Multi-Product Systems with Both Setup Times and Costs: Fluid Bounds and Schedules

Wei-Min Lan
IOE Department
University of Michigan
1205 Beal Avenue
Ann Arbor, MI 48109-2117
weimin@ethome.net.tw

Tava Lennon Olsen
John M. Olin School of Business
Washington University in St. Louis
One Brookings Drive
St. Louis, MO 63130-4899
olsen@olin.wustl.edu

May, 2001; Revised: October 2003

Abstract

This paper considers a multi-product, single-server production system where both setup times and costs are incurred whenever the server changes product. The system is make-to-order with a per unit backlogging cost. The objective is to minimize the long-run average cost per unit time. Using a fluid model, we provide a closed-form lower bound on system performance. This bound is also shown to provide a lower bound for stochastic systems when scheduling is static, but is only an approximation when scheduling is dynamic. Heavy-traffic analysis yields a refined bound that includes second-moment terms. The fluid bound suggests both dynamic and static scheduling heuristics, which are tested via a simulation study.

1 Introduction

In this paper we consider a production environment where a number of different products are produced on a single machine and setup activities are necessary when switches of product type are made. These setup activities require both time and cost that depend on the specific product type. Throughout the paper we assume that the setups do not depend on the previous product produced (sequence-independent setups) but Section 6 discusses how the work may be extended to sequence-dependent setups. In general, the arrival and processing rates are likely to differ across product types. As a consequence of these differences, production scheduling will have a great impact on production costs. This paper discusses performance bounds for evaluating scheduling heuristics for these systems and shows how these bounds lead to good heuristic schedules. Examples of appropriate production environments for such models may be found, for example, in Olsen [59].

The scheduling of systems with setups has a vast literature. However, the majority of this literature is for deterministic environments. In particular, solution techniques for the Economic Lot Scheduling Problem (ELSP) consume a large portion of this work. Elmaghraby [26] and Carstensen [16] provide surveys of the ELSP. In addition, Drexl and Kimms [22] survey the lot sizing and scheduling literature for deterministic capacitated systems where demand may change over time (but is known). Surveys of batching decisions are provided by Kuik, Salomon, and van Wassenhove [42] and Potts and Kovalyov [66]. Baker [1] reviews the deterministic scheduling literature for scheduling job families with setups and due dates as well as presenting new heuristic procedures.

Here we focus on systems that are both stochastic and dynamic. That is, arrivals continue to occur (in a random fashion) and decisions are on an ongoing basis, based on the state of the system. Stochastic make-to-stock systems are often referred to as the stochastic ELSP (see, e.g., Federgruen and Katalan [28], Markowitz and Wein [50], and Sox et al. [70]). Here we focus on systems that are make-to-order rather than make-to-stock. Single-server make-to-order queueing systems where a setup time is incurred whenever the server changes product class are known as *polling models*. Polling models have been heavily studied in the queueing and telecommunications literature. The interested reader is referred to Takagi [71], [72], and [73] for surveys of this literature and Levy and Sidi [46] for a survey of polling model applications.

In general, polling table work considers setup times but not costs; whereas there is a large operations literature for systems with setup costs but not times. In general both may be relevant in a production environment. The presence of a setup time to produce a changeover is clear. Setup costs are generated by the need to hire crews to perform the setups (where a per setup cost is an approximation to this cost) and by the use of solvents, etc., for cleaning (where a per setup cost is very realistic). Note that solvent costs can indeed be quite high particularly in industries such as painting. Other setup costs relate to the need to perform warm-up tests on real product before actual production begins and increased scrap rates once initial production has begun.

In general, scheduling rules can be partitioned into two classes, namely, *static* and *dynamic* strategies. Under dynamic strategies, the scheduler determines which product type needs to be produced at any point of time by employing the information gathered from the whole system. On the other hand, static strategies use only the information that pertains to the product currently being produced. Static schedules have the advantage that they are very hands-off, with few informational requirements. Dynamic schedules have the advantage of being very flexible. Here we present both types of scheduling heuristics. Our dynamic scheduling heuristic will have the strong advantage of being readily recoverable if a deviation is made by the controller for any reason; it will therefore the heuristic that we recommend if the informational requirements (which are relatively minor) are met. Indeed, a large difference between telecommunication and manufacturing applications for polling models is that in manufacturing it usually reasonable to assume complete state information, whereas in telecommunications it is not. This is a likely reason why telecommunication polling papers have generally focused on static, rather than dynamic, policies.

Two well known static scheduling methods are *cyclic* scheduling and *periodic* scheduling. In the latter, the server visits the queues following a pre-specified sequence, called the polling table. When the last queue in the table is served, a cycle is said to have ended and the process is repeated from the first queue of the table. Cyclic scheduling (sometimes referred to as rotation schedules) is a special case of periodic scheduling where each queue is visited exactly once in each cycle. Any scheduling method also needs to define when the server should leave the current queue. Three well known static rules for polling systems include *exhaustive, gated,* and *k-limited* (see, e.g., Takagi [71]). This paper deals exclusively with the *exhaustive* service discipline, which requires the server to serve all customers in the current queue until there are no more customers left in that queue. This includes

those customers that arrive during the service period of that queue. Motivation for this restriction is given in the body of the paper. In make-to-stock models, exhaustive service corresponds to a base-stock policy where inventory is replenished until a given base-stock level is reached.

Optimal dynamic schedules for polling systems are extremely difficult to obtain. Even systems with only two queues present significant challenges (see, e.g., Boxma and Down [11], Hofri and Ross [34], Koole [41], Nobel [56] [57], Nobel and Tijms [58], and Reiman and Wein [68]). For systems with more than two queues one is struck by the "curse of dimensionality." While there has been some numerical work on solving such systems (see, e.g., Kim et al. [40]), the schedules that are produced are often not very implementable. For example, if there are few items the queue might be cleared, for a medium number of items a switch might occur, and for a large number of items the server may again stay at the queue (because the system is now heavily loaded). Given the limited availability and utility of optimality results, we focus on heuristic schedules. We also restrict our attention to rules that are readily implementable (e.g., via a spreadsheet).

The primary methodology that we have applied in this paper is a fluid model approach. In a fluid model, the queueing network is approximated by a piecewise-linear fluid system where the workload arrives and is depleted in a deterministic and continuous manner in each queue. In other words, workload is represented as a fluid that flows continuously through the system. Setups are taken to be deterministic. The motivation to model the system as a fluid model is twofold. First, a fluid model is a useful model for systems that process continuous flows, such as those in the chemical industry. Second, a wide range of queueing networks will converge to fluid networks asymptotically (see the following section for literature). Therefore, a fluid model can serve well as an approximation for a stochastic queueing system, especially when approximating first moments such as mean waiting time.

In Section 4, we find a closed-form lower bound on the long-run average production cost per unit time in the fluid system. This lower bound is then shown to be a lower bound on performance for stochastic systems with Poisson arrivals operating under a static scheduling rule. Numerical studies are done to examine the tightness of the bound. In systems which converge to the fluid system, the performance gap between the bound and the optimal value will asymptotically become the gap between the bound and the optimal fluid value. When this gap is small, then the lower bound may serve as a good approximation for the performance of the stochastic system. The

relevant literature on bounds for similar systems is described in Section 2. Section 4 also presents a lower bound under traditional heavy-traffic scalings for dynamic scheduling heuristics.

In addition, the bound leads to scheduling heuristics for the stochastic system. Section 5 presents what we believe to be the first dynamic scheduling heuristic for stochastic make-to-order production systems with both setup times and costs. We also present two static scheduling heuristics, which are provided primarily for benchmarking purposes. The relevant literature on scheduling heuristics for such systems is surveyed in Section 2.

This paper is organized as follows. Section 2 surveys the relevant literature on system bounds and scheduling. Section 3 introduces notation and assumptions used in the rest of the paper. Section 4 presents the lower bounds and Section 5 presents the scheduling heuristics. Finally, Section 6 concludes the paper and discusses avenues for future research.

2 Literature

Due to the fact that fluid systems are interesting in their own right, there has been some work on bounding the performance of fluid systems. In particular, Perkins and Kumar [65] provided a lower bound on the average production cost per unit time for a similar type of fluid model to the one that we consider in Section 4.1. Chase and Ramadge [17] extended this work by introducing the notion of idling and gave a refined lower bound. Our fluid bound extends this previous work to cases where setup times are allowed to be different for each product type and a setup cost is incurred when a setup is performed. Our bound simplifies to Chase and Ramadge's bound for the case where setup costs are zero and all the setup times are identical. In addition, we correct one of the idling conditions derived by Chase and Ramadge. Perkins and Kumar [65] (but not Chase and Ramadge [17]) present a scheduling heuristic for the fluid system based on their bound.

Dai and Weiss [21] present bounds and schedules utilizing a fluid system for minimizing makespan in jobshops. Bounds for systems with both setups times and costs in stochastic systems have been considered by Federgruen et al. [27] in parallel work to ours. However, their work focuses the effect of product variety for make-to-stock systems; they provide both upper and lower bounds for the performance of periodic base-stock policies without idling. Because they are studying product variety, they use their bounds to provide insights on the growth rate of system cost. One of these

insights is described in Section 4.3.

Perhaps most related to the work here, Bertsimas and Xu [8] have provided a lower bound for systems with sequence-dependent setup times and no setup costs. Assuming Poisson arrivals they derive a lower bound on weighted mean delay over all static policies. This lower bound is given as the maximum of two terms. The first term is the optimal weighted mean delay under the optimal M/G/1 priority system and the second is the solution to a convex programming problem. They also provide an explicit solution to the convex programming problem once one of the flow constraints has been relaxed. The relationship of this convex programming problem to our work is discussed in Section 4.2. Further, Bertsimas and Xu provide refined bounds for homogeneous systems when service must be exhaustive and discuss how to design effective static policies that are based on the ratio of switch-overs from one station to another. They provide numerical experiments testing the performance of their bounds.

The lower bounds described above and the bounds in this paper are for stochastic systems under static service policies. Lower bounds for weighted mean delay under dynamic policies have been found in Bertsimas and Niño-Mora [6]. In this work, external arrivals are assumed to be Poisson but Bernoulli routing is allowed. Setups are sequence-dependent. Using the achievable region approach, Bertsimas and Niño-Mora provide a series of math programming problems of increasing relaxation and increasing ease of solution that lower bound performance. However, none of the solutions are explicit and it is very unlikely that explicit solutions could be found given the complexity of the problems. Bertsimas and Niño-Mora also outline (but do not test) how their work can be used for performance evaluation and to design efficient static heuristic policies. This work is extended to systems with multiple servers in Bertsimas and Niño-Mora [7].

In heavy-traffic limit theorems there are two predominant methods of scaling, namely strong-law (fluid) scalings and central-limit (diffusion) scalings. The latter is the more traditional scaling for performance approximations because in simple systems without setups strong-law scalings often result in a zero limit. However, such a zero limit is often highly useful in proving stability (see, e.g., Dai [19] and Rybko and Stolyar [69]). Indeed, Dai and Jennings [20] and Jennings [38] explicitly use fluid models of systems with setups to prove stability.

In systems with setups, strong-law scalings, where both work and time are scaled by one minus utilization, can result in systems with non-zero limits. In such systems the resulting limit looks like

a fluid system. For any dynamic systems that is converging to a fluid system in heavy-traffic the fluid lower bound should indeed hold under the limiting regime. Note that Bertsimas and Xu [8] also conjecture that their lower bound for static policies should hold for dynamic policies under heavy-traffic conditions.

In systems with setups, it is also of interest to perform strong-law type scalings with respect to setup times. In other words, mean setup times are increased to infinity while both work and time are scaled down in proportion to mean setup time. Olsen [60] and van der Mei [75] consider such heavy-setup scalings. Most relevant to the work here, [60] considers systems with polling tables under exhaustive service in steady-state. By noting that the waiting time distribution for a randomly sampled particle of work in a steady-state fluid system has distribution equal to a uniform [0,1] random variable multiplied by intervisit time (see, e.g., Olsen and van der Mei [63]), the limits correspond to what would be derived from a fluid system with appropriate flows.

From this prior work, one might expect that a fluid model should be a good approximation for the stochastic system when:

1. The system is in heavy-traffic (i.e., utilization is close to 100%),
2. Setup times are large and have low variance, or
3. There are a large number of queues each with similar setup times.

In these cases, the lower bound is likely to hold even for stochastic dynamic systems and scheduling methods that work well for the fluid system should also work well in the stochastic setting.

We are far from the first authors to use fluid systems for scheduling. Some of the relevant work has been described above in the context of bounds. There is also work that views the fluid model as the limit of a stochastic system. For example, fluid limit models are explicitly considered in Meyn [52] and [53]. The optimal policy for the fluid model provides a policy for a stochastic network model that is "almost optimal in heavy traffic" (see [53] for details). Also, Maglaras [48] uses fluid limits that result from heavy-traffic limits to derive schedules for a system without setups. Recent surveys of the use of fluid models for scheduling may be found in Bertsimas et al. [4] and Hasenbein [32].

There have been some limited optimality results for multi-queue systems like those in this paper. For example, Liu et al. [47] show that for symmetric single machine systems, with some

mild independence assumptions, policies that are exhaustive and serve the next class as that with the most work stochastically minimize the unfinished work and the number of jobs in the system at any time. Duenyas and Van Oyen [24] show that in order to minimize total expected throughput it is optimal to serve exhaustively at the class with the smallest processing time. Markowitz et al. [49] show that, in cyclic service systems under heavy-traffic, it is optimal to serve all but the lowest priority class exhaustively. Van der Mei and Levy [76] analyze cyclic systems under branching-type service. They derive closed form expressions for the expected delay under standard heavy-traffic scalings in terms of a single "exhaustiveness" parameter. One can use their expressions to show that there exists a value of total expected setup around a cycle such that all queues should be served exhaustively for setup values greater than or equal to this threshold. Further, if the backlogging cost per queue equals the queue's utilization (so the objective is to minimize expected work) then their results show work is minimized when service is exhaustive. If setups are set to zero then their results imply that expected work is indifferent to the level of exhaustiveness.

Given the scarcity of optimality results, there have been a number of heuristics designed for stochastic, dynamic, make-to-order systems with either setup times or costs. However, we are aware of none that are available for both. Boxma et al. [12] find sequences for periodic service to minimize expected waiting cost where costs are assumed to be linear. Browne and Yechiali [14] derive quasi-dynamic index policies where each queue is visited exactly once per cycle but the order in which queues are visited within a cycle changes dynamically. Duenyas and Van Oyen [23] provide a dynamic heuristic for scheduling systems with setup costs that is based on a dynamic programming formulation and Duenyas and Van Oyen [24] provides similar results for systems with setup times. Olsen [59] provides a dynamic heuristic that is designed to produce flow times with a light tailed distribution and moderate mean.

Other relevant models include: Setups in the context of lotsizing (see, e.g., Hodgson et al. [33], Karmarkar et al. [39], and references there-in), models with no arrivals but some stochastic element (see, e.g., Van Oyen et al. [77] and Van Oyen and Pichitlamken [78]), polling systems with fixed times for beginning and ending service (see, e.g., Borst et al. [10]), polling systems where the queues are in sequence and the customer is routed through each station in turn (see, e.g., Duenyas et al. [25] and Iravani et al. [36]), scheduling systems with finite buffers (see, e.g., Browne and Yechiali [15], Hwang and Chang [35], Kim and Van Oyen [40], and Marsan et al. [51]), optimization within cyclic

schedules (see, e.g., Blanc and van der Mei [9] and Levy [45]), transportation applications (see, e.g., Gandhi and Cassandras [31]), workload allocation or customer routing across parallel polling models (see, e.g., Benjaafar and Gupta [3] and Rajan and Agrawal [67]) and practical applications of setup scheduling (see, e.g., Fransoo et al. [29], Olson and Schniederjans [64], and Tayur [74]).

3 Notation and Assumptions

The systems we consider are composed of N types of product (classes of jobs) served by a single machine (server) in a make-to-order fashion. Orders for product i arrive at the machine with rate equal to λ_i, $1 \leq i \leq N$. We let μ_i represent the processing rate of product i, $1 \leq i \leq N$. A fixed setup cost k_i and a setup time with mean s_i are incurred before the server can produce product i, regardless of which product was last produced, $1 \leq i \leq N$. We will make different assumptions about the inter-arrival, service, and setup time distributions in different sections and so leave discussion of such assumptions to the appropriate section.

