Optimal Server Assignment in Multi-Server Queueing Systems with Random Connectivities

Hassan Halabian, Ioannis Lambadaris, and Yannis Viniotis

Abstract: **In this paper, we provide complementary results on delay-optimal server allocation in multi-queue multi-server (MQMS) systems with random connectivities. More specifically, we consider an MQMS system where each queue is limited to get service by at most one server during each time slot. It is known that maximum weighted matching (MWM) is a throughput-optimal server assignment policy for such a system. In this paper, using dynamic coupling argument we prove that for a system with i.i.d. Bernoulli arrivals and connectivities, MWM minimizes, in stochastic ordering sense, a range of cost functions of the queue lengths such as total queue occupancy (which implies minimization of average queueing delay). Finally, we propose a low complexity heuristic server assignment policy for MQMS systems namely least connected server first/longest connected queue (LCSF/LCQ) and through simulations we show that it performs very closely compared with the optimal policy in terms of average queueing delay.**

Index Terms: **Delay-optimal server allocation, dynamic coupling, maximum weighted matching, multi-server queueing systems.**

I. INTRODUCTION

MULTI-SERVER queueing models with random connectivities have been used to study optimal resource allocation in wireless networks [1]–[6]. While various performance criteria including the stable throughput region and utility functions of the admitted traffic rates have been studied in several papers [1]–[3], [5], [7]–[9], average queueing delay has received less attention. The inherent randomness in wireless channels makes delay-optimal resource allocation a challenging problem in wireless networks. In this paper, we focus on delay-optimal server assignment in a time-slotted, multi-queue, multi-server system with random connectivities. Random connectivities can model unreliable and randomly varying wireless channels. Although this model is a simplified representation of a real wireless system, nevertheless it does provide valuable intuition for the performance optimization of real systems. Similar modeling approaches have already appeared in [2], [3], [5], [6], [10]–[12].

Throughput-optimal server allocation in queueing networks has been studied and understood well in queueing theory. It is known that back-pressure algorithm is a throughput-optimal algorithm for resource allocation and routing in queueing networks. While in throughput-optimality the objective is to determine a policy that maximizes the stable throughput region [1], [7], in *delay-optimality* the goal is to determine a policy that minimizes the average queueing delay. A server allocation policy may be throughput-optimal but not delay-optimal; however, a delay-optimal policy (for all the arrival rates) is always throughput-optimal. In [2], the authors proved that for a *multi-queue, single-server* system with i.i.d. Bernoulli arrival and connectivity processes, longest connected queue (LCQ) policy is both throughput-optimal and delay-optimal. The extension of this result for non-i.i.d. case is still an open problem.

In generalizing the results to *multi-queue, multi-server* (MQMS) systems, various systems have been studied [3]–[6]. The authors in [3] studied the throughput-optimal policy in two MQMS models. In the first model, there is no limitation on the number of servers allocated to a queue during each time slot. This model is denoted as MQMS-Type1 in our paper. In the second model which is a specific case of MQMS-Type1 model, the queues are limited to get service from at most one server per time slot. This model is denoted as MQMS-Type2 in our paper. The authors in [3] studied the stability region of the MQMS systems in the case of infrequent channel state information. In [5], the authors provided an explicit characterization of the network stability region of MQMS-Type1 system for general stationary channel processes. While the work in [3], [5] focus on stability region analysis of multi-server queueing models, research in [4], [6], [13] considers delay-optimal server allocation in such systems. The authors in [6] considered a queueing model with a set of parallel queues and i.i.d. Bernoulli packet arrivals that are competing to attract service from K identical servers forming a *server-bank*. The connectivities of the queues to the *entire* server-bank are assumed to be i.i.d. Bernoulli processes. Each queue is restricted to receive service from *at most one server during each time slot*. The authors proposed LCQ policy in which the servers of the server-bank are allocated to the K longest connected queues at each time slot. Using dynamic coupling and stochastic ordering, they proved the delay optimality of LCQ policy for such a system. The work in [4] focuses on delay-optimal server allocation problem in MQMS-Type1 system. In [4], the authors argue that in general, achieving instantaneous throughput and load balancing is impossible in a general MQMS system. However, they showed that this goal is attainable in the special case with ON-OFF channel processes. They also introduced maximum-throughput load-balancint (MTLB) policy and using dynamic programming showed that this policy minimizes a class of cost functions including total average delay for the case of *two queues* with i.i.d., Bernoulli-distributed

Manuscript received September 12, 2018 approved for publication by Hyoil Kim, Division III, February 20, 2019.

This work was supported by Mathematics of Information Technology and Complex Systems (MITACS) and Natural Sciences and Engineering Research Council of Canada (NSERC).

H. Halabian and I. Lambadaris are with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, e-mail: {hassanh, ioannis}@sce.carleton.ca.

Y. Viniotis is with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina, e-mail: candice@ncsu.edu.

H. Halabian is the corresponding author.

Digital Object Identifier: 10.1109/JCN.2019.000023

1229-2370/19/$10.00 © 2019 KICS

arrivals and connectivities. More precisely, the work in [4] proves the optimality of MTLB policy via a series of lemmas which establish a set of properties for the optimal value function inductively. The authors show the optimality of MTLB policy when there are $N = 2$ users, using dynamic programming (DP) arguments and the properties of the DP value function. In contrast to [4], in our paper we use stochastic ordering and dynamic coupling arguments to prove the optimality of maximum weighted matching (MWM), in stochastic ordering sense, for a class of cost functions of the queue length process. In our approach, we use Proposition 1 of [4] which proves that MWM results in the most balanced queue state in MQMS-Type2 systems and use it in our dynamic coupling arguments in Lemma 2, Lemma 3, Theorem 1 and Theorem 2 to show the delay optimality of MWM. Moreover, we show the optimality of the MWM policy for a broad class of cost functions including L^r-norm of the queue occupancy vector. A special case of the cost function would be the total queue occupancy of the system which results in minimum queueing delay. The work in [13] considers delay and throughput-optimal server allocation in MQMS-Type1 system in the case of many-server many-queue asymptotic regime where the number of servers (and queues) goes to infinity. The authors developed new easy-to-verify sufficient conditions for rate-function delay optimality in the asymptotic regime. Using this result, they proved rate-function delay optimality for a class of oldest packets first (OPF) policies and throughput optimality for a large class of maximum weight in the fluid limit (MWF) policies. As opposed to [13], our analysis in this paper for MQMS-Type2 is provided for non-asymptotic case (similar to [2], [4], [6]). In this paper, we focus on MWM policy in MQMS-Type2 system and prove that this throughput-optimal policy *is also delay-optimal* for an MQMS-Type2 system with i.i.d. arrival and connectivity processes. Our work provides incremental results for MQMS-Type2 system using the results derived previously in [4] for MQMS-Type1 system. On the other hand, our work extends the system model and the analytical methodology used in [2], [6] for single-server system to MQMS-Type2 system. In particular, the researchers in [2], [6] have considered queueing models where a *single server* or a *server-bank* is randomly connected to a set of parallel queues. In this paper, we consider a more general model where *each individual server is randomly connected* to each queue.

Our contributions in this paper are summarized as follows: First, for an MQMS-Type2 system we conclude that during each time slot, maximum weighted matching (MWM) policy will result in the most balanced queue vector in the system. Second, using this result in conjunction with the notions of stochastic ordering and dynamic coupling, we prove the delay optimality of MWM policy for an MQMS-Type2 system with i.i.d. Bernoulli arrivals and connectivities. More specifically, we prove that MWM minimizes, in *stochastic ordering* sense, a range of cost functions of queue lengths including total queue occupancy[1]. The optimality of MWM can be easily extended for MQMS systems with imperfect services where the service of a scheduled packet fails randomly with a certain probability and systems with more general connectivity and arrival processes which follow conditional permutation invariant distributions. Finally, we propose a low complexity heuristic algorithm called least connected server first/longest connected queue (LCSF/LCQ) as an alternative for the optimal policy in MQMS systems (MQMS-Type1 and MQMS-Type2). In LCSF/LCQ the servers are selected sequentially for assignment based on the number of connectivities incident to them. Using simulations, we compare the delay performance of LCSF/LCQ policy with the optimal one in both MQMS-Type1 and MQMS-Type2 and show how closely they perform.

While preliminary presentation of the initial results of the paper were presented in two conference papers [14], [15], none of them provide detailed theoretical analysis of the results particularly the proofs of the lemmas and the theorems. Furthermore, in this paper we complement the results of [14], [15] by adding Lemma 3 and Theorem 2 to show the optimality of *any MWM policy* for MQMS-Type2 system. In other words, we show that the optimal policy is not a unique policy and any policy following the Maximum Weight principle is delay optimal for MQMS-Type2 system. Moreover, as mentioned earlier we introduce LCSF/LCQ algorithm as a low complexity heuristic server allocation algorithm for MQMS systems (MQMS-Type1 and MQMS-Type2) whose performance is compared with the optimal policies through simulations.

The rest of this paper is organized as follows. In Section II, we introduce the queueing model and the required notation. In Section III, we describe the MWM server assignment policy. In Section IV, we prove the delay optimality of MWM server assignment policy. In Section V, we present the simulation results where we evaluate the performance of the optimal policies for MQMS-Type1 and MQMS-Type2, i.e., MWM and MTLB policies, respectively. Furthermore, we propose a heuristic policy (LCSF/LCQ) for each system and compare its performance with the optimal one in terms of average queue occupancy (or equivalently average queueing delay). Finally, we summarize our conclusions in Section VI.

II. MODEL DESCRIPTION

Throughout the paper, random variables are represented by CAPITAL letters and lower case letters are used to represent sample values of the random variables. Moreover, we use boldface font to represent matrices and vectors.

