

Modeling and Analysis of Flexible Queueing Systems

Suri Gurumurthi,¹ Saif Benjaafar²

¹ *Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

² *Graduate Program in Industrial Engineering, Department of Mechanical Engineering, University of Minnesota, Minneapolis, Minnesota 55455*

Received 6 November 2001; revised 3 October 2003; accepted 23 March 2004
DOI 10.1002/nav.20020

Abstract: We consider queueing systems with multiple classes of customers and heterogeneous servers where customers have the flexibility of being processed by more than one server and servers possess the capability of processing more than one customer class. We provide a unified framework for the modeling and analysis of these systems under arbitrary customer and server flexibility and for a rich set of control policies that includes customer/server-specific priority schemes for server and customer selection. We use our models to generate several insights into the effect of system configuration and control policies. In particular, we examine the relationship between flexibility, control policies and throughput under varying assumptions for system parameters. © 2004 Wiley Periodicals, Inc. *Naval Research Logistics* 51: 755–782, 2004.

Keywords: queueing systems; flexibility; dynamic routing and sequencing; markov chains; call centers

1. INTRODUCTION

In this paper, we consider the modeling and analysis of flexible queueing systems. We use the term flexible queueing systems to refer to systems with heterogeneous servers and multiple customer classes where customers have the flexibility of being processed by more than one server and servers possess the capability of processing more than one customer class. Customer classes can vary in demand rate and routing flexibility. Servers can vary in service rates and service flexibility. The assignment of customers to servers is determined by a server selection rule, that can be customer-specific, and the selection of the next customer to serve is determined by a queue selection rule, that can be server-specific. An example of a flexible queueing system is shown in Figure 1.

Flexible queueing systems arise in a variety of contexts, including manufacturing (Buzacott and Shanthikumar [8]), telecommunication networks (Ross [23]), computer systems (Kleinrock

Correspondence to: S. Benjaafar (saif@ie.umn.edu); S. Gurumurthi (suri@mit.edu).

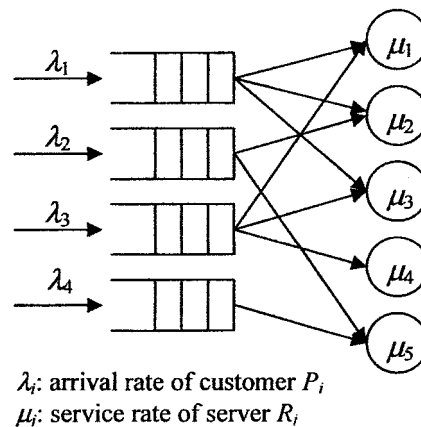


Figure 1. An example of a flexible queueing system.

[19]), and service operations (Hall [15]). In manufacturing, there is often flexibility in routing jobs to functionally equivalent machines or production lines. These machines may vary by speed or cost. In telecommunication, flexibility arises from the availability of multiple links to which incoming calls can be routed. Different links may carry different capacities or provide different response times. Similar issues arise in large computer systems with multiple users and multiple servers. Flexibility derives from the ability to dynamically route users to different servers and to share computing capacity among different customers. Call centers are an important application in the service sector. Incoming customer calls vary by need, duration, and level of difficulty. Call centers are staffed by operators with varying skills who are capable of handling some or all of the call types (Koole and Mandelbaum [20]; Whitt [33]).

In this paper, we provide a modeling framework for the analysis of general systems with an arbitrary number of server and customer types and arbitrary customer and server flexibility. We consider a rich set of control policies that includes strict priority schemes for server selection (customer routing) and queue selection, and dynamic policies such as longest queue first. Our models are applicable to systems with finite queue capacity. This queue capacity may be customer-specific or in the form of a global bound on the total number of customers in the system. An important contribution of this work is an alternative state representation scheme that not only is flexible but also yields significant economies in the size of the required state space. To our knowledge, this work is the first to provide exact methods for the analysis of systems with general customer and server flexibility and heterogeneous servers.