The workload at time $t \geq 0$ for product i, $v_i(t)$, is the expected amount of time that the server must spend working only on product i to clear all current jobs that are waiting, $1 \leq i \leq N$. Consequently, $\rho_i = \lambda_i/\mu_i$ is the type i workload arrival rate, $1 \leq i \leq N$. Further, define $\rho = \sum_{i=1}^{N} \rho_i$ to be the system utilization rate excluding setups. Throughout we assume that $\rho < 1$. There is a workload backlogging cost of c_i per unit time, $1 \leq i \leq N$. For systems with discrete items, we convert this to a per unit backlogging cost, $\tilde{c}_i$, by dividing each c_i by μ_i (i.e., $\tilde{c}_i = c_i/\mu_i$), $1 \leq i \leq N$. In the stochastic (or discrete-item) system, let $L_i(t)$ represent the number of jobs in queue i at time $t \geq 0$, $1 \leq i \leq N$. Thus workload $v_i(t) = L_i(t)/\mu_i$; note that we have defined workload in terms of expected time to complete all jobs so that it is observable at any time. Under any policy γ we let L_i^{γ} be the steady-state variable representing the number of customers in queue i (if that quantity exists). We also define the cost terms $w_i = c_i\rho_i(1-\rho_i)$, $1 \leq i \leq N$.

In the fluid system we deal exclusively with workload. When the server is serving queue i (and there is backlog), workload is being depleted at rate $1-\rho_i$ and otherwise workload is building up at rate ρ_i, $1 \leq i \leq N$ (both in a deterministic fashion). Setups are also deterministic. Under our scheduling policies we will allow the server to *cruise* at a queue. This is where the server stays at a queue that has been exhausted and continues to serve work as it arrives. This is the same idea as

"idling" as presented by Chase and Ramadge [17]. We choose to use a different term because the server in the fluid system is not ever idle during this time, but instead is working continuously at the workload arrival rate. A cruising server in the stochastic system will have periods of idleness alternating with periods of working. As in Perkins and Kumar [65] we give the following definition of stability for the fluid system.

Definition 1 *A scheduling policy is defined to be stable for the given fluid system if there exists a constant C such that for all queues $i \in [1, N]$, $\sup_t v_i(t) \leq C$.*

If the server is at queue i, then a *static* policy will only use information on $v_i(t)$ to determine whether to (a) continue service at queue i or (b) begin a setup at a new queue (and if so, at which queue). Cruising is not allowed under static service. A *dynamic* policy may at any time t use information on all $v_j(t)$, $1 \leq j \leq N$, to determine whether to (a) continue service at the current queue, (b)"cruise" at the current queue, or (c) begin a setup at a new queue (and if so, at which queue).

The *intervisit* time for a queue is the time between last serving that queue and the next time service commences at that queue (following a setup). The j^{th} intervisit time at queue i is represented by $I_{i,j}$, $1 \leq i \leq N$, $j = 1, 2, \ldots$. Under any policy γ we let I_i^γ be the steady-state variable representing the intervisit time of queue i (if that quantity exists). Define $n_i(t)$ at the number of setup completions for type i, $1 \leq i \leq N$, up to time t. If a constant limit exists almost surely under some policy γ then define

$$n_i^\gamma = \lim_{t \to \infty} n_i(t)/t.$$

In the stochastic system we define the waiting time of the j^{th} job at queue i, $W_{i,j}$, to be the time from a job's arrival to its commencement of service, $1 \leq i \leq N$, $j = 1, 2, \ldots$. Under any policy γ we let W_i^γ be the steady-state waiting time at queue i (if that quantity exists).

Definition 2 *We define a policy γ to be* stable *in a stochastic system if the underlying Markov process governing the queueing dynamics is positive Harris recurrent (see, e.g., Meyn and Tweedie [54]) yielding steady-state random variables L_i^γ, W_i^γ, and I_i^γ, $1 \leq i \leq N$. Further we require that $E[L_i^\gamma]$, $E[W_i^\gamma]$, $E[I_i^\gamma]$, and n_i^γ exist and are finite, $1 \leq i \leq N$.*

Note that, for ease of exposition, we have chosen a relatively strong and slightly redundant definition

of stability (e.g., $E[W_i^\gamma] < \infty$ implies $E[L_i^\gamma] < \infty$). There is no reason to believe that the results hinge on this definition.

4 Lower Bounds

This section provides lower bounds for system performance; it is organized as follows. Section 4.1 provides a closed-form lower bound on the long-run average production cost in the fluid system. Section 4.2 then shows that this lower bound is also a lower bound on performance for stochastic systems with Poisson arrivals operating under a static scheduling rule. Section 4.3 presents a brief numerical study of the tightness of the bound and discusses insights available from the bound. Last, Section 4.4 uses heavy-traffic analysis to refine the bound.

4.1 Fluid Bounds

In this section, we consider a fluid system and give a lower bound on its performance. In our approach, we find properties of the optimal trajectory of the workload for a queue. Based on this knowledge, we build a convex programming model to represent the system. By solving this convex programming problem, we derive a lower bound on the long-run average production cost for the fluid system. In doing so, we derive two necessary conditions for the machine to use cruising at a particular queue.

The following lemma contains terms n_i and d_i which have a good intuitive explanation as the number of setups in queue i per unit time (setup frequency) and the fraction of time that the machine spends cruising in queue i, respectively. We use the word "intuitive" because under our definition of stability we have not proven that such fractions exist. In Boxma et al. [12], [13] and Federgruen and Katalan [28], polling tables are determined by first finding appropriate n_i, $1 \leq i \leq N$. Our solution to n_i can serve this purpose as well. This idea is explored further in Section 5.

Lemma 1 *The solution of the following non-linear programming model is a lower bound on the average production cost of all stable policies in the fluid system.*

$$\textbf{Minimize:} \qquad \textit{Average production cost (AC)} = \sum_{i=1}^{N} \frac{w_i(1-d_i)^2}{2n_i} + \sum_{i=1}^{N} n_i k_i \tag{1}$$

$$\textbf{Subject to:} \qquad (1-\rho) = \sum_{i=1}^{N} [n_i s_i + d_i(1-\rho_i)] \tag{2}$$

$$n_i, d_i \geq 0 \qquad \forall\ i \in [1, N]$$

where $w_i = c_i \rho_i (1-\rho_i)$.

Proof: The proof is rather tedious and is relegated to the online appendix; an outline is as follows. We first show that, when working, the server will either work at rate 1 when serving a queue with a backlog or at the workload arrival rate (i.e., cruise) when backlog is zero. We then show that for each queue (without considering the other queues), the optimal trajectory of workload realization should have two properties: 1) the workload should be served exhaustively in each production run; and 2) the optimal workload level that accumulates between production runs should be the same. This results in the optimal average cost for a single queue. Note that such a trajectory may not actually be feasible once all queues are considered but this is allowable as we are constructing a lower bound rather than a feasible schedule. At the next step, we consider the balance of workload in all queues under a given time period T and derive a constraint for this system. We then let T go to infinity. □

The math programming problem of Lemma 1 can be solved analytically to give a lower bound on the long-run average production cost of the fluid production system. This closed-form expression is given in the following theorem. The lower bound depends upon whether cruising is to be used or not. Cruising condition 1 (CC_1) finds which queues (of which there may be more than one) are most cost effective to cruise at; cruising condition 2 (CC_2) evaluates whether there is sufficient capacity to make cruising cost effective. Note that it is possible for multiple queues with different parameters to satisfy both the cruising conditions. Nevertheless, the given cost bound remains the same regardless of which queue that satisfies both cruising conditions is chosen to be queue i in the expression. In other words, while the solution to the lower bound is unique, the solution to the values $\{d_i : 1 \leq i \leq N\}$ is not. A proof of this is given in the online appendix that accompanies this paper.

Theorem 1 *For any stable scheduling policy in a fluid system:*

$$\liminf_{T\to\infty} \sum_{i=1}^{N} \frac{1}{T} \left[\int_0^T c_i v_i(t)\, dt + n_i(T) k_i \right] \geq FB_{sk}$$

where

$$FB_{sk} = \begin{cases} \sum_{j=1, j\neq i}^{N} \sqrt{2w_j(\delta_i s_j + k_j)} + \delta_i(\rho - \rho_i) \\ \qquad \textit{if the CCs hold for some queue } i \\ \sum_{j=1}^{N} \sqrt{\frac{w_j}{2}} \left[\frac{k_j}{\sqrt{\beta s_j + k_j}} + \sqrt{\beta s_j + k_j} \right] \\ \qquad \textit{if none of the queues satisfy the CCs} \end{cases}$$

and β *is the solution to* $\sum_{j=1}^{N} s_j \sqrt{\frac{w_j}{2(\beta s_j + k_j)}} = (1-\rho)$.
The conditions for cruising at queue i *(CCs) are*

$$CC_1 \quad : \quad \delta_i \triangleq \left[\frac{s_i w_i}{(1-\rho_i)^2} + \frac{w_i}{(1-\rho_i)^2} \sqrt{s_i^2 + \frac{2k_i(1-\rho_i)^2}{w_i}} \right] \geq \delta_j \qquad \forall j \in [1, N]$$

$$CC_2 \quad : \quad \sum_{j=1}^{N} s_j \sqrt{\frac{w_j}{2(\delta_i s_j + k_j)}} < (1-\rho).$$

Proof: The proof follows from solving the KKT conditions for the convex programming problem in Lemma 1 and is given in the Appendix. □

One consequence of the proof of Theorem 1 is that there are closed-form expressions for the values of n_i in Lemma 1. If the cruising conditions do not hold then, for $1 \leq i \leq N$,

$$n_i = \sqrt{\frac{w_i}{2(\beta s_i + k_i)}} = \sqrt{\frac{c_i \rho_i (1-\rho_i)}{2(\beta s_i + k_i)}}.$$

Thus visit frequencies may be expected to scale up with backlogging costs and down with setup times and costs. If the cruising conditions do hold then the expressions are slightly more complicated and are given as equations (12), (13), and (14) in the Appendix.

In the remainder of this section, we discuss some special cases for the system.

Corollary 1 *In a system without any setup costs, for any stable scheduling policy in a fluid system:*

$$\sum_{i=1}^{N} \liminf_{T\to\infty} \frac{1}{T} \int_0^T c_i v_i(t)\, dt \geq FB_s$$

where

$$FB_s = \begin{cases} 2(\rho - \rho_i)\left(\frac{s_i c_i \rho_i}{1-\rho_i}\right) + \sum_{j=1, j\neq i}^{N} 2\sqrt{\frac{w_j s_j s_i c_i \rho_i}{1-\rho_i}} & \textit{if the CCs hold for queue } i \\ \frac{1}{2(1-\rho)} \left[\sum_{i=1}^{N} \sqrt{w_i s_i} \right]^2 & \textit{otherwise} \end{cases}$$

and the cruising conditions are

$$\begin{aligned} CC_1 \quad &: \quad \frac{s_i c_i \rho_i}{1-\rho_i} \geq \frac{s_j c_j \rho_j}{1-\rho_j}, \ \forall \ j \in [1, N], \\ CC_2 \quad &: \quad \sum_{j=1}^{N} \sqrt{w_j s_j} < 2\left(\frac{1-\rho}{1-\rho_i}\right)\sqrt{w_i s_i}. \end{aligned}$$

Proof: Let $k_i = 0$, it follows that $\delta_i = \dfrac{2 s_i c_i \rho_i}{1-\rho_i}$ and $\beta = (\sum_{i=1}^{N} \sqrt{w_i s_i})^2 / 2(1-\rho)^2$. The result follows. □

Chase and Ramadge [17] provide a similar lower bound for systems with zero setup costs and identical setup times. If we let the s_i all be the same, the lower bound in Corollary 1 indeed reduces to the lower bound derived by Chase and Ramadge in their Theorem 1. However, our CC_1 is $\dfrac{c_i \rho_i}{1-\rho_i} \geq \dfrac{c_j \rho_j}{1-\rho_j}$ which corrects the first cruising condition from Chase and Ramadge [17]. Further, as described in Lan [43], although there is a flaw in Chase and Ramadge's proof that at most one queue should have cruising, the result is still true (unlike in systems with both setup times and costs or with setup costs alone where multiple queues may have cruising).

Corollary 2 *In a system without any setup times, for any stable scheduling policy in a fluid system:*

$$\sum_{i=1}^{N} \liminf_{T \to \infty} \frac{1}{T} \left[\int_0^T c_i v_i(t)\, dt + n_i(T) k_i \right] \geq FB_k$$

where

$$FB_k = \sum_{j=1}^{N} \sqrt{2 k_j w_j} - (1-\rho)\sqrt{\frac{2 k_i c_i \rho_i}{1-\rho_i}},$$

and i can be any queue that satisfies $\dfrac{k_j c_j \rho_j}{1-\rho_j} \leq \dfrac{k_i c_i \rho_i}{1-\rho_i}, \forall j \in [1, N]$.

Proof: Let $s_i = 0$ so that $\delta_i = \sqrt{\dfrac{2 k_i c_i \rho_i}{1-\rho_i}}$, then the result follows. □

In Corollary 2, CC_2 from Theorem 1 has disappeared and CC_1 is always true for some queue, therefore cruising is inevitable under this scenario. Indeed, as there are no setup times, cruising must occupy $1-\rho$ of the time. An important property associated with systems with setup costs only is that, in FB_k, the average setup costs equals the average workload backlogging costs in each production run for each product. This is similar to the results in a traditional EOQ model (see, e.g., [55]) in which average setup costs equals average inventory costs for a single product system. Note that, without cruising, it would not be possible to let the backlogging costs equal setup costs for each product in each production run.

4.2 Lower Bound Under the Stochastic Setting

In this section, we show that the lower bound that was derived from the fluid model is a valid lower bound for static policies under a stochastic setting. Product i arrives to the system according to a Poisson process with rate λ_i, while unit production times for product i are $i.i.d$ with mean $1/\mu_i$ and finite second moment σ_i^2, $1 \leq i \leq N$. The machine serves the jobs in each queue by following a First-Come-First-Serve (FCFS) policy. The setup times for queue i are also $i.i.d$ with mean s_i and finite second moment, $1 \leq i \leq N$. Recall that exhaustive service was defined in the introduction.

Lemma 2 *Pick any queue i, $1 \leq i \leq N$. For a given time period $[0, T]$, the average number of jobs in queue i is minimized under an exhaustive service policy in queue i for each production run.*

Proof: Let policy π be an arbitrary non-exhaustive server allocation policy. Let ω be an arbitrary sample path of interarrival, service, and setup times (each queue receives its own sequence of interarrival, service, and setup times). Suppose that there are $n_i^{\pi}(T)$ setup completions for type i, $1 \leq i \leq N$, up to time T. Let τ be any instant such that $L_i(\tau) > 0$ and the server switches to any other queue under policy π. Let time $\tau + h$ be the first time that the jobs left in queue i at time τ are served and $\tau + h + \varepsilon$ be the time point that one job has been served since $\tau + h$. If such a time epoch exists, consider a policy π^* that mimics policy π up to time τ then serves this particular job until $\tau + \varepsilon$. It then follows exactly the same path as policy π does in $[\tau, \tau + h]$ and then mimics policy π again from time $\tau + h + \varepsilon$ to time T. Let $L_i^*(t)$ represent the jobs in queue i under policy π^*. Because $L_i(t) = L_i^*(t)$ for $t < \tau$ and $t \geq \tau + h + \varepsilon$, we need only to compare the number of jobs in queue i between τ to $\tau + h + \varepsilon$ under policies π and π^*. By the way that policy π^* is defined,

$$L_i(t) - 1 = L_i^*(t) \qquad t \in [\tau, \tau + h).$$

and

$$L_i(t) = L_i^*(t) \qquad t \in [\tau + h, \tau + h + \varepsilon).$$

Therefore the average number of jobs under π^* is less and policy π cannot be optimal for queue i.

If such a time epoch $\tau + h$ does not exist, i.e., the jobs left in time τ haven't been served before time T, consider a policy π^* that mimics policy π up to time τ, then serves this particular job until $\tau + \varepsilon$ or T. It then follows exactly the same path as policy π does in $[\tau, T - \varepsilon]$. Since $L_i(t) = L_i^*(t)$

for $t < \tau$ and $L_i(t) - 1 = L_i^*(t)$ for $t \in [\tau, T]$. The average number of jobs under π^* is less and policy π cannot be optimal for queue i.

