 2(s-l) \cdot \mathbb{1}_{\{\hat{x}_{[l]}(n) \geq \hat{x}_{[s]}(n)+2\}} \tag{28}$$

*where $l$ (respectively $s$) is the order of queue $f$ (respectively $t$) in $\hat{x}(n)$ when ordered in descending order*[4].

[4]Intuitively, we use $s$ (respectively $l$) to refer to the order of the "shorter" (respectively the "longer") queue of the two queues used in the interchange.

The proof is a direct consequence of Lemma A.1.1 in Appendix A.1 and the fact that, by definition of the balancing interchange, we have $s > l$.

In words, Equation (28) states that an interchange $I(f,t)$, when balancing, results in a cost reduction of $2(s-l)$ when $\hat{x}_f(n) = \hat{x}_{[l]}(n) \geq \hat{x}_{[s]}(n)+2 = \hat{x}_t(n)+2$ and keeps it unchanged otherwise, i.e., when $\hat{x}_f(n) = \hat{x}_t(n)+1$. The latter case agrees with intuition, since the balancing interchange in this case will result in permuting the lengths of queues $f$ and $t$ which does not change the total sum of differences (and hence the imbalance index) in the resulting queue length vector.

## 4.5 The difference vector D

The vector $\mathbf{D}$ defined below (Equation (29)) is a measure of how much an arbitrary policy $\pi$ differs from a given MB policy during a given time slot $n$.

**Definition:** Consider a given state $(\mathbf{x}(n), \mathbf{g}(n))$ and a policy $\pi \in \Pi_{n-1}$ that chooses the feasible withdrawal vector $\mathbf{y}(n)$ during time slot $n$. Let $\mathbf{y}^{MB}(n)$ be a withdrawal vector chosen by an MB policy during the same time slot $n$. We define the $(L+1) \times 1$-dimensional vector $\mathbf{D} \in \mathcal{Z}^{L+1}$ as

$$\mathbf{D} = \mathbf{y}^{MB}(n) - \mathbf{y}(n), \tag{29}$$

where, for notational simplicity, we omit the dependence of $\mathbf{D}$ on the policy $\pi$, and the time index. Intuitively, if element $D_f$ of vector $\mathbf{D}$ is positive, this indicates that more packets than necessary have been removed from queue $f$ under policy $\pi$. Lemma 4 provides a way to systematically select balancing (and hence improving) interchanges. Lemma 5 provides a bound on the number of interchanges needed to convert any policy into an MB one. The proofs of the two lemmas are given in Appendix A.3.

**Lemma 4.** *Consider a given state $(\mathbf{x}(n), \mathbf{g}(n))$ and a feasible withdrawal vector $\mathbf{y}(n)$. Any feasible interchange $I(f,t)$ with indices $f$ and $t$ such that $D_f \geq +1$, $D_t \leq -1$ is a balancing interchange.*

For any fixed $n \geq 1$, let $\Pi_n$ denote the set of policies that have the MB property at time slots $t = 1, 2, \ldots, n$. We can easily see that these sets form a monotone sequence, with $\Pi_n \subseteq \Pi_{n-1}$. Then the set $\Pi^{MB}$ in Equation (11) can be defined as $\Pi^{MB} = \bigcap_{n=1}^{\infty} \Pi_n$. Furthermore, we denote $h = \sum_{i=0}^{L} |D_i|/2$. We will show in the Appendix (equation (A-50)) that $h$ is integer valued and $0 \leq h \leq K$. Consider a sequence of balancing

interchanges, $I(f_1, t_1), I(f_2, t_2), \ldots$ Let $\pi^*$ denote the policy that results from applying the sequence of these interchanges.

**Lemma 5.** *For any policy $\pi \in \Pi_{n-1}$, at most $h$ balancing interchanges are required at time slot $n$ to ensure that the resulting policy $\pi^* \in \Pi_n$.*

Lemma 4 can be used to identify queues $f$ and $t$ during time slot $n$ such that the interchange $I(f, t)$ is balancing. Lemma 5 shows that performing a sequence of $h$ such interchanges, starting from $\mathbf{y}(n)$, will result in a withdrawal vector that is balancing during that time slot. Both lemmas are crucial for the proof of our main result, since they indicate how a given policy can be improved (by reducing its difference from an MB policy, i.e., $|\mathbf{D}|$) using one balancing interchange at a time.

# 5 Optimality of MB Policies

In this section, we present the main result of this paper, that is, the optimality of the Most Balancing (MB) policies. We will establish the optimality of MB policies for a wide range of performance criteria including the minimization of the total number of packets in the system. We introduce the following definition.

## 5.1 Definition of Preferred Order

Lets define the relation $\preceq$ on $\mathcal{Z}_+^{(L+1)}$ first; we say $\tilde{\mathbf{x}} \preceq \mathbf{x}$ if:

1- $\tilde{x}_i \leq x_i$ for all $i$ (i.e., point wise comparison),

2- $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by permuting two of its components; the two vectors differ only in two components $i$ and $j$, such that $\tilde{x}_i = x_j$ and $\tilde{x}_j = x_i$, or

3- $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a *"balancing interchange"*, as defined in Equation (26).

To prove the optimality of MB policies, we will need a methodology that enables comparison of the queue lengths under different policies. Towards this end, we define a "preferred order" as follows:

**Definition: (Preferred Order)**. The transitive closure of the relation $\preceq$ defines a partial order (which we call *preferred order* and use the symbol $\prec_p$ to represent) on the set $\mathcal{Z}_+^{(L+1)}$. $\square$

The transitive closure [19], [6] of $\preceq$ on the set $\mathcal{Z}_+^{(L+1)}$ is the smallest transitive relation on $\mathcal{Z}_+^{(L+1)}$ that contains the relation $\preceq$. From the engineering point of view, $\tilde{\mathbf{x}} \prec_p \mathbf{x}$ if $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a sequence of *reductions*, *permutations of two components* and/or *balancing interchanges*.

For example, if $\tilde{\mathbf{x}} = (3, 4, 5)$ and $\mathbf{x} = (4, 5, 3)$ then $\tilde{\mathbf{x}} \prec_p \mathbf{x}$ since $\tilde{\mathbf{x}}$ can be obtained from $\mathbf{x}$ by performing the following two consecutive two-component permutations: first swap the second and third components of $\mathbf{x}$, yielding $\mathbf{x}^1 = (4, 3, 5)$ then swap the first and second components of $\mathbf{x}^1$, yielding $\mathbf{x}^2 = (3, 4, 5) = \tilde{\mathbf{x}}$.

Suppose that $\tilde{\mathbf{x}}, \mathbf{x}$ represent queue size vectors for our model. Case (3-) in this case describes moving a packet from one real, large queue $i$ to another smaller one $j$ (note that the queue with index $j = 0$ is not excluded since a balancing interchange may represent the allocation of an idled server). We say that $\tilde{\mathbf{x}}$ is *more balanced* than $\mathbf{x}$ when (3-) is satisfied. For example, if $L = 2$ and $\mathbf{x} = (0, 5, 2)$ then a balancing interchange (where $i = 1$ and $j = 2$) will result in $\tilde{\mathbf{x}} = (0, 4, 3)$. In summary, the queue size vector $\tilde{\mathbf{x}}$ is preferred over $\mathbf{x}$ ($\tilde{\mathbf{x}} \prec_p \mathbf{x}$) if $\tilde{\mathbf{x}}$ can be obtained from $\mathbf{x}$ by performing a sequence of packet removals, permutations or balancing interchanges.

## 5.2 The class $\mathcal{F}$ of cost functions

Let $\tilde{\mathbf{x}}, \mathbf{x} \in \mathcal{Z}_+^{(L+1)}$ be two vectors representing queue lengths. Then we denote by $\mathcal{F}$ the class of real-valued functions on $\mathcal{Z}_+^{(L+1)}$ that are monotone, non-decreasing with respect to the partial order $\prec_p$; that is, $f \in \mathcal{F}$ if and only if

$$\tilde{\mathbf{x}} \prec_p \mathbf{x} \Rightarrow f(\tilde{\mathbf{x}}) \leq f(\mathbf{x}) \tag{30}$$

From (30) and the definition of preferred order, it can be easily seen that the function $f(\mathbf{x}) = x_1 + x_2 + \ldots + x_L$ belongs to $\mathcal{F}$. This function corresponds to the total number of queued packets in the system[5].

[5]Another example is the function $f'(\mathbf{x}) = \max\{x_1, \ldots, x_L\}$ which also belongs to the class $\mathcal{F}$.

## 5.3 Definition of the subsets $\Pi_n^h, 0 \leq h \leq K$

Consider a policy $\pi \in \Pi_{n-1}$; let $h = \sum_{i=0}^{L} |D_i|/2$ with respect to a given MB policy $\pi^{MB}$, where the vector $\mathbf{D}$ is defined in Equation (29).

**Definition:** For any given time $n$ and $0 \leq h \leq K$, define the set $\Pi_n^h, 0 \leq h \leq K$ as the set that contains all feasible policies $\pi \in \Pi_{n-1}$, such that '*at most*' $h$ balancing interchanges are needed to make $\pi \in \Pi^n$.

We define $\Pi_n^0 = \Pi_n$. Note that the set $\Pi_n^1$ is not empty, since MB policies are elements of it; Note also that $\pi \in \Pi_n^K$ by default. $\{\Pi_n^h\}_{h=0}^K$ forms a monotone sequence, with $\Pi_n^0 \subseteq \cdots \subseteq \Pi_n^h \subseteq \cdots \subseteq \Pi_n^K$. We can easily check that $\Pi_n^K = \Pi_{n-1}$; note that the set $\Pi$ of all policies can be denoted as $\Pi = \Pi_0 = \cup_{h=0}^K \Pi_1^h$.

In the rest of this paper, we say that a policy $\sigma$ *dominates* another policy $\pi$ if

$$f(\mathbf{X}^\sigma(t)) \leq_{st} f(\mathbf{X}^\pi(t)), \quad \forall\ t = 1, 2, \ldots \tag{31}$$

for all cost functions $f \in \mathcal{F}$.

We will need the following lemma to complete the proof of our main result presented in Theorem 2.

**Lemma 6.** *For any policy $\pi \in \Pi_\tau^h$ and $h > 0$, a policy $\tilde{\pi} \in \Pi_\tau^{h-1}$ can be constructed such that $\tilde{\pi}$ dominates $\pi$.*

The proof of Lemma 6 is given in Appendix A.5.

## 5.4 The main result

In the following, $\mathbf{X}^{MB}$ and $\mathbf{X}^\pi$ represent the queue sizes under a MB and an arbitrary policy $\pi$. For two real-valued random variables $A$ and $B$, $A \leq_{st} B$ defines the usual stochastic ordering [2].

**Theorem 2.** *Consider a system of $L$ queues served by $K$ identical servers, as shown in Figure 1 with the assumptions of Sections 1. A Most Balancing (MB) policy dominates any arbitrary policy when applied to this system, i.e.,*

$$f(\mathbf{X}^{MB}(t)) \leq_{st} f(\mathbf{X}^\pi(t)), \quad \forall\ t = 1, 2, \ldots \tag{32}$$

*for all $\pi \in \Pi$ and all cost functions $f \in \mathcal{F}$.*

*Proof.* From (30) and the definition of stochastic dominance, it is sufficient to show that $\mathbf{X}^{MB}(t) \prec_p \mathbf{X}^{\pi}(t)$ for all $t$ and all sample paths in a suitable sample space. The sample space is the standard one used in stochastic coupling methods [1]; see Appendix A.5 for more details.

To prove the optimality of an MB policy, $\pi^{MB}$, we start with an arbitrary policy $\pi$ and apply a series of modifications that result in a sequence of policies $(\pi_1, \pi_2, \ldots)$. The modified policies have the following properties:

(a) $\pi_1$ dominates the given policy $\pi$,

(b) $\pi_i \in \Pi_i$, i.e., policy $\pi_i$ has the MB property at time slots $t = 1, 2, \ldots, i$, and,

(c) $\pi_j$ dominates $\pi_i$ for $j > i$ (and thus $\pi_j$ has the MB property for a longer period of time than $\pi_i$).