To illustrate the usefulness of our models, we carry out a numerical study to examine the relationship between performance, as measured by throughput, and customer and server flexibility. Counter to intuition, we show that higher flexibility does not always improve throughput. For systems where it is beneficial, we show that the value of flexibility exhibits diminishing returns, with most of this value realized with relatively limited flexibility. In systems with identical rates and identical service rates, we find that a special configuration called *flexibility chaining* yields most of the benefits of full flexibility. This is not the case in asymmetric systems (i.e., systems where the demand rates are different for different customer type or service rates are different for different servers), where we find that an asymmetric allocation of flexibility (i.e., nonidentical allocation of flexibility among customers and servers) is generally superior. We examine the impact of control policies under different conditions of demand asymmetry. We

show that there is a range of asymmetry in which the difference in throughput due to different control policies is maximum. For systems with asymmetric flexibility, we illustrate the value of well designed control policies, especially those that take customer and server flexibility into consideration.

The remainder of this paper is organized as follows. In Section 2, we provide a brief review of relevant literature. In Section 3, we present our model. In Section 4, we discuss numerical results and several insights. In Section 5, we summarize our results and offer some concluding comments.

2. LITERATURE REVIEW

The literature on queueing systems with multiple servers can be broadly classified as pertaining to either design or control of these systems. For design, the literature can be grouped around four central questions: (1) How many servers should we have, (2) how should capacity be allocated to these servers, (3) how much flexibility should each server have, and (4) how much routing flexibility should we provide to each customer class. Issues pertaining to questions (1) and (2) are generally referred to as capacity allocation. The literature on this subject is voluminous. A review of important results and applications can be found in Kleinrock [19] and Buzacott and Shanthikumar [8]. A review of related literature on call center staffing can be found in Gans, Koole, and Mandelbaum [12] and Whitt [33]. The available literature related to questions 3 and 4 focuses mostly on comparing two extreme scenarios: *dedication* versus *pooling*. In a dedicated system, each customer class can be routed to only one server and each server can be routed to only one customer class. Hence, the system operates as a set of independent single server queues. In a pooled system, the customers are grouped in a single queue and can be routed to any server. In this case, the system operates as a multi-server queueing system. In systems with homogeneous service and demand distributions, it has been shown that a pooled system always outperforms a dedicated one (Smith and Whitt [27], Benjaafar [4]). Calabrese [9] shows that when average load is held constant in an $M/M/m$ queueing system, average delay is strictly decreasing in m . Benjaafar [4] offers performance bounds on the effectiveness of several pooling scenarios. Stecke and Solberg [28] study pooling in the context of closed queueing network models of Flexible Manufacturing Systems (FMS) and show that throughput increases in the number of pooled servers.

In heterogeneous systems, pooling is not always superior. Buzacott [7] considers a variety of pooling scenarios. He shows that when the service times of different customer classes are not identical in distribution, pooling can lead to longer queueing delays. He shows that the difference in performance between different pooling scenarios is sensitive to service time variability, the size of demand from each class, and the routing policy. These results support the observations made by Smith and Whitt [27], who show that a pooled system can be made arbitrarily worse than a dedicated one through the introduction of rare customers with long service times. Related discussion can be found in Mandelbaum and Reiman [22].

Sheikhzadeh, Benjaafar, and Gupta [25] propose *server chaining* as an alternative design strategy to both dedication and pooling. In a chained configuration, each customer can be routed to one of two neighboring servers and each server can process customers from two neighboring classes (see Section 5). By maintaining an overlap of one customer class between neighboring servers, Sheikhzadeh et al. [25] show that chained systems, under the assumption of homogeneous demand and service times, achieve most of the benefits of total pooling. Similar insights were obtained by Jordan and Graves [18] in a production planning context and by Hopp et al. [17] for worker cross-training in serial production systems.

In this paper, we extend the analytical framework developed by Sheikhzadeh, Benjaafar, and Gupta [25] to include a broader class of system configurations and control policies and a less restrictive set of assumptions. In particular, Sheikhzadeh et al. [25] limit their analysis to three system configurations: dedication, chaining, and full flexibility. They consider a symmetric system with equal number of customer classes and servers and identical bounds on queue sizes. They also consider control policies that do not allow for a full specification of a priority scheme for customer routing and queueing selection.