Iterating this procedure, we finally end with an exhaustive policy such that average number of jobs is always less than or equal to an arbitrary non-exhaustive policy. The proof is then concluded by removing the conditioning on ω. □

Note that Lemma 2 does not imply that an exhaustive service discipline is globally optimal because other queues are not taken into consideration at the same time.

Theorem 2 *Let π be some stable static policy that does not allow idling. The lower bound on the long-run average production cost derived from the fluid model in Theorem 1 without allowing cruising is a lower bound for the average steady-state cost for the stochastic system operated under π.*

Proof: Define each π_i^*, $i = 1, \ldots, N$ as the exhaustive policy laid out in the proof of Lemma 2. Note that, by how π_i^* is defined in the proof, along any coupled sample path ω, $I_{i,j}^{\pi}(\omega) = I_{i,j}^{\pi_i^*}(\omega)$. Further, under any given sample path ω and for any $t \geq 0$, $n_i^{\pi}(t,\omega) - 1 \leq n_i^{\pi_i^*}(t,\omega) \leq n_i^{\pi}(t,\omega)$. Thus, $n_i^{\pi_i^*} = n_i^{\pi}$, where existence of $n_i^{\pi_i^*}$ is implied by existence of n_i^{π}. As π is stable and $L_i^{\pi}(t)$ is integer valued there exists a constant integer c such that $\liminf_{t\to\infty} L_i^{\pi}(t) = c$, almost surely. Each time the system under π hits c the coupled system under π_i^* will have at most c customers in queue i and if it has c customers then it will have the identical number of customers in all queues at the identical time point. Therefore, by the stability of π, $L_i^{\pi_i^*}$, $W_i^{\pi_i^*}$, and $I_i^{\pi_i^*}$ are all well defined. Because $E[L_i^{\pi_i^*}] \leq E[L_i^{\pi}] < \infty$,

$$\sum_{i=1}^{N} \frac{c_i}{\mu_i} E[L_i^{\pi}] \geq \sum_{i=1}^{N} \frac{c_i}{\mu_i} E[L_i^{\pi_i^*}] = \sum_{i=1}^{N} c_i \rho_i E[W_i^{\pi_i^*}],$$

where the last equivalence follows from Little's law.

The requirements of the Fuhrmann-Cooper decomposition [30] for $M/G/1$ queues with generalized vacations and exhaustive service hold for queue i under π_i^*. In particular, the number of customers that arrive during an intervisit time is, by definition, independent of the number in queue i at the start of the intervisit time (as that number is zero). Further, the waiting time of a customer at queue i is independent of the arrival process to queue i that occurs after that customer's arrival epoch for the following reasons. If the customer arrives during an exhaustive service period its wait

only depends on customers ahead of it in the queue; if the customer arrives during an intervisit period then its wait depends on the customers ahead of it an the length of the intervisit period. However, because π is a static policy, any intervisit period for queue i is independent of the arrival process to queue i which occurs after that intervisit period has begun. As noted above, along any coupled sample path ω, $I_{i,j}^{\pi}(\omega) = I_{i,j}^{\pi_i^*}(\omega)$. Further, commencement of the intervisit time under π_i^* is after that under π, so that independence of the intervisit period and the arrival process (after its commencement) will continue to hold. Therefore, from the Fuhrmann-Cooper decomposition,

$$E[W_i^{\pi_i^*}] \quad = \quad E[I_i^{\pi_i^*}]\frac{1+cv(I_i^{\pi_i^*})^2}{2} + E[Y_i] \tag{3}$$

where $cv(I_i^{\pi_i^*})^2$ is the squared coefficient of variance of $I_i^{\pi_i^*}$ and $E[Y_i] = 1/\mu_i + (1+\mu_i^2\sigma_i^2)(2\mu_i^2(1-\rho_i))$ is the expected waiting time of queue i in an ordinary $M/G/1$ system without setups.

The fraction of time that the machine is serving queue i plus the fraction of time that the machine is not serving queue i must equal 1. Therefore,

$$1 = \rho_i + n_i^{\pi_i^*} E[I_i^{\pi_i^*}],$$

and hence

$$E[I_i^{\pi_i^*}] = \frac{(1-\rho_i)}{n_i^{\pi_i^*}} = \frac{(1-\rho_i)}{n_i^{\pi}}. \tag{4}$$

Therefore, since $cv(I_i^{\pi_i^*})^2 \geq 0$

$$\sum_{i=1}^{N} \frac{c_i}{\mu_i} E[L_i^{\pi}] \geq \sum_{i=1}^{N} \frac{c_i\rho_i(1-\rho_i)}{2n_i^{\pi}} + \sum_{i=1}^{N} c_i\rho_i E[Y_i] \geq \sum_{i=1}^{N} \frac{w_i}{2n_i^{\pi}} \tag{5}$$

Observing that constraint (2) with $d_i = 0$ must also hold for stochastic systems and (5) is the same as the first term in (1), the proof is complete. Note that a better lower bound can be obtained simply by adding back the $E[Y_i]$ in (5), which are independent of the scheduling policy. □

Remark 1: The only place the proof hinges on the assumption of Poisson arrivals is in applying the Fuhrmann-Cooper decomposition in equation (3). However, Bertsimas and Mourtzinou [5] have generalized the Fuhrmann-Cooper decomposition results. Therefore this result also extends to systems with Mixed Generalized Erlang arrival process using Theorem 4 in [5] or, as in Theorem 7 in [5], becomes asymptotically exact for systems with general renewal arrival processes under heavy-traffic conditions (see [5] for rigorous definitions of these terms). Note that Bertsimas and

Mourtzinou require that service is static and that the polling order does not depend on the number of customers in the system (see assumptions A.1 - A.7 in [5]).

Remark 2: The assumption of a static service policy precludes schedules with state-dependent idling or cruising. However, it is fairly straight-forward to extend Theorem 2 to systems with state-independent idling; in this case the lower bound needs to be the bound with cruising. This may be useful for systems with prescheduled preventive maintenance, etc.

Remark 3: It is shown in the numerical study that the lower bound is not a lower bound for systems under dynamic scheduling schemes. One reason for this lies with the inspection paradox (see, e.g., [79]). For example, if one could show that the probability that a type i job does not see the machine cruising in queue i is $(1 - d_i^{\pi})$ (as one may intuitively expect), then the lower bound would follow for static scheduling with cruising. However, this is not the case because of the inspection paradox. Note that the fluid bound is still useful for dynamic systems because it can guide scheduling methods (see Section 5) and will be asymptotically correct for any system where the stochastic system converges to the fluid system. A non-fluid approach (perhaps like that in Bertsimas and Niño-Mora [6]) will likely have to be taken to obtain bounds for dynamic stochastic systems unless such bounds are only considered to be true bounds in some limiting regime (e.g., in heavy-traffic or under heavy-setups).

Remark 4: As mentioned in Section 2, there are a couple of (as yet unpublished) works that provide bounds that are similar in spirit to the above. In particular, Federgruen et al. [27] provide a bound that is similar in form to that in Lemma 1, translated to systems that are make-to-stock and disallowing idling. However, instead of solving the convex programming problem explicitly, they first relax the problem to provide their bound. The non-linear programming model is also similar to that developed by Bertsimas and Xu [8]. In particular, if setup costs and cruising are eliminated and setups are sequence-independent then the bounds become equivalent if the $E[Y_i]$ that were found in (5) are added back in.

4.3 Numerical Results and Insights

This section discusses insights available from the lower bounds. We use simulation to study the fluid bound with respect to changes in setup times and system utilization rates. The simulation code is written in the C programming language and the simulation run length required for each case was at least that needed for 5,000,000 job arrivals. To avoid statistical problems due to "initial transience" (i.e., the system is not in steady-state when it is started), the first 10% of data is discarded. The statistics are collected using the method of "batch-means". Due to space considerations we do not quote confidence intervals. However, they were in general less than 5% of the estimated value. We also use common random numbers across the simulations so that the estimates are positively correlated. The same simulation is used in Section 5.4.

One observation from Theorem 1 and Corollaries 1 and 2 is that the lower bound need not grow rapidly as ρ tends to one if the cruising conditions are satisfied by some queue. In other words, the usual scenario that costs must grow in proportion to $1/(1-\rho)$ is only true for systems where idling or cruising is not currently being implemented. Therefore, if we seek to use the bounds in heavy-traffic performance approximations such approximations may only be valid for systems without cruising. Note that, for systems with non-zero setup times, there is a value $\hat{\rho} < 1$ such that the cruising conditions are not satisfied for $\rho \geq \hat{\rho}$.

For systems that consider setup costs or setup times only, it can be deduced from Corollaries 1 and 2 that the lower bound grows m times when the s_i are all scaled by a factor m and grows $\sqrt{m}$ times when the k_i are scaled by a factor m. This is similar to one of the effects detailed in Federgruen et al. [27]. Such linear scaling in setups is similar to the heavy-setups results described in Section 2.

In the following data set, we study the system under a "perfect" asymmetric case. A perfect asymmetric system is a non-symmetric system that can space queues evenly according to the optimal n_i. In particular, consider a system with 4 queues, if the ratio of the optimal n_i is 3:1:1:1 for queues 1 to 4, respectively, and the utilization rates and mean setup times for queues 2 to 4 are the same, then the queues in the system can be served sequentially in the order of 1, 2, 1, 3, 1, 4 and so on. The expected intervisit time for queue 1 will be the same in all production runs. Moreover, for a fluid system, such a schedule is in fact optimal so that there is no difference between the lower bound and an optimally scheduled fluid system.

We simulate a system with parameters that result in the optimal sequence described above. The data set contains 4 queues, the workload backlogging costs and setup costs are $c_i = 1$ and $k_i = 50$ for all $1 \leq i \leq 4$. Further $\mu_1 = 9$, $\mu_i = 1$ for $i \in [2, 4]$, $\lambda_1 = 1.125,\ 1.575$, or 2.025, and $\lambda_i = 0.125,\ 0.175$, or 0.225 for $i \in [2, 4]$ with ρ equal to 0.5, 0.7, or 0.9, respectively. We simulate a periodic exhaustive service system with a polling table equal to the optimal fluid sequence. We assume exponential service and interarrival times. Setup times are taken to be either deterministic or exponential, as indicated. The simulation results are reported in Table 1.

		$\rho = 0.5$			$\rho = 0.7$			$\rho = 0.9$		
		Sim	*FB*	*% diff*	*Sim*	*FB*	*% diff*	*Sim*	*FB*	*% diff*
det	$s_i = 1$	26.6	15.9	67.7	29.9	21.4	39.8	42.9	36.4	18.0
	$s_i = 10$	48.9	41.9	16.8	91.1	88.1	3.4	326.2	314.4	3.8
	$s_i = 100$	395.7	394.0	0.4	869.7	866.4	0.4	3148.4	3138.9	0.3
exp	$s_i = 1$	27.4	15.9	72.6	30.6	21.4	42.7	49.8	36.4	36.8
	$s_i = 10$	49.2	41.9	17.4	99.0	88.1	12.3	336.4	314.4	7.0
	$s_i = 100$	456.5	394.0	15.9	951.2	866.4	9.8	3277.9	3138.9	4.4

Table 1: Convergence to the Lower Bound in the Perfect Asymmetric Case

Table 1 reports that average cost per unit time for a variety of ρ and s_i (as shown). Notice that the simulation gets very close to the fluid bound as setups and/or ρ get large. This corresponds well to the convergence results discussed in Section 2. That the percentage differences are not always strictly decreasing in ρ is probably more due to simulation error than actual properties of convergence.

4.4 Heavy-Traffic Bounds

As discussed earlier, fluid models can result from strong-law type scalings. Frequently, more refined results are possible from central-limit type scalings. In this section we suggest some heavy-traffic performance bounds. To keep notation and technical details to a minimum we sometimes use heuristic verbal explanations in place of rigorous mathematics. We feel this is appropriate given that the statements in this section are conjectures rather than theorems.

Our heavy traffic performance bounds are based on the work of Coffman et al. [18] (CPR), who provide the heavy-traffic averaging principle (HTAP) for cyclic exhaustive service polling systems. This principle implies that during the course of a cycle total scaled workload is effectively constant and the individual queues' workloads are defined by a deterministic trajectory. This trajectory

may be thought of as a fluid model where work is flowing in at constant rate ρ_i/ρ to each queue i, $(1 \leq i \leq N)$, and flows out at rate 1 (switch-overs become negligible under this scaling). Total work in the system is governed by a Bessel process.

Here we suppose that the HTAP holds for some dynamic system without idling. In particular we assume that, under appropriate heavy-traffic scalings, the behavior at the queue level corresponds to a fluid model, where this fluid model has well defined long-run average visit frequencies, $\{\hat{n}_i(x) : 1 \leq i \leq N\}$, that scale with total workload x such that $\hat{n}_i(x)/\hat{n}_j(x)$ remain constant, $1 \leq i, j \leq N$.

From the proof of Theorem 1, we see that performance of the fluid model is bounded below by a fluid model that serves all queues exhaustively and each intervisit time for a queue is the same. This type of system would correspond to a "perfect" polling table where queues are evenly spaced.

Recently, Olsen and van der Mei have provided heavy-traffic limit theorems for the behavior of polling tables under exhaustive (and gated) service. Olsen and van der Mei [62] prove results for systems in steady-state with Poisson arrivals and Olsen and van der Mei [63] (OVDM) conjecture results for both transient and steady-state systems with renewal arrivals. These conjectures are based on the work of CPR and the steady-state results correspond exactly to those rigorously proven in [62] for systems with Poisson arrivals. We can use this work to analyze systems operating under perfect polling tables.

Consider a polling table of length M. Let s be the total expected setup time in a cycle. Let m_i be the number of visits to queue i in the table. Define

$$\sigma^2 = \sum_{i=1}^{N} \lambda_i (Var[B_i] + \rho_i^2 Var[A_i]),$$

where $Var[B_i]$ and $Var[A_i]$ are the variances of the service and interarrival times, respectively. When there are a total of x units of scaled work in the fluid system, let $c(x)$ be the corresponding cycle time (as a function of x). Then let ϱ be such that $x = \varrho c(x)$ (where ϱ in our notation is represented by δ in OVDM). See OVDM for an exact expression for ϱ, which in general requires the solution to $M - N$ equations.

A unique pseudo-queue is associated with each entry in the polling table. Denote by PQ_j the pseudo-queue associated with the j-th entry in the polling table, $1 \leq j \leq M$; its corresponding queue has index $T(j)$. Let $\hat{\pi}_j$ be the fraction of customers at queue $T(j)$ that are served at PQ_j, $j = 1, \ldots, M$. Then defining $\mathbf{I}_j^{(PQ)}$ as the length-biased (or time-averaged) intervisit time for

pseudo-queue, OVPM states that

$$(1-\rho)\mathbf{I}_j^{(PQ)} \rightarrow_d \tilde{\mathbf{I}}_j^{(PQ)} \quad (\rho \uparrow 1),$$

where $\tilde{\mathbf{I}}_j^{(PQ)}$ has a gamma-distribution with parameters

$$\alpha := 2s\varrho/\sigma^2 + 1 \text{ and } \mu_j := 2\varrho/((1-\rho_{T(j)})\hat{\pi}_j\sigma^2).$$

Further,

$$(1-\rho)W_j^{(PQ)} \rightarrow_d \tilde{W}_j^{(PQ)} \quad (\rho \uparrow 1),$$

where $W_j^{(PQ)}$ is the wait at pseudo-queue j,

$$\tilde{W}_j^{(PQ)} =_d U\tilde{\mathbf{I}}_j^{(PQ)},$$

and where U is a uniform[0,1] random variable that is independent of $\tilde{\mathbf{I}}_j^{(PQ)}$.