Let $\pi$ be any arbitrary policy; then $\pi \in \Pi_0 = \Pi_1^K$. Using Lemma 6 we can construct a policy $\tilde{\pi} \in \Pi_1^{K-1}$ that dominates the original policy $\pi$. Repeating this operation recursively we can construct policies that belong to $\Pi_1^{K-1}, \Pi_1^{K-2}, \ldots, \Pi_1^0 = \Pi_1$ such that all dominates the original policy $\pi$. This sequence of construction steps will result in a policy $\pi_1$ that is MB at $t = 1$, i.e., $\pi_1 \in \Pi_1$, and dominates $\pi$. Therefore, by construction $\pi_1 \in \Pi_2^K$. We repeat the construction steps above for time slot $t = 2$, by improving on $\pi_1$, to obtain a policy $\pi_2 \in \Pi_2$ that dominates $\pi_1$, and recursively for $t = 3, 4, \ldots$ to obtain policies $\pi_3, \pi_4, \ldots$ . From the construction of $\pi_n, n = 1, 2, \ldots$, we can see that it satisfies properties (a), (b) and (c) above.

Denote the limiting policy as $n \longrightarrow \infty$ by $\pi^*$. One can see that $\pi^*$ is an MB policy. Furthermore, $\pi^*$ dominates $\pi_i$, for all $i < \infty$, as well as the original policy $\pi$. □

**Remark 2.** *The optimality of MB policies is intuitively apparent; any such policy will tend to reduce the probability that any server idles. This is because an MB policy distribute the servers among the connected queues in the system such that it keeps packets spread among all the queues in a "uniform" manner. The MB policies also outperform a* Longest Connected Queue *(LCQ) policy which assigns all $K$ servers to the longest connected queue at each time slot.* □

# 6 The Least Balancing Policies

The *Least Balancing* (LB) policies are the server allocation policies that at every time slot ($n = 1, 2, \ldots$), choose a packet withdrawal vector $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})$ that "maximizes the differences" between queue lengths in the system (i.e., maximizes $\kappa_n(\pi)$ in Equation (10)). In other words, if $\Pi^{LB}$ is the set of all LB policies and $\Pi^{WC}$ is the set of all work conserving policies then

$$\Pi^{LB} = \left\{\pi : \underset{\mathbf{y}(n)\in\mathcal{Y}(\mathbf{x},\mathbf{g})}{\operatorname{argmax}} \kappa_n(\pi), \pi \in \Pi^{WC}, \quad \forall n\right\} \tag{33}$$

Maximizing the imbalance among the queues in the system will result in maximizing the number of empty queues at any time slot, thus maximizing the chance that servers are forced to idle in future time slots. This intuitively suggests that LB policies will be outperformed by any work conserving policy. Furthermore, a non-work conserving policy can by constructed such that it will perform worse than LB policies, e.g., a policy that idles all servers. The next theorem states this fact. It's proof is analogous to that of Theorem 2 and will not be given here.

**Theorem 3.** *Consider a system of $L$ queues served by $K$ identical servers, under the assumptions described in Sections 1. A Least Balancing (LB) policy is dominated by any arbitrary work conserving policy when applied to this system, i.e.,*

$$f(\mathbf{X}^{\pi}(t)) \leq_{st} f(\mathbf{X}^{LB}(t)), \quad \forall\ t = 1, 2, \ldots \tag{34}$$

*for all $\pi \in \Pi^{WC}$ and all cost functions $f \in \mathcal{F}$.*

An LB policy has no practical significance, since it maximizes the cost functions presented earlier. Intuitively, it should also worsen the system stability region and hence the system throughput. However, it is interesting to study the worst possible policy behavior and to measure its performance. The LB and MB policies provide lower and upper limits to the performance of any work conserving policy. The performance of any policy can be measured by the deviation of its behavior from that of the MB and LB policies.

# 7 Heuristic Implementation Algorithms For MB and LB Policies

In this section, we present two heuristic policies that approximate the behavior of the MB and LB policies respectively. We present an implementation algorithm for each one of them.

## 7.1 Approximate Implementation of MB Policies

We introduce the *Least Connected Server First/Longest Connected Queue* (LCSF/LCQ) policy, a low-overhead approximation of MB policy, with $O(L\times K)$ computational complexity. We show that it results in a feasible withdrawal vector. The policy is stationary and depends only on the current state $(\mathbf{X}(n), \mathbf{G}(n))$ during time slot $n$.

The LCSF/LCQ implementation during a given time slot is described as follows: The least connected server is identified and is allocated to its longest connected queue. The queue length is updated (i.e., decremented). We proceed accordingly to the next least connected server until all servers are assigned. In algorithmic terms, the LCSF/LCQ policy can be described/implemented as follows:
Let $\mathbb{Q}_j = \{i : i = 1, 2, \ldots, L; g_{i,j}(t) = 1\}$ denote the set of queues that are connected to server $j$ during time slot $t$; we omit the dependence on $t$ to simplify notation. Let $\mathbb{Q}_{[i]}$ be the $i^{th}$ element in the sequence $(\mathbb{Q}_1, \ldots, \mathbb{Q}_K)$, when ordered in ascending manner according to their size (set cardinality), i.e., $|\mathbb{Q}_{[l]}| \geq |\mathbb{Q}_{[m]}|$ if $l > m$. Ties are broken arbitrarily. Then under the LCSF/LCQ policy, the $K$ servers are allocated according to the following algorithm:

**Algorithm 2** (LCSF/LCQ Implementation)**.**

1. $for \quad t = 1, 2, \ldots \quad do \quad \Big\{$
2. $\quad$ *Input:* $\mathbf{X}(t), \mathbf{G}(t)$. *Calculate* $\quad \mathbb{Q}_{[l]}$, $l = 1, \ldots, K$.
3. $\quad \mathbf{X}' \longleftarrow \mathbf{X}(t), \ \mathbf{Y} \longleftarrow \mathbf{0}, \ \mathbf{Q} \longleftarrow \mathbf{0}$
4. $\quad for \ j = 1 \quad to \quad K \quad \Big\}$ *; allocate servers sequentially*
5. $\quad\quad Q_{[j]} = \min \left( l : l \in \left\{ \underset{k:k \in \mathbb{Q}_{[j]}}{\operatorname{argmax}} (X'_k | X'_k > 0) \right\} \right)$
6. $\quad\quad for \ i = 1 \quad to \quad L \quad \Big\{$
7. $\quad\quad\quad Y_i = Y_i + \mathbb{1}_{\{i = Q_{[j]}\}}$
8. $\quad\quad\quad X'_i = X_i(t) - Y_i \quad \Big\} \quad \Big\}$
9. $\quad$ *Output:* $\mathbf{y}(t) \longleftarrow \mathbf{Y}, \mathbf{q}(t) \longleftarrow \mathbf{Q} \quad$ *; report outputs*
10. $\Big\} \quad$ *; End of Algorithm 2.*

Note that in line 5 of Algorithm 2, if the set $\mathbb{Q}_{[j]}$ is empty, then the argmax returns the empty set. In this case, the $j^{th}$ order server will not be allocated (i.e., will be idle during time slot $t$). Algorithm 2 produces two outputs, when it is run at $t = n$: $\mathbf{y}(n)$ and $\mathbf{q}(n)$ as shown in line 9 of the algorithm. In accordance to the definition of a policy in Equation (9), the LCSF/LCQ policy can be formally defined as the sequence of time-independent mappings $u(\mathbf{x}(n), \mathbf{g}(n))$ that produce the withdrawal vector $\mathbf{y}(n)$ described in line 9 above. The following lemma asserts that the mapping defines feasible controls.

**Lemma 7.** *The policy obtained by applying Algorithm 2 results in a feasible withdrawal vector at every time slot $n$ and any state $(\mathbf{x}(n), \mathbf{g}(n))$.*

*Proof.* Let $\mathbf{y}(n)$ and $\mathbf{q}(n)$ denote the outputs of Algorithm 2; the inputs are $(\mathbf{x}(n), \mathbf{g}(n))$. Let $\mathbf{V}(n)$ be the matrix with elements

$$V_{ij}(n) = \mathbb{1}_{\{i = q_j(n)\}} \cdot g_{i,j}(n). \tag{35}$$

We must show that the output $\mathbf{y}(n)$ can be written as

$$\mathbf{y}(n) = \mathbf{V}(n) \cdot \mathbf{I}_K \tag{36}$$

and that $\mathbf{V}(n)$ satisfies the feasibility constraints (2) and (3).

From Algorithm 2, line 5, it can be seen that for every server $[j]$, only the set of queues that are connected to server $[j]$ are considered as candidates for allocating this server. Therefore, $V_{i[j]}(n) = 1$ is true only when $g_{i,[j]}(n) = 1$ and $\mathbb{1}_{\{i=q_{[j]}(n)\}} = 1$ are true, establishing Equation (35). From Equations (36) and (3) we can easily see that

$$\mathbf{y}(n) \leq \mathbf{x}(n) \tag{37}$$

is a sufficient condition for Inequality (3) to hold. Note that queue $i$ will be selected in Algorithm 2, line 5 (to be served by server $[j]$) only if its current size $X_i'$ is strictly positive. This will ensure that the number of servers allocated to any queue is no larger than the number required to empty that queue. Therefore, $y_i(n) \leq x_i(n), i = 1, \ldots, L$, proving Inequality (37).

Constraints (2) are satisfied. To prove that, fix a server $[j]$; the initialization step 3 assigns this server to the dummy queue. Observe that even though the inner for-loop in Algorithm 2 is executed $L+1$ times, the indicator function $\mathbb{1}_{\{i=Q_{[j]}\}}$ in line 7 is nonzero for only one value of $i \in \{0, 1, \ldots, L\}$; each server is allocated to one queue only, either the dummy queue or the queue with the minimum index out of the outcome of the argmax function in line 5 of Algorithm 2. Therefore the statement $\sum_{i=0}^{L} \mathbb{1}_{\{i=q_{[j]}(t)\}} = 1$ is true for all $j$, proving equality (2). □

Although allocating the available servers to their longest connected queues in the order specified by Algorithm 2, i.e., starting from the least connected server first to the most connected server last, may not be "most balancing" in some occasions, the LCSF/LCQ is expected to perform very close to any MB policy.

Our intuition suggests that in a sequential server allocation, such as that of Algorithm 2, one of the $K!$ possible server orderings (with LCQ server allocation) may actually result in an MB policy. The construction and proof of such an algorithm is out of the scope of this paper and hence it is left for future research work.

**Lemma 8.** *LCSF/LCQ is not an MB policy.*

To prove lemma 8 we present the following counter example. Consider a system with $L = 4$ and $K = 7$. At time slot $n$ the system has the following configuration:

The queue state at time slot $n$ is $\mathbf{x}(n) = (5, 5, 5, 4)$. Servers 1 to 6 are connected to queues 1, 2 and 3 and server 7 is connected to queues 1 and 4 only.

Under this configuration, we can show that the LCSF/LCQ algorithm will result in $\hat{\mathbf{x}}(n) = (0, 2, 3, 3, 4)$ (where the first element represents the dummy queue that by assumption holds no real packets) and $\kappa_n(LCSF/LCQ) = 18$. A policy $\pi$ can be constructed that selects the feasible server allocation $\mathbf{q} = (1, 2, 3, 1, 2, 3, 4)$ which yields the state $\hat{\mathbf{x}}(n) = (0, 3, 3, 3, 3)$ and $\kappa_n(\pi) = 12 < \kappa_n(LCSF/LCQ)$. Therefore, the LCSF/LCQ does not belong to the class of MB policies.

The LCSF/LCQ policy is of particular interest for the following reasons: (a) It follows a particular server allocation ordering (LCSF) to their longest connected queues (LCQ) and thus it can be implemented using simple sequential server allocation with low computation complexity. (b) the selected server ordering (LCSF) and allocation (LCQ) intuitively tries to maximize the opportunity to target and reduce the longest connected queue in the system thus minimizing the imbalance among queues, and (c) as we will see in Section 8, the LCSF/LCQ performance is statistically indistinguishable from that of an MB policy (implying that the counterexamples similar to the one in Lemma 8 proof have low probability of occurrence under LCSF/LCQ system operation).

## 7.2 Approximate Implementation of LB Policies

In this section, we present the MCSF/SCQ policy as a low complexity, easy to implement approximation of LB policies. We also provide an implementation algorithm for MCSF/SCQ using sequential server allocation principle that we used in the previous algorithms.