Control of flexible queueing systems can be broadly categorized as either static or dynamic. Under static control, customers are preassigned to specific servers according to a static partitioning scheme and served according to a fixed sequencing policy. Under dynamic control, routing and sequencing decisions are made based on the state of the system. Most of the literature on static control falls under the category of what is called *workload allocation*. In workload allocation demand that arises from different customer classes is partitioned among the different servers. The partitioning can be either discrete or continuous. In discrete partitioning, all of the demand from each customer class must be allocated to one server. This gives rise to a combinatorial problem that is NP-hard in most cases. In continuous partitioning, fractional assignments are possible. The challenge in this case is to identify the optimal fractions of the demand from each customer class to assign to each server. A review of this literature can be found in Wang and Morris [30]. Related discussion can be found in Benjaafar [5] and Benjaafar and Gupta [6].

The literature on dynamic control is vast (Sennot [24]). Much of this literature deals with systems consisting of a single class of arrivals and multiple servers with each server maintaining a separate queue. For systems with identical servers, Poisson arrivals and exponential processing times, Winston [31] shows that the policy of routing an arriving customer to the shortest queue maximizes the discounted number of service completions in any finite interval $[0, T]$. Weber [32] extends this result to systems with generally distributed interarrival times and servers with nondecreasing hazard rates. Hajeck [14] treats a related problem with two servers and shows that the optimal policy is described by a switching function. A review of the dynamic routing literature can be found in Stidham and Weber [29].

There is related emerging literature on call centers with *skills-based routing* (SBR). Call centers with SBR are centers with multi-skilled agents handling a variety of calls. Calls are routed to an agent based on the nature of the call and the skill of the agent. The exact analysis and optimal control of call centers with SBR are notoriously difficult, and few general results exist (Gans, Koole, and Mandelbaum [12]; Whitt [33]). Research to date has instead attempted to reduce the difficulty of the problem by either (1) treating simple flexibility configurations, (2) simplifying the control policies (e.g., by assuming fixed priority policies), or (3) analyzing the system under heavy traffic assumptions. Examples include Garnett and Mandelbaum [13], Shumsky [26], Aksin and Karaesemen [2], Harrisson and Lopez [16], and Armony [3]. Reviews of this literature can be found in Gans, Koole, and Mandelbaum [12] and Whitt [33].

This paper deals with systems with finite buffers of which loss systems are a special case. Loss systems are queueing systems where waiting is not allowed. Loss systems with multiple servers and multiple customer classes have been widely studied in the context of telecommunication networks (Cooper [10]). A recent review of relevant problems and literature can be found in Ross [23].

3. A MODELING FRAMEWORK FOR FLEXIBLE QUEUEING SYSTEMS

Consider a system consisting of m servers and n customer classes. Customers of class i ($i = 1, \dots, n$) arrive to the system according to an independent Poisson process with rate λ_i .

Processing times at server j ($j = 1, \dots, m$) are exponentially distributed and *i.i.d.* with mean $1/\mu_j$. Each server is capable of processing one or more customer classes, and each customer class can be processed by one or more servers. Let $M = \{R_1, R_2, \dots, R_m\}$ be the set of servers in the system and $P = \{P_1, P_2, \dots, P_n\}$ be the set of customer types. Possible customer-server assignments are denoted by an $n \times m$ matrix $A = [a_{ij}]$, where

$$a_{ij} = \begin{cases} 1, & \text{if part } P_i \text{ can be processed by server } R_j, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

We define a set of servers Q_i associated with each customer type P_i , such that this customer type can be processed by any of the servers in Q_i :

$$Q_i = \{R_{i(1)}, R_{i(2)}, \dots, R_{i(m)}\},$$

where $i(k)$ denotes the index of the k th server assigned to customer type P_i . We let $m_i \equiv |Q_i| = \sum_{j=1}^m a_{ij}$ denote the cardinality of the set Q_i . Similarly, we define T_j to be the set of customer types that can be processed by server R_j such that

$$T_j = \{P_{j(1)}, P_{j(2)}, \dots, P_{j(n)}\},$$

where $j(k)$ denotes the index of the k th customer type assigned to server R_j and $n_j \equiv |T_j| = \sum_{i=1}^n a_{ij}$ as the cardinality of set T_j .