Now if the polling table is perfect then all intervisit and waiting times for a given queue will be identically distributed and we can suppress the pseudo-queue notation. Further, if there are m_i visits to queue i in a cycle then $\hat{\pi}_i = 1/m_i$. Also, simple algebra yields that,

$$\varrho = \frac{1}{2}\sum_{i=1}^{N} \frac{\rho_i(\rho-\rho_i)}{m_i}.$$

Therefore, the wait under any perfect polling table may be approximated by,

$$(1-\rho)W_i \approx_d U\tilde{\mathbf{I}}_i \tag{6}$$

where $\tilde{\mathbf{I}}_i$ has a gamma-distribution with parameters

$$\alpha := 2s\varrho/\sigma^2 + 1 \text{ and } \mu_i := 2\varrho m_i/((1-\rho_i)\sigma^2).$$

and where U is a uniform[0,1] random variable that is independent of $\tilde{\mathbf{I}}_i$.

If a perfect polling table generates long-run average visit frequencies for the queues of $\{\hat{n}_i : 1 \leq i \leq N\}$ and if the expected cycle length under this table equals c, then $m_i = \hat{n}_i c$ and $s = c(1-\rho)$. Substituting this in the above we have that the distribution of wait may be approximated as in (6) where $\tilde{\mathbf{I}}_i$ has a gamma-distribution with parameters

$$\hat{\alpha} := 2\hat{\varrho}(1-\rho)/\sigma^2 + 1 \text{ and } \hat{\mu}_i := 2\hat{\varrho}\hat{n}_i/((1-\rho_i)\sigma^2)$$

and where

$$\hat{\varrho} = \frac{1}{2}\sum_{i=1}^{N} \frac{\rho_i(\rho - \rho_i)}{\hat{n}_i}.$$

The above suggests that a lower bound on the expected cost per unit time, that should become asymptotically bounding in heavy-traffic, is

$$\begin{aligned} & \sum_{i=1}^{N} \left(\frac{\tilde{c}_i \lambda_i \hat{\alpha}}{2(1-\rho)\hat{\mu}_i} + k_i \hat{n}_i \right) \\ = & \sum_{i=1}^{N} \left(\frac{\tilde{c}_i \lambda_i (2\hat{\varrho}(1-\rho)/\sigma^2 + 1)}{2(1-\rho)2\hat{\varrho}\hat{n}_i/((1-\rho_i)\sigma^2)} + k_i \hat{n}_i \right) \\ = & \sum_{i=1}^{N} \left(\frac{c_i \rho_i (1-\rho_i)(1 + \sigma^2/2\hat{\varrho}(1-\rho))}{2\hat{n}_i} + k_i \hat{n}_i \right) \end{aligned} \tag{7}$$

If $\sigma^2 = 0$ (as in a deterministic system) then this is the same cost function as in (1).

For any given set of visit frequencies we have a heavy-traffic performance bound that will be increasingly tight the closer one can get to a perfect polling table with the given frequencies. For example, if all queues and visit frequencies are identical then the bound is achievable in heavy-traffic. Further, if we minimize (7) subject to $\sum_{i=1}^{N} s_i \hat{n}_i = 1 - \rho$ then we have a lower bound over all dynamic policies that satisfy the HTAP. Unfortunately, as $\hat{\varrho}$ depends on the visit frequencies, this optimization is a non-trivial non-convex optimization.

One question that immediately arises is: how large is the class of policies that satisfy the HTAP? Jennings [37] hypothesizes, but does not prove, that his two-queue dynamic scheduling heuristic satisfies the HTAP. Even for cyclic systems with more than two queues the HTAP is just hypothesized to hold in CPR. For systems with polling tables OVDM present a strong hypothesis that the HTAP holds, but the result is not proven. Classifying such policies is left as an open research question. However, one reasonable hypothesis is that the class of systems is the same as the (unknown) class of systems where a strong-law limit of the system converges to a unique fluid system. In this case, the fluid lower bound and the heavy-traffic lower bounds would both hold asymptotically (as $\rho \to 1$) for the same set of dynamic scheduling policies.

5 Scheduling Heuristics

The fluid bounds of Section 4.1 yield values for n_i, how frequently queue i should be visited, as well as the optimal (in the ideal fluid system) maximum workload that should be seen by queue

i, $1 \leq i \leq N$. These values lead to heuristic scheduling algorithms, which are presented here. This section is organized as follows. Three heuristics are presented in Sections 5.1 to 5.3. The first heuristic is dynamic and is the primary one for the paper, whereas the heuristics in Sections 5.2 and 5.3 are static and are provided primarily for benchmarking purposes. All heuristics are appropriate for general arrival and service processes as they depend only on the first moments of the system.

5.1 Heuristic 1

Perkins and Kumar [65] propose a heuristic for a fluid system where setup times are identical and setup costs and cruising are not allowed. Under their heuristic, each time a queue has been exhausted, the server checks the workload that is currently accumulated in each queue. After adding the workload that will accumulate during setup, this quantity is then divided by the optimal maximal workload that should accumulate in each queue and the next queue served is the one with the greatest of these values.

Although this heuristic was originally designed for fluid systems, we find it attractive and useful for stochastic systems as well. Further, the results in Section 4.1 allow us to schedule a more general system where both setup times and setup costs co-exist and are allowed to be different for each queue. In addition, we also consider the implementation of cruising into our scheduling method. We propose a heuristic that depends on whether the cruising conditions are satisfied or not. The heuristic is easy to implement and requires few calculations.

Consider the case where none of the queues satisfy the cruising conditions. From the proof of Lemma 1, the optimal maximal workload that should accumulate in each queue is:

$$v_i = \frac{\sqrt{2\rho_i(1-\rho_i)(\beta s_i + k_i)}}{\sqrt{c_i}}, 1 \leq i \leq N. \tag{8}$$

Let t be the current time at a decision epoch. The index of the queue to be served next is:

$$i^*(t) = \arg\max_i \left\{ \frac{(v_i(t) + \rho_i s_i)\sqrt{c_i}}{\sqrt{2\rho_i(1-\rho_i)(\beta s_i + k_i)}} \right\}. \tag{9}$$

It is possible with large setup times that the current queue has the highest ratio in (9) (even though there are no jobs in the current queue); in this case, the switch will be made to the queue with the second highest ratio.

If the cruising conditions hold for some queues then we slightly modify our scheduling method. The implementation of cruising is extremely important when setup times are negligible. Recall that in a system with discrete parts "cruising" corresponds to serving orders in the current queue as they arrive. From the proof of Lemma 1, we obtain the optimal maximal accumulated workload, which equals

$$v_i = \frac{\sqrt{2\rho_i(1-\rho_i)(\delta^* s_i + k_i)}}{\sqrt{c_i}}, 1 \leq i \leq N. \tag{10}$$

At each decision epoch, if the workload at any queue exceeds the optimal maximal workload (when the expected workload to arrive during setups is included) then the machine will switch to the queue with the highest ratio of accumulated workload divided by optimal maximal workload level; otherwise, the machine will stay at the current queue until some queue meets this criteria. Although, from Theorem 1 for the fluid system, only cruising at queues that satisfy both cruising conditions leads to optimality, from simulation experiments we found that, for stochastic systems, allowing cruising at all queues generally yields a better result in terms of the average production cost. In stochastic systems the workload is not deterministic, as it is in the fluid system. Therefore, by allowing cruising in all queues, we take advantage of times when the workload levels in all other queues are lower than they should be on average. In addition, the heuristic manages to maintain the maximal workload level in each queue close to its optimal maximal level. The formula in (11) states the rule,

$$(i^*, t^*) = \arg\min_{t'} \arg\max_{i} \left\{ \frac{(v_i(t') + \rho_i s_i)\sqrt{c_i}}{\sqrt{2\rho_i(1-\rho_i)(\delta^* s_i + k_i)}} \geq f \right\}. \tag{11}$$

where t^* is the point of time that the machine switches to queue i^*; of course, (i^*, t^*) cannot be predicted ahead of time in a stochastic system and therefore must be reevaluated whenever the system changes state.

Note that instead of using one in the right hand side of (11), we allow a factor f between zero and one. If $f = 0$ then no cruising is implemented, whereas $f = 1$ corresponds to cruising as described above. Numerical experiments indicate that, when there are only a few queues in the system (say fewer than six) and the utilization rate is high (say above 0.8), the machine tends to cruise too long when f is set to one. An f of around 0.7 seemed to perform best for the few tests we performed, but this is rather an arbitrary number. In practice a manager may well use his or her best judgment, based on workloads in all the queues, on when to call an end to cruising.

In general, whether cruising should be implemented in a system depends on whether the cruising conditions are satisfied or not. However, for stochastic systems with light traffic, we find that implementing cruising tends to performs slightly better than not even when none of the queues satisfy the cruising conditions. Therefore, it maybe desirable to implement cruising in systems with light traffic.

The scheduling rules given in (9) and (11) are for the most general cases. These rules can be further simplified when we consider systems with only setup costs or setup times. In particular, when there are no setup times in the system and if $f = 1$, the term inside the brackets in (11) can be re-written as $\frac{c_i v_i(t)^2}{2\rho_i(1-\rho_i)} \geq k_i$ which means that the average backlogging cost must be greater than or equal to average setup costs if a switch to queue i at time t is made. This, in general, will ensure that the average backlogging cost is close to the average setup cost. Recall that, from the results of the fluid model for systems without setup times, the lower bound is achieved when average setup cost is equal to average backlogging cost.

The heuristic is guaranteed to result in a stable system if implemented in a fluid environment. This is shown in the following theorem. Let $\bar{s} = \max_i s_i$, $\bar{v} = \max_i v_i$, $\underline{v} = \min_i v_i$, $\underline{\rho} = \min_i \rho_i$ and $\bar{h} = \max_i \left\{\frac{v_i f}{\rho_i}\right\}$, where v_i is obtained from (10). Note that $\frac{v_i f}{\rho_i}$ is the maximal length of time for queue i to accumulate workload that is greater than the switching threshold with the factor f given in (11). Therefore, $\bar{h}$ is an upper bound on the cruising period.

Theorem 3 *The proposed heuristic is stable for the fluid system and the system workload at any time is bounded above by*

$$\begin{aligned} \textit{No Cruising} &: U(0) + \rho\bar{s}\frac{2-\epsilon}{1-\epsilon}, \\ \textit{With Cruising} &: U(0) + [\rho\bar{s} + (\rho - \underline{\rho}), \bar{h}]\frac{2-\epsilon}{1-\epsilon} \end{aligned}$$

where $\epsilon = \frac{\underline{v}(1-\rho)}{\bar{v}(N-1)(1-\underline{\rho})}$ *and* $U(0)$ *is the initial system workload.*

Proof: The proof is primarily algebraic in nature and is relegated to the Online Appendix.

Note that, as discussed in Section 2, stability of the fluid system frequently implies stability in the corresponding stochastic system. Therefore, the above result strongly suggests that, for $\rho < 1$, the heuristic is stable in the stochastic setting. A rigorous proof however is non-trivial and left as the subject of future research.

5.2 Heuristic 2

The only work we are aware of that combines both setup times and costs in stochastic dynamic (queueing) systems is Federgruen and Katalan [28] (FK). This work is designed for systems with inventory but by setting holding costs to be a very large number in their system we may use it to design polling tables for systems with only backlogging. Given that it is not explicitly designed for such systems, poor performance in these cases should not be extrapolated to poor performance for make-to-stock systems.

In FK, polling tables for periodic service are found using a three phase approach. First the relative frequencies for production of the different items are found, second a polling table size is chosen and the relative frequencies are translated into absolute frequencies, and last the items are sequenced in the table. The same three phase approach was also used in Boxma et al. [12] and [13]. In our numerical study we will benchmark our results against the tables produced by the FK algorithm. In addition, we use our n_i in the algorithm (instead of those found by FK) and refer to this as Heuristic 2.

5.3 Heuristic 3

For all cases we tested, when Heuristic 1 is implemented in a deterministic system, it converges to a repeating cycle (in less than a second of computational time). This cycle can then be read off as a polling table that has visit frequencies approximately equal to the desired n_i. Thus our final suggested heuristic is a polling table policy utilizing the cycle found this way. Note that, proving that this algorithm will always converge is non-trivial and left as the subject of future research.

5.4 Numerical Study

In this section we test our three heuristics. We compare them to each other, to the lower bound, and to other heuristics from the literature. The only other truly dynamic heuristics for stochastic systems like those in this paper that we are aware of are the papers by Duenyas and Van Oyen [23] and [24]. Paper [23] presents a heuristic (DVO1) for systems with setup costs only and paper [24] presents a heuristic (DVO2) for systems with setup times only. They do not present a heuristic for systems with both setups costs and times. The only heuristic we are aware of for systems with both setup costs and times is the static heuristic of Federgruen and Katalan (FK) that was described

above and is designed for inventory systems. Below we compare our heuristics with DVO1, DVO2, and FK. Note that DVO1 and DVO2 are not exhaustive service heuristics, whereas FK is.

5.4.1 Setup Costs Only

We tested our heuristic (with cruising) against DVO1 using the data in their paper. Detailed results from these tests may be found in Lan [43]; below we outline the main findings.

The first instances in their paper are for two queues, where our service order is cyclic serve-to-exhaustion (CSTE) and is therefore not worth simulating (CSTE is tested in their paper). For the data in Table 2 of [23], which are systems consisting of three queues, our heuristic is on average 1.6% higher than DVO1; which is 3.1% suboptimality (for systems of three or fewer queues their paper finds the optimal cost). The heuristic ranges from underperforming DVO1 by 7.4% to outperforming it by 2.8%. Note that their work also shows poor performance of CSTE, cyclic gated service, and cμ rules, all of which are defined in their paper.

We also tested our heuristics versus the four queue examples of [23] (instances 41 - 92 in their paper). The results suggest that heuristic DVO1 clearly outperforms our heuristic if the setup costs are small. This is not surprising, since without the setup costs their heuristic is the $c\mu$ rule which is known to be optimal under the scenario. However, on average there is little performance difference between DVO1 and our heuristic, as our heuristic yields a 1.7% higher average production costs than DVO1 for these instances. In fact, among these 52 instances, our heuristic yields a lower average production cost in 29 instances. Both DVO1 and our heuristic clearly outperform all other tested heuristics in almost all cases.

5.4.2 Setup Times Only

We now test heuristics designed for systems with setup times only. We first perform tests against the data in Duenyas and Van Oyen [24]. Detailed results from these tests may be found in Lan [43]. We test all examples from their paper except the two queue examples. We first test instances 15 - 68 (the three queue examples). Generally speaking, DVO2 has the best performance. Our heuristic, which performs better than the other benchmark heuristics of the paper, yields a 15% higher average production cost than DVO2 on average. The percentage gaps between HEUR1 and DVO2 are especially large when all the setup times are very small (e.g., the instances where

setup times all equal 0.1). Under this scenario, even the $c\mu$ rule outperforms HEUR1. However, when setup times are increased, the gaps between HEUR1 and DVO2 are reduced and the $c\mu$ rule becomes unstable. Moreover, if we eliminate instances with setup times all equal to 0.1, the average percentage difference reduces to 9%.

One of the main reasons that DVO2 outperforms HEUR1 for these instances is that they are small systems. Such small systems makes our scheduling heuristic less flexible. In other words, to space the queue evenly according to the optimal frequencies for each queue is more difficult with fewer queues. Indeed, to construct a polling table based on a specific visit frequencies for such small systems is usually impossible, which also causes the simulation results for the heuristic to be far away from the fluid bounds. In most cases, the fluid bounds are unachievable.

The next 7 instances are also taken from Duenyas and Van Oyen [24], in this case they correspond to examples 69 - 75. We also test FK on these systems. These seven instances all have 6 queues in the system with backlogging costs per part equaling 5, 2, 0.5, 0.4, 0.3, 0.2 for queues 1 to 6 respectively. The processing rates equal 1 for all the queues while the arrival rates equal 0.2 for queues 1 and 2 and 0.1 for all other queues (so $\rho = 0.8$). Setup time varies from 0.1 to 5.0 with all but the 2nd test having at least one queue with setup time of 0.1 (the second test has setup equal to 1.0 at all queues). The specific data may be found in [24]. Table 2 gives the results of the simulation tests. The first column gives the instance number and the second column gives the simulated mean cost under HEUR1. Columns three to eight give the percentage difference between HEUR1 and DVO2, FK, HEUR2, HEUR3, the fluid bound, and the heavy-traffic bound, respectively.