The *Most Connected Server First/Shortest Connected Queue* (MCSF/SCQ) policy is the server allocation policy that allocates each one of the $K$ servers to its shortest connected queue (not counting the packets already scheduled for service) starting with the most connected server first. The MCSF/SCQ implementation algorithm is analogous to Algorithm 2 except for lines 4 and 5 which are described next:

**Algorithm 3** (MCSF/SCQ Implementation)**.**

1. $for \quad t = 1, 2, \ldots \quad do \quad \Big\{$

$\vdots$

4. $\quad for\ j = K \quad to \quad 1 \quad \Big\}$ *; Servers in descending order*

5. $\qquad Q_{[j]} = \min \left( l : l \in \left\{ \underset{k:k \in \mathbb{Q}_{[j]}}{\operatorname{argmin}} (X'_k | X'_k > 0) \right\} \right)$

$\vdots$

10. *; End of Algorithm 3.*

Comments analogous to the ones valid for Algorithm 2 are also valid for Algorithm 3.

# 8 Performance Evaluation and Simulation Results

We used simulation to study the performance of the system under the MB/LB policies and to compare against the system performance under several other policies. The metric we used in this study is $EQ \triangleq E(\sum_{i=1}^{L} X_i)$, the average of the total number of packets in the system.

We focused on two groups of simulations. In the first, we evaluate the system performance with respect to number of queues ($L$) and servers ($K$) as well as channel connectivity (Figures 3 to 7). Random arrivals to queues are assumed to be i.i.d. Bernoulli. In the second group of simulations (Figures 8(a) to 8(c)) we consider batch arrivals with random (uniformly distributed) burst size.

The policies used in this simulation are: LCSF/LCQ, as an approximation of an MB policy; MCSF/SCQ, as an approximation of an LB policy. An MB policy was implemented using full search, for the cases specified in this section, and its performance was indistinguishable from that of the LCSF/LCQ. Therefore, in the simulation graphs the MB and LCSF/LCQ are represented by the same curves. Other policies that were simulated include the randomized, Most Connected Server First/Longest Connected Queue (MCSF/LCQ), and Least Connected Server First/Shortest Connected Queue (LCSF/SCQ)

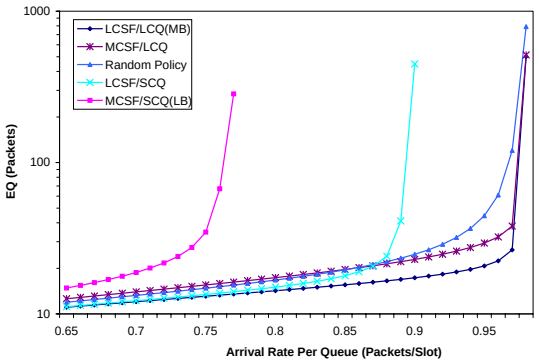



Figure 3: Average total queue occupancy, $EQ$, versus load under different policies, $L = 16, K = 16$ and $p = 0.2$.

policies. The randomized policy is the one that at each time slot allocates each server, randomly and with equal probability, to one of its connected queues. The MCSF/LCQ policy differs from the LCSF/LCQ policies in the order that it allocates the servers. It uses the exact reverse order, starting the allocation with the most connected server and ending it with the least connected one. However, it resembles the LCSF/LCQ policies in that it allocates each server to its longest connected queue. The LCSF/SCQ policy allocates each server, starting from the one with the least number of connected queues, to its shortest connected queue. The difference from an LCSF/LCQ policy is obviously the allocation to the shortest connected queue. This policy will result in greatly unbalanced queue lengths and hence a performance that is closer to the LB policies.

Figure 3 shows the average total queue occupancy versus arrival rate under the five different policies. The system in this simulation is a symmetrical system with 16 parallel queues ($L = 16$), 16 identical servers ($K = 16$) and i.i.d. Bernoulli queue-to-server (channel) connectivity with parameter $p = P[G_{i,j}(t) = 1] = 0.2$.

The curves in Figure 3 follow a shape that is initially almost flat and ends with a rapid increase. This abrupt increase happens at a point where the system becomes unstable. In this case, the queue lengths in the system will grow fast and the system becomes unstable. The graph shows that LCSF/LCQ, the MB policy approximation outperforms[6] all other policies.

[6]99% confidence intervals are very narrow and would affect the readability of the graphs. Therefore they are not included.

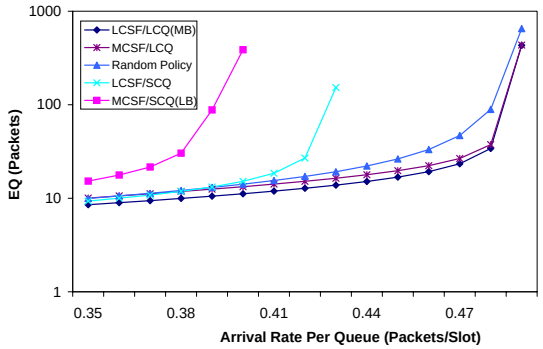



Figure 4: Average total queue occupancy, $EQ$, versus load, $L = 16, K = 8$ and $p = 0.2$.

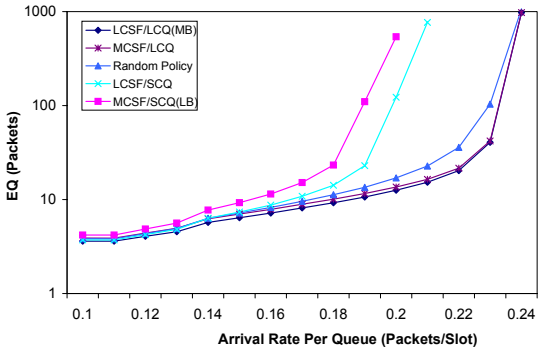



Figure 5: Average total queue occupancy, $EQ$, versus load, $L = 16, K = 4$ and $p = 0.2$.

It minimizes, $EQ$ and hence the queuing delay. We also noticed that it maximizes the system stability region and hence the system throughput as well. As expected, the performance of the other three policies lies within the performance of the MB and LB policies.

The MCSF/LCQ and LCSF/SCQ policies are variations of the MB and LB policies respectively. The performance of MCSF/LCQ policy is close to that of the MB policy. The difference in performance is due to the order of server allocation. On the other hand, the LCSF/SCQ policy shows a large performance improvement on that of the LB policy. This improvement is a result of the reordering of server allocations.

Figure 3 also shows that the randomized policy performs reasonably well. Moreover, its performance improves as the number of servers in the system decreases, as the next set of experiments shows.

## 8.1 The Effect of The Number of Servers

In this section, we study the effect of the number of servers on policy performance. Figures 4 ($K = 8$) and 5 ($K = 4$) show $EQ$ versus arrival rate per queue under the five policies, in a symmetrical system with $L = 16$ and $p = 0.2$. Comparing these two graphs to the one in Figure 3, we notice the following:

First, the performance advantage of the LCSF/LCQ (an hence of an MB policy) over the other policies increases as the number of servers in the system increases. The presence of more servers implies that the server allocation action space is larger. Selecting the optimal (i.e., MB) allocation, over any arbitrary policy, out of a large number of options will result in reduced system occupancy as compared to the case when the number of server allocation options is less.

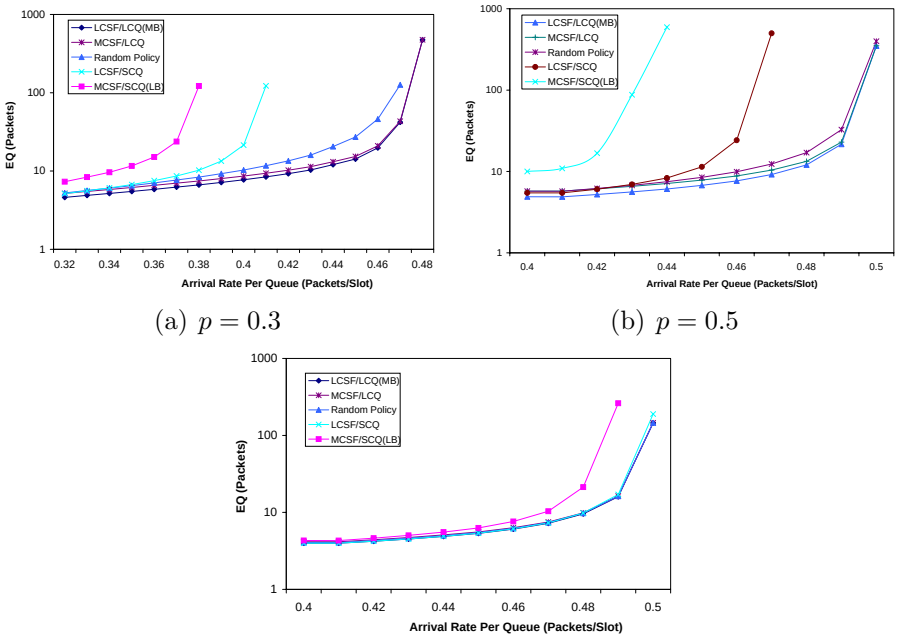
Second, the stability region of the system becomes narrower when less servers are used. This is true because fewer resources (servers) are available to be allocated by the working policy in this case.

Finally, we notice that the MCSF/LCQ performs very close to the LCSF/LCQ policy in the case of $K = 4$. Apparently, when $K$ is small, the order of server allocation does not have a big impact on the policy performance.

## 8.2 The Effect of Channel Connectivity

In this section we investigate the effect of channel connectivity on the performance of the previously considered policies. Figures 6 and 7 show this effect for two choices of $L$ and $K$. We observe the following:

First, we notice that for larger channel connection probabilities ($p \geq 0.9$), the effect of the policy behavior on the system performance becomes less significant. Therefore, the performance difference among the various policies is getting smaller. The LCSF/LCQ policy still has a small advantage over the rest of the policies, even though statistically difficult to distinguish. MCSF/SCQ continues to have the worst performance. As $p$ increases, the probability that a server will end up connected to a group of empty queues will be very small regardless of the policy in effect. In fact, when the servers have full connectivity to all queues (i.e., $p = 1.0$) we expect that any work conserving policy will minimize the total number of packets in a symmetrical homogeneous system of queues since, any (work-conserving) policy will be optimal in a system with full connectivity.




(a) $p = 0.3$ (b) $p = 0.5$

(c) $p = 0.9$

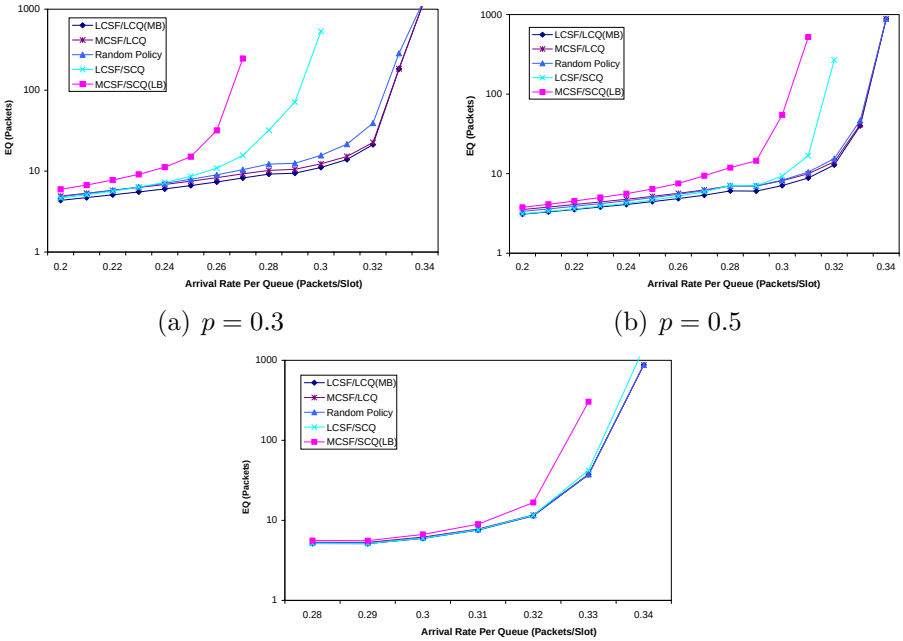
Figure 6: Average total queue occupancy, $EQ$, versus load under different policies, $L = 8$ and $K = 4$.

Second, from all graphs we observe that there is a maximum input load that results in a stable system operation (maximum stable throughput). An upper bound (for stable system operation) for the arrival rate $\alpha$ for each queue can be easily shown to be

$$\alpha < \frac{K}{L}\big(1 - (1-p)^L\big) \tag{38}$$

I.e., the average number of packets entering the system ($\alpha L$) must be less than the rate they are being served. When $p = 1.0$, the stability condition in Inequality (38) will be reduced to $\alpha L < K$, which makes intuitive sense in such a system.