We consider a time-slotted, MQMS-Type2 system consisting of a set of parallel queues $\mathcal{N} = \{1, 2, \cdots, N\}$ with infinite buffer space for each queue (see Figure 1). Packets in this system are assumed to have constant length and require one time slot to complete service. The service to this set of queues is provided by a set of identical servers $\mathcal{K} = \{1, 2, \cdots, K\}$. The connectivity of each queue $n \in \mathcal{N}$ to each server $k \in \mathcal{K}$ at each time slot t is random and varying across time slots. We denote the connectivity of queue n to server k at time slot t by $C_{n,k}(t) \in \{0, 1\}$. When $C_{n,k}(t) = 1$ ($C_{n,k}(t) = 0$), queue n is connected to (disconnected from) server k at time slot t. The connectivity variables $C_{n,k}(t)$ are assumed to be i.i.d. Bernoulli random variables with a fixed parameter p.

At any time slot, each server can serve at most one packet from a connected, non-empty queue. We do not allow server

[1] The optimality of MWM is proven among all causal server assignment policies.

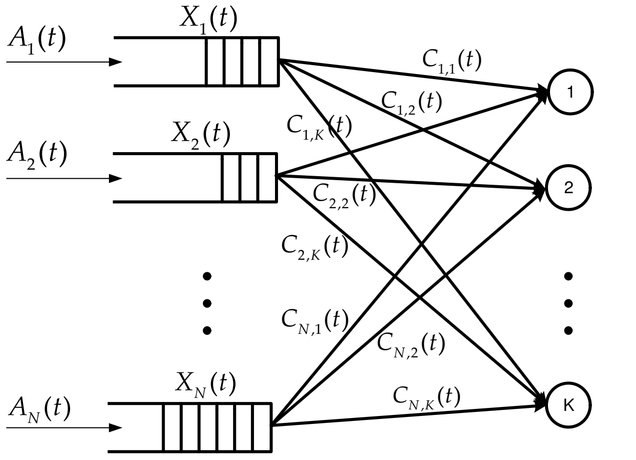

Fig. 1. Discrete-time MQMS-Type2 system with N parallel queues and K servers.

sharing in the system, i.e., a server can serve at most one queue per time slot. We also assume that *at most one server* can be assigned to any connected queue during a time slot.

Let $A_n(t)$ denote the number of packet arrivals to queue n at time slot t. We assume that new arrivals at each time slot are added to the queues at the end of the time slot. The arrival variables $A_n(t)$ are assumed to be i.i.d. Bernoulli random variables with the same parameter λ for all n and t[2].

We denote the length of queue n at the end of time slot t (i.e., after adding the new arrivals) by $X_n(t)$. Hence, $X_n(t)$ represents the number of packets in the nth queue at the end of time slot t (or beginning of time slot $t+1$).

Any server assignment policy π in MQMS-Type2 system is a bipartite matching between sets $\mathcal{N}$ and $\mathcal{K}$ and is fully determined by its indicator variables $M_{n,k}^{(\pi)}(t)\ \forall n \in \mathcal{N}, \forall k \in \mathcal{K}, t = 1, 2, \cdots$ defined to be 1 if server k is assigned to queue n by policy π at time slot t and zero otherwise. The $N \times K$ matrix $\boldsymbol{M}^{(\pi)}(t) = (M_{n,k}^{(\pi)}(t)), \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$ is defined as the *employed matching by policy* π at time slot t.

III. MAXIMUM WEIGHTED MATCHING (MWM) POLICY

In [1], [7], it was shown that *back-pressure* algorithm maximizes the stable throughput region of a general data network, i.e., it is throughput-optimal. The reader may refer to [1], [7] for more information regarding the back-pressure algorithm. For the model introduced in Section II, the back-pressure algorithm reduces to the following optimization problem at each time slot t [3]. In the optimization problem (1), $M_{n,k}(t)$'s are the optimization variables and $X_n(t-1)$ and $C_{n,k}(t)$ are known parameters.

$$\begin{aligned}
&\underset{M_{n,k}(t),\,\forall n,\,k}{\texttt{Maximize:}} && \sum_{n=1}^{N} X_n(t-1) \sum_{k=1}^{K} M_{n,k}(t) C_{n,k}(t) && (1)\\
&\texttt{Subject to:} && \sum_{k=1}^{K} M_{n,k}(t) \le 1 \quad \forall n \in \mathcal{N} \\
& && \sum_{n=1}^{N} M_{n,k}(t) \le 1 \quad \forall k \in \mathcal{K} \\
& && M_{n,k}(t) \in \{0,1\} \quad \forall n \in \mathcal{N}, \forall k \in \mathcal{K}.
\end{aligned}$$

Finding the solution of problem (1) is equivalent to finding a maximum weighted matching in the $N \times K$ bipartite graph $G_t = (\mathcal{N}, \mathcal{K}, \mathcal{E})$. In G_t, $\mathcal{N}$ and $\mathcal{K}$ are the two sets of vertices in each part of the graph and $\mathcal{E} = \{e_{n,k}, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}\}$ is the set of edges between these two parts. In G_t, the associated weight to each edge $e_{n,k}$ is $X_n(t-1)C_{n,k}(t)$. A matching in graph G_t is a sub-graph of G_t in which no two edges share a common vertex. Any matching $\boldsymbol{M}^{(\pi)}(t)$ at any time slot t is corresponding to a sub-graph of G_t namely $G_t^{(\pi)} = (\mathcal{N}, \mathcal{K}, \mathcal{E}^{(\pi)})$ in which $e_{n,k} \in \mathcal{E}^{(\pi)}$ if and only if $M_{n,k}^{(\pi)}(t) = 1$. Maximum weighted matching in bipartite graphs can be determined in polynomial time using the Hungarian algorithm whose complexity is $O((\min\{N,K\})(\max\{N,K\})^2)$ [16].

Assume that $\boldsymbol{M}^{(\text{MWM})}(t) = (M_{n,k}^{(\text{MWM})}(t))\ \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$ is the matching whose indicator variables are the solution of the optimization problem (1). $\boldsymbol{M}^{(\text{MWM})}(t)$ has the following properties:

(a) $\boldsymbol{M}^{(\text{MWM})}(t)$ always exists at all time slots.

(b) $\boldsymbol{M}^{(\text{MWM})}(t)$ may not be unique.

Definition 1: A MWM server assignment policy is defined as a policy that employs maximum weighted matching $\boldsymbol{M}^{(\text{MWM})}(t)$ at all time slots, i.e., $\pi^{(\text{MWM})} = \{\boldsymbol{M}^{(\text{MWM})}(t)\}_{t=1}^{\infty}$. By construction, the MWM policy is causal.

Definition 2: We denote the set of all policies that employ maximum weighted matching at all time slots by Π^{MWM}.

According to property *(a)* above, the set Π^{MWM} is not empty. Moreover, according to property *(b)*, we conclude that Π^{MWM} may contain an infinite number of policies.

IV. DELAY OPTIMALITY OF MWM POLICY

In this section, we prove the delay optimality of MWM policy. This result is formally presented in Theorem 2. More specifically, we show that in an MQMS-Type2 system with i.i.d. Bernoulli arrival and connectivity processes, any MWM policy is optimal in minimizing, *in stochastic ordering sense*, a class of cost functions of queue length processes including total queue occupancy. According to Little's law, minimization of total queue occupancy is equivalent to minimization of average queueing delay. For brevity we will use the term "*delay optimality*" to refer to the optimality of MWM in this sense.

A. *Equivalence of Queue Length Balancing and Maximum Weighted Matching*

We start this section by introducing the *intermediate* queue state in the following definition.

[2] The values of λ and p do not involve in our analysis. We only rely on the fact that the arrivals and connectivities are i.i.d. Bernoulli processes.

Definition 3: Let $\boldsymbol{X}'(t) = (X_1'(t), X_2'(t), \cdots, X_N'(t))$ denote the queue length vector at time slot t exactly *after serving the queues according to a server assignment policy* π *and before adding the new arrivals* of time slot t, i.e.,

$$X_n'(t) = \left(X_n(t-1) - \sum_{k=1}^{K} C_{n,k}(t) M_{n,k}^{(\pi)}(t) \right)^+ . \qquad (2)$$

We call this vector as a the *intermediate* queue state. Recall that the final state of queue n at time slot t is determined after adding the new arrivals.

Given $\boldsymbol{x}'(t)$ as a sample value of random vector $\boldsymbol{X}'(t)$, we define a *balancing server reallocation* at time slot t as follows.

Definition 4: Assume that the employed matching at time slot t (assignment of servers to the queues at time slot t) will result in the intermediate queue vector $\boldsymbol{x}'(t)$. A balancing server reallocation at this time slot is *a new matching* resulting in intermediate vector $\tilde{\boldsymbol{x}}'(t)$ such that one of the following conditions is satisfied.

(**C1**) $\tilde{x}_n'(t) \le x_n'(t)$ for all $n = 1, 2, \cdots, N$ and there exists an $m \in \{1, 2, \cdots, N\}$ such that $\tilde{x}_m'(t) < x_m'(t)$.

(**C2**) $\tilde{\boldsymbol{x}}'(t)$ and $\boldsymbol{x}'(t)$ are different in only two elements n and m such that $x_n'(t) < \tilde{x}_n'(t) \le \tilde{x}_m'(t) < x_m'(t)$ and the following constraints are satisfied: $\tilde{x}_n'(t) = x_n'(t) + 1$ and $\tilde{x}_m'(t) = x_m'(t) - 1$.

The balancing reallocation defined above balances the intermediate queue vector step-by-step. This step-by-step queue balancing is required in our stochastic ordering arguments later. It is worth mentioning that each balancing reallocation will result in a queue length which is more balanced according to the balancing definition provided in [4] which is based on lexicographic ordering.

Example: Consider a system with three queues and three servers. Assume that $\boldsymbol{x}(t-1) = (3, 2, 5)$ is the queue length vector right at the end of time slot $t-1$ (or at the beginning of time slot t). We consider two distinct examples to show the definition of balancing server reallocations corresponding to each of the cases **C1** and **C2** in Definition 4. Figs. 2(a) and 2(b) show these examples of balancing server reallocations. In each case, we also show the weight of each edge (n, k) which is equal to $c_{n,k}(t)x_n(t-1)$. In these figures, since none of the queues is empty, the edges with weight 0 are the ones which are disconnected. We have specified the original allocations by solid lines and the balancing ones by dashed lines. For the system in Fig. 2(a), the original allocation will result in the intermediate vector $\boldsymbol{x}'(t) = (3, 1, 4)$ while the balancing server reallocation will result in the intermediate vector $\tilde{\boldsymbol{x}}'(t) = (2, 1, 4)$. The vectors $\boldsymbol{x}'(t)$ and $\tilde{\boldsymbol{x}}'(t)$ satisfy Condition **C1**. For the system in Fig. 2(b), the original allocation will result in the intermediate vector $\boldsymbol{x}'(t) = (2, 1, 5)$ while the balancing server reallocation will result in $\tilde{\boldsymbol{x}}'(t) = (3, 1, 4)$. The vectors $\boldsymbol{x}'(t)$ and $\tilde{\boldsymbol{x}}'(t)$ satisfy Condition **C2**.