In addition to specifying feasible customer-server assignments, the analysis of flexible queueing systems requires the specification of a control policy. The control policy is applied at each decision epoch. Decision epochs are triggered by either the arrival of a customer or the completion of service by a server. When a customer arrives and finds multiple idle servers, the control policy specifies which server is selected. Similarly, when a customer completes service and finds more than one customer in the queue, the control policy specifies which customer is selected next for service. Hence a control policy is defined by a server selection rule and a customer selection rule. This view is consistent with the treatment found in the literature on skills-based routing (see, for example, Section 5 of Gans, Koole, and Mandelbaum [12]), where the server selection rule is referred to as the *agent* selection rule (how does an arriving call select an idle agent, if there is one) and the customer selection rule as the *call* selection rule (how does an idle agent select a waiting call, if there is one).

In this paper, we consider static server selection rules, where server preferences can be specified in terms of a priority scheme for each customer class. For each customer class and for each server, we associate a priority $\alpha(P_i, R_j) \in \{1, 2, \dots, m\}$, which, for notational compactness, we shall heretofore denote as α_{ij} , where priority is higher for lower values of α_{ij} . If there is competition between two or more idle servers for a customer of type P_i , the customer is assigned to the server with the lower value of α_{ij} . Special cases of the priority scheme include the strict priority (SP) rule where $\alpha_{ij} \neq \alpha_{ik}$ for all values of i, j and k for which $a_{ij} = a_{ik} = 1$, and the random routing (RR) rule where $\alpha_{ij} = \alpha_{ik}$ for all values of i, j , and k for which $a_{ij} = a_{ik} = 1$. In all cases, ties are broken arbitrarily.

For customer selection, we consider a dynamic rule under which a server, upon becoming available, always selects a customer from the class with the longest queue from the set of feasible customer classes. Among customers from the same class, customers are served on a first in first out (FIFO) basis. We term this rule the longest queue first (LQF) rule. We also consider

static customer selection rules, where customers are selected based on a priority scheme. Specifically, for each customer class and for each server, we associate a priority $\gamma(P_i, R_j) \in \{1, 2, \dots, n\}$, or more simply γ_{ij} . Upon becoming idle, a server R_j selects a customer from the class with the lowest value of γ_{ij} . Within each class, customers are again ordered on a first in first out (FIFO) basis. Special cases of priority schemes include the *strict priority* (SP) rule, where $\gamma_{ij} \neq \gamma_{kj}$ for all values of i, j , and k for which $a_{ij} = a_{kj} = 1$, and the *random service* (RS) rule, where $\gamma_{ij} = \gamma_{kj}$ for all values of i, j , and k for which $a_{ij} = a_{kj} = 1$.

Although static, fixed priority rules allow us to represent a rich set of control policies, including those that take into account differences in processing rate and flexibility among servers, and demand rates and routing flexibility among customers. For example, customers may assign priorities to servers based on their processing speed (e.g., always select the fastest available server). Alternatively, customers may assign servers priorities based on their flexibility (e.g., always select the least flexible available server). Similarly, servers may associate priorities with customers based on their demand rate or their routing flexibility. For instance, customers are assigned priorities based on their arrival rate (e.g., always select the customer with the highest arrival rate) or alternatively based on their flexibility (e.g., select the customer with the fewest feasible number of servers). In Section 4, we illustrate in more detail how such control policies can be constructed and examine their effect on performance.

In this paper, we are concerned with the analysis of systems with finite queue capacity. We define queue capacity in one of two ways. A maximum queue size, denoted by w_i , where $w_i \geq 1$ for $i = 1, \dots, n$, is associated with each customer class. Hence, customers of class i are admitted as long as the number of class i customers already in the queue is less than or equal to w_i . Alternatively, we may specify a global bound on the maximum number of customers, regardless of type, that can be allowed in the system.