Finally, we observe that the MCSF/LCQ policy performance is very close to the one from LCSF/LCQ. However, its performance deteriorates in systems with higher number of servers and lower channel connectivity probabilities. It is intuitive that with more servers available, the effect of the order of




(a) $p = 0.3$ (b) $p = 0.5$

(c) $p = 0.9$

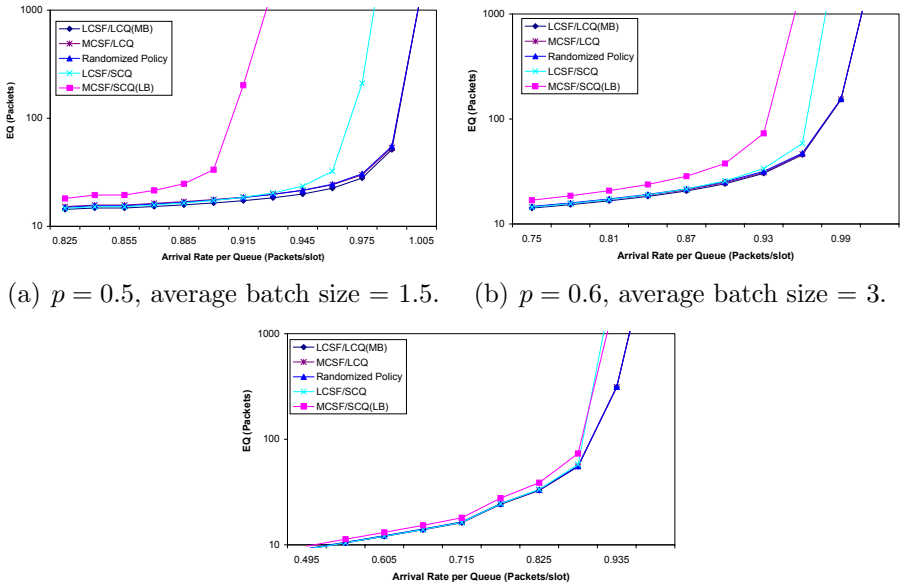
Figure 7: Average total queue occupancy, $EQ$, versus load under different policies, $L = 12$ and $K = 4$.

server allocations on the performance will increase. Since MCSF/LCQ differs from LCSF/LCQ only by the order of server allocation, therefore, more servers implies larger performance difference. Also, the lower the connectivity probability, the higher the probability that a server will end up with no connectivity to any non-empty queue, and hence be forced to idle.

## 8.3 Batch Arrivals With Random Batch Sizes

We studied the performance of the presented policies in the case of batch arrivals with uniformly distributed batch size, in the range $\{1, \ldots, U\}$. Figure 8 shows $EQ$ versus load for three cases with $U = 2, 5, 10$, and hence average batch sizes 1.5, 3, and 5.5. The LCSF/LCQ policy clearly dominates all the other policies. However, the performance of the other policies, including MCSF/SCQ (LB approximation) approaches that of the LCSF/LCQ policy as the average batch size increases. The performance of all the policies

deteriorates when the arrivals become burstier, i.e., the batch size increases.




(a) $p = 0.5$, average batch size $= 1.5$. (b) $p = 0.6$, average batch size $= 3$.

(c) $p = 0.8$, average batch size $= 5.5$.

Figure 8: Average total queue occupancy, $EQ$, versus load, batch arrivals, $L = 16$ and $K = 16$.

# 9 Conclusion

In this work, we presented a model for dynamic packet scheduling in a multi-server systems with random connectivity. This model can be used to study packet scheduling in emerging wireless systems. We modeled such systems via symmetric queues with random server connectivities and general arrival distributions. We introduced the class of Most Balancing policies. These policies distribute the service capacity between the longest connected queues in the system in an effort to "equalize" the queue occupancies. A theoretical proof of the optimality of MB policies using stochastic coupling arguments was presented. Optimality was defined as minimization, in stochastic ordering sense, of a range of cost functions of the queue lengths. The LCSF/LCQ policy was proposed as good, low-complexity approximation for MB policies.

A simulation study was conducted to study the performance of five different policies. The results verified that the MB approximation outperformed all other policies (even when the arrivals became bursty). However, the performance of all policies deteriorate as the mean burst size increases. Furthermore, we observed (through simulation) that the performance gain of the optimal policy over the other policies is reduced greatly in this case. Finally, we observed that a randomized policy can perform very close to the optimal one in several cases.

# APPENDIX:

## A.1 Balancing Interchanges And The Imbalance Index

In this section, we present a lemma that quantifies the effect of performing a balancing interchange on the imbalance index $\kappa_n(\pi)$. The "balancing interchange" is defined in Section 5.3.

**Lemma A.1.1.** *Let* $\mathbf{x}$ *be an* $L$*-dimensional ordered vector (in descending order); suppose that* $\mathbf{x}^*$ *is obtained from* $\mathbf{x}$ *by performing a balancing interchange of the* $l^{th}$ *and the* $s^{th}$ *components, s.t.,* $s > l, x_l > x_a, \forall a > l$ *and* $x_s < x_b, \forall b < s$*. Then*

$$\sum_{i'=1}^{L}\sum_{j'=i'+1}^{L}(x^*_{i'} - x^*_{j'}) = \sum_{i=1}^{L}\sum_{j=i+1}^{L}(x_i - x_j) \\ -2(s-l)\cdot \mathbb{1}_{\{x_l \geq x_s+2\}} \tag{A-1}$$

*Proof.* We generate the vector $\mathbf{x}^*$ by performing a *balancing interchange* of two components (the $l^{th}$ and the $s^{th}$ largest components) in the vector $\mathbf{x}$. The resulted vector $\mathbf{x}^*$ is characterized by the following:

$$x^*_{l'} = x_l - 1, \quad x^*_{s'} = x_s + 1, \quad x_l > x_s \\ x^*_k = x_k, \quad \forall k \neq l, s, l', s' \tag{A-2}$$

where $l'$ (respectively $s'$) is the new order (i.e., the order in the new vector $\mathbf{x}^*$) of the $l^{th}$ (respectively $s^{th}$) component in the original vector $\mathbf{x}$.

From Equation (A-2) we can easily show that

$$\sum_{i'=1}^{L}\sum_{j'=i'+1}^{L}(x^*_{i'} - x^*_{j'}) \;=\; \sum_{i=1}^{L}\sum_{j=i+1}^{L}(x_i - x_j), \\ \textit{for all} \quad i,j \notin \{l,s\} \quad ; \quad i',j' \notin \{l',s'\} \tag{A-3}$$

For the remaining cases, i.e., when at least one of the indices $i, j$ belongs to $\{l, s\}$ and/or $i', j'$ belongs to $\{l', s'\}$, we pair the index $i'$ (respectively $j'$) on the left hand side with the index $i$ (respectively $j$) on the right hand side of Equation (A-1). We first assume that $x_l \geq x_s + 2$, then we can easily show that $l' \leq s'$. In this case, we have the following five, mutually exclusive, cases to consider:

1. When $i' = l', i = l, j' = s'$ and $j = s$. This case occurs only once, i.e., when decomposing the double sum in Equation (A-1) we can find only one term that satisfies this case. From Equation (A-2) we have

$$x^*_{l'} - x^*_{s'} = x_l - x_s - 2 \tag{A-4}$$

2. When $i' = l', i = l, j' \neq s'$ and $j \neq s$. There are $L - l - 1$ terms that satisfy this case. Analogous to case 1) we determined that

$$x^*_{l'} - x^*_{j'} = x_l - x_j - 1 \tag{A-5}$$

3. When $i' \neq l', i \neq l, j' = s'$ and $j = s$. There are $s-2$ terms that satisfy this case. In this case we can show that

$$x^*_{i'} - x^*_{s'} = x_i - x_s - 1 \tag{A-6}$$

4. When $i' \neq l', s', i \neq l, s, j' = l'$ and $j = l$. There are $l - 1$ terms that satisfy this case. In this case we can show that

$$x^*_{i'} - x^*_{l'} = x_i - x_l + 1 \tag{A-7}$$

5. When $i' = s', i = s, j' \neq l', s'$ and $j \neq l, s$. There are $L - s$ terms that satisfy this case. In this case we have

$$x^*_{s'} - x^*_{j'} = x_s - x_j + 1 \tag{A-8}$$

The above cases (i.e., Equations (A-3)-(A-8)) cover all the terms in Equation (A-1) when $x_{[l]} > x_{[s]} + 1$. Combining all these terms yields:

$$\begin{aligned} \sum_{i'=1}^{L} \sum_{j'=i'+1}^{L} (x^*_{i'} - x^*_{j'}) &= \sum_{i=1}^{L} \sum_{j=i+1}^{L} (x_i - x_j) \\ -2 \cdot (1) - 1 \cdot (L-l-1) - 1 \cdot (s-2) + 1 \cdot (l-1) + 1 \cdot (L-s) & \\ &= \sum_{i=1}^{L} \sum_{j=i+1}^{L} (x_i - x_j) - 2(s-l) \cdot \mathbb{1}_{\{x_l \geq x_s + 2\}} \end{aligned} \tag{A-9}$$

Furthermore, if $x_l = x_s + 1$, then from Equation (A-2) it is clear that $x^*_{l'} = x_s$ and $x^*_{s'} = x_l$, i.e., the resulted vector is a permutation of the original

one. Therefore, the sum of differences will be the same in both vectors and Equation (A-1) will be reduced to

$$\sum_{i'=1}^{L}\sum_{j'=i'+1}^{L}(x^*_{i'}-x^*_{j'})=\sum_{i=1}^{L}\sum_{j=i+1}^{L}(x_i-x_j) \tag{A-10}$$

□

## A.2 Proofs for Results in Section 3

*Proof for Lemma 1 of Section 3.* We will show that at time slot $n$ a policy $\pi^*$ (as described in the lemma statement) will always result in an imbalance index that is equal to that resulted from applying the original policy $\pi^\theta$, i.e.,

$$\sum_{i'=1}^{L+1}\sum_{j'=i'+1}^{L+1}(\hat{x}^*_{[i']}(n)-\hat{x}^*_{[j']}(n))=\sum_{i=1}^{L+1}\sum_{j=i+1}^{L+1}(\hat{x}_{[i]}(n)-\hat{x}_{[j]}(n)) \tag{A-11}$$

where $\hat{x}^*_{[k]}(n)$ (respectively $\hat{x}_{[k]}(n)$) is the size of the $k^{th}$ longest queue after applying the policy $\pi^*$ (respectively $\pi^\theta$) at time slot $n$.

We start by constructing the policy $\pi^*$ at time slot $n$, such that it uses the same allocation order as $\pi^\theta$ (i.e., $\theta$), allocates the first $K-1$ servers (call them $s_{[1]},\ldots,s_{[K-1]}$) similar to $\pi^\theta$, and allocates the $K^{th}$ server ($s_{[K]}$) to its longest connected queue (instead of a NLCQ as in $\pi^\theta$). Let $\hat{\mathbf{x}}^*(n,k)$ (respectively $\hat{\mathbf{x}}(n,k)$) represent the updated queue length vector (or state vector) after allocating the $k^{th}$ server during a sequential server allocation. Then $\hat{\mathbf{x}}^*(n,K-1)=\hat{\mathbf{x}}(n,K-1)$, i.e., both policies will result in the same queue sizes after the $(K-1)^{st}$ server has been allocated, since the two policies have the same server allocation (by construction) up to this point.

At the $(K-1)^{st}$ allocation step (just before the $K^{th}$, i.e., the last, server allocation), we assume that the queues have been ordered such that the longest queue lies on top, i.e., the vector $\hat{\mathbf{x}}(n,K-1)$ is arranged in a descending order). We identify two queues of interest. The first is the $K^{th}$ server longest connected queue (LCQ) which has an order $l$ (i.e., $\hat{x}_{[l]}(n,K-1)\geq\hat{x}_{[i]}(n,K-1),\forall i>l,i\in\mathbb{Q}_{[K]}$). This queue will receive service in step $K$ (when the $K^{th}$, i.e., *last*, server is allocated) under policy $\pi^*$. Therefore, $\hat{x}^*_{[l]}(n,K)=\hat{x}_{[l]}(n,K)-1$.