Intuitively, a more balanced system minimizes the server waste in future time slots. This is due to the fact that the probability of having an empty queue which is disconnected from the servers is more in an unbalanced system, i.e., a more balanced system will have higher average server utilization in time.

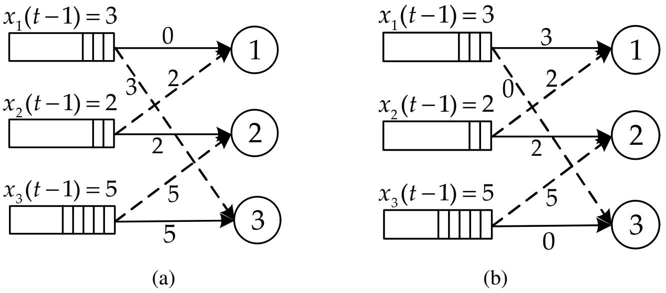

Fig. 2. Examples of balancing server reallocations (the weight $c_{n,k}(t)x_n(t-1)$ of each edge (n, k) is also shown): (a) Satisfying condition **C1** and (b) satisfying condition **C2**.

Definition 5: For a server assignment policy π with the allocation variables $\{M_{n,k}^{(\pi)}(t)\}_{t=1}^{\infty}$, $\forall k \in \mathcal{K}$ and $\forall n \in \mathcal{N}$, we define matching weight (MW) index at time slot t by

$$\mathsf{MW}_\pi(t) = \sum_{n=1}^{N} X_n(t-1) \sum_{k=1}^{K} C_{n,k}(t) M_{n,k}^{(\pi)}(t). \qquad (3)$$

MW index is exactly the objective of the optimization problem (1). $\mathsf{MW}_\pi(t)$ is an index associated with policy π at time slot t whose value is dependent on the state of the system (queue lengths and connectivities) as well as the matching employed by policy π at time slot t. In [4], the authors has shown that a maximum weight load balancing matching is nothing but a maximum weight matching on an equivalent bipartite graph of an MQMS-Type1 system. In other words, the MWM on the equivalent bipartite graph of MQMS-Type1 system is equivalent to balancing the queues. In our model, since the number of servers allocated per queue is limited to one, the equivalent bipartite graph is exactly the original graph. Thus the same result follows for MQMS-Type2 as well. Moreover, using similar proof as in Proposition 1 of [4], we can show that any balancing reallocation will result in an improved matching weight index and when the matching weight index is not maximized there exists a balancing server reallocation. This result is formally stated in this paper in the following corollary.

Corollary 1: For a given policy π employing matching $\boldsymbol{M}^{(\pi)}(t)$ at time slot t, by applying a balancing server reallocation at time slot t (if there exists any), we can create a new policy $\tilde{\pi}$ (differing from π only at time slot t) such that $\mathsf{MW}_\pi(t) < \mathsf{MW}_{\tilde{\pi}}(t)$. Moreover, for a given policy π at time slot t, if $\mathsf{MW}_\pi(t)$ is not maximized, i.e., if $\mathsf{MW}_\pi(t) < \mathsf{MW}_{\mathsf{MWM}}(t)$, then there exists a balancing server reallocation at that time slot.

B. Background on Stochastic Ordering and Dynamic Coupling

In this section, we briefly review the concepts of stochastic ordering (stochastic dominance) and dynamic coupling techniques. These concepts are needed in the proof of delay optimality of MWM policy in the rest of our discussion. The reader is encouraged to consult [17]–[19] for more details about stochastic ordering and dynamic coupling.

Definition 6: Consider two real-valued, discrete-time stochastic processes $A = \{A(t)\}_{t=1}^{\infty}$ and $B = \{B(t)\}_{t=1}^{\infty}$ in $\mathbb{R}$. We say A is stochastically smaller than B and we write $A \le_{st} B$ if

$\Pr(A(t) > r) \leq \Pr(B(t) > r)$ *for all* $t = 1, 2, \cdots$ and *all* $r \in \mathbb{R}$ [17], [18].

The following two properties of stochastic ordering are useful: If $A \leq_{st} B$, then

(a) $E[A(t)] \leq E[B(t)]$

(b) $f(A) \leq_{st} f(B)$ for all non-decreasing functions f.

Process A is stochastically smaller than B, if there exists a process $\tilde{A} = \{\tilde{A}(t)\}_{t=1}^{\infty}$ defined on the same probability space as B, has the same probability distribution as A and satisfies $\tilde{A}(t) \leq B(t)$ almost surely (a.s.) for every $t = 1, 2, \cdots$ [6]. The last statement is known as coupling of A and $\tilde{A}$. When applying coupling technique, given the process A, we construct a *coupled* process $\tilde{A}$ with the same distribution as A and $\tilde{A}(t) \leq B(t)$ a.s. *for all* t. This gives us a tool for comparing the processes A and B stochastically when it is infeasible to derive the distributions of A and B (e.g., in our queueing model when comparing the total occupancy process for different server assignment policies).

C. Delay Optimality of MWM

In this subsection, we will elaborate on proving the delay optimality of any MWM policy. We first introduce some definitions. We denote by $\mathbb{Z}_+$ the set of non-negative integers and by $\mathbb{Z}_+^{\mathbb{N}}$ the N dimensional Cartesian space of non-negative integers. We define the relation "$\preceq$" on $\mathbb{Z}_+^{\mathbb{N}}$ as follows.

Definition 7: For two vectors $\boldsymbol{x}$, $\tilde{\boldsymbol{x}} \in \mathbb{Z}_+^{\mathbb{N}}$, we write $\tilde{\boldsymbol{x}} \preceq \boldsymbol{x}$ if one of the following relations holds:

D1: $\tilde{x}_n \leq x_n$ for all $n = 1, 2, \cdots, N$.

D2: $\tilde{\boldsymbol{x}}$ is obtained by permutation of two distinct elements of $\boldsymbol{x}$, i.e., $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are different in only two elements n and m such that $\tilde{x}_n = x_m$ and $\tilde{x}_m = x_n$. In this case, we say $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are *equal in permutation* and we write $\tilde{\boldsymbol{x}} \overset{p}{=} \boldsymbol{x}$.

D3: $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are different in only two elements n and m such that $x_n < \tilde{x}_n \leq \tilde{x}_m < x_m$ and the following constraints are satisfied: $\tilde{x}_n = x_n + 1$ and $\tilde{x}_m = x_m - 1$.

The three relations **D1**, **D2** and **D3** are mutually exclusive. In **D3**, we say that $\tilde{\boldsymbol{x}}$ is more balanced than $\boldsymbol{x}$ and can be obtained by decreasing a larger element of $\boldsymbol{x}$ (i.e., m) by one and increasing a smaller element (i.e., n) by one. We call such an interchange as a *balancing interchange* on vector $\boldsymbol{x}$. Thus, the result of a balancing interchange on a vector $\boldsymbol{x}$ would be a vector $\tilde{\boldsymbol{x}}$ such that $\tilde{\boldsymbol{x}} \preceq \boldsymbol{x}$. According to Definition 4, a balancing server reallocation satisfying Condition **C2**, will result in a balancing interchange between $\boldsymbol{x}'(t)$ and $\tilde{\boldsymbol{x}}'(t)$.

We define the partial order "$\preceq_p$" on $\mathbb{Z}_+^{\mathbb{N}}$ as the transitive closure of relation" $\preceq$" [20]. In other words, $\tilde{\boldsymbol{x}} \preceq_p \boldsymbol{x}$ if and only if $\tilde{\boldsymbol{x}}$ is obtained from $\boldsymbol{x}$ by performing a sequence of reductions (i.e., reducing an element of the vector $\boldsymbol{x}$ such that $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ satisfy **D1**), permutations of two elements (permutation of two elements of the vector $\boldsymbol{x}$ such that $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ satisfy **D2**) and/or balancing interchanges (such that $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ satisfy **D3**). When $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are two queue length vectors, we write $\tilde{\boldsymbol{x}} \preceq_p \boldsymbol{x}$ if and only if queue length vector $\tilde{\boldsymbol{x}}$ is obtained from $\boldsymbol{x}$ by applying a sequence of packet removals, two-queue permutations and balancing interchanges.

Definition 8: We define $\mathcal{F}$ as the class of real-valued functions on $\mathbb{Z}_+^{\mathbb{N}}$ that are monotone and non-decreasing with respect to the partial order $\preceq_p$, i.e.,

$$f \in \mathcal{F} \iff \tilde{\boldsymbol{x}} \preceq_p \boldsymbol{x} \Rightarrow f(\tilde{\boldsymbol{x}}) \leq f(\boldsymbol{x}). \tag{4}$$

Lemma 1: Function $f(\boldsymbol{x}) = \sum_{n=1}^{N} x_n^r$ for any non-negative integer r belongs to $\mathcal{F}$. Consequently L^r-norm of queue length vector $\boldsymbol{x}$ belongs to $\mathcal{F}$.