From the results in Table 2, the performance difference between DVO2 and HEUR1 is quite small. The only instance where DVO2 significantly outperforms HEUR1 is the first instance, which has $s_i = 0.1$ for all the queues. This is consistent with the same trend that we have seen from previous results. Nevertheless, our heuristic actually slightly outperforms heuristic DVO2 in most of the other instances. The FK heuristic does not perform well. However, in all cases the FK heuristic produced a table of length 6 which corresponds to cyclic service; this was not the case in later examples. Note that the bounds are quite imprecise particularly for examples 4-7. In these four examples, the queues with large backlogging costs have small setup times and those with small backlogging costs have large setup times.

	HEUR1	DVO2	FK	HEUR2	HEUR3	FB	HT
1	6.2	-12.9%	14.6%	-32.6%	3.7%	-80.6%	-11.2%
2	16.3	1.2%	47.6%	-5.9%	30.0%	-28.2%	-1.8%
3	30.4	4.9%	20.0%	10.0%	26.0%	-23.4%	1.8%
4	12.8	0.8%	181.3%	111.2%	33.9%	-63.3%	-45.9%
5	14.6	1.4%	180.9%	-6.1%	17.8%	-73.3%	-58.9%
6	15.5	0.0%	163.6%	-9.9%	15.5%	-65.8%	-46.8%
7	23.8	8.0%	204.5%	-1.9%	15.4%	-56.7%	-47.8%
average		0.5%	116.1%	9.3%	20.3%	-55.9%	-30.1%

Table 2: Simulation Results for DVO2 Examples 69-75.

Up to now, all the systems that we have been tested have been rather small. We now test our heuristics using larger systems. The next 12 ten-queue instances are created by us. The data for these instances may be found in Table 4.10 in Lan [43]. Backlogging costs for these 10 queues range from 1 to 10; the individual queue utilization rates range from 0.01 to 0.3 with mean processing times ranging from 0.1 to 1; setup times range from 5 to 40 in these instances. Tests are done for system utilization rates equal to 0.6 and 0.9. Table 3 shows the results of the simulations. The columns are as in Table 2. Detailed numbers are reported for $\rho = 0.9$ and averages are reported for $\rho = 0.6$.

It can be seen the HEUR1 significantly outperforms all other tested heuristics. The static heuristic HEUR3 performs well in the last four instances but very poorly in instance three, suggesting variable performance. The average polling table lengths over the twelve instances were 55.2, 70.2, and 441.6, for heuristics FK, HEUR2, and HEUR3, respectively. The significantly greater length of HEUR3's table probably explains its better performance for most instances. The high variability in the polling table results, compared with a relatively low variability in HEUR1's performance as compared to the fluid bound, is clearly an advantage of dynamic heuristics over static polling table heuristics. The underperformance of DVO2 suggests that while it is an attractive heuristic for small queue systems, particularly those with small setup times, it may not be as attractive for larger systems. HEUR1 comes relatively close to the fluid bound and even closer to the heavy-traffic bound (which was formed using the n_i from the fluid lower bound). Indeed, the heavy-traffic bound

	HEUR1	DVO2	FK	HEUR2	HEUR3	FB	HT
1	3113.2	4.43%	57.29%	8.09%	5.69%	-5.38 %	-3.96%
2	1123.1	37.72%	49.17%	22.39%	12.27%	-3.36 %	-1.29%
3	1235.1	50.37%	15.43%	22.22%	121.15%	-7.44%	-5.04%
4	1290.1	21.39%	24.42%	12.15%	15.12%	-6.07 %	-4.58%
5	5806.9	8.67%	66.43%	22.94%	10.35%	-4.85 %	-3.60%
6	1811.1	56.87%	72.56%	10.30%	25.22%	-1.80 %	-0.06%
7	2105.5	40.56%	16.08%	17.21%	29.73%	-9.60 %	-7.19%
8	1731.4	44.89%	30.19%	18.18%	35.60%	-6.35 %	-4.96%
9	4758.1	4.54%	64.94%	12.58%	-0.58%	-5.35 %	-4.79%
10	1524.3	27.41%	74.75%	64.69%	-0.21%	-4.73 %	-3.05%
11	1754.5	57.78%	18.82%	8.15%	0.18%	-7.86 %	-6.89%
12	1677.2	18.04%	85.71%	21.42%	-2.63%	-9.31 %	-7.68%
Average	2327.5	31.06%	47.98%	20.03%	20.99%	-6.01%	-4.42%
Avg. $\rho = 0.6$	417.3	21.77%	59.39%	20.59%	16.94%	-7.14%	-4.77%

Table 3: Simulation Results for Ten-Queue Examples

may be an attractive performance approximation for systems operated using HEUR1.

5.4.3 Setup Costs and Times

The following tests are described fully in Lan [43]. They are based on a data set taken from Olsen [59], which comes originally from a fence-making operation. It consists of 19 different products, with an average service time of 0.1 hours and a range from 0.02 to 0.27. The average mean setup time over all products is 2 hours with a range from 0.17 to 11.3. The arrival rates at the different queues are very diverse and range from 0.17% to 29.9%. There are no setup costs in the original data set. However, for the sake of more thorough testing, we assign setup costs which are randomly generated from a uniform distribution on [0, 80] for each queue. The whole data set is given in Lan [43]. We take all distributions to be exponential, although in practice we found that the distributions do not make much difference to the observed trends.

The full search technique for the FK heuristic cannot be performed on so large a data set so instead we run the Normal, Scarf, and Limited Search heuristics as suggested in [28] and then

pick the best as the benchmark to use. Table 4 gives the results of the simulation tests. The first column gives instance number, the second the value of ρ (arrival rates are scaled accordingly), and the third the simulated mean cost under HEUR1. Columns four to eight give the percentage difference between HEUR1 and FK, HEUR2, HEUR3, the fluid bound, and the heavy-traffic bound, respectively.

	ρ	HEUR1	FK	HEUR2	HEUR3	FB	HT
1	0.5	154.5	12.8%	55.2%	-9.2%	-16.4%	-12.9%
2	0.7	301.5	33.8%	30.0%	1.1%	-12.0%	-9.7%
3	0.9	1055.8	69.7%	16.7%	1.6%	-8.4%	-6.9%
average			38.7%	34.0%	-2.2%	-12.3%	-9.9%

Table 4: Simulation Results for 19 Queue System

It can be seen that for these examples HEUR1 significantly outperforms FK and HEUR2. Performance is close to that from HEUR3. The average table lengths across the heuristics are 188, 175, and 346. One must question how practical table lengths of these magnitudes would be.

Finally by using scale factors, 0, 1, 2, and 8 to scale the original s_i and k_i, we generated 15 scenarios based on the original data set used in Table 4. Each scenario is tested using ρ equal to 0.5, 0.7, and 0.9 (arrival rates are scaled accordingly). The heuristic is simulated for both a fluid system and for the stochastic system. Table 5 reports the results. The percentage differences between the simulation results and the fluid bound are reported in the columns labeled *%diff1*. In addition, the percentage differences between the simulation results under the stochastic and fluid settings are reported in the columns labeled *%diff2*. This allows for insight into what part of the distance from the lower bound is caused by the heuristic not being able to achieve the lower bound schedule, and what part of the distance is caused by the stochastic setting. Finally the columns labeled *%diff3* compare HEUR1 to the heavy-traffic bound.

The percentage different between the simulation results and the fluid bound ranges from -2.7% to 23.6%. As predicted in Section 2, the percentage difference between the fluid bound (and heavy-traffic bound) and the stochastic system decreases as system utilization increases. Unlike in Section 4.3, the bound does not generally become tighter as setups increase. This is likely because as setups increase the system in a sense becomes "lumpier" and scheduling the queues in line with

	$\rho = 0.5$			$\rho = 0.7$			$\rho = 0.9$		
	%dif1	*%dif2*	*%dif3*	*%dif1*	*%dif2*	*%dif3*	*%dif1*	*%dif2*	*%dif3*
$(k_i,\ 0)$	2.4	13.9	n/a	2.7	6.9	n/a	1.5	3.4	n/a
$(2k_i,\ 0)$	-0.8	11.1	n/a	0.5	4.8	n/a	-0.1	0.0	n/a
$(8k_i,\ 0)$	-4.6	7.6	n/a	-2.6	1.8	n/a	-1.0	-0.9	n/a
$(0,\ s_i)$	-15.9	-5.8	-12.4	-11.5	3.4	-9.3	-9.9	-4.2	-8.3
$(k_i,\ s_i)$	-16.4	-8.7	-12.9	-12.0	1.3	-9.7	-8.4	-2.7	-6.9
$(2k_i,\ s_i)$	-11.5	-5.2	-9.3	-11.7	0.5	-9.5	-9.6	-3.9	-8.0
$(8k_i,\ s_i)$	-7.8	-5.0	-6.6	-7.1	-2.7	-5.9	-7.7	-2.5	-6.4
$(0,\ 2s_i)$	-22.7	-12.0	-17.9	-13.3	-5.4	-11.2	-8.9	-3.2	-7.8
$(k_i,\ 2s_i)$	-19.3	-9.5	-15.5	-12.8	-5.2	-10.9	-9.4	-2.6	-7.3
$(2k_i,\ 2s_i)$	-18.5	-9.3	-15.0	-12.9	-5.3	-10.9	-8.3	-2.6	-7.3
$(8k_i,\ 2s_i)$	-16.0	-9.4	-13.3	-10.4	-3.8	-8.9	-7.9	-2.2	-6.9
$(0,\ 8s_i)$	-21.1	-10.5	-17.2	-12.8	-4.9	-11.2	-10.3	-4.5	-9.3
$(k_i,\ 8s_i)$	-23.3	-12.6	-18.8	-15.1	-6.9	-13.0	-11.3	-5.5	-10.1
$(2k_i,\ 8s_i)$	-23.6	-13.0	-18.9	-12.8	-4.9	-11.2	-9.3	-3.5	-8.4
$(8k_i,\ 8s_i)$	-23.1	-12.8	-18.6	-14.5	-6.6	-12.5	-8.7	-2.9	-7.9
average	-14.8	-5.4	-14.7	-9.8	-1.8	-10.4	-7.3	-2.5	-7.9

Table 5: Simulation Results for 19 Queue System with Scaled Setup Costs and Times

the desired frequencies becomes more difficult.

The negative numbers on the table show that, as explained in Section 4.1, the lower bound for the fluid system is not guaranteed to be a lower bound for general stochastic systems. Indeed for systems without setups the stochastic system can significantly outperform a deterministic fluid system. This indicates that, while the fluid bound may be a useful approximation for system performance, it cannot be treated as a strict lower bound. Unfortunately, it also indicates that finding tight closed-form lower bounds for system performance under dynamic schedules may be very difficult. The achievable region approach of Bertsimas and Niño-Mora [6] is probably the most promising approach available.

5.4.4 Conclusions From Numerical Study

We have presented three heuristics with Heuristic 1 being the most appealing. The other two were provided primarily as benchmarks. Heuristic 1 is clearly competitive with other heuristics, particularly for systems with a moderate to large number of queues. It is also simpler to implement than the other dynamic algorithms benchmarked. It is easily implementable on a spreadsheet and requires no advanced programming (unlike the static heuristics tested). It requires only first moment information and is practical for very general distributions. We have also presented a relatively accurate heavy-traffic bound that works well as a performance approximation; this bound also assumes general distributions and only requires the additional information of the variance of the interarrival and service times, respectively. The only real-time informational requirements are the current backlog of orders in each queue.

We envision a manager having a spreadsheet that states the maximum desired workload in each queue, the current workload, and the ratio of the two. Unlike in a static system, if for some reason a queue is chosen because the company has agreed to rush that order, or a queue is not exhausted for some reason, then the algorithm is not thrown off. Indeed the manager may choose a queue that is slightly less backlogged than that with the maximum ratio because of qualitative factors.

This work will also help managers make decisions on overtime. This is a difficult decision for managers as they will always face a backlog, the question is when is that backlog "too big." With this work, they have a benchmark for average work in the fluid system (simply by adding up the column of maximum workloads and dividing by two for systems without cruising). Therefore they

may calculate at any time how far they are ahead or behind this benchmark. Further, the amount they are behind represents how long it would take without setups and ignoring new arrivals to get back to the ideal workload. Experience may dictate what an acceptable amount "in the hole" to be would be, or they may decide to run overtime whenever there is a full shift's worth of work to complete. It would also help make the converse decision of when to shut down for maintenance by showing how far ahead of the benchmark they are.

6 Conclusions and Future Research

In this paper we provided a closed-form lower bound on the performance of a fluid system. We also showed that this bound applies to a fairly broad class of stochastic systems. The bound is not very tight for small systems but becomes increasingly tight as the number of queues increases or as traffic increases.

Extensions of the lower bound to make-to-stock systems and systems with sequence dependent setups are possible and may be found in Lan [43]. In the case of make-to-stock systems the lower bound is also closed-form and contains terms to account for holding as well as backlogging costs. For sequence-dependent setups the bound is only found in closed form for the two-queue case and otherwise requires the solution of a convex programming problem. Such a convex programming problem is similar to that in Bertsimas and Xu [8] once setup costs and cruising have been added. It is likely that systems with sequence-dependent setups will perform better under static rather than dynamic schedules. Developing such schedules is left as the subject of future research.

We also obtained heavy-traffic bounds for the system. These bounds are attractive as performance approximations for systems with a moderate number of queues in moderate to heavy traffic. The method of obtaining these bounds is similar in spirit to the work of Markowitz et al. [49] and Markowitz and Wein [50]. While the cyclic schedules of these papers were not appropriate for the make-to-order systems of this paper, the use of Coffman et al. [18]'s HTAP to find inventory levels suggests a natural way to extend the work here to make-to-stock systems. This is left as the subject of future research.

Finally, we used the fluid bound to derive dynamic and static scheduling heuristics and tested the heuristics via a numerical study. The dynamic heuristic was found to perform very well against

the benchmark heuristics and quite well against the lower bound, particularly for moderate to large systems. The static heuristics were sometime quite competitive and other times not, suggesting that their performance is very sensitive to good polling table design. Further study of effective polling table design is left as the subject of future research.

Acknowledgments

We are grateful to Mike Harrison, Shane Henderson, and Peter Glynn for discussions on this topic. We would also like to thank Wiremu Solomon for suggesting the term "cruising". This work was supported in part by NSF CAREER grant DMI-0196513. Zhenxin (Dennis) Yu performed some of the numerical tests for us.

Appendix

Proof of Theorem 1:

To prove our theorem we need to find the optimal solution for the model in Lemma 1. Observe that the problem in Lemma 1 is convex and hence it suffices to solve the KKT conditions (see, e.g., [2]). The KKT conditions for this model are (2) and

$$\begin{aligned} \alpha_i + \mu s_i + k_i &= \frac{(1-d_i)^2 w_i}{2n_i^2} \qquad \forall\ i \\ \theta_i + \mu(1-\rho_i) &= \frac{w_i(1-d_i)}{n_i} \qquad \forall\ i \\ \alpha_i n_i = 0, &\qquad \theta_i d_i = 0 \qquad \forall\ i \end{aligned}$$

where $\mu > 0$ and $\alpha_i, \theta_i \leq 0$ for all i. Note that a feasible solution requires that $n_i > 0$, therefore $\alpha_i = 0$, $1 \leq i \leq N$.

Define a candidate set I as a set of queues such that, $\forall\ i \in I$, queue i satisfies both the cruising conditions. Let $\bar{I}$ represent all the queues that are not in I.