The second queue of interest is the non-LCQ that server $s_{[K]}$ was allocated to under the policy $\pi^\theta$, and which has an order $s$ (where $s > l$, since $\hat{x}_{[s]}(n, K-1) < \hat{x}_{[l]}(n, K-1)$). Therefore, $\hat{x}^*_{[s]}(n, K) = \hat{x}_{[s]}(n, K)+1$. The remaining queue lengths remain the same, i.e., $\hat{x}^*_{[i]}(n, K) = \hat{x}_{[i]}(n, K)$, $\forall i \neq l, s$. In other words, $\hat{\mathbf{x}}^*(n)$ is obtained from $\hat{\mathbf{x}}(n)$ by performing a "balancing interchange". Using lemma A.1.1 we can show that

$$\sum_{i'=1}^{L+1} \sum_{j'=i'+1}^{L+1} (\hat{x}^*_{[i']}(n) - \hat{x}^*_{[j']}(n)) \leq \sum_{i=1}^{L+1} \sum_{j=i+1}^{L+1} (\hat{x}_{[i]}(n) - \hat{x}_{[j]}(n)) \tag{A-12}$$

Since $\pi^\theta$ has the MB property at time slot $n$ (by assumption), then Equation (A-12) can only be satisfied with strict equality (i.e., $\pi^*$ has the MB property at time slot $n$) and Lemma 1 follows.
**Note:** Stated differently, $\pi^\theta$ should allocate its last server either to its longest connected queue or to a connected queue that has one packet less than the longest connected queue. Otherwise, it may not be an MB policy. □

*Proof for Lemma 2 of Section 3.* During time slot $n$, denote the two swapped servers by $s_1$ (NLCQ allocated) and $s_2$ (LCQ allocated server that is next in order). Denote the queue that $s_1$ (respectively $s_2$) is allocated to by $q_1$ (respectively $q_2$). Denote the new order (resulted from swapping the allocation orders of $s_1$ and $s_2$) by $\theta^*(n)$ and denote all quantities that result from using that order by the superscript $(*)$.

The LCQ property of $(s_2, q_2)$ will not hold only if $q_2$ becomes shorter than the LCQ due to the swapping. To prove lemma 2, we must show that this is not possible. To do that we consider the following two mutually exclusive cases:

Case 1) If $\mathcal{M}^n_{s_1} \cap \mathcal{M}^n_{s_2} = \{\phi\}$ (i.e., the two servers are not connected to common queues), then $s_2$ will definitely retain its LCQ property since the swap in such case will not affect the order of queues in the set $\mathcal{M}^n_{s_2}$ (i.e., their relative lengths). The swap does not change the queue length of $q_2$, i.e.,

$$\hat{x}^*_{q_2}(n, k-1) = \hat{x}_{q_2}(n, k), \textit{and } \hat{x}^*_{q_1}(n, k) = \hat{x}_{q_1}(n, k-1) \tag{A-13}$$

where $k$ is the step where $s_1$ is allocated under the order $\theta(n)$. Step $k$ is also the step where $s_2$ is allocated under the order $\theta^*(n)$ (see Figure A-1 for the allocation order).

Case 2) If $\mathcal{M}^n_{s_1} \cap \mathcal{M}^n_{s_2} \neq \{\phi\}$, then the only case that needs to be considered is when $q_1$ is connected to both servers, $s_1$ and $s_2$, (Figure A-2).

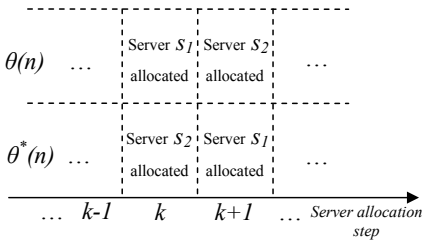



Figure A-1: Sequence of servers' allocation during one time slot.

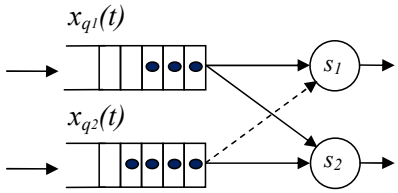



Figure A-2: Connectivity pattern for case 2.

In that case and under the sequential server allocation order $\theta(n)$, queue $q_1$ must be shorter than queue $q_2$ by step $k$ (i.e., the moment just before allocating server $s_2$ to its longest connected queue). This is true because $s_1$ is a NLCQ allocated server by assumption. Therefore,

$$\hat{x}_{q_1}(n,k) < \hat{x}_{q_2}(n,k) \tag{A-14}$$

Recall that server $s_2$ is connected to both $q_1$ and $q_2$. If $s_2$ is allocated before $s_1$ (as in $\theta^*(n)$) then

$$\hat{x}^*_{q_1}(n,k-1) \leq \hat{x}^*_{q_2}(n,k-1) \tag{A-15}$$

Therefore, the LCQ property of $(s_2, q_2)$ pair *remains valid* and lemma 2 follows. □

## A.3 Proof of Lemma 4

Before we present the proof, we need to introduce a few intermediate lemmas. They describe useful properties of $I(f,t)$ and $\mathbf{D}$.

**Lemma A.3.1.** *The feasible interchange $I(f,0), f > 0$ is a balancing interchange.*

*Proof.* By definition, $x_0(n) = 0$. Since $y_0(n) \geq 0$, therefore $\hat{x}_0(n) = x_0(n) - y_0(n) \leq 0$. According to the feasibility constraint (23), the interchange $I(f, 0)$ is feasible only when $\hat{x}_f(n) \geq 1$. Therefore, $\hat{x}_f(n) \geq \hat{x}_0(n) + 1$, and it follows that $I(f, 0)$ is a balancing interchange. □

**Lemma A.3.2.** *For a given policy $\pi \in \Pi_{n-1}$ and a time slot $n$,*

$$\sum_{i=0}^{L} D_i \cdot \mathbb{1}_{\{D_i > 0\}} = -\sum_{j=0}^{L} D_j \cdot \mathbb{1}_{\{D_j < 0\}} \tag{A-16}$$

*i.e., the sum of all positive elements of* $\mathbf{D}$ *equals the sum of all negative elements of* $\mathbf{D}$*.*

*Proof.* For any withdrawal vector $\mathbf{y}(n)$, we have

$$\sum_{i=0}^{L} y_i(n) = K,$$

where $K$ is the number of servers. From equation (29), we have then:

$$\begin{aligned} \sum_{i=0}^{L} D_i &= \sum_{i=0}^{L} y_i^{MB}(n) - \sum_{i=0}^{L} y_i(n) \\ &= K - K = 0, \end{aligned}$$

and Equation (A-16) follows. □

**Lemma A.3.3.** *Consider a given state $(\mathbf{x}(n), \mathbf{g}(n))$ during time slot $n$. Let $f, t \in \{0, 1, \ldots, L\}$ be any two queues such that $I(t, f)$ is feasible. A policy $\pi \in \Pi$ that results in $\hat{x}_f(n) \leq \hat{x}_t(n) - 2$ is not an MB policy.*

*Proof.* The interchange $I(t, f)$ is a balancing interchange by definition. Since $\hat{x}_f(n) \leq \hat{x}_t(n) - 2$, then the balancing interchange $I(t, f)$ reduces the imbalance index by a factor of two according to Equation (28). Therefore, $\pi$ does not achieve the minimum imbalance index during time slot $n$, and hence, is not an MB policy. □

**Lemma A.3.4.** *Consider a given state $(\mathbf{x}(n), \mathbf{g}(n))$ and a policy $\pi$. If $\mathbf{D} = 0$, then $\pi$ has the MB property. Conversely, if $\pi$ has the MB property, the vector $\mathbf{D}$ has components that are $0, +1,$ or $-1$ only.*

*Proof.* Assume that $\mathbf{D} = 0$; then, using Equation (29), we have:

$$\begin{aligned} \mathbf{y}(n) &= \mathbf{y}^{MB}(n) \\ \mathbf{x}(n) - \mathbf{y}(n) &= \mathbf{x}(n) - \mathbf{y}^{MB}(n) \\ \hat{\mathbf{x}}(n) &= \hat{\mathbf{x}}^{MB}(n) \end{aligned} \tag{A-17}$$

From Equations (A-17) and (10), we have that $\kappa_n(\pi) = \kappa_n(\pi^{MB})$ and thus $\pi$ has the MB property during time slot $n$.

To prove the converse part of the lemma, assume that $\pi$ has the MB property. Therefore, $\kappa_n(\pi) = \kappa_n(\pi^{MB})$. From Lemma A.1.1 this is only possible if either: (i) $\hat{\mathbf{x}}(n) = \hat{\mathbf{x}}^{MB}(n)$, or (ii) $\hat{\mathbf{x}}(n)$ is obtained by performing a balancing interchange between the pair of the $l^{th}$ and the $s^{th}$ longest queues ($l < s$) in $\hat{\mathbf{x}}^{MB}(n)$ such that $\hat{x}_{[l]}(n) = \hat{x}_{[s]}(n) + 1$, is satisfied; note that there may be multiple such queue pairs. The balancing interchange in case (ii) will affect the length of two queues only (call them $i$ and $j$) such that $\hat{x}_i(n) = \hat{x}_i^{MB}(n) - 1$ and $\hat{x}_j(n) = \hat{x}_j^{MB}(n) + 1$, where $i = [l]$ and $j = [s]$ (for each given pair). Therefore,

$$\begin{aligned} y_i(n) &= x_i(n) - \hat{x}_i(n) = x_i(n) - (\hat{x}_i^{MB}(n) - 1) \\ &= y_i^{MB}(n) + 1, \end{aligned} \tag{A-18}$$

and,

$$\begin{aligned} y_j(n) &= x_j(n) - \hat{x}_j(n) = x_j(n) - (\hat{x}_j^{MB}(n) + 1) \\ &= y_j^{MB}(n) - 1, \end{aligned} \tag{A-19}$$

while withdrawals from the remaining queues will be the same, i.e.,

$$y_b(n) = y_b^{MB}(n), \forall b \neq i, j. \tag{A-20}$$

From Equations (A-18) through (A-20), we conclude that the vector $\mathbf{D}$ has components that are $0, +1$, or $-1$ only. □

**Lemma A.3.5.** *Given the state* $(\mathbf{x}(n), \mathbf{g}(n))$ *and a feasible withdrawal vector* $\mathbf{y}(n)$ *then a withdrawal vector* $\mathbf{y}'(n)$ *that results from performing any sequence of feasible server reallocations on* $\mathbf{y}(n)$ *is feasible.*

*Proof.* It suffices to show that a single feasible server reallocation will result in a feasible withdrawal vector $\mathbf{y}^*(n)$. Starting from the feasible withdrawal

vector $\mathbf{y}^*(n)$, we can use the same argument to show that the second server reallocation will also result in a feasible withdrawal vector. The subsequent server reallocations will also have the same result.

Let $\mathbf{q}(n)$ be a server allocation vector that results in the withdrawal vector $\mathbf{y}(n)$. Then $\mathbf{y}(n)$ is related to $\mathbf{q}(n)$ as follows

$$y_i(n) = \sum_{j=1}^{K} \mathbb{1}_{\{i=q_j(n)\}}, \quad i = 0, 1, 2, \ldots, L. \tag{A-21}$$

A feasible reallocation of server $k$ from queue $q_k(n)$ to queue $b$ will result in a withdrawal vector $\mathbf{y}^*(n)$, where

$$y_b^*(n) = y_b(n) + 1$$

$$y_{q_k}^*(n) = y_{q_k}(n) - 1$$

$$y_i^*(n) = y_i(n), \forall i \neq q_k, b$$

From the above, we conclude that

$$\sum_{i=0}^{L} y_i^*(n) = \sum_{i=0}^{L} y_i(n) = K \tag{A-22}$$

The resulted server allocation vector $\mathbf{q}^*(n)$ is given by

$$q_i^*(n) = \begin{cases} b & \text{if } i = k \\ q_i(n) & \text{Otherwise} \end{cases} \tag{A-23}$$

Using the definition of feasible server reallocation, we conclude that $g_{b,k}(n) = 1$ and $\hat{x}_b(n) \geq 1$. Therefore, feasibility constraints (2) and (3) (in the main paper) are satisfied and the feasibility of $\mathbf{y}^*(n)$ follows. Using the same argument for subsequent server reallocations results in the feasible withdrawal vector $\mathbf{y}'(n)$. □

**Lemma A.3.6.** *Consider the state $(\mathbf{x}(n), \mathbf{g}(n))$ and any two feasible withdrawal vectors $\mathbf{y}(n)$ and $\mathbf{y}'(n)$. Then, starting from $\mathbf{y}(n)$, the vector $\mathbf{y}'(n)$ can be obtained by performing a sequence of feasible server reallocations.*

*Proof.* To prove this lemma we construct one such sequence next.