Proof: Consider two queue length vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ such that $\tilde{\boldsymbol{x}} \preceq_p \boldsymbol{x}$. We can easily observe that if $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ satisfy **D1** or **D2**, $\sum_{n=1}^{N} \tilde{x}_n^r \leq \sum_{n=1}^{N} x_n^r$ and the result follows. If $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ satisfy **D3**, $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are different in only two elements n and m such that $x_n < \tilde{x}_n \leq \tilde{x}_m < x_m$ and the following constraints are satisfied: $\tilde{x}_n = x_n + 1$ and $\tilde{x}_m = x_m - 1$. Thus, $f(\boldsymbol{x}) = x_m^r + x_n^r + \sum_{i=1, i \neq n,m}^{N} x_i^r = (\tilde{x}_m + 1)^r + (\tilde{x}_n - 1)^r + \sum_{i=1, i \neq n,m}^{N} \tilde{x}_i^r$. Using Binomial expansion of $(\tilde{x}_m + 1)^r$ and $(\tilde{x}_n - 1)^r$, we have $f(\boldsymbol{x}) = \sum_{i=1}^{N} \tilde{x}_i^r + \sum_{k=1}^{r} \binom{r}{k} (\tilde{x}_m^{r-k} + \tilde{x}_n^{r-k}(-1)^k)$. Note that $\sum_{i=1}^{N} \tilde{x}_i^r = f(\tilde{\boldsymbol{x}})$. Since $\tilde{x}_n \leq \tilde{x}_m$, the term $\sum_{k=1}^{r} \binom{r}{k} (\tilde{x}_m^{r-k} + \tilde{x}_n^{r-k}(-1)^k) \geq 0$, and therefore $f(\boldsymbol{x}) \geq f(\tilde{\boldsymbol{x}})$ and the result follows. □

In Lemma 1, $r = 1$ results in function $f(\boldsymbol{x}) = \sum_{n=1}^{N} x_n$. This function represents the total queue occupancy of the system. According to Little's law ($E[D] = \frac{E[\sum X_n]}{N \times \lambda}$), minimization of average total queue occupancy ($E[\sum X_n]$) is equivalent to minimization of average queueing delay ($E[D]$).

Definition 9: We define $\Pi_t, t = 1, 2, \cdots$, as the set of all policies that employ maximum weighted matching in every time slot $\tau = 1, \cdots, t$.

We observe that $\Pi_{t-1} \supseteq \Pi_t$ and $\Pi^{\text{MWM}} = \bigcap_{t=1}^{\infty} \Pi_t$.

Consider a policy $\pi \in \Pi_{t-1}$ which is using an arbitrary matching $\boldsymbol{M}^{(\pi)}(t)$ at time slot t. If $\boldsymbol{M}^{(\pi)}(t)$ is not a maximum weighted matching, then from Corollary 1 we conclude that by applying a sequence of balancing server reallocations we can create a policy $\pi^\star \in \Pi_t$. Let h_t^π denote the number of balancing server reallocations required to convert the employed matching in policy π at time slot t to a maximum weighted matching.

Definition 10: We define the *distance* of policy $\pi \in \Pi_{t-1}$ from the set Π_t to be h_t^π balancing server reallocations.

According to Corollary 1, since by applying each server reallocation, the matching weight index *strictly* increases, the number of balancing server reallocations needed to convert π to a maximum weighted matching is bounded, i.e., $h_t^\pi \leq H < \infty$ for all t, π. Hence, after applying the first balancing server reallocation at time slot t we reach a policy $\tilde{\pi}_1$ whose distance from Π_t is $h_t^\pi - 1$ balancing server reallocations. By repeating this procedure we finally identify a policy whose distance to Π_t is zero, i.e., it belongs to Π_t.

Definition 11: By Π_t^h $(0 \leq h \leq H)$ we denote the set of all server assignment policies in Π_{t-1} whose distance from Π_t is h balancing server reallocations. Recall that $\Pi_t^0 = \Pi_t$.

Definition 12: For any two policies π and $\tilde{\pi}$ with queue length processes $\boldsymbol{X} = \{\boldsymbol{X}(t)\}_{t=1}^{\infty}$ and $\tilde{\boldsymbol{X}} = \{\tilde{\boldsymbol{X}}(t)\}_{t=1}^{\infty}$, respectively, we say $\tilde{\pi}$ *dominates* π, if $f(\tilde{\boldsymbol{X}}) \leq_{st} f(\boldsymbol{X})$, $f \in \mathcal{F}$, i.e., the queue length cost (delay) of policy $\tilde{\pi}$ is stochastically less than that of policy π.

If $\tilde{\pi}$ dominates π we have $E[f(\tilde{\boldsymbol{X}})] \leq E[f(\boldsymbol{X})]$. In the following lemma, we will interconnect the notions of "maximizing the matching weight index" and "delay optimality" and show that maximization of the matching weight index (at any given time t) will improve the delay performance (will decrease the queue length cost function $f(\boldsymbol{X})$ stochastically). The key element in the interconnection is the notion of balancing server reallocation. In particular, we show that, for any given policy $\pi \in \Pi_t^h$, $h = h_t^\pi$ that does not employ a maximum weighted matching at time slot t (i.e., $h > 0$), there exists a balancing server reallocation at time slot t. In the following lemma, we show that by using such a balancing server reallocation at time slot t we can construct a new policy $\tilde{\pi}$ that dominates the original policy π. For the detailed proof, please refer to Appendix A. We used stochastic ordering and dynamic coupling to prove this lemma.

Lemma 2: For any policy $\pi \in \Pi_t^h$ where $h = h_t^\pi > 0$, we can construct a policy $\tilde{\pi} \in \Pi_t^{h-1}$ such that $\tilde{\pi}$ dominates π. Thus, $\tilde{\pi}$ outperforms π in terms of average queueing delay.

Using Lemma 2, we can prove the following theorem which states that any MWM policy outperforms any non-MWM policy in terms of average queueing delay.

Theorem 1: For any server assignment policy $\pi \notin \Pi^{\text{MWM}}$, there exists an MWM policy $\pi^* \in \Pi^{\text{MWM}}$ such that π^* dominates π.

Proof: Let π be any arbitrary non-MWM policy. Then $\pi \in \Pi_1^{H_1}$ where $H_1 = h_1^\pi$. By applying Lemma 2 repeatedly, we can construct a sequence of policies such that each policy dominates the previous one. Thus, we obtain policies that belong to $\Pi_1^{H_1}, \Pi_1^{H_1-1}, \Pi_1^{H_1-2}, \cdots, \Pi_1^0 = \Pi_1$. The last policy is called π_1 for which we have $\pi_1 \in \Pi_2^{H_2}$ where $H_2 = h_2^{\pi_1}$. By continuing such an argument, we obtain a sequence of policies $\pi_t \in \Pi_t, t = 1, 2, \cdots$ such that π_j dominates π_i for $j > i$. This sequence of policies defines a limiting policy π^* that agrees with MWM at all time slots. Thus, π^* is an MWM policy that dominates all the previous policies, including the starting policy π. This proves that the delay-optimal policy is an MWM policy in Π^{MWM}. □

As we mentioned before, the set Π^{MWM} may contain an infinite number of policies. In the following, we show that *any* MWM policy is delay-optimal. More specifically, we show that the queue length costs of all the maximum weighted matching policies in Π^{MWM} are equal in distribution. To show this result we first show that the intermediate queue lengths resulted from any MWM matching are permutations of each other.

Lemma 3: Multiple distinct maximum weighted matchings result in intermediate queue length vectors which are *equal in permutation*, i.e., one is a permutation of the others.
The proof of the lemma is given in Appendix B.

Using this lemma in conjunction with the notion of dynamic coupling and stochastic ordering we can show that any two MWM policy dominate each other, i.e., for any $\pi_1, \pi_2 \in \Pi^{\text{MWM}}$, we have $f(\boldsymbol{X}^{(\pi_1)}) \leq_{st} f(\boldsymbol{X}^{(\pi_2)})$ and $f(\boldsymbol{X}^{(\pi_2)}) \leq_{st} f(\boldsymbol{X}^{(\pi_1)})$. Therefore, according to the definition of "$\leq_{st}$", we can conclude that $f(\boldsymbol{X}^{(\pi_1)})$ and $f(\boldsymbol{X}^{(\pi_2)})$ are equal in distribution, i.e., $f(\boldsymbol{X}^{(\pi_1)}) \overset{\mathcal{D}}{=} f(\boldsymbol{X}^{(\pi_2)})$. Details of the analysis are skipped here since it follows a similar argument we used for the proof of Lemma 2.

Using the result stated above and Theorem 1, we can conclude the main result of this section in the following theorem.

Theorem 2: Any MWM policy dominates any server assignment policy, i.e., any MWM policy is delay-optimal.

D. Discussion

The optimality of MWM in MQMS-Type2 system can be also concluded from the results in [4] presented for MQMS-Type1 system. More precisely, in MQMS-Type1 system where the number of servers per queue are unconstrained, [4] established the optimality of MTLB policies via a series of lemmas which establish a set of properties for the optimal value function inductively. These lemmas, except for Lemmas 4 and 5 in [4], hold in full generality for $N > 1$. Although the final result on delay optimality of MTLB policy was shown only for the case of two queues in MQMS-Type1 system, the optimality of MTLB holds as long as Observation 1 in [4] holds. Such a dependency of the results in [4] on the number of queues is due to the fact that for MQMS-Type1 system, Observation 1 only holds for the case of two symmetric queues (as stated in Remark 1). However, for the case of MQMS-Type2 system, Observation 1 in [4] holds for the general N and thus, Lemmas 4 and 5 in [4] would hold true for general N. Recall that MTLB and MWM policies are the same for the case of MQMS-Type2 system. In contrast with [4], in this paper we used dynamic coupling arguments to prove the optimality of MWM while the results in [4] rely on using dynamic programming formulation. Using this approach we are able to prove the optimality of the MWM policy for a vast range of cost functions of the queue lengths such as L^r-norm of the queue occupancy vector. The class of cost functions for which the MWM optimality is shown (defined in Def. 8) is determined based on the definition of partial orderings " $\preceq$ " and " $\preceq_p$ " in Def. 7.

Although the results in the previous section have been shown for a limited case with i.i.d. Bernoulli arrival and connectivity processes, we may conclude some standard extensions as previously appeared in the literature (that used similar analytical approach). In particular, we can easily conclude the optimality of MWM for MQMS-Type2 systems with imperfect services where the service of a scheduled packet fails randomly with a certain probability. Moreover, the results can be extended for arrival and connectivity processes with permutation invariant distribution. Similar extensions has been provided in [6] for the case of a multi-queue system with a single server bank. For the completeness of the results we also provide such extensions in the following sections.