CASE 1:I is not empty

From the first cruising condition, it is clear that if both i and $i' \in$ I, δ_i equals $\delta_{i'}$. Let i' be an arbitrary queue in I. The solution to the KKT conditions are $\mu = \delta_{i'}$, for all $j \in \bar{I}$, $d_j = 0$,

$$\theta_j = (1-\rho_j)\left[\sqrt{\frac{2(\delta_{i'} s_j + k_j)c_j\rho_j}{1-\rho_j}} - \delta_{i'}\right]$$

and

$$n_j = \sqrt{\frac{w_j}{2(\delta_{i'} s_j + k_j)}}. \tag{12}$$

Note that satisfying the first cruising condition, $\delta_i' > \delta_j$, implies that $\theta_j \leq 0$. In addition, for all $i \in I$, $\theta_i = 0$,

$$n_i = (1-d_i)\frac{c_i\rho_i}{\delta_i}, \tag{13}$$

and the d_i solve

$$\sum_{i\in I} \frac{s_i c_i \rho_i}{\delta_i} + \sum_{j\in\bar{I}} n_j s_j + \sum_{i\in I}\left[(1-\rho_i) - \frac{s_i c_i \rho_i}{\delta_i}\right] d_i = (1-\rho). \tag{14}$$

With a little algebra, it can be shown that for all $i \in\ [1, N]$,

$$\delta_i = \sqrt{\frac{2(\delta_i s_i + k_i)c_i\rho_i}{1-\rho_i}}. \tag{15}$$

By (12), (15), and the second cruising condition $\sum_{i\in I}\left[(1-\rho_i)-\frac{s_i c_i \rho_i}{\delta_i}\right] d_i > 0$. Observe that $\delta_i > \frac{2s_i c_i \rho_i}{1-\rho_i}$; therefore $(1-\rho_i) > \frac{2s_i c_i \rho_i}{\delta_i}$. It follows that $\sum_{i\in I} d_i > 0$ and hence cruising is necessary in this case. Unlike the result in [17], which suggests that only one queue could satisfy both the cruising conditions, it is possible to have more than one queue satisfy both the cruising conditions here. Note that the solution for n_i and d_i is not unique when I contains more than one queue and different queues may be chosen to cruise. To derive the lower bound as in the statement of this theorem requires significant algebra. For a detailed proof, see the Online Appendix.

CASE 2:I is empty

Suppose that queue i satisfies only the second cruising condition; therefore, there exists an $i' \in [1, N]$ that satisfies the first cruising condition, such that $\delta_{i'} > \delta_i$ and hence

$$\sum_{j=1}^{N} s_j \sqrt{\frac{w_j}{2(\delta_{i'} s_j + k_j)}} < \sum_{j=1}^{N} s_j \sqrt{\frac{w_j}{2(\delta_i s_j + k_j)}} < (1-\rho).$$

It follows that i' satisfies the second cruising condition too and I is not empty. Hence, if I is empty, then none of the queues satisfy the second cruising condition.

The solutions for the KKT conditions when I is empty are $\mu = \beta$, where β is defined in the statement of this theorem. In addition, for all $j \in [1, N]$, $d_j = 0$,
$\theta_j = (1-\rho_j)\left[\sqrt{\frac{2(\beta s_j + k_j) c_j \rho_j}{1-\rho_j}} - \beta\right]$, and

$$n_j = \sqrt{\frac{w_j}{2(\beta s_j + k_j)}}. \tag{16}$$

We will use the second cruising condition to show that, if none of the queues satisfies the second cruising condition, such a β will imply $\theta_j \leq 0$ and the KKT conditions are satisfied. Let $\delta^* = \max_{j\in[1,N]} \delta_j$. Because none of the queues satisfy the second cruising condition, $\sum_{j=1}^{N} s_j \sqrt{\frac{w_j}{2(\delta^* s_j + k_j)}} \geq (1-\rho)$. Since $\sum_{j=1}^{N} s_j \sqrt{\frac{w_j}{2(\beta s_j + k_j)}} = (1-\rho)$, it follows that $\beta \geq \delta^* \geq \delta_j$. With some algebra, we can show that this implies that $\theta_j \leq 0$. It follows that if none of the queues satisfy both the cruising conditions, then

$$AC = \sum_{j=1}^{N} \sqrt{\frac{w_j}{2}} \left[\frac{k_j}{\sqrt{\beta s_j + k_j}} + \sqrt{\beta s_j + k_j}\right]. \tag{17}$$

This completes the proof of the theorem. □

References

[1] K. R. Baker. Heuristic procedures for scheduling job families with setups and due dates *Naval Research Logistics*, 46 (8): 978-991, December 1999.

[2] M. S. Bazaraa, H. D. Sherali and C. M. Shetty. *Nonlinear Programming Theory and Algorithms.* New York: John Wiley & Sons, 1993.

[3] S. Benjaafar and D. Gupta. Workload allocation in multi-product, multi-facility production systems with setup times. *IIE Transactions,* 31 (4): 339-352, April 1999.

[4] D. Bertsimas, D. Gamarnik, and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the holding cost objective. *Operations Research,* 51 (5): 798-813, September–October 2003.

[5] D. Bertsimas and G. Mourtzinou. Decomposition results for general polling systems and their applications. *Queueing Systems*, 31 (3-4): 295–316, 1999.

[6] D. Bertsimas and J. Niño-Mora. Optimization of Multiclass Queueing Networks with Changeover Times Via the Achievable Region Approach: Part 1, the Single-Station Case. *Mathematics of Operation Research*, 24 (2): 306-330, May 1999.

[7] D. Bertsimas and J. Niño-Mora. Optimization of Multiclass Queueing Networks with Changeover Times Via the Achievable Region Approach: Part II, the Multi-Station Case. *Mathematics of Operation Research*, 24 (2): 331–361, May 1999.

[8] D. Bertsimas and H. Xu. Optimization of Polling Systems and Dynamic Vehicle Routing Problems on Networks. Working Paper, Operations Research Center, MIT, 1993.

[9] J. P. C. Blanc and R. D. van der Mei. Optimization of polling systems with Bernoulli schedules. *Performance Evaluation,* 22 (2): 139-158, April 1995.

[10] S. C. Borst, O. J. Boxma, J. H. A. Harink, and G. B. Huitema. Optimization of fixed time polling schemes. *Telecommunication Systems,* 3 (1): 31-59, 1994.

[11] O. J. Boxma and D. G. Down. Dynamic server assignment in a two-queue model. *European Journal of Operational Research*, 103 (3): 595-609, December, 1997.

[12] O. J. Boxma, H. Levy, and J. A. Westrate. Efficient visit frequencies for polling tables: minimization of waiting cost. *Queueing Systems Theory and Applications*, 9 (1-2): 133–162, October 1991.

[13] O. J. Boxma, H. Levy and J. A. Westrate. Efficient Visit Orders for Polling Systems. *Performance Evaluation*, 18 (2): 103–123, 1993.

[14] S. Browne and U. Yechiali. Dynamic Priority Rules for Cyclic Type Queues. *Advances in Applied Probability*, 21: 432-450, 1989.

[15] S. Browne and U. Yechiali. Dynamic scheduling in single-server multiclass service systems with unit buffers. *Naval Research Logistics,* 38 (3): 383-396, June 1991.

[16] P. Carstensen. The economic lot scheduling problem - survey and LP-based method. *OR Spektrum,* 21 (4): 429-460, October 1999.

[17] C. Chase and P. J. Ramadge. On Real-time Scheduling Policies for Flexible Manufacturing Systems. *IEEE Transactions on Automatic Control*, 37 (4): 491–496, April 1992.

[18] E. Coffman, Jr., A. Puhalskii, and M. Reiman. Polling systems in heavy-traffic: A Bessel process limit. *Mathematics of Operations Research*, 23 (2): 257–304, 1998.

[19] J. G. Dai. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Annals of Applied Probability*, 5: 49-77, 1995.

[20] J. G. Dai and O. B. Jennings. Stabilizing Queueing Networks with Setups. Working Paper, Fuqua School of Business, Duke University, Durham, NC, 2002.

[21] J. G. Dai and G. Weiss. A fluid heuristic for minimizing makespan in job shops. *Operations Research*, 50 (4): 692-707, July-August, 2002.

[22] A. Drexl and A. Kimms. Lot sizing and scheduling - Survey and extensions. *European Journal of Operational Research*, 99:221-235, 1997.

[23] I. Duenyas and M. P. Van Oyen. Stochastic Scheduling of Parallel Queues with Setup Costs. *Queueung Systems*, 19: 421-444, 1995.

[24] I. Duenyas and M. P. Van Oyen. Heuristic Scheduling of Parallel Heterogeneous Queues with Setups. *Management Science*, 42: 814-829, 1996.

[25] I. Duenyas, D. Gupta, and T. L. Olsen. Control of a single-server tandem queueing system with setups. *Operations Research,* 46 (2): 218-230, March-April 1998.

[26] S. E. Elmaghraby. The economic lot scheduling problem (ELSP): review and extensions. *Management Science*, 24 (6): 587–598, February 1978.

[27] A. Federgruen, G. Guillermo and Z. Katalan. The Effect of Product Variety on Manufacturing Performance. Working paper, Columbia University, NY, June 1999.

[28] A. Federgruen and Z. Katalan. Determining Production Schedules Under Base-Stock Policies in Single Facility Multi-Item Production Systems. *Operations Research*, 46 (6): 883–898, 1998.

[29] J. C. Fransoo, V. Sridharan, and J. W. M. Bertrand. A hierarchical approach for capacity coordination in multiple products single-machine production systems with stationary stochastic demands. *European Journal of Operational Research,* 86 (1): 57-72, October 1995.

[30] S. W. Fuhrmann and R. B. Cooper. Stochastic decompositions in the $M/G/1$ queue with generalized vacations. *Operations Research*, 33 (5): 1117–1129, September-October 1985.

[31] A. D. Gandhi and C. G. Cassandras. Optimal control of polling models for transportation applications. *Mathematical and Computer Modelling,* 23 (11-12): 1-23, June 1996.

[32] J. Hasenbein. Scheduling Multiclass Queueing Networks via Fluid Models. *Tutorial given at INFORMS Conference,* San Jose, California, November 2002. Downloads available at http://www.me.utexas.edu/$\tilde{\text{h}}$as/Tutorial.html.

[33] T. J. Hodgson, G. Ge, R. E. King, and H. Said. Dynamic lot size/sequencing policies in a multi-product, single-machine system. *IIE Transactions*, 29 (2): 127–137, February 1997.

[34] M. Hofri and K. W. Ross. On the optimal control of two queues with server set-up times and its analysis. *SIAM Journal of Computing*, 16: 399–420, 1987.

[35] L. C. Hwang and C. J. Chang. Optimal-design of a finite-buffer polling network with mixed service discipline and general service order sequence. *IEE Proceedings - Communications,* 142 (1): 1-6, February 1995.

[36] S. M. R. Iravani, M. J. M. Posner, and J. A. Buzacott. A two-stage tandem queue attended by a moving server with holding and switching costs. *Queueing Systems,* 26 (3-4): 203-228, 1997.

[37] O. B. Jennings. *Generalized Round Robin Service Disciplines in Stochastic Networks with Setups: Stability Analysis and Diffusion Approximation.* Ph.D. Thesis, Georgia Institute of Technology, Atlanta GA, 1999.

[38] O. B. Jennings. Positive Harris Recurrence of Multiclass Queueing Networks with Setups. Working Paper, Fuqua School of Business, Duke University, Durham, NC, 2000.

[39] U. S. Karmarkar, S. Kekre, and S. Kekre. Multi-item batching heuristics for minimization of queuing delays. *European Journal of Operational Research*, 58 (1): 99–111, April 1992.

[40] E. Kim, M. P. Van Oyen, and M. Rieders. General Dynamic Programming Algorithms Applied to Polling Systems. *Communications in Statistics: Stochastic Models,* (14) :5, 1998.

[41] G. Koole. Assigning a single server to inhomogeneous queues with switching costs. *Theoretical Computer Science,* 182 (1-2): 203-216, August 1997.

[42] R. Kuik, M. Salomon and L. N. van Wassenhove. Batching decisions - structure and models. *European Journal of Operational Research*, 75 (2): 243–263, June 1994.

[43] W.-M. Lan. *Dynamic Scheduling of Multi-Product Systems: Bounds and Heuristics.* PhD dissertation, University of Michigan, 2000.

[44] W.-M. Lan and T. L. Olsen. "Heuristics for the Stochastic Scheduling of Multi-Product Systems with Both Setup Times and Costs", *Working Paper*, Washington University in St. Louis, 2001.

[45] H. Levy. Binomial-gated service - a method for effective operation and optimization of polling systems. *IEEE Transactions on Communications,* 39 (9): 1341-1350, September 1991.

[46] H. Levy and M. Sidi. Polling systems: Applications, modeling, and optimization. *IEEE Transactions on Communications*, 38 (10): 1750–1760, October 1990.

[47] Z. Liu, P. Nain, and D. Towsley. On optimal polling policies. *Queueing Systems Theory and Applications*, 11: 59–84, 1992.

[48] C. Maglaras. Discrete-review policies for scheduling stochastic networks: Trajectory tracking and fluid-scale asymptotic optimality. *Annals of Applied Probability,* 10 (3): 897-929, 2000.

[49] D. M. Markowitz, M. I. Reiman, and L. M. Wein. The stochastic economic lot scheduling problem: heavy traffic analysis of dynamic cyclic policies. *Operations Research*, 48 (1): 136-154, January-February 2000.

[50] D. M. Markowitz and L. M. Wein. Heavy traffic analysis of dynamic cyclic policies: A unified treatment of the single machine scheduling problem. *Operations Research*, 49 (2): 246-270, March-April 2001.

[51] M. A. Marsan, S. Donatelli, F. Neri, and U. Rubino. Dynamic polling orders in multiserver multiqueue systems. *IEE Proceedings-Communications,* 142 (2): 75-86, April 1995.

[52] S. P. Meyn. Sequencing and Routing in Multiclass Queueing Networks. Part I: Feedback Regulation. *SIAM Journal on Control and Optimization,* 40 (3): 741–776, 2001.

[53] S. P. Meyn. Sequencing and Routing in Multiclass Queueing Networks. Part II: Workload Relaxations. *SIAM Journal on Control and Optimization,* 42 (1): 178-217, 2003.

[54] S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability.* Springer-Verlag, London, 1993.

[55] S. Nahmias. *Production and Operations Analysis.* Irwin Inc, Homewood, IL, Second edition, 1993.

[56] R. D. Nobel. Optimal control for two queues with one switching server. Working Paper, Vrije Universiteit, The Netherlands, 2001.

[57] R. D. Nobel. Optimal control for two finite-buffer queues with one switching server. Working Paper, Vrije Universiteit, The Netherlands, 2001.

[58] R. D. Nobel and H. C. Tijms. Optimal control for an M-X/G/1 queue with two service modes. *European Journal of Operational Research*, 103 (3): 610-619, March 1999.

[59] T. L. Olsen. A Practical Scheduling Method for Multiclass Production Systems with Setups. *Management Science*, 45 (1): 116–130, January 1999.

[60] T. L. Olsen. Limit Theorems for Polling Models with Increasing Setups. *Probability in the Engineering and Informational Sciences*, 15: 35–55, 2001.

[61] T. L. Olsen. "On Multi-Item Production Systems with Setups: Review and Intuition", Working Paper, Washington University in St. Louis, 2001.

[62] T. L. Olsen and R. D. van der Mei. Polling Systems with Periodic Server Routing in Heavy-Traffic: Distribution of the Delay. *Journal of Applied Probability*, 40: 305-326, 2003.

[63] T. L. Olsen and R. D. van der Mei. Polling Systems with Periodic Server Routing in Heavy-Traffic: Renewal Arrivals Under submission: *OR Letters*, 2003

[64] J. R. Olson and M. J. Schniederjans. A heuristic scheduling system for ceramic industrial coatings. *Interfaces,* 30 (5): 16-22, September-October 2000.

[65] J. R. Perkins and P. R. Kumar. Stable, Distributed, Real-time Scheduling of Flexible Manufacturing/Assembly/Disassembly Systems. *IEEE Transactions on Automatic Control*, 34 (2): 139–148, February 1989.

[66] C. N. Potts and M. Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120 (2): 228–249, January 2000.

[67] R. Rajan and R. Agrawal. Server allocation and routing in homogeneous queues with switching penalties. *IEEE Transactions on Automatic Control,* 41 (11): 1657-1661, November 1996.

[68] M. I. Reiman and L. M. Wein. Dynamic scheduling of a two-class queue with setups. *Operations Research*, 46(4): 532–547, July-August 1998.

[69] A. N. Rybko and A. L. Stolyar. Ergodicity of stochastic processes describing the operation of open queueing networks. *Problems of Information Transmission,* 28: 199-220, 1992.

[70] C. R. Sox, P. L. Jackson, A. Bowman, and J. A. Muckstadt. A review of the stochastic lot scheduling problem. *International Journal of Production Economics* 62 (3): 181-200, September 1999.

[71] H. Takagi. *Analysis of Polling Systems.* The MIT Press, Cambridge, MA, 1986.