Let $\mathbf{q}(n), \mathbf{q}'(n)$ denote two implementations of $\mathbf{y}(n), \mathbf{y}'(n)$ respectively. We can write

$$\mathbf{y}'(n) = \mathbf{y}(n) + \sum_{k=1}^{K} I(q'_k(n), q_k(n)) \tag{A-24}$$

where, by definition, server $k$ is connected to both queues $q_k(n)$ and $q'_k(n)$. Therefore, each interchange $I(q'_k(n), q_k(n))$ is equivalent to a feasible server reallocation. Note that $q_k(n) = q'_k(n)$ is possible, for some $k$, in which case $I(q'_k(n), q_k(n)) = 0$. By construction, all the interchanges in the right hand side of Equation (A-24) are feasible. □

We are now ready to prove Lemma 4 of Section 5.3.

*Proof.* We observe that $f \neq t$ must be true. Otherwise, we arrive at a contradiction, i.e., $+1 \leq D_f \leq -1$. This leaves three cases to consider:

Case 1: $f = 0$. This case is not possible by contradiction. By assumption, $D_0 \geq +1$, which means that $y_0^{MB}(n) \geq y_0(n)+1$. This case states that an MB policy idled at least one more server than $\pi$. Therefore, $\hat{x}_0^{MB}(n) \leq -1$. This makes queue 0 the shortest queue. Allocating the idled server to queue $t$, i.e., the interchange $I(t, 0)$, is both feasible (since $\mathbf{y}(n)$ is feasible by assumption) and balancing (by Lemma A.3.1). The interchange $I(t, 0)$ will result in a withdrawal vector $\mathbf{y}'(n) = \mathbf{y}^{MB}(n) + I(t, 0)$. Let $s$ be the order of queue $f = 0$ when ordering the vector $\hat{\mathbf{x}}^{MB}(n)$ in a descending manner. Therefore, $s = L+1$. Furthermore, in order for $I(t, 0)$ to be feasible queue $t$ must not be empty (according to feasibility constraint (23)) which implies that $\hat{x}_t^{MB}(n) \geq 1$ and the order of queue $t$ is $l < s$. Therefore, $\hat{x}_f^{MB}(n) \leq \hat{x}_t^{MB}(n) - 2$ and the interchange $I(t, 0)$ will reduce the imbalance index by $2(s - l)$ according to Equation (28). This implies that the new policy has less imbalance index than an MB policy. This contradicts the fact that any MB policy minimizes the imbalance index.

Case 2: $t = 0$. When $t = 0$ then the interchange $I(f, t)$ is the process of allocating an idled server to queue $f > 0$. This, according to Lemma A.3.1, is a balancing interchange.

Case 3: $t, f > 0$. We will show that this case will also result in a balancing interchange. Let $\mathbf{y}(n)$ be the original withdrawal vector. Let $\mathbf{y}^*(n)$ be the withdrawal vector resulted from the interchange $I(f, t)$, i.e.,

$$\mathbf{y}^*(n) = \mathbf{y}(n) + I(f, t)$$

Using the assumption $D_t \leq -1$ and Equation (16), we arrive at the following:

$$\begin{aligned} y_t^{MB}(n) - y_t(n) &\leq -1 \\ y_t^{MB}(n) &\leq y_t(n) - 1 = y_t^*(n) \\ y_t^{MB}(n) &\leq y_t^*(n) = y_t(n) - 1 \end{aligned} \tag{A-25}$$

and,

$$\begin{aligned} x_t(n) - y_t^{MB}(n) &\geq x_t(n) - y_t^*(n) = x_t(n) - (y_t(n) - 1) \\ \hat{x}_t^{MB}(n) &\geq \hat{x}_t^*(n) = \hat{x}_t(n) + 1, \quad t > 0 \end{aligned} \tag{A-26}$$

Similarly, using the assumption $D_f \geq +1$ and Equation (15), we have

$$\begin{aligned} y_f^{MB}(n) - y_f(n) &\geq +1 \\ y_f^{MB}(n) &\geq y_f(n) + 1 = y_f^*(n) \\ y_f^{MB}(n) &\geq y_f^*(n) = y_f(n) + 1 \end{aligned} \tag{A-27}$$

and,

$$\begin{aligned} x_f(n) - y_f^{MB}(n) &\leq x_f(n) - y_f^*(n) = x_f(n) - (y_f(n) + 1) \\ \hat{x}_f^{MB}(n) &\leq \hat{x}_f^*(n) = \hat{x}_f(n) - 1, \quad f > 0 \end{aligned} \tag{A-28}$$

To show that $I(f,t)$ in this case is a balancing interchange, we have to show that $\hat{x}_f(n) \geq \hat{x}_t(n) + 1$. Suppose to the contrary that $\hat{x}_f(n) \leq \hat{x}_t(n)$; then, from Equations (A-26) and (A-28), we have

$$\begin{aligned} \hat{x}_f(n) &\leq \hat{x}_t(n) \\ \hat{x}_f^*(n) + 1 &\leq \hat{x}_t^*(n) - 1 \\ \hat{x}_f^*(n) &\leq \hat{x}_t^*(n) - 2 \end{aligned} \tag{A-29}$$

From (A-26) and (A-28) we have,

$$\begin{aligned} \hat{x}_f^{MB}(n) &\leq \hat{x}_f^*(n) \leq \hat{x}_t^*(n) - 2 \leq \hat{x}_t^{MB}(n) - 2 \\ \hat{x}_f^{MB}(n) &\leq \hat{x}_t^{MB}(n) - 2 \end{aligned} \tag{A-30}$$

The differences $D_f \geq +1$ and $D_t \leq -1$ by assumption, i.e., there is at least one more (respectively one less) server allocated to queue $f$ (respectively queue $t$) under the MB policy. Therefore, $y^*(n) = y^{MB}(n) + I(t,f)$ is feasible. Therefore, Inequality (A-30) is a contradiction according to Lemma A.3.3. Therefore, $\hat{x}_f(n) \geq \hat{x}_t(n) + 1$ and by definition the interchange $I(f,t)$ is a balancing interchange. □

## A.4 Proof of Lemma 5

We present the proof of Lemma 5 of Section 5.3 in this section. Lemma A.4.1, stated below, guarantees the existence of a feasible interchange.

**Lemma A.4.1.** *Consider a given state* $(\mathbf{x}(n), \mathbf{g}(n))$ *and a policy* $\pi \in \Pi_{n-1}$ *that selects a withdrawal vector* $\mathbf{y}(n)$ *during time slot* $n$. *Let* $F \neq \{\phi\}$ *and* $T \neq \{\phi\}$, *where* $\{\phi\}$ *is the empty set, denote the sets of queues for which* $D_f \geq +1$ *and* $D_t \leq -1$, *respectively. Then, there exist at least two queues* $f \in F$ *and* $t \in T$ *such that the interchange* $I(f,t)$ *is feasible.*

*Proof.* Let $\pi^* \in \Pi^{MB}$ be an MB policy that selects the withdrawal vector $\mathbf{y}^*(n)$ during time slot $n$. Let $\mathbf{D} = \mathbf{y}^*(n) - \mathbf{y}(n)$. Furthermore, let $\mathbf{q}^*(n)$ and $\mathbf{q}(n)$ be two implementations of $\mathbf{y}^*(n)$ and $\mathbf{y}(n)$ respectively. From Lemma A.3.6 we have:

$$\mathbf{y}^*(n) = \mathbf{y}(n) + \sum_{k=1}^{K} I(q_k^*(n), q_k(n)) \tag{A-31}$$

The summation on the right-hand side of Equation (A-31) composed of $K$ terms, each represents a reallocation of a server $k$ from queue $q_k(n)$ to queue $q_k^*(n)$. Such server reallocation can be formulated as an interchange $I(q_k^*(n), q_k(n))$.

In the following, we will selectively use $i$ out of the $K$ terms (or equivalently, $K$ server reallocations) in Equation (A-31) to construct a feasible interchange $I(r_1, r_{i+1}) = I(r_1, r_2) + I(r_2, r_3) + \cdots + I(r_i, r_{i+1})$ for some $i \leq K$, s.t. $r_1 \in F$ and $r_{i+1} \in T$. Note that this interchange is composed of $i$ server reallocations each represents a term in the summation in Equation (A-31) above. We will show that such a queue $r_{i+1}, i \leq K$ that belongs to $T$ does exist and the interchange $I(r_1, r_{i+1})$ is feasible.

Let $r_1 \in F, r_1 \in \{1, 2, \ldots L\}$ then using Equation (A-31) we can write

$$y_{r_1}^*(n) = y_{r_1}(n) + \sum_{k=1}^{K} \mathbb{1}_{\{q_k^*(n)=r_1\}} - \mathbb{1}_{\{q_k(n)=r_1\}} \tag{A-32}$$

Since $r_1 \in F$ by assumption, then we have

$$y_{r_1}^*(n) - y_{r_1}(n) \geq 1 \tag{A-33}$$

From Equations (A-32) and (A-33) we conclude

$$\sum_{k=1}^{K} \mathbb{1}_{\{q_k^*(n)=r_1\}} \geq \sum_{k=1}^{K} \mathbb{1}_{\{q_k(n)=r_1\}} + 1 \tag{A-34}$$

In words, there is at least one more server allocated to queue $r_1$ under $\pi^*$ than the servers allocated to queue $r_1$ under $\pi$. Let $k_1$ be one such server. From (A-34) we conclude that one of the $K$ terms in Equation (A-31) must be $I(q^*_{k_1}(n), q_{k_1}(n))$ such that $q^*_{k_1}(n) = r_1, q_{k_1}(n) = r_2, k_1 \in \{1, 2, \ldots K\}$. In other words, a server $k_1$ and two queues $r_1 = q^*_{k_1}(n)$ and $r_2 = q_{k_1}(n)$ must exist such that the interchange $I(r_1, r_2)$ is feasible.

The feasibility of $I(r_1, r_2)$ stems from the fact that server $k_1$ is allocated to queues $r_1$ and $r_2$ under two different feasible policies, namely $\pi^*$ and $\pi$. This is possible only if

$$g_{r_1,k_1}(n) = g_{r_2,k_1}(n) = 1 \tag{A-35}$$

Furthermore, using Equation (A-33) we can write

$$\begin{aligned} y^*_{r_1}(n) &\geq y_{r_1}(n) + 1 \\ x_{r_1}(n) - y^*_{r_1}(n) &\leq x_{r_1}(n) - y_{r_1}(n) - 1 \\ 0 \leq \hat{x}^*_{r_1}(n) &\leq \hat{x}_{r_1}(n) - 1 \end{aligned} \tag{A-36}$$

From Equation (A-36) we conclude

$$\hat{x}_{r_1}(n) \geq 1 \tag{A-37}$$

Equations (A-35) and (A-37) are sufficient for the feasibility of the interchange $\mathbf{y}(n) + I(r_1, r_2)$.

Consider queue $r_2$ above. One of the following two cases may apply:

Case (1) $r_2 \in T$: The proof of the lemma in this case is completed by letting $f = r_1$ and $t = r_2$. The resulted interchange $I(f, t), f \in F, t \in T$ is feasible by construction and the lemma follows.

Case (2) $r_2 \notin T$: Define $\mathbf{y}^1(n)$ as follows:

$$\mathbf{y}^1(n) = \mathbf{y}(n) + I(r_1, r_2) \tag{A-38}$$

Using Lemma A.3.5 we conclude that $\mathbf{y}^1(n)$ is a feasible withdrawal vector. From Equations (A-38) and (13) we can write

$$y^1_{r_2}(n) = y_{r_2}(n) - 1 \tag{A-39}$$

From Equation (A-39) and since $r_2 \notin T$ we conclude

$$y^*_{r_2}(n) \geq y_{r_2}(n) \geq y^1_{r_2}(n) + 1 \tag{A-40}$$

Therefore, one of the terms of the summation in Equation (A-31) must be a server reallocation of some server $k_2$ from queue $r_3 = q_{k_2}(n)$ to queue $r_2 = q^*_{k_2}(n)$, i.e., the interchange $I(r_2, r_3)$ is a feasible server reallocation. It follows that $\mathbf{y}^2(n)$ is feasible, where

$$\begin{aligned} \mathbf{y}^2(n) &= \mathbf{y}^1(n) + I(r_2, r_3) \\ &= \mathbf{y}(n) + I(r_1, r_2) + I(r_2, r_3) \\ &= \mathbf{y}(n) + I(r_1, r_3) \end{aligned} \tag{A-41}$$

If $r_3 \in T$ then we complete the proof using the argument in case (1) above. Otherwise, we repeat the argument in case (2) again.