D.1 Imperfect Services

We can extend Theorems 1 and 2 for the case where the service of a scheduled packet by a connected server fails randomly with a certain probability. This can model the operation of realistic wireless networks where service failures usually occur due to unexpected and unpredictable effects of noise, interference, etc. In the case of a packet service failure, the packet will be kept in the queue and will be rescheduled and retransmitted in future time slots.

By the random variable $Q_{n,k}(t) \in \{0, 1\}$, we denote the successful/unsuccessful service of queue n provided by server k at time slot t; a value of 1 (resp. 0) denotes that the service is successful (resp. unsuccessful). We assume that $Q_{n,k}(t)$, $\forall n \in \mathcal{N}, \forall k \in \mathcal{K}$ are i.i.d. Bernoulli random variables with the same success probability q. The parameter q (similar to parameters λ and p) is not explicitly involved in our analysis other than the fact that $E[Q_{n,k}(t)] = q, \forall n, k, t$. The queue lengths are then updated at the end of each time slot by the following rule.

$$X_n(t) = \left(X_n(t-1) - \sum_{k=1}^{K} C_{n,k}(t) M_{n,k}^{(\pi)}(t) Q_{n,k}(t) \right)^+ + A_n(t) \quad \forall n \in \mathcal{N} \qquad (5)$$

The network scheduler (that performs server assignment process) cannot observe the variables $Q_{n,k}(t)$ and from its perspective they are assumed to be random. The random vector $\boldsymbol{X}'(t)$ is defined similar to (2). Hence, $\boldsymbol{X}'(t)$ represents the queue lengths before adding the new arrivals of time slot t as if all the services at that time slot are successful.

For such a system, we can extend Lemma 2 for the system with service failures by considering the random variables $Q_{n,k}(t)$ in our dynamic coupling arguments. The proof is followed by using the same approach as in Lemma 2 and is omitted here due to space limitations. By applying the same approach as in the proof of Theorem 1, Lemma 3 and Theorem 2, we can similarly prove the delay optimality of MWM policy for the system with imperfect services.

D.2 Extensions for Connectivity and Arrival Processes

The arguments in Lemmas 2 and Theorem 1 remain valid if the i.i.d. assumption for connectivity and arrival processes is relaxed as follows; we will consider connectivity and arrival processes which follow conditional permutation invariant distributions. Given event $\mathcal{H}$ (which is used to denote the history of the system), we define a conditional multivariate probability distribution $f(y_1, y_2, \cdots, y_n \mid \mathcal{H})$ to be permutation invariant if for any permutation of the variables $y_1, y_2, \cdots, y_n$ namely $y_1', y_2', \cdots, y_n'$ we have $f(y_1, y_2, \cdots, y_n \mid \mathcal{H}) = f(y_1', y_2', \cdots, y_n' \mid \mathcal{H})$. We can readily see that for all the connectivity and arrival processes whose joint distributions at each time slot given the history of the system[3] (i.e., $f_{\boldsymbol{A}(t)}(a_1, a_2, \cdots, a_N \mid \mathcal{H})$ and $f_{\boldsymbol{C}(t)}(c_{1,1}, c_{1,2}, \cdots, c_{N,K-1}, c_{N,K} \mid \mathcal{H})$) are permutation invariant, Lemma 2 and Theorem 1 are still valid and therefore MWM is delay-optimal.

We also consider the generalization of Theorems 1 and 2 for non-Bernoulli arrival processes. Suppose that the number of arrivals to each queue can be represented by the summation of some i.i.d. Bernoulli random variables, i.e., has Binomial distribution. Also suppose that $A_n(t) \leq A_{\max}$ *for all* $n \in \mathcal{N}$ and *all* t. In this case, we can create a new (virtual) system in which after each time slot we append $A_{\max} - 1$ virtual time slots and put the connectivities all equal to zero, i.e., for each virtual time slot t, $C_{n,k}(t) = 0, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$. We then distribute the arrivals of the actual time slot among these $A_{\max}$ time slots (one actual time slot and $A_{\max} - 1$ virtual time slots) randomly such that at each time slot at most one packet arrival occurs. Since the connectivities and the arrivals in both systems are *permutation invariant*, we can still prove Theorems 1 and 2 for the virtual system. We observe that the operation of the two systems (the original system and the virtual system) are the same. Therefore, we can conclude that Theorem 1 is also valid for a multi-server system with Binomial arrival processes. As the Poisson process is approximated by a sequence of Binomial distributions, same arguments can be used to show the validity of the results for Poisson arrival process.

V. SIMULATION RESULTS

In this section, we compare the delay performance of the optimal policies for MQMS-Type1 and MQMS-Type2 with two alternative server assignment policies; a random policy and a heuristic policy called LCSF/LCQ. For each system, we will observe that LCSF/LCQ policy performs very closely to the optimal policy and therefore it can be used as a good approximation for the optimal policy for practical implementation in wireless systems. In the following, we will introduce the random and LCSF/LCQ policies for both MQMS-Type1 and MQMS-Type2 systems:

- *Random policy*: Each server is randomly allocated to one of the queues which is connected to the server. In MQMS-Type2 system, both the server and the queue to which the server is allocated are removed from the list of the servers and queues. This process is repeated for all the servers.
- *LCSF/LCQ policy*: The servers are sorted based on the number of connectivities incident to them. We start from the server with the minimum number of connectivities (least connected server first). We allocate the server to its longest connected queue. The queue length is updated (i.e., decremented). In MQMS-Type2, the server and the queue to which the server was allocated will be removed from the list of queues and servers. We proceed accordingly to the next least connected server until all servers are assigned.

Recall that LCQ algorithm was proposed for server allocation in single server systems in [2], [6]. In single-server systems, we only deal with one resource/server and the problem is how to allocate that single server to the competing queues. It is proven in [2] that if we assign the server to the longest connected queue, the average queue length is minimized. In MQMS systems we are dealing with a set of servers such that each server has an independent random connectivity to each queue. In these systems, LCQ cannot be applied alone since the delay performance of the system would be different for different orders of servers to choose and then applying LCQ on them. LCSF/LCQ server allocation algorithm is composed of two phases; LCSF and LCQ. In LCSF phase we try to determine a proper order of servers to choose for applying the second phase, i.e., LCQ. Therefore, in LCSF phase, it chooses the least connected server first and then applies an LCQ on that server. It then updates the queue lengths and continues with the second least connected server and applies LCQ on that. This process continues until all the servers are assigned. In LCSF/LCQ policies the idea is that by serving the servers with the least number of connectivities we are trying to maximize the chance to use those servers. In fact, least connected servers are considered scarce resources and utilization of them in the system will increase the total service utilization in the system. If we differ using those servers, the probability of missing the service of those servers increases since the queues

[3] By history of the system we mean all the channel states, arrivals and matchings of the previous time slots up to time slot t.

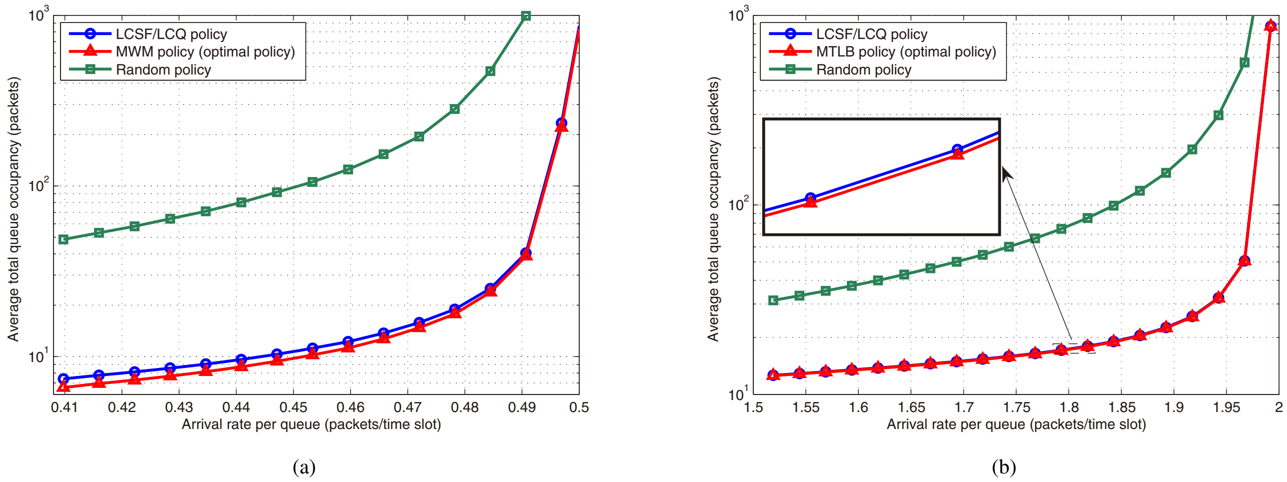

Fig. 3. Performance comparison of the optimal policy with the heuristic policy (LCSF/LCQ) and the random policy: (a) Delay performance of MWM, LCSF/LCQ and Random policies and (b) delay performance of MTLB, LCSF/LCQ and random policies

to which the server is connected may become unavailable (become empty or be removed due to the service of another server). In summary, LCSF phase of the algorithm tries to maximize the server utilization while LCQ phase tries to balance the queue lengths and minimize the probability of service waste in future time slots.

Note that LCSF/LCQ does not provide the most balance state for the intermediate queue state and thus it cannot result in the optimal policy. To show this fact, we provide a counter example. Consider an MQMS-Type1 system with $N = 4$ and $K = 7$ with the following configuration at time slot t: The queue state at time slot t is $x(t) = (5, 5, 5, 4)$. 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 the intermediate queue length $x'(t) = (2, 3, 3, 4)$ while MTLB will result in $x'(t) = (3, 3, 3, 3)$ which is a more balanced vector. Although LCSF/LCQ is not optimal, as we will see in the simulations it performs very closely to the MTLB policy in terms of average queueing delay.