[72] H. Takagi. Queueing analysis of polling models: An Update. In *Stochastic analysis of computer and communication systems*, Chapter 1, H. Takagi (editor). Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1990.

[73] H. Takagi. Queueing analysis of polling models: progress in 1990-1994. In *Frontiers in Queueing: Models and Applications in Science and Engineering*, Chapter 5, J. H. Dshalalow (editor), CRC Press, New York, 1997.

[74] S. Tayur. Improving operations and quoting accurate lead times in a laminate plant. *Interfaces* 30 (5): 1-15, September-October 2000.

[75] R. D. van der Mei. Delay in polling systems with large switch-over times. *Journal of Applied Probability*, 36: 232–243, 1999.

[76] R. D. van der Mei and H. Levy. Polling Systems in heavy traffic: Exhaustiveness of service policies. *Queueing Systems Theory and Applications*, 27: 227-250, 1997.

[77] M. P. Van Oyen, D. G. Pandelis, and D. Teneketzis. Optimality of index policies for stochastic scheduling with switching penalties. *Journal of Applied Probability*, 29 (4): 957–966, December 1992.

[78] M. P. Van Oyen and J. Pichitlamken. Properties of optimal-weighted flowtime policies with a makespan constraint and set-up times. *Manufacturing & Service Operations Management*, 2 (1): 84–99, Winter 2000.

[79] R. W. Wolff, *Stochastic Modeling and the Theory of Queues.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

Online Appendix

Proof of Lemma 1:

We begin by noting that if there are no orders waiting for the product that is currently being served, then the machine should work at rate equal to that product's workload arrival rate and otherwise it should work at rate 1. To prove that this is optimal, note that any period of time that the machine is working at a rate between the workload arrival rate and one may be replaced by a period of the same length where the machine first works at rate one and then cruises at the workload arrival rate. The average workload under the latter is less than that of the former and thus the former cannot be optimal. Further, if the machine works at less than the workload arrival rate workload the average workload will again be higher than if the machine were to first cruise and then idle and the former again cannot be optimal.

For a given time period T, construct a workload realization at queue i with $n_i(T)$ production runs. In each production run j $(1 \leq j \leq n_i(T))$, the workload starts with workload equal to $v_{i,j}$ and ends with workload equal to $l_{i,j}$. Let the unfinished workload at time T be equal to $v_{i,n_i(T)+1}$ and the workload at time 0 be equal to $l_{i,0}$. Let $d_{i,j}$ represent the time for the server to stay in queue i after queue i has been exhausted after the j^{th} run. During this time interval, the machine continues to work on queue i and no workload is being accumulated at queue i. The output rate for this time period is equal to ρ_i, the workload arrival rate of queue i. This is the cruising period. An example of a realization of workload accumulated at some queue i is given in Figure 1, where $n_i(T) = 3$, $d_{i,1} = d_{i,3} = 0$, and $l_{i,2} = 0$. The average workload for queue i up to time T is equal

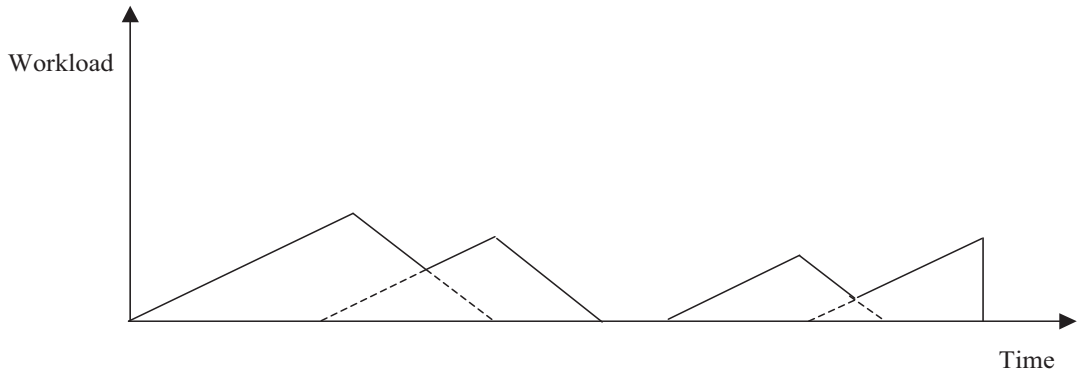

Figure 1: A realization of workload in queue i.

to the area under $v_i(t)$ from 0 to T divided by T. From Figure 1, the area under $v_i(t)$ is equal to sum of all the triangular areas, represented by the solid line in Figure 1, that have height equal to $v_{i,j}$ for $j = 1$ to $n_i(T) + 1$ minus the overlapping triangular areas, of height $l_{i,j}$, for $j = 0$ to $n_i(T)$. Because the slopes of the realization of workload at queue i are equal to $1 - \rho_i$ when the workload is decreasing and ρ_i when the workload is increasing, it follows that the area under $v_i(t)$ from 0 to T equals

$$\frac{1}{2}\left[\frac{\sum_{j=1}^{n_i(T)}(v_{i,j}^2 - l_{i,j}^2)}{\rho_i(1-\rho_i)} + \frac{v_{i,n_i(T)+1}^2}{\rho_i} - \frac{l_{i,0}^2}{\rho_i}\right] \tag{18}$$

and

$$T = \frac{\sum_{j=1}^{n_i(T)}(v_{i,j} - l_{i,j})}{\rho_i(1-\rho_i)} + \frac{(v_{i,n_i(T)+1} - l_{i,0})}{\rho_i} + \sum_{j=1}^{n_i(T)} d_{i,j}. \tag{19}$$

Hence, the average weighted workload carrying cost for this time period at queue i is equal to:

$$\frac{1}{T}\int_0^T c_i v_i(t)\,dt = \frac{c_i\left[\sum_{j=1}^{n_i(T)}(v_{i,j}^2 - l_{i,j}^2) + (1-\rho_i)(v_{i,n_i(T)+1}^2 - l_{i,0}^2)\right]}{2\sum_{j=1}^{n_i(T)}(v_{i,j} - l_{i,j}) + (1-\rho_i)(v_{i,n_i(T)+1} - l_{i,0}) + \rho_i(1-\rho_i)d_i(T)}, \tag{20}$$

where $d_i(T) = \sum_{j=1}^{n_i(T)} d_{i,j}$.

Fix queue i, $1 \le i \le N$, we now show that for each pair of fixed $n_i(T)$ and $d_i(T)$ $(0 \le d_i(T) < T)$ the average production cost for queue i is minimized when the $v_{i,j}$ are all the same (i.e., $v_{i,1} = v_{i,2} = \ldots = v_{i,n_i(T)+1}$) and $l_{i,j} = 0$ for all j. To show that the $l_{i,j} = 0$, consider any realization of a production run at queue i, such that the $l_{i,j}$ are not all equal to 0. Construct a new workload realization for queue i as follows. Let $v'_{i,j} = v_{i,j} - (1-\rho_i)l_{i,j-1} - \rho_i l_{i,j}$, $l'_{i,j} = 0$, and $d'_{i,j} = d_{i,j}$ for $j = 1$ to $n_i(T)$; also, let $v'_{i,n_i(T)+1} = v_{i,n_i(T)+1} - l_{i,n_i(T)}$. Note that since $v_{i,j}$ is greater than both $l_{i,j-1}$ and $l_{i,j}$, it follows that $v'_{i,j} \ge 0$. Moreover, the length of each production run is the same as in the original system. This is a feasible realization of workload at queue i. It has all the $l'_{i,j} = 0$ and finishes the production of queue i at the same point of time as the original realization (i.e., $T' = T$). The new realization of workload at queue i has a lower workload carrying cost given that

T, $d_i(T)$, and $n_i(T)$ are still the same; therefore $l_{i,j}$ must be equal to 0 for all j if the workload is optimal realized at queue i.

An example of the new realization of workload corresponding to the old workload realization of Figure 1 is given in Figure 2. In Figure 2, the old workload realization is represented by the solid line while the new workload realization is represented by the dashed line. It can be seen that the interval of each production run is still the same while the new realization yields a smaller area under each production run which will lead to a smaller production cost at queue i.

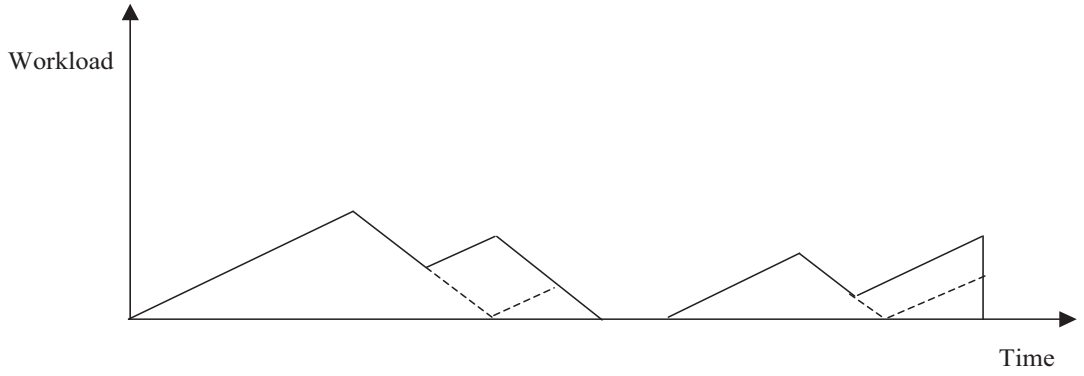

Figure 2: Workload in queue i with $l_{i,j}$ all equal 0.

Since the $l_{i,j}$ must all equal 0, the average weighted workload carrying cost up to time T, under an optimal workload realization, reduces to

$$\frac{1}{T}\int_0^T c_i v_i(t)\,dt = \frac{c_i\left[\sum_{j=1}^{n_i(T)} v_{i,j}^2 + (1-\rho_i)v_{i,n_i(T)+1}^2\right]}{2\left[\sum_{j=1}^{n_i(T)} v_{i,j} + (1-\rho_i)v_{i,n_i(T)+1} + \rho_i(1-\rho_i)d_i(T)\right]}. \tag{21}$$

Using Equation (19) and letting $l_{i,j} = 0$ for all j, for a fixed time T, the denominator in Equation (21) is fixed such that

$$v_{i,1} + v_{i,2} + \ldots + (1-\rho_i)v_{i,n_i(T)+1} + \rho_i(1-\rho_i)d_i(T) = \rho_i(1-\rho_i)T. \tag{22}$$

It can be verified that $v_{i,1}^2 + v_{i,2}^2 + \ldots + (1-\rho_i)v_{i,n_i(T)+1}^2$, is minimized, subject to (22) and $d_i(T)$ remaining fixed, when the $v_{i,j}$ are all the same. Hence (21), the average workload carrying cost of queue i, is also minimized when the $v_{i,j}$ are all the same. Therefore, using (22), the optimal

workload in queue i at the beginning of each production run, as a function of $n_i(T)$ and $d_i(T)$, is

$$v_i = \frac{\rho_i(1-\rho_i)(T-d_i(T))}{n_i(T)+1-\rho_i}. \tag{23}$$

Therefore, along the optimal trajectory of the workload for a queue, that queue should be served exhaustively, and the maximum workload should be the same at the start of each production run.

Up to now, the only restriction on $d_i(T)$ and $n_i(T)$ is $d_i(T) < T$. Thus in the optimal workload trajectory for queue i, the long-run average production cost, which equals the average workload carrying cost of queue i plus the average setup cost at queue i, is

$$AC_i(T) = \frac{w_i(T-d_i(T))^2}{2T(n_i(T)+1-\rho_i)} + \frac{n_i(T)k_i}{T}. \tag{24}$$

Let $n_i^T = \frac{n_i(T)}{T}$ and $d_i^T = \frac{d_i(T)}{T}$. With a little algebra, we can rewrite (24) as

$$AC_i(T) = \frac{w_i(1-d_i^T)^2}{2n_i^T} + n_i^T k_i - r_i(T), \tag{25}$$

where

$$r_i(T) = \frac{w_i(1-d_i^T)^2(1-\rho_i)}{2n_i^T(n_i(T)+1-\rho_i)}. \tag{26}$$

If the policy is unstable then the cost may easily be shown to be greater than a straw stable policy. Therefore, we may assume without loss of generality that the policy is stable. Using this, $\sup_t v_i(t) \le C$, for some C, and hence, from (23),

$$n_i(T) + (1-\rho_i) \ge \frac{\rho_i(1-\rho_i)}{C}(T-d_i(T)). \tag{27}$$

In addition, let $i' = i+1$ and $i' = 1$ if i equals N, then $\sup_t v_{i'}(t) \le C$ implies that $\rho_{i'}\frac{d_i(T)}{n_i(T)} \le C$. The inequality comes from the fact that $\sup_t v_{i'}(t)$ must be greater than the arrivals of type i' workload in any given type i cruising period and at least one of these periods is of length at least $\frac{d_i(T)}{n_i(T)}$. Combining this result with (27), with a little algebra, it can be shown that $n_i(T) \ge a_i T - b_i$ where $a_i = \frac{\rho_i\rho_{i'}(1-\rho_i)}{C[\rho_{i'}+\rho_i(1-\rho_i)]}$ and $b_i = \frac{\rho_{i'}(1-\rho_i)}{\rho_{i'}+\rho_i(1-\rho_i)}$. As a direct result from (26)

$$r_i(T) \le \frac{w_i(1-\rho_i)}{(2a_i - b_i T^{-1})(a_i T - b_i + 1 - \rho_i)}. \tag{28}$$

Note that $r_i(T) \to 0$ as $T \to \infty$.

After looking at each queue i, we need to consider the relationship among all the queues. For a single queue, the total workload that is depleted from queue i, should equal the total workload that arrived at queue i minus the unfinished workload at queue i at time T. Therefore, along an optimal trajectory,

$$\frac{n_i^T v_i}{(1-\rho_i)} + \rho_i d_i^T = \rho_i - \frac{v_i}{T}. \tag{29}$$

In addition, the total time that the machine spends in setup, cruising, and processing, plus the added time during which the unfinished workload at time T is being accumulated in each queue, should equal T. As a result,

$$\sum_{i=1}^{N} \left[n_i^T s_i + d_i^T + \frac{n_i^T v_i}{(1-\rho_i)} \right] = 1 - \min_{i\in[1,N]} \frac{v_i}{\rho_i T}, \tag{30}$$

where the last term in the RHS of (30) comes from the minimum time to accumulate the unfinished workload over all queues divided by T. Summing up (29) over all queues and subtracting from (30) yields a new constraint

$$\sum_{i=1}^{N} [n_i^T s_i + d_i^T (1-\rho_i)] = (1-\rho) - R_1(T) + R_2(T), \tag{31}$$

where $R_1(T) = \min_{i\in[1,N]} \frac{v_i}{\rho_i T}$ and $R_2(T) = \sum_{i=1}^{N} \frac{v_i}{T}$. It is easy to verify that for any $\epsilon > 0$, we can always find T_ϵ big enough such that $r_i(T_\epsilon)$, $\forall\ i \in [1, N]$, $R_1(T_\epsilon)$, and $R_2(T_\epsilon)$ are all less than ϵ. Taking $\epsilon \downarrow 0$ completes the proof of the lemma. □

Proof of Uniqueness of the Lower Bound:

It is possible to have more than one queue satisfy both cruising conditions. Under this scenario, the solution for n_i and d_i is not unique. However, the lower bound stated in Theorem 1 is always true regardless of which queue i is chosen so long as it satisfies the two cruising conditions. In the following, we show that when more than one queue satisfies both of the cruising conditions, and therefore more than one queue can have cruising, the lower bound can always be simplified to the form in Theorem 1. An outline of the proof is as follows. First, we show that the sum of the average production costs of the queues that do not satisfy the cruising conditions is fixed. After that, we show that so long as the KKT conditions are satisfied, the sum of the long-run average production costs of the queues that satisfy the cruising conditions will be fixed no matter how the

values of n_i and d_i change. To do this, we first show the following proposition which is related to the first cruising condition and will be used in the proof below.