Repeating the previous argument $i$ times, $1 \leq i \leq K$, we arrive at the following relationship:

$$\mathbf{y}^i(n) = \mathbf{y}(n) + \sum_{j=1}^{i} I(r_j, r_{j+1}), \tag{A-42}$$

where by construction, each one of the $i$ terms in Equation (A-42) above corresponds uniquely to one of the terms of the summation in Equation (A-31). For every $i$ we check to see whether $r_{i+1} \in T$ or not. If not then we have

$$y^*_{r_{i+1}}(n) \geq y_{r_{i+1}}(n) \geq y^i_{r_{i+1}}(n) + 1 \tag{A-43}$$

Repeating the argument $K$ times (one for each term of the summation in Equation (A-31)) we show that a queue $r_{i+1} \in T$, where $I(r_i, r_{i+1})$ is one of the terms in Equation (A-31), must exist.

In order to do that, we assume to the contrary that $r_{i+1} \notin T, \forall i = 1, 2, \ldots K$. The $K^{th}$ (last) server reallocation $I(r_K, r_{K+1}), r_K = q^*_{k_K}(n), r_{K+1} = q_{k_K}(n)$ will result in the withdrawal vector $\mathbf{y}^K(n)$, such that,

$$\mathbf{y}^K(n) = \mathbf{y}(n) + \sum_{j=1}^{K} I(r_j, r_{j+1}) \tag{A-44}$$

Since there is one-to-one correspondence between the summation terms in Equation (A-44) and those in Equation (A-31) by construction, then we can write

$$\mathbf{y}^K(n) = \mathbf{y}(n) + \sum_{k=1}^{K} I(q^*_k(n), q_k(n)) \tag{A-45}$$

However, since $r_{K+1} \notin T$ then

$$y^*_{r_{K+1}}(n) \geq y_{r_{K+1}}(n) \geq y^K_{r_{K+1}}(n) + 1 \tag{A-46}$$

Using Equations (A-31), (A-45) and (A-46) we arrive at the following contradiction:

$$\sum_{k=1}^{K} I(q^*_k(n), q_k(n)) \neq \sum_{k=1}^{K} I(q^*_k(n), q_k(n)) \tag{A-47}$$

Therefore, we conclude that there must exist a queue $r_{i+1} \in T$ such that server $k_i$ reallocation $I(r_i, r_{i+1}), r_i = q^*_{k_i}(n), r_{i+1} = q_{k_i}(n)$ is feasible. Let $f = r_1$ and $t = r_{i+1}$. It follows that the interchange $I(f,t) = I(r_1, r_{i+1})$ is feasible and the lemma follows. □

*Proof for Lemma 5.* The policy $\pi \in \Pi_{n-1} \subseteq \Pi$ and therefore $\mathbf{y}(n)$ is a feasible withdrawal vector. A necessary feasibility condition is the one given by Equation (6), i.e.,

$$\sum_{i=0}^{L} y_i(n) = K \tag{A-48}$$

Therefore, the total difference between the two vectors is bounded by:

$$0 \leq \sum_{i=0}^{L} |y^{MB}_i(n) - y_i(n)| \leq 2K, \tag{A-49}$$

or equivalently,

$$0 \leq \sum_{i=0}^{L} |D_i| \leq 2K \tag{A-50}$$

When the sum equals 0, then $\pi$ has the MB property during time slot $n$ according to Lemma A.3.4. When the sum equals $2K$, then one can conclude using Lemma A.3.2 that the sum of positive terms is equal to the sum of negative terms in the summation above. To put it differently, there are at most $K$ (+1)'s and $K$ (-1)'s in the difference vector $\mathbf{D}$.

According to Lemma 4 and Lemma A.4.1, a pair of queues, $f$ and $t$, where $I(f,t)$, $D_f \geq +1$ and $D_t \leq -1$, is a feasible balancing interchange does exist. Since we have $\sum_i |D_i|/2$ such pairs of queues, then applying the balancing interchange described by Lemma 4 for $\sum_i |D_i|/2$ times will result in a new difference vector $\mathbf{D}^* = 0$, i.e., $\mathbf{y}^*(n) = \mathbf{y}^{MB}(n)$.

Therefore, using Lemma 4 we conclude that for any arbitrary feasible policy $\pi \in \Pi_{n-1}$ and a corresponding withdrawal vector $\mathbf{y}(n)$, at most $\sum_i |D_i|/2$ feasible balancing interchanges are required to make $\mathbf{y}^*(n) = \mathbf{y}^{MB}(n)$ and hence the resulting policy $\pi^* \in \Pi_n$. □

## A.5 Coupling Method and the Proof of Lemma 6

### A.5.1 The Coupling Method

If we want to compare probability measures on a measurable space, it is often possible to construct random elements, with these measures as their distributions, on a common probability space, such that the comparison can be carried out in terms of these random elements rather than the probability measures. The term *stochastic coupling* (or coupling) is often used to refer to any such construction. In the notation of [1], a formal definition of coupling of two probability measures on the measurable space $(E, \mathcal{E})$ (the state space, e.g., $E = \mathcal{R}, \mathcal{R}^d, \mathcal{Z}_+, etc.$) is given below; see [1] for more details.

A random element in $(E, \mathcal{E})$ is a quadruple $(\Omega, \mathscr{F}, \mathbf{P}, X)$, where $(\Omega, \mathscr{F}, \mathbf{P})$ is the sample space and $X$ is the class of measurable mappings from $\Omega$ to $E$ ($X$ is an $E$-valued random variable, s.t. $X^{-1}(B) \in \mathscr{F}$ for all $B \in \mathcal{E}$).

**Definition:** *A coupling of the two random elements* $(\Omega, \mathscr{F}, \mathbf{P}, \mathbf{X})$ *and* $(\Omega', \mathscr{F}', \mathbf{P}', \mathbf{X}')$ *in* $(E, \mathcal{E})$ *is a random element* $(\hat{\Omega}, \hat{\mathscr{F}}, \hat{\mathbf{P}}, (\hat{\mathbf{X}}, \hat{\mathbf{X}}'))$ *in* $(E^2, \mathcal{E}^2)$ *such that*

$$\mathbf{X} \stackrel{\mathscr{D}}{=} \hat{\mathbf{X}} \quad \text{and} \quad \mathbf{X}' \stackrel{\mathscr{D}}{=} \hat{\mathbf{X}}', \tag{A-51}$$

*where* $\stackrel{\mathscr{D}}{=}$ *denotes 'equal in distribution'.*

**Remark 3.** *The above definition makes no assumption about the distribution of the collection of random variables* $\mathbf{X}$*; for example,* $\mathbf{X}$ *may be a sequence of non-i.i.d. random variables, as is the case with the arrival process in our model.* □

We apply the coupling method to our proof as follows: Let $\omega$ and $\pi$ be a given sample path of the system state process and server allocation policy. The values of the sequences $\{X(n)\}$ and $\{Y(n)\}$ can be completely determined by $\omega$ and $\pi$. We denote the ensemble of all random variables as system $S$. A new sample path, $\tilde{\omega}$ and a new policy $\tilde{\pi}$ are constructed as we specify in detail in the proof. We employ the tilde notation in all random variables that belong to the new system; we denote the ensemble of

all random variables (in the new construction) as system $\tilde{S}$. Then, in the coupling definition, $\hat{\omega} = (\omega, \tilde{\omega})$ and the "coupled" processes of interest in Equation (A-51) will be the queue sizes $\hat{\mathbf{X}} = \{X(n)\}$ and $\hat{\mathbf{X}}' = \{\tilde{X}(n)\}$.

We define $\omega$ as the sequence of sample values of the random variables $(\mathbf{X}(1), \mathbf{G}(1), \mathbf{Z}(1), \mathbf{G}(2), \mathbf{Z}(2), \ldots)$, i.e., $\omega \equiv (\mathbf{x}(1), \mathbf{g}(1), \mathbf{z}(1), \mathbf{g}(2), \mathbf{z}(2), \ldots)$. The sample path $\tilde{\omega} \equiv (\tilde{\mathbf{x}}(1), \tilde{\mathbf{g}}(1), \tilde{\mathbf{z}}(1), \tilde{\mathbf{g}}(2), \tilde{\mathbf{z}}(2), \ldots)$ is constructed such that (a) $\tilde{\mathbf{x}}(1) = \mathbf{x}(1)$, (b) $\tilde{\mathbf{g}}(n)$ the same as $\mathbf{g}(n)$ except for two elements that are exchanged, (c) $\tilde{\mathbf{z}}(n)$ the same as $\mathbf{z}(n)$ except for two elements that are exchanged. Which elements are exchanged is detailed in the proof. In the symmetrical system we are studying, $\{\tilde{\mathbf{G}}(n), \tilde{\mathbf{Z}}(n)\}$ has the same distribution as $\{\mathbf{G}(n), \mathbf{Z}(n)\}$, since the distributions of $\mathbf{G}(n)$ and $\mathbf{Z}(n)$ will not change when reordering their elements. The mappings from $\mathbf{G}(n)$ to $\tilde{\mathbf{G}}(n)$ and from $\mathbf{Z}(n)$ to $\tilde{\mathbf{Z}}(n)$ are one-to-one.

The new policy $\tilde{\pi}$ is constructed (by showing how $\tilde{\pi}$ chooses the withdrawal vector $\tilde{\mathbf{y}}(\cdot)$) as detailed in the proof. Then using Equation (7), the new states $\mathbf{x}(\cdot), \tilde{\mathbf{x}}(\cdot)$ are determined under $\pi$ and $\tilde{\pi}$. The goal is to prove that the relation

$$\tilde{\mathbf{x}}(t) \prec_p \mathbf{x}(t) \tag{A-52}$$

is satisfied at all times $t$. Towards this end, the preferred order (introduced in Section 5.1) can be described by the following property:
**Property D:** $\tilde{\mathbf{x}}$ is preferred over $\mathbf{x}$ ($\tilde{\mathbf{x}} \prec_p \mathbf{x}$) if and only if one of the following statements holds:

(D1) $\tilde{\mathbf{x}} \leq \mathbf{x}$: the two vectors are component-wise ordered;

(D2) $\tilde{\mathbf{x}}$ is a two-component permutation of $\mathbf{x}$ as described in (2-) in Section 5.1.

(D3) $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a *"balancing interchange"* as described in (3-) in Section 5.1.

### A.5.2 Proof of Lemma 6 of Section 5

*Proof.* Fix an arbitrary policy $\pi \in \Pi_\tau^h$ and a sample path $\omega = (\mathbf{x}(1), \mathbf{g}(1), \mathbf{z}(1), \ldots)$, where $\mathbf{x}(.), \mathbf{g}(.)$ and $\mathbf{z}(.)$ are sample values of the random variables $\mathbf{X}(.), \mathbf{G}(.)$ and $\mathbf{Z}(.)$. Let , $\pi^* \in \Pi^{MB}$ be an MB policy that works on the same system. The policy $\pi^*$ chooses a withdrawal vector $y^*(t), \forall t$.

The proof has two parts; Part 1 provides constructions for $\tilde{\omega}$ and $\tilde{\pi}$ (as defined by Lemma 6 statement) for times up to $t = \tau$. Part 2 does the same for $t > \tau$.

***Part 1:*** For the construction of $\tilde{\omega}$, we let the arrivals and channel states be the same in both systems at all time slots before $\tau$, i.e., $\tilde{\mathbf{z}}(t) = \mathbf{z}(t)$ and $\tilde{\mathbf{g}}(t) = \mathbf{g}(t)$ for all $t < \tau$. We construct $\tilde{\pi}$ such that it chooses the same withdrawal vector as $\pi$, i.e., we set $\tilde{\mathbf{y}}(t) = \mathbf{y}(t)$ for all $t < \tau$. In this case, at $t = \tau$, the resulting queue sizes are equal, i.e., $\tilde{\mathbf{x}}(\tau) = \mathbf{x}(\tau)$. Therefore at $t = \tau$, property (D1) (of the preferred order) holds true.