We have evaluated the performance of the above mentioned policies and compared with the optimal policy in terms of average total queue occupancy. For MQMS-Type2 system we considered a system with $N = 8$, $K = 4$ and $p = 0.5$. For the MQMS-Type1 we considered a system with $N = 8$, $K = 16$ and $p = 0.5$ where p is the probability of connectivity. The arrival processes into the queues are Poisson processes. The results for Bernoulli arrival process were very comparable with the Poisson process. Therefore, we only present the result for Poisson arrival which is a more general and well accepted process for packet arrivals. Figs. 3a and 3b show the results. We observe that the optimal policy outperforms the LCSF/LCQ and the random policy for both MQMS-Type1 and MQMS-Type2. We can also see that the LCSF/LCQ performs very closely to the optimal policy in both systems. For the case of MQMS-Type1 system, the difference is indiscernible (maximum difference was measured 0.2).

The random policy is the worst performer as expected since both the optimal and LCSF/LCQ policies try to balance the load and maximize the service utilization while the random policy is randomly allocating the servers to the connected queues.

The LCSF/LCQ policy is of particular interest for the following reasons: (a) It follows a particular server allocation ordering (LCSF) to their LCQ and thus it can be implemented using a sequential server allocation with low computation complexity of $O(K(N + \log K))$[4], (b) the selected server ordering (LCSF) and allocation (LCQ) intuitively attempt to reduce the size of the longest connected queue thus reducing the imbalance among queues, Therefore, LCSF/LCQ can be proposed as an approximate heuristic for the implementation of the optimal policy.

VI. CONCLUSIONS

In this paper, we provided some complementary results on delay-optimality of MWM policy for MQMS-Type2 system in which the queues are restricted to get service from at most one server per time slot. We used dynamic coupling arguments to show the optimality of MWM for a class of cost functions of the queue length vector. More specifically, we showed that MWM policy is delay optimal for a symmetric MQMS-Type2 system with permutation invariant connectivity and arrival processes. Furthermore, we introduced a low complexity heuristic server assignment policy for both MQMS-Type1 and MQMS-Type2 systems and through simulations we showed how it performs closely compared with the optimal policy.

APPENDIX A
PROOF OF LEMMA 2

Proof: Fix any arbitrary policy $\pi \in \Pi_t^h$ where $h = h_t^\pi > 0$, and any arbitrary sample path $\boldsymbol{\omega} = (\boldsymbol{x}(0), \boldsymbol{c}(1), \boldsymbol{a}(1), \boldsymbol{x}(1), \boldsymbol{c}(2), \boldsymbol{a}(2), \boldsymbol{x}(2), \cdots)$ of the underlying random variables $(\boldsymbol{X}(0), \boldsymbol{C}(1), \boldsymbol{A}(1), \boldsymbol{X}(1), \boldsymbol{C}(2), \boldsymbol{A}(2), \boldsymbol{X}(2), \cdots)$. We apply the coupling method to construct from $\boldsymbol{\omega}$ a new sample path $\tilde{\boldsymbol{\omega}} = (\tilde{\boldsymbol{x}}(0), \tilde{\boldsymbol{c}}(1), \tilde{\boldsymbol{a}}(1), \tilde{\boldsymbol{x}}(1), \tilde{\boldsymbol{c}}(2), \tilde{\boldsymbol{a}}(2), \tilde{\boldsymbol{x}}(2), \cdots)$ resulting in a new sequence of random variables $(\tilde{\boldsymbol{X}}(0), \tilde{\boldsymbol{C}}(1), \tilde{\boldsymbol{A}}(1), \tilde{\boldsymbol{X}}(1), \tilde{\boldsymbol{C}}(2), \tilde{\boldsymbol{A}}(2), \tilde{\boldsymbol{X}}(2), \cdots)$

[4] Compare with the MTLB complexity which is $O(K^2(N + \log K))$ [4].

with $\boldsymbol{X}(0) = \tilde{\boldsymbol{X}}(0)$. Recall that $\boldsymbol{X}(0)$ is the queue length vector in which the system starts. We denote the policy defined on the new sample path $\tilde{\boldsymbol{\omega}}$ by $\tilde{\pi}$. In fact, we construct $\tilde{\boldsymbol{\omega}}$ and $\tilde{\pi} \in \Pi_t^{h-1}$ in such a fashion that*for all* the sample paths we have $\tilde{\boldsymbol{x}}(t') \preceq_p \boldsymbol{x}(t')$ for all $t' = 1, 2, \cdots$. Therefore, for any $f \in \mathcal{F}$ we have $f(\tilde{\boldsymbol{x}}(t')) \le f(\boldsymbol{x}(t'))$ for all t'. As it will be shown, the processes $\{(\boldsymbol{C}(t'), \boldsymbol{A}(t'))\}_{t'=1}^{\infty}$ and $\{(\tilde{\boldsymbol{C}}(t'), \tilde{\boldsymbol{A}}(t'))\}_{t'=1}^{\infty}$ are the same in distribution (these processes are permutation invariant). Thus, the process $f(\tilde{\boldsymbol{X}}) = \{f(\tilde{\boldsymbol{X}}(t'))\}_{t'=1}^{\infty}$ obtained by applying policy $\tilde{\pi}$ to the system is stochastically smaller than $f(\boldsymbol{X}) = \{f(\boldsymbol{X}(t'))\}_{t'=1}^{\infty}$, i.e., $f(\tilde{\boldsymbol{X}}) \le_{st} f(\boldsymbol{X})$) and $\tilde{\pi}$ dominates π.

Therefore, in the following, our goal will be to construct $\tilde{\pi}$ and $\tilde{\boldsymbol{\omega}}$ such that $\tilde{\boldsymbol{x}}(t') \preceq_p \boldsymbol{x}(t')$ for all time slots. In the proof, we always use the tilde notation for all random variables that belong to the new system. The construction of $\tilde{\pi}$ is done in two steps:

Step 1: Construction of $\tilde{\pi}$ for $\tau \le t$: To construct the new sample path $\tilde{\boldsymbol{\omega}}$ we let the arrival, connectivity and the policy be the same as the first system until time slot $t-1$, i.e., $\tilde{\boldsymbol{c}}(\tau) = \boldsymbol{c}(\tau)$, $\tilde{\boldsymbol{a}}(\tau) = \boldsymbol{a}(\tau)$ and $\boldsymbol{M}^{(\pi)}(\tau) = \boldsymbol{M}^{(\tilde{\pi})}(\tau)$ for $\tau \le t-1$. Thus, the resulting queue lengths at the beginning of time slot t (or at the end of time slot $t-1$) are equal, i.e., $\tilde{\boldsymbol{x}}(t-1) = \boldsymbol{x}(t-1)$.

We now consider the construction of $\tilde{\boldsymbol{\omega}}$ and $\tilde{\pi}$ for time slot t. Since $\pi \in \Pi_t^h$ and $h > 0$, according to Corollary 1 there exists a balancing server reallocation such that either **C1** or **C2** is satisfied. Thus, we consider two cases:

Case 1: After applying the balancing server reallocation, condition **C1** is satisfied. In other words, there exists a matching such that if applied on the queue length $\tilde{\boldsymbol{x}}(t-1) = \boldsymbol{x}(t-1)$ at time slot t, we get $\tilde{\boldsymbol{x}}'(t)$ such that $\tilde{\boldsymbol{x}}'(t) \le \boldsymbol{x}'(t)$. We denote such a matching by $\boldsymbol{M}^{(\tilde{\pi})}(t)$. In this case, we let $\tilde{\boldsymbol{c}}(t) = \boldsymbol{c}(t)$ and $\tilde{\boldsymbol{a}}(t) = \boldsymbol{a}(t)$ and we apply $\boldsymbol{M}^{(\tilde{\pi})}(t)$ at time slot t, i.e., arrivals and connectivities are the same in both systems and policy $\tilde{\pi}$ acts at time slot t. So, we can easily check that $\tilde{\boldsymbol{x}}(t) \le \boldsymbol{x}(t)$ and therefore $\tilde{\boldsymbol{x}}(t) \preceq \boldsymbol{x}(t)$.

Case 2: After applying the balancing server reallocation, condition **C2** is satisfied. In other words, there exists a matching such that if applied on the system at time slot t, we get $\tilde{\boldsymbol{x}}'(t)$ which is different from $\boldsymbol{x}'(t)$ in two elements m and n such that $x'_n(t) < \tilde{x}'_n(t) \le \tilde{x}'_m(t) < x'_m(t)$ and the following constraints are satisfied: $\tilde{x}'_n(t) = x'_n(t)+1$ and $\tilde{x}'_m(t) = x'_m(t)-1$. We call such a matching by $\boldsymbol{M}^{(\tilde{\pi})}(t)$. In this case, we let $\tilde{\boldsymbol{c}}(t) = \boldsymbol{c}(t)$ and $\tilde{\boldsymbol{a}}(t) = \boldsymbol{a}(t)$ and we apply $\boldsymbol{M}^{(\tilde{\pi})}(t)$ at time slot t, i.e., arrivals and connectivities are the same in both systems and policy $\tilde{\pi}$ acts at time slot t. We consider all the following conditions for arrivals to queues m and n as follows:

- If there is no arrival or there is an arrival to both queues m and n (i.e., $a_m(t) = a_n(t) = 0$ or $a_m(t) = a_n(t) = 1$), we conclude that $\tilde{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t)$ satisfy condition **D3**. Thus, $\tilde{\boldsymbol{x}}(t) \preceq \boldsymbol{x}(t)$.
- If there is an arrival to queue m but not n (i.e., $a_m(t) = 1$, $a_n(t) = 0$), we conclude that $\tilde{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t)$ satisfy condition **D3**. Thus, $\tilde{\boldsymbol{x}}(t) \preceq \boldsymbol{x}(t)$.
- If there is an arrival to queue n but not m (i.e., $a_m(t) = 0$, $a_n(t) = 1$) and $\tilde{x}_m(t) = \tilde{x}_n(t)$, we conclude that $\tilde{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t)$ satisfy condition **D2**. Thus, $\tilde{\boldsymbol{x}}(t) \preceq \boldsymbol{x}(t)$.
- If there is an arrival to queue n but not m (i.e., $a_m(t) = 0$, $a_n(t) = 1$) and $\tilde{x}_n(t) < \tilde{x}_m(t)$, we conclude that $\tilde{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t)$ satisfy condition **D3**. Thus, $\tilde{\boldsymbol{x}}(t) \preceq \boldsymbol{x}(t)$.