Proposition 1 *(i): $\delta_i \geq \delta_l$ implies that*

$$\sqrt{\frac{2(\delta_i s_l + k_l)w_l}{(1-\rho_l)^2}} \leq \delta_i. \tag{32}$$

(ii): δ_i satisfies

$$\sqrt{\frac{2(\delta_i s_i + k_i)w_i}{(1-\rho_i)^2}} = \delta_i. \tag{33}$$

Proof: Since $\delta_i \geq \delta_l$, we have

$$\delta_i - \left[\frac{s_l w_l}{(1-\rho_l)^2} + \frac{w_l}{(1-\rho_l)^2}\sqrt{s_l^2 + \frac{2k_l(1-\rho_l)^2}{w_l}}\right] \geq 0. \tag{34}$$

Observing that, in (34), the second term inside the brackets is positive, therefore

$$\delta_i - \left[\frac{s_l w_l}{(1-\rho_l)^2} - \frac{w_l}{(1-\rho_l)^2}\sqrt{s_l^2 + \frac{2k_l(1-\rho_l)^2}{w_l}}\right] \geq 0. \tag{35}$$

Multiplying together the left hand sides of (34) and (35) yields

$$\delta_i^2 - \frac{2s_l w_l \delta_i}{(1-\rho_l)^2} - \frac{2k_l w_l}{(1-\rho_l)^2} \geq 0, \tag{36}$$

and (32) is verified.

The proof of (33) is similar to the proof of (32). Using the fact that

$$\delta_i - \left[\frac{s_i w_i}{(1-\rho_i)^2} + \frac{w_i}{(1-\rho_i)^2}\sqrt{s_i^2 + \frac{2k_i(1-\rho_i)^2}{w_i}}\right] = 0, \tag{37}$$

we then multiply (37) with

$$\delta_i - \left[\frac{s_i w_i}{(1-\rho_i)^2} - \frac{w_i}{(1-\rho_i)^2}\sqrt{s_i^2 + \frac{2k_i(1-\rho_l)^2}{w_i}}\right]. \tag{38}$$

This changes the inequality sign in (36) to a equality sign and (33) is verified.

□

Define AC_I as the sum of the average production cost AC_i, $\forall\ i \in I$. The average production cost of the system AC equals $AC_I + AC_{\bar{I}}$. Notice that $\forall\ l \in \bar{I}$, $n_l = \sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}}$, where

$\delta^* = \max_{l\in[1,Q]} \delta_l$, only depends on the system parameters. Substituting the value of n_l into (1) yields

$$AC_l = \frac{w_l}{2n_l} + n_l k_l = \sqrt{\frac{w_l}{2}}\left[\frac{k_l}{\sqrt{\delta^* s_l + k_l}} + \sqrt{\delta^* s_l + k_l}\right]. \tag{39}$$

It follows that $AC_{\bar{I}} = \sum_{l\in\bar{I}} AC_l$ is fixed and only depends on the system parameters.

To prove that AC_I is fixed and only depends on the system parameters, we start by deriving the general solution form of n_i and d_i, $\forall i \in I$. From (14)

$$\sum_{i\in I} d_i\left[1-\rho_i - \frac{s_i w_i}{\delta^*(1-\rho_i)}\right] = \left[(1-\rho) - \sum_{l=1}^{Q} s_l\sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}}\right]. \tag{40}$$

Moving all the terms except d_i to the right hand side yields

$$\begin{aligned} d_i = {} & \frac{\delta^*(1-\rho_i)}{(1-\rho_i)^2\delta^* - s_i w_i}\left[(1-\rho) - \sum_{l=1}^{Q} s_l\sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}}\right] \\ & - \sum_{i'\in I, i'\neq i} \frac{(1-\rho_{i'})\delta^* - s_{i'}c_{i'}\rho_{i'}}{(1-\rho_i)\delta^* - s_i c_i \rho_i} d_{i'}. \end{aligned}$$

Let $H_{i',i} = \dfrac{[(1-\rho_{i'})^2\delta^* - s_{i'}w_{i'}](1-\rho_i)}{[(1-\rho_i)\delta^* - s_i w_i](1-\rho_{i'})}$, it follows that, $\forall\ i \in I$,

$$\begin{aligned} 1-d_i &= \sum_{i'\in I, i'\neq i} H_{i',i} d_{i'} + \frac{\delta^*}{(1-\rho_i)\delta^* - s_i c_i \rho_i}\left[\rho + \sum_{l=1,l\neq i}^{Q} s_l\sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}}\right] \\ &- \frac{\delta^*}{(1-\rho_i)\delta^* - s_i c_i \rho_i}\left[1 - s_i\sqrt{\frac{w_i}{2(\delta^* s_i + k_i)}} - \frac{(1-\rho_i)\delta^* - s_i c_i \rho_i}{\delta^*}\right] \end{aligned} \tag{41}$$

From Proposition 1, Equation (33), and using the fact that $\delta^* = \delta_i$, it follows that

$$\sqrt{\frac{w_i}{2(\delta^* s_i + k_i)}} = \frac{w_i}{\delta^*(1-\rho_i)}. \tag{42}$$

By substituting (42) into the third term of the right hand side of (41), Equation (41) simplifies to

$$\begin{aligned} 1-d_i &= \sum_{i'\in I, i'\neq i} H_{i',i} d_{i'} \\ &+ \frac{\delta^*(1-\rho_i)}{(1-\rho_i)^2\delta^* - s_i w_i}\left[\rho - \rho_i + \sum_{l=1,l\neq i}^{Q} s_l\sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}}\right]. \end{aligned} \tag{43}$$

Applying (44) in $n_i = (1-d_i)\dfrac{w_i}{\delta^*(1-\rho_i)}$ gives that

$$n_i = \frac{w_i}{\delta^*(1-\rho_i)} \sum_{i'\in I, i'\neq i} H_{i',i} d_{i'} + \frac{w_i}{(1-\rho_i)^2\delta^* - s_i w_i}\left[(\rho-\rho_i) + \sum_{l=1,l\neq i}^{Q} s_l\sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}}\right]. \tag{44}$$

Substituting $1 - d_i = \dfrac{n_i \delta^* (1-\rho_i)}{w_i}$ into Equation (1) yields

$$AC_I = \sum_{i \in I} n_i \left[\frac{(\delta^*)^2 (1-\rho_i)^2}{2w_i} + k_i \right]. \tag{45}$$

To simplify the AC_I above, we multiply term $\left[\dfrac{(\delta^*)^2 (1-\rho_i)^2}{2w_i} + k_i \right]$ by $\left[\dfrac{w_i}{(1-\rho_i)^2 \delta^* - s_i w_i} \right]$ and show that the result equals δ^*. Assuming that

$$\delta^* \neq \left[\frac{(\delta^*)^2 (1-\rho_i)^2}{2w_i} + k_i \right] \left[\frac{w_i}{(1-\rho_i)^2 \delta^* - s_i w_i} \right],$$

therefore $(1-\rho_i)^2 (\delta^*)^2 - 2 s_i w_i \delta^* - 2 k_i w_i \neq 0$, and hence

$$(\delta^* - \delta_i) \left[\delta^* - \delta_i + \frac{2w_i}{(1-\rho_i)^2} \sqrt{s_i^2 + \frac{2k_i (1-\rho_i)^2}{w_i}} \right] \neq 0.$$

Note that $\forall\ i \in I$, $\delta^* = \delta_i$ and the above equation must equal 0, which contradicts the assumption. Therefore,

$$\delta^* = \left[\frac{(\delta^*)^2 (1-\rho_i)^2}{2w_i} + k_i \right] \left[\frac{w_i}{(1-\rho_i)^2 \delta^* - s_i w_i} \right]. \tag{46}$$

Using the fact that $\dfrac{w_i}{(1-\rho_i)} H_{i',i} = \left[\dfrac{w_i}{(1-\rho_i)^2 \delta^* - s_i w_i} \right] \left[\dfrac{(1-\rho_{i'})^2 \delta^* - s_{i'} w_{i'}}{(1-\rho_{i'})} \right]$ and combining with the result from (46), AC_I from (45) can be written as

$$\begin{aligned} AC_I &= \delta^* \sum_{i \in I} \left[\rho - \rho_i + \sum_{l=1, l \neq i}^{Q} s_l \sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}} \right] \\ &+ (P-1) \delta^* \sum_{i \in I} d_i \left[(1-\rho_i) - \frac{s_i w_i}{\delta^* (1-\rho_i)} \right], \end{aligned} \tag{47}$$

where P is the number of queues in set I.

By substituting the left hand side of (40) into (47) and simplifying, this yields

$$AC_I = \delta^* \left[\rho - \sum_{i \in I} \rho_i + (P-1) + \sum_{l=1, l \in \bar{I}} s_l \sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}} \right]. \tag{48}$$

It is clear that AC_I depends on the system parameters only and is fixed. Combining $AC_{\bar{I}}$ and AC_I yields

$$\begin{aligned} AC &= \sum_{l=1, l \in \bar{I}} \frac{w_l}{2n_l} + n_l k_l \\ &= \sqrt{\frac{w_l}{2}} \left[\frac{k_l}{\sqrt{\delta^* s_l + k_l}} + \sqrt{\delta^* s_l + k_l} \right] \\ &+ \delta^* \left[\rho - \sum_{i \in I} \rho_i + (P-1) + \sum_{l=1, l \in \bar{I}} s_l \sqrt{\frac{w_l}{2(\delta^* s_l + k_l)}} \right]. \end{aligned} \tag{49}$$

Equation (49) can be further simplified by adding all the terms that are related to $\bar{I}$ which is reduced to

$$AC = \sum_{l=1, l\in\bar{I}} \sqrt{2w_l(\delta^* s_l + k_l)} + \delta^* \left[\rho - \sum_{i\in I} \rho_i + (P-1)\right]. \tag{50}$$

To show that AC can also take the solution form in Theorem 1 is the same as showing that

$$\delta^* \left[(P-1) - \sum_{i'\in I, i'\neq i} \rho_{i'}\right] = \sum_{i'\in I, i'\neq i} \sqrt{2w_l(\delta^* s_l + k_l)}. \tag{51}$$

By modifying the left hand side of (51) to $\sum_{i'\in I, i'\neq i} \delta_{i'}(1-\rho_{i'})$ and using the second result from Proposition 1, the proof is complete. Note that letting $d_{i'} = 0, i' \in I, i' \neq i$ does not violate the KKT conditions, as long as (40) is satisfied. □

Proof of Theorem 3

Let τ_n be the time epoch of the beginning of the n^{th} setup and $\zeta(n)$ be the index of the queue that the machine is set up for at time τ_n. Let t_n be the length of the n^{th} service period before queue $\zeta(n)$ has been exhausted and h_n be the length of the holding period after the n^{th} service period. Finally, let $v_i(t)$ represent the workload level at queue i during time t and $V(t) = \sum_{i=1}^{N} v_i(t)$ be the total workload level in the system at time t. In the proof, we show stability of the heuristic with cruising, stability without cruising can be simply deduced by letting $h_n = 0$ for all n.

By definition,

$$\tau_{n+1} = \tau_n + s_{\zeta(n)} + t_n + h_n.$$

The total workload at time τ_{n+1} equals the total workload at time τ_n plus the total arrived workload minus the workload that is depleted from queue $\zeta(n)$ inside this time period. This yields that

$$\begin{aligned} V(\tau_{n+1}) &= V(\tau_n) + \rho(\tau_{n+1} - \tau_n) - t_n - \rho_{\zeta(n)} h_n \\ &= V(\tau_n) - (1-\rho)t_n + (\rho - \rho_{\zeta(n)})h_n + \rho s_{\zeta(n)}. \end{aligned} \tag{52}$$

In the next step, we show that the policies proposed in previous section guarantee that

$$(1-\rho)t_n \geq \epsilon V(\tau_n) \qquad \forall\ n. \tag{53}$$

Define $g_i(\tau_n) = \dfrac{v^*(\tau_n) + \rho_i s_i}{v_i}$, where $v^*(\tau_n) = \max_i v_i(\tau_n)$. In addition, let $i^* = \arg\max_i \dfrac{v_i(\tau_n) + \rho_i s_i}{v_i}$ and $r = \min_i \rho_i s_i$. Note that, from formulas (9) and (11), index i^* is the queue that is chosen for

setup at time τ_n. Following from these definitions,

$$g_{i^*}(\tau_n) \geq \frac{\max_i(v_i(\tau_n)+\rho_i s_i)}{\bar{v}} \geq \frac{v^*(\tau_n)+r}{\bar{v}} \geq \frac{V(\tau_n)}{\bar{v}(N-1)} + \frac{r}{\bar{v}} \tag{54}$$

where the last inequality comes from the fact that at time τ_n, there are $N-1$ queues that can have accumulated workload inside and $(N-1)v^*(\tau_n) \geq V(\tau_n)$. Therefore,

$$\frac{v_{i^*}(\tau_n)+\rho_{i^*}s_{i^*}}{v_{i^*}} \geq \frac{V(\tau_n)}{\bar{v}(N-1)} + \frac{r}{\bar{v}}$$

and hence

$$v_{i^*}(\tau_n)+\rho_{i^*}s_{i^*} \geq \frac{\underline{v}V(\tau_n)}{\bar{v}(N-1)} + \frac{\underline{v}r}{\bar{v}}.$$

This implies that the workload at queue i^* after setting up is at least $\frac{\underline{v}V(\tau_n)}{\bar{v}(N-1)}$, which implies that

$$t_n \geq \frac{\underline{v}V(\tau_n)}{\bar{v}(N-1)(1-\rho_{i^*})} \geq \frac{\underline{v}V(\tau_n)}{\bar{v}(N-1)(1-\underline{\rho})},$$

and equation (53) is verified.

It follows that

$$V(\tau_{n+1}) \leq (1-\epsilon)V(\tau_n) + (\rho-\rho_{\zeta(n)})h_n + \rho s_{\zeta(n)}. \tag{55}$$

Let $\varepsilon = 1-\epsilon$ and observe that both $\frac{\underline{v}}{\bar{v}}$ and $\frac{(1-\rho)}{(1-\rho_{i^*})}$ are less than 1, this yields that $0 < \varepsilon < 1$. Recall the definitions of $\underline{\rho}$, $\bar{h}$ and $\bar{s}$ at the beginning, as a result,

$$V(\tau_{n+1}) \leq \varepsilon V(\tau_n) + (\rho-\underline{\rho})\bar{h} + \rho\bar{s} \tag{56}$$

Taking (56) back to time 0 by induction yields that

$$V(\tau_{n+1}) \leq \varepsilon^{n+1}V(\tau_0) + \frac{1-\varepsilon^{n+1}}{1-\varepsilon}[(\rho-\underline{\rho})\bar{h} + \rho\bar{s}] \tag{57}$$

Note that $0 < \varepsilon < 1$, so that

$$V(\tau_{n+1}) \leq V(\tau_0) + \frac{(\rho-\underline{\rho})\bar{h} + \rho\bar{s}}{1-\varepsilon} \equiv C_0. \tag{58}$$

To demonstrate that our heuristic is stable, it suffices to show that $V(t)$ is bounded above for $t \in [\tau_n, \tau_{n+1})$. Observe that

$$\begin{aligned} V(t) &= V(\tau_n) + \rho(t-\tau_n) \leq C_0 + \rho\bar{s} \equiv C_1 \qquad\qquad t \in [\tau_n, \tau_n + s_{\zeta(n)}), \\ V(t) &= V(\tau_n + s_{\zeta(n)}) - (1-\rho)(t-\tau_n-s_{\zeta(n)}) \end{aligned} \tag{59}$$

$$\leq V(\tau_n + s_{\zeta(n)}) \leq C_1 \qquad t \in [\tau_n + s_{\zeta(n)}, \tau_n + s_{\zeta(n)} + t_n), \tag{60}$$

$$\begin{aligned} V(t) &= V(\tau_n + s_{\zeta(n)} + t_n) + (\rho - \rho_{\zeta(n)})(t - \tau_n - s_{\zeta(n)} - t_n) \\ &\leq V(\tau_n + s_{\zeta(n)} + t_n) + (\rho - \underline{\rho})\bar{h_n} \\ &\leq C_1 + (\rho - \underline{\rho})\bar{h} \qquad t \in [\tau_n + s_{\zeta(n)} + t_n, \tau_{n+1}), \end{aligned} \tag{61}$$

therefore $V(t) \leq C_1 + (\rho - \underline{\rho})\bar{h}$, $t \in [\tau_n, \tau_{n+1})$. Let $UB_{heur2} = C_1 + (\rho - \underline{\rho})\bar{h}$, this completes the proof. □