At time slot $\tau$, let $\tilde{\omega}$ have the same channel connectivities and arrivals as $\omega$, i.e., let $\tilde{\mathbf{g}}(\tau) = \mathbf{g}(\tau)$ and $\tilde{\mathbf{z}}(\tau) = \mathbf{z}(\tau)$. Furthermore, let $\mathbf{D} = y^*(\tau) - y(\tau)$. Recall that $h = \sum_{i=0}^{L} |D_i|/2$. Then one of the following two cases may apply:

1- During time slot $t = \tau$, the original policy $\pi$ differs from $\pi^*$, the MB policy, by *strictly* less than $h$ balancing interchanges. Then $\pi \in \Pi_\tau^{h-1}$ as well, so we set $\tilde{\mathbf{y}}(\tau) = \mathbf{y}(\tau)$. In this case, the resulting queue sizes $\tilde{\mathbf{x}}(\tau+1), \mathbf{x}(\tau+1)$ will be equal, property (D1) holds true and (A-52) is satisfied at $t = \tau + 1$.

2- During time slot $t = \tau$, $\pi$ differs from the MB policy $\pi^*$ by *exactly* $h$ balancing interchanges. Since $\pi \in \Pi_\tau^h$ and $h > 0$, we can identify two queues $l$ and $s$ such that: (a) $D_l \geq 1$, (b) $D_s \leq -1$, and (c) $I(l, s)$ is feasible. Lemma 5 states that such queues must exist when $h > 0$. The construction of $\tilde{\pi}$ is completed in this case by performing the interchange $I(l, s)$, i.e.,

$$\tilde{\mathbf{y}}(\tau) = \mathbf{y}(\tau) + I(l, s), \tag{A-53}$$

or equivalently,

$$\tilde{\hat{\mathbf{x}}}(\tau) = \hat{\mathbf{x}}(\tau) - I(l, s) \tag{A-54}$$

According to Lemma 4, this interchange is balancing. Therefore, the queue lengths at the beginning of time slot $(\tau + 1)$ under the two policies satisfy property (D3), and (A-52) is satisfied at $t = \tau + 1$.

***Part 2:*** The above concluded the construction of $\tilde{\omega}, \tilde{\pi}$ during $t = \tau$. The next step is to construct $\tilde{\omega}, \tilde{\pi}$ for times $n > \tau$, such that the partial order (A-52) is preserved. To achieve this, we will use induction. We assume that $\tilde{\pi}$ and $\tilde{\omega}$ are defined up to time $n - 1$ and such that $\tilde{\mathbf{x}}(n) \prec_p \mathbf{x}(n)$. We will prove that at time slot $n$, $\tilde{\pi}$ can be constructed such that $\tilde{\mathbf{x}}(n+1) \prec_p \mathbf{x}(n+1)$, i.e., (A-52) holds at $t = n + 1$. In order for (A-52) to hold, we have to show that either D1, D2 or D3 holds at time slot $n + 1$.

The following three cases, that correspond to properties (D1), (D2) and (D3) introduced earlier will be considered next.

Case (1) $\tilde{\mathbf{x}}(n) \leq \mathbf{x}(n)$. The construction of $\tilde{\omega}$ is straightforward in this case. We set $\tilde{\mathbf{z}}(n) = \mathbf{z}(n)$ and $\tilde{\mathbf{g}}(n) = \mathbf{g}(n)$. We construct $\tilde{\pi}$ such that $\tilde{\mathbf{y}}(n) = \mathbf{y}(n)$. In this case, its obvious that $\tilde{\mathbf{x}}(n+1) \leq \mathbf{x}(n+1)$ and (A-52) holds at $t = n+1$.

Case (2) $\tilde{\mathbf{x}}(n)$ is a permutation of $\mathbf{x}(n)$, such that $\tilde{\mathbf{x}}(n)$ can be obtained from $\mathbf{x}(n)$ by permuting components $i$ and $j$ (as described in property D2 of the preferred order). For the construction of $\tilde{\omega}$, we set $\tilde{g}_{i,c}(n) = g_{j,c}(n)$ and $\tilde{g}_{j,c}(n) = g_{i,c}(n)$, for all $c = 1, 2, \ldots, K$; $\tilde{z}_i(n) = z_j(n)$ and $\tilde{z}_j(n) = z_i(n)$; the connectivities and arrivals for each one of the remaining queues are the same as in $\omega$. We construct $\tilde{\pi}$ such that $\tilde{y}_i(n) = y_j(n)$, $\tilde{y}_j(n) = y_i(n)$ and $\tilde{y}_m(n) = y_m(n)$ for all $m \neq i, j$. As a result, $\tilde{\mathbf{x}}(n+1)$ and $\mathbf{x}(n+1)$ satisfy property (D2) again and (A-52) is satisfied at $t = \tau + 1$.

Case (3) $\tilde{\mathbf{x}}(n)$ is obtained from $\mathbf{x}(n)$ by performing a balancing interchange for queues $i$ and $j$ as defined in property (D3). In this case $x_i(n) \geq x_j(n) + 1$, by the definition in (D3)[7]. There are two cases to consider:

(3.a) $x_i(n) = x_j(n) + 1$. Therefore, $\tilde{x}_i(n) = x_j(n)$ and $\tilde{x}_j(n) = x_i(n)$, i.e., the vectors $\mathbf{x}(n)$ and $\tilde{\mathbf{x}}(n)$ have components $i$ and $j$ permuted and all other components are the same. This case corresponds to case (2) above.

(3.b) $x_i(n) > x_j(n) + 1$. Depending on whether $\pi$ empties queues $i, j$ or not, the construction of $\tilde{\omega}$, $\tilde{\pi}$ will follow one of the following two arguments:

(i) $y_i(n) < x_i(n)$, i.e., *$\pi$ does not empty queue $i$*. We construct $\tilde{\omega}$ as in case (1). Furthermore, let $\tilde{y}_m(n) = y_m(n), \forall m \neq j$. If $\pi$ does not empty queue $j$, then let $\tilde{y}_j(n) = y_j(n)$ and property (D3) will be preserved and (A-52) is satisfied at $t = n+1$. If, on the other hand, policy $\pi$ empties queue $j$, then if under policy $\pi$ all the servers connected to queue $j$ are allocated, then let $\tilde{y}_j(n) = y_j(n)$ (i.e., $\tilde{\pi}$ is identical to $\pi$ at $t = n$). Therefore, (D3) holds and (A-52) satisfied at $t = n+1$.

In the event that $\pi$ empties queue $j$ without exhausting all the servers connected to queue $j$, then $\tilde{\pi}$ will be constructed such that one of these idling servers is allocated to queue $j$, i.e., $\tilde{y}_j(n) = y_j(n) + 1$, so that $\tilde{\pi}$ preserves the work conservation property at $t = n$. Since $\tilde{x}_j(n) = x_j(n) + 1$ by property (D3) and $\tilde{z}_j(n) = z_j(n)$ by assumption, then we have

$$\tilde{x}_j(n+1) = x_j(n+1) = z_j(n)$$

Since $\tilde{x}_i(n) = x_i(n) - 1$ by property (D3) and $\tilde{y}_i(n) = y_i(n)$ by construction,

[7]By definition, we have $x_i(n) > x_j(n)$, $\tilde{x}_i(n) = x_i(n) - 1$ and $\tilde{x}_j(n) = x_j(n) + 1$.

we have

$$\tilde{x}_i(n+1) = x_i(n+1) - 1$$

The rest of the queues will have the same lengths in both systems at $t = n+1$. Therefore, (D1) is satisfied at $t = n+1$ and (A-52) follows.

(ii) $y_i(n) = x_i(n)$, i.e., $\pi$ *empties queue* $i$. In this case, there are at least $x_i(n)$ servers connected to queue $i$ at $t = n$,

$$\sum_{c=1}^{K} g_{i,c}(n) \geq x_i(n).$$

For the construction of $\tilde{\omega}$, we set $\tilde{\mathbf{z}}(n) = \mathbf{z}(n)$, $\tilde{g}_{m,c}(n) = g_{m,c}(n)$ for all $m \neq i, j$, and for all $c$. For queues $i$ and $j$ we do the following:

Let $\mathcal{M}_i^n$ be the set of servers connected to queue $i$ at time slot $n$. Then let server $r \in \mathcal{M}_i^n$ be such that $q_r(n) = i$ (i.e., server $r$ is allocated to queue $i$ by policy $\pi$ at $t = n$). Now, we switch the connectivity of server $r$ to queue $i$ and that of server $r$ to queue $j$, i.e., we set $\tilde{g}_{j,r}(n) = g_{i,r}(n)$ and $\tilde{g}_{i,r}(n) = g_{j,r}(n)$.

We construct $\tilde{\pi}$ such that $\tilde{q}_r(n) = j$ and $\tilde{q}_c(n) = q_c(n), \forall c \neq r$. This means that $\tilde{\pi}$ differs from $\pi$, at $t = n$, by one server allocation (server $r$) that is allocated to queue $j$ (under $\tilde{\pi}$) rather than queue $i$ (under $\pi$). From equation (1), we can easily calculate that resulting queue lengths will be:

$$\begin{aligned} \tilde{x}_i(n+1) &= x_i(n+1) = z_i(n), \\ \tilde{x}_m(n+1) &= x_m(n+1), \qquad \forall\, m \neq i. \end{aligned}$$

In this case, it is obvious that property (D1) is satisfied and therefore (A-52) is again satisfied at $t = n+1$.

Cases (i) and (ii) are the only possible ones, since $\pi$ cannot allocate more servers to queue $i$ than its length. Note that policy $\tilde{\pi}$ belongs to $\Pi_\tau^{h-1}$ by construction in Part 1; its dominance over $\pi$ follows easily from relation (30). $\square$

# References


[1] T. Lindvall, *Lectures on the coupling method,* New York: Wiley(1992).

[2] D. Stoyan, *Comparison Methods for Queues and other Stochastic Models*, J. Wiley and Sons, Chichester, 1983.

[3] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, 39(2): 466 - 478, March 1993.

[4] N. Bambos and G. Michailidis, "On the stationary dynamics of parallel queues with random server connectivities," *Proceedings of 34th Conference on Decision and Control*, (CDC), New Orleans, LA (1995).

[5] N. Bambos and G. Michailidis, "On parallel queueing with random server connectivity and routing constraints," *Probability in Engineering and Informational Sciences,* 16: 185-203, 2002.

[6] A. Ganti, E. Modiano and J. N. Tsitsiklis, "Optimal transmission scheduling in symmetric communication models with intermittent connectivity," *IEEE Transactions on Information Theory*, 53(3): 998-1008, March 2007.

[7] S. Kittipiyakul, T. Javidi, "Delay-optimal server allocation in multiqueue multi-server systems with time-varying connectivities," *IEEE Transactions on Information Theory*, 55(5): 2319-2333, May 2009.

[8] C. Lott and D. Teneketzis, "On the optimality of an index rule in multichannel allocation for single-hop mobile networks with multiple service classes," *Probability in the Engineering and Information Services*, 14(3): 259-297, July 2000.

[9] G. Koole, Z. Liu and R. Righter, "Optimal transmission policies for noisy channels", *Operations Research*, 49(6): 892-899, Nov. 2001.

[10] X. Liu, E.K.P. Chong, N.B. Shroff, "Optimal opportunistic scheduling in wireless networks", *IEEE 58th Vehicular Technology Conference,* Vol.3, Oct. 2003.

[11] X. Liu, E. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks" *Computer Networks*, 41(4): 451-474, Mar. 2003.

[12] R. Agrawal and V. Subramanian, "Optimality of certain channel aware scheduling policies", *The 40th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, Oct. 2002.

[13] M. Andrews, "Instability of the proportional fair scheduling algorithm for HDR", *IEEE Transactions on Wireless Communications*, 3(5): 14221426, 2004.

[14] M. Andrews and L. Zhang, "Scheduling over a time-varying user-dependent channel with applications to high speed wireless data", *Journal of the ACM* (JACM), 52(5): 809-834, Sep. 2005.

[15] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control", *IEEE INFOCOM 05*, Miami, FL, Mar. 2005.

[16] A. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation", *Operations Research*, 53: 1225, 2005.

[17] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks", *IEEE/ACM Transactions on Networking*, 7(4): 473-489, 1999.

[18] S. Shakkottai and A. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule", *American Mathematical Society Translations*, Series 2, Vol. 207, 2002.

[19] R. Lidl and G. Pilz, *Applied abstract algebra, 2nd edition,* Undergraduate Texts in Mathematics, Springer,(1998).