In all the cases we can see that $\tilde{\boldsymbol{x}}(t) \preceq \boldsymbol{x}(t)$. The obtained policy $\tilde{\pi}$ belongs to Π_t^{h-1} since we applied a balancing server reallocation to the matching employed in π at time slot t.

Step 2: Construction of $\tilde{\pi}$ for $\tau > t$: In this step, we focus on construction of $\tilde{\boldsymbol{\omega}}$ and $\tilde{\pi}$ for $\tau > t$. We will employ mathematical induction to achieve this goal. In particular, we assume that $\tilde{\boldsymbol{\omega}}$ and $\tilde{\pi}$ are constructed up to time slot τ ($\tau \ge t$) such that $\tilde{\boldsymbol{x}}(\tau) \preceq \boldsymbol{x}(\tau)$ i.e., one of the conditions **D1**, **D2** and **D3** is satisfied for $\boldsymbol{x}(\tau)$ and $\tilde{\boldsymbol{x}}(\tau)$. We will prove that policy $\tilde{\pi}$ and sample path $\tilde{\boldsymbol{\omega}}$ can be constructed such that $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$ (i.e., one of the conditions **D1**, **D2** and **D3** is satisfied for $\boldsymbol{x}(\tau+1)$ and $\tilde{\boldsymbol{x}}(\tau+1)$). Accordingly, we consider three cases corresponding to each condition **D1**, **D2** or **D3** at time slot τ:

Case 1: $\tilde{\boldsymbol{x}}(\tau) \le \boldsymbol{x}(\tau)$. In this case, the construction of $\tilde{\boldsymbol{\omega}}$ and $\tilde{\pi}$ at time slot $\tau+1$ is straightforward. We let $\tilde{\boldsymbol{c}}(\tau+1) = \boldsymbol{c}(\tau+1)$ and $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$ and $\boldsymbol{M}^{(\tilde{\pi})}(\tau+1) = \boldsymbol{M}^{(\pi)}(\tau+1)$. Thus, $\tilde{\boldsymbol{x}}(\tau+1) \le \boldsymbol{x}(\tau+1)$ and therefore $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.

Case 2: $\tilde{\boldsymbol{x}}(\tau)$ is obtained from $\boldsymbol{x}(\tau)$ by permutation of two distinct elements m and n. In this case, we let $\tilde{c}_{n,k}(\tau+1) = c_{m,k}(\tau+1)$ and $\tilde{c}_{m,k}(\tau+1) = c_{n,k}(\tau+1)$ for $k = 1, 2, \cdots, K$; $\tilde{c}_{i,k}(\tau+1) = c_{i,k}(\tau+1)$ for all $i \in \mathcal{N}$, $i \ne n, m$ and $k = 1, 2, \cdots, K$; $\tilde{a}_n(\tau+1) = a_m(\tau+1)$, $\tilde{a}_m(\tau+1) = a_n(\tau+1)$ and $\tilde{a}_i(\tau+1) = a_i(\tau+1)$ for $i \in \mathcal{N}$, $i \ne n, m$. Suppose that $\boldsymbol{M}^{(\pi)}(\tau+1) = (M_{n,k}^{(\pi)}(\tau+1))\ \forall n \in \mathcal{N}, k \in \mathcal{K}$ be the employed matching by policy π at time slot $\tau+1$. We construct $\boldsymbol{M}^{(\tilde{\pi})}(\tau+1)$ as follows: Let $M_{i,k}^{(\tilde{\pi})}(\tau+1) = M_{i,k}^{(\pi)}(\tau+1)$ for $i \in \mathcal{N}$, $i \ne n, m$ and also let $M_{n,k}^{(\tilde{\pi})}(\tau+1) = M_{m,k}^{(\pi)}(\tau+1)$ and $M_{m,k}^{(\tilde{\pi})}(\tau+1) = M_{n,k}^{(\pi)}(\tau+1)$ for $k = 1, 2, \cdots, K$. As a result, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D2** at time slot $\tau+1$ and therefore $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.

Case 3: $\tilde{\boldsymbol{x}}(\tau)$ is obtained from $\boldsymbol{x}(\tau)$ by performing a balancing interchange of two distinct elements m and n as defined in condition **D3**. In particular, $\tilde{\boldsymbol{x}}(\tau)$ and $\boldsymbol{x}(\tau)$ are different in only two elements n and m such that $x_n(\tau) < \tilde{x}_n(\tau) \le \tilde{x}_m(\tau) < x_m(\tau)$ and the following constraints are satisfied: $\tilde{x}_n(\tau) = x_n(\tau)+1$ and $\tilde{x}_m(\tau) = x_m(\tau)-1$. In this case, we consider the following sub-cases:

Sub-case 3.1: $\tilde{x}_n(\tau) < \tilde{x}_m(\tau) - 1$: In this case, we let $\tilde{\boldsymbol{c}}(\tau+1) = \boldsymbol{c}(\tau+1)$ and $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$ and we let $\boldsymbol{M}^{(\tilde{\pi})}(\tau+1) = \boldsymbol{M}^{(\pi)}(\tau+1)$. Thus, if $x_n(\tau) = 0$ and queue n is serviced, condition **D1** is satisfied at $\tau+1$. Otherwise, $\tilde{\boldsymbol{x}}(\tau+1)$ is obtained from $\boldsymbol{x}(\tau+1)$ by performing a balancing interchange of elements m and n. Therefore $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.

Sub-case 3.2: $\tilde{x}_n(\tau) = \tilde{x}_m(\tau) - 1$: In this case again we let $\tilde{\boldsymbol{c}}(\tau+1) = \boldsymbol{c}(\tau+1)$ and $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$ and we let $\boldsymbol{M}^{(\tilde{\pi})}(\tau+1) = \boldsymbol{M}^{(\pi)}(\tau+1)$. Thus, if $x_n(\tau) = 0$ and queue n is serviced, condition **D1** is satisfied at $\tau+1$. If queue m gets service, queue n does not get service, there is an arrival to queue n and no arrival to queue m, then $\tilde{x}_n(\tau+1) = x_m(\tau+1)$ and $\tilde{x}_m(\tau+1) = x_n(\tau+1)$. Therefore, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D2** and $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$. Otherwise, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D3** and $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.

Sub-case 3.3: $\tilde{x}_n(\tau) = \tilde{x}_m(\tau)$: In this case, we let $\tilde{\boldsymbol{c}}(\tau+1) = \boldsymbol{c}(\tau+1)$ and $\boldsymbol{M}^{(\tilde{\pi})}(\tau+1) = \boldsymbol{M}^{(\pi)}(\tau+1)$. Now, we consider the following cases to determine the arrivals at time slot $\tau + 1$.

- If $x_n(\tau) > 0$ and both queues m and n or none of them get service at time slot $\tau+1$, we let $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$. Therefore, if $a_m(\tau+1) = 0$ and $a_n(\tau+1) = 1$, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D2** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$. Otherwise, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D3** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.
- If $x_n(\tau) > 0$ and queue n gets service at time slot $\tau+1$ and queue m does not get service at time slot $\tau+1$, we let $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$. Therefore, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D3** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.
- If $x_n(\tau) > 0$ and queue m gets service at time slot $\tau+1$ and queue n does not get service at time slot $\tau+1$, we let $\tilde{a}_m(\tau+1) = a_n(\tau+1)$ and $\tilde{a}_n(\tau+1) = a_m(\tau+1)$ and $\tilde{a}_i(\tau+1) = a_i(\tau+1)$ for $i \in \mathcal{N}$ and $i \neq m, n$. Therefore, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D2** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.
- If $x_n(\tau) = 0$ and queue n gets service at time slot $\tau+1$ (although it does not have any packet to be served), we let $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$. Therefore, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D1** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.
- If $x_n(\tau) = 0$ and queue m gets service at time slot $\tau+1$ and queue n does not get service at time slot $\tau+1$, we let $\tilde{a}_m(\tau+1) = a_n(\tau+1)$ and $\tilde{a}_n(\tau+1) = a_m(\tau+1)$ and $\tilde{a}_i(\tau+1) = a_i(\tau+1)$ for $i \in \mathcal{N}$ and $i \neq m, n$. Therefore, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D2** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.
- If $x_n(\tau) = 0$ and neither queue m nor n gets service at time slot $\tau+1$, we let $\tilde{\boldsymbol{a}}(\tau+1) = \boldsymbol{a}(\tau+1)$. If $a_m(\tau+1) = 0$ and $a_n(\tau+1) = 1$, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D2** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$. Otherwise, $\tilde{\boldsymbol{x}}(\tau+1)$ and $\boldsymbol{x}(\tau+1)$ satisfy condition **D3** and thus $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.

The above cases cover all the possible cases for all of which we constructed $\tilde{\boldsymbol{\omega}}$ and $\tilde{\pi}$ such that $\tilde{\boldsymbol{x}}(\tau+1) \preceq \boldsymbol{x}(\tau+1)$.

According to steps 1 and 2, from any sample path $\boldsymbol{\omega}$ and any arbitrary policy $\pi \in \Pi_t^h$, $h = h_t^\pi > 0$, we can construct a sample path $\tilde{\boldsymbol{\omega}}$ and a policy $\tilde{\pi} \in \Pi_t^{h-1}$ such that at all time slots we have $\tilde{\boldsymbol{x}}(t') \preceq_p \boldsymbol{x}(t')$. Therefore, $f(\tilde{\boldsymbol{x}}(t')) \leq f(\boldsymbol{x}(t'))$. Consequently, the process $f(\tilde{\boldsymbol{X}}) = \{f(\tilde{\boldsymbol{X}}(t'))\}_{t'=1}^{\infty}$ obtained by applying policy $\tilde{\pi}$ to the system is stochastically smaller than $f(\boldsymbol{X}) = \{f(\boldsymbol{X}(t'))\}_{t'=1}^{\infty}$, i.e., $f(\tilde{\boldsymbol{X}}) \leq_{st} f(\boldsymbol{X})$ and therefore $\tilde{\pi} \in \Pi_t^{h-1}$ dominates $\pi \in \Pi_t^h$. □

APPENDIX B
PROOF OF LEMMA 3

Proof: We need to show that for any two maximum weighted matchings $\boldsymbol{M}^{(\text{MWM1})}(t)$ and $\boldsymbol{M}^{(\text{MWM2})}(t)$, if queue n is being served under $\boldsymbol{M}^{(\text{MWM1})}(t)$ but not under $\boldsymbol{M}^{(\text{MWM2})}(t)$, then there exists a queue m with $x_n(t) = x_m(t)$ which is being served under $\boldsymbol{M}^{(\text{MWM2})}(t)$ but not under $\boldsymbol{M}^{(\text{MWM1})}(t)$. We define a perfect graph G_t' of size $\max\{N, K\}$ over which we define sub-graphs $G_t'^{(\text{MWM1})}$ and $G_t'^{(\text{MWM2})}$ corresponding to $\boldsymbol{M}^{(\text{MWM1})}(t)$ and $\boldsymbol{M}^{(\text{MWM2})}(t)$. We build two directed sub-graphs D_t^{MWM1} and D_t^{MWM2} using sub-graphs $G_t'^{(\text{MWM1})}$ and $G_t'^{(\text{MWM2})}$ as follows: D_t^{MWM1} is the same as $G_t'^{(\text{MWM1})}$ with *positive* edges directed from queues to the servers. D_t^{MWM2} is the same as $G_t'^{(\text{MWM2})}$ with *negative* edges directed from servers to the queues. Similarly we define graph U as the union of these two sub-graphs, i.e., $U = D_t^{(\text{MWM1})} \bigcup D_t^{(\text{MWM2})}$.

Graph U is the union of a number of even cycles. Assume that queue n belongs to cycle ℓ (with W nodes) in graph U. Each node n_i in cycle ℓ is incident to one edge from D_t^{MWM1} and one edge from D_t^{MWM2} and therefore we can associate a pair $r_{n_i} = (e_{k_1,n_i}, e_{n_i,k_2})$ to it, where e_{k_1,n_i} is a directed edge from server k_1 to node n_i (with negative weight) and e_{n_i,k_2} is a directed edge from node n_i to server k_2 (with positive weight). The weights associated to the edges of each pair r_{n_i}, $1 \leq i \leq W$ are either $(0,0)$, $(0, x_{n_i}(t-1))$,$(-x_{n_i}(t-1), 0)$ or $(-x_{n_i}(t-1), x_{n_i}(t-1))$. Accordingly, we will specify three types of edge pairs as follows:

- **Type 0 (T0)**: pairs with edge weights $(0,0)$ or $(-x_{n_i}(t-1), x_{n_i}(t-1))$, $1 \leq i \leq W$.
- **Type 1 (T1)**: pairs with edge weights $(-x_{n_i}(t-1), 0)$, $1 \leq i \leq W$ and $x_{n_i}(t-1) > 0$.
- **Type 2 (T2)**: pairs with edge weights $(0, x_{n_i}(t-1))$, $1 \leq i \leq W$ and $x_{n_i}(t-1) > 0$.

If queue n is being served under $\boldsymbol{M}^{(\text{MWM1})}(t)$ but not under $\boldsymbol{M}^{(\text{MWM2})}(t)$, then the edges incident to queue n make a **T2** pair $r_n = (0, x_n(t-1))$. We now trace forward over cycle ℓ. The edge pairs after queue n cannot be all **T0** pairs, since by using the allocations of MWM1 in MWM2 for the queues in ℓ, we can increase the matching weight index of MWM2 which is a contradiction. Thus, we consider the following two cases:

Case 1: Assume that the first non-**T0** pair after r_n is a **T2** pair denoted by r_m. By using the allocations used in MWM1 for queues n to the one right before queue m in cycle ℓ in MWM2 and not serving queue m, we will obtain a matching whose matching weight index is larger than that of MWM2 which makes a contradiction.

Case 2: Assume that the first non-**T0** pair after r_n is a **T1** pair denoted by r_m. If $x_m(t-1) > x_n(t-1)$, by using the allocations used in MWM2 for the queues right after n in the cycle ℓ to queue m in MWM1 and not serving queue n, we obtain a matching whose matching weight index is larger than that of MWM1 which makes a contradiction. If $x_m(t-1) < x_n(t-1)$, by using the allocations used in MWM1 for the queue n to the queue right before queue m in cycle ℓ in MWM2 and not serving queue m, we will obtain a matching whose matching weight index is larger than that of MWM2. This is a contradiction too. Thus, the only valid case is $x_m(t-1) = x_n(t-1)$. □

REFERENCES

[1] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Trans. Auto. Control*, vol. 37, no. 12, pp. 1936–1949, Dec. 1992.

[2] L. Tassiulas and A. Ephremides, “Dynamic server allocation to parallel queues with randomly varying connectivity,” *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.

[3] K. Kar, X. Luo, and S. Sarkar, “Throughput-optimal scheduling in multichannel access point networks under infrequent channel measurements,” *IEEE Trans. Wireless Commun.*, vol. 7, no. 7, pp. 2619–2629, July 2008.

[4] S. Kittipiyakul and T. Javidi, “Delay-optimal server allocation in multiqueue multi-server systems with time-varying connectivities,” *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2319–2333, May 2009.

[5] H. Halabian, I. Lambadaris, and C.-H. Lung, “Explicit characterization of stability region for stationary multi-queue multi-server systems,” *IEEE Trans. Auto. Control*, vol. 59, no. 2, pp. 355–370, Feb. 2014.

[6] A. Ganti, E. Modiano, and J. N. Tsitsiklis, “Optimal transmission scheduling in symmetric communication models with intermittent connectivity,” *IEEE Trans. Inf. Theory*, vol. 53, no. 3, pp. 998–1008, Mar. 2007.

[7] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross Layer Control in Wireless Netw.*, Now Publisher, 2006.

[8] V. Tsibonis, L. Georgiadis, and L. Tassiulas, "Exploiting wireless channel state information for throughput maximization," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2566–2582, Nov. 2004.

[9] N. Pappas, A. Traganitis, and A. Ephremides, "Stability and performance issues of a relay assisted multiple access scheme," in *Proc. of GLOBECOM* , Miami, USA, Dec. 2010.

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

[11] G. Koole, Z. Liu, and R. Righter, "Optimal transmission policies for noisy channels," *Operations Research*, vol. 49, no. 6, pp. 892–899, Nov. 2001.

[12] T. Javidi, "Rate stable resource allocation in ofdm systems: from waterfilling to queue-balancing," in *Proc. Allerton Commun., Control, and Comput.*, Oct. 2004.

[13] B. Ji, G. R. Gupta, X. Lin, and N. B. Shroff, "Low-complexity scheduling policies for achieving throughput and asymptotic delay optimality in multichannel wireless networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1911–1924, Dec. 2014.

[14] H. Halabian, I. Lambadaris, and C.-H. Lung, "Delay optimal server assignment to symmetric parallel queues with random connectivities," in *Proc. IEEE CDC-ECC*, Orlando, USA, Dec. 2011.

[15] H. Halabian, I. Lambadaris, and C.-H. Lung, "Optimal server assignment in multi-server parallel queueing systems with random connectivities and random service failures," in *Proc. IEEE ICC*, Ottawa, Canada, June 2012.

[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955.

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

[18] S. M. Ross, *Stochastic Processes, 2nd ed.* New York: J. Wiley and Sons, 1996.

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

[20] R. Lidl and G. Pilz, *Applied abstract algebra, 2nd edition*. New York: Springer, 1998.

Hassan Halabian received the B.Sc. and M.Sc. degrees in Electrical Engineering from Isfahan University of Technology in 2005 and 2008, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Carleton University, Ottawa, in 2012. His research interests include stochastic network optimization, queueing systems, optimal resource allocation, and distributed systems. He was a recipient of a number of fellowships and awards, including the MITACS Elevate Strategic Post-Doctoral Fellowship, the Ontario Graduate Scholarship, the Queen Elizabeth II Scholarship in Science and Technology, the Ontario Graduate Scholarships in Science and Technology, and the Carleton Dean of Graduate Studies Academic Excellence Scholarship.

Ioannis Lambadaris was born in Thessaloniki, Greece. He received the Diploma degree in Electrical Engineering from the Polytechnic School, Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1984, the M.Sc. degree in engineering from Brown University, Providence, RI, USA, in 1985, and the Ph.D. degree in Electrical Engineering from the Department of Electrical Engineering, Systems Research Center (SRC), Institute for Systems Research (ISR), University of Maryland, College Park, MD, USA, in 1991. After finishing his graduate education, he was a Research Associate with Concordia University, Montreal, QC, Canada, from 1991 to 1992. Since September 1992, he has been with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, where he is currently a Professor. His research interests include applied stochastic processes and their application for modeling and performance analysis of computer communication networks and wireless networks, quality-of-service (QoS) control for IP and evolving optical networks architectures and stochastic control/optimization in emerging wireless networks. His research is done in close collaboration with his students and colleagues in the Broadband Networks Laboratory. He was the recipient of a Fellowship from the National Fellowship Foundation of Greece (1980-1984) during his undergraduate studies, and a Fulbright Fellowship (1984-1985) for graduate studies in the U.S. He was also the recipient of the Technical Chamber of Greece Award (ranked first in graduating class). While at Carleton University, he received the Premiers Research Excellence Award, and the Carleton University Research Excellence Award (2000-2001), for his research achievements in the area of modeling and performance analysis of computer networks.

Yannis Viniotis received his Ph.D. from the University of Maryland, College Park, in 1988 and is currently a Professor with the Department of Electrical and Computer Engineering at North Carolina State University (http://www.ece.ncsu.edu/people/candice). Dr. Viniotis is the author of over one hundred technical publications, including two engineering textbooks. He has served as the cochair of two international conferences in computer networking. His research interests include virtualization, service engineering, IoT and design and analysis of stochastic algorithms as they apply to network management. Dr. Viniotis was the cofounder of Orologic, a successful startup networking company in Research Triangle Park, NC, that specialized in ASIC implementation of integrated traffic management solutions for high-speed networks.