3.1. The State Space

The state of the system in a flexible queueing system can be described completely by specifying (i) the number of customers in queue for each customer class, and (ii) the state of every server in the set M . Since we do not model server failures, there can only be two possible states, 0 and 1, for each server. The state of the system can thus be described using a vector $\mathbf{N} \equiv (n_1, n_2, \dots, n_{n+m})$, where n_i is the number of customers of type i for $1 \leq i \leq n$ and n_i is the state of server i for $n+1 \leq i \leq n+m$. We denote the state space generated by such a representation as S_1 . Although this state space representation could be used, it requires evaluating a large number of states even for small values of n and m . In order to avoid such enumeration, Sheikhzadeh, Benjafer, and Gupta [25] took advantage of the simple structure of the systems they studied to redefine their state space. Since they considered only the three symmetric systems, specialization, chaining, and full flexibility, where the number of customers equals the number of servers ($n = m$), they were able to represent states of the system using a vector of state variables $\mathbf{N} = (n_1, n_2, \dots, n_n)$, where $n_i = q_i + s_i$, with q_i ($q_i = 0, 1, \dots, w_i$) being the number of customers of the i th type and s_i ($s_i = 0, 1$) being the state of the i th server. This modified state space representation yields considerable economies in the number of states that need to be considered since it reduces the number of state variables from $n + m$ to n .

In our case, because we allow the number of customers to be different from the number of servers and allow for asymmetries in the routing flexibility of customers and the customer flexibility of servers, the modified state space representation cannot be immediately used. However, in what follows we show that an alternative representation that yields significant reduction in the size of the state space is also possible here. The state representation we

introduce shares some similarities with that of Sheikhzadeh, Benjaafar, and Gupta [25] in that every state variable refers to either: (i) the number in queue for a customer type i (q_i), (ii) the state of a server j ($s_j = 0, 1$), or (iii) the sum of the two, $q_i + s_j$, for a pair customer i -server j .

Our approach is based on a graph-theoretic view of flexible queueing systems. In particular, we view our queueing system as a connected bipartite undirected graph, $G(V, E)$, where V is the set of vertices and E is the set of edges. The vertices of the graph lie in two disjoint subsets. The first subset, V_1 , corresponds to the set of customer types, and the second, V_2 , corresponds to the set of servers. The edges of the graph connect the vertices in the set of customers to the vertices in the set of servers, where an edge e_{ij} between customer i and server j exists only if $a_{ij} = 1$. We limit our attention to connected graphs since a disconnected graph leads to two or more queueing systems that can be analyzed independently of each other. Before we present our state space representation scheme, we need the following two definitions and lemma.

DEFINITION 1: A subgraph of $G(V, E)$ in which every vertex has a degree of at most one (i.e., every vertex has at most one edge) is called a matching. The problem of finding such a subgraph is also sometimes called matching.

DEFINITION 2: A maximum matching (or a matching of maximum cardinality) of graph $G(V, E)$, is a matching $G_x(V, E_x)$ such that $|E_x| \geq |E_y|$ for any other matching $G_y(V, E_y)$, where $|E_x|$ and $|E_y|$ refer to the cardinality of the set of edges E_x and E_y , respectively.

LEMMA 1: Consider an undirected and connected bipartite graph $G(V, E)$ whose vertices can be partitioned into two disjoint sets V_1 with n vertices and V_2 with m vertices. Then, there exists a graph $G_x(V, E_x)$ that has the following properties:

1. $G_x(V, E_x)$ is a subgraph of $G(V, E)$.
2. $G_x(V, E_x)$ is a maximum matching of $G(V, E)$.
3. $G_x(V, E_x)$ has k edges, where $1 \leq k \leq \min(m, n)$ and $m + n - 2k$ unmatched vertices.

A proof of lemma 1 can be found in Ahuja, Magnanti, and Orlin [1]. A maximum matching can be obtained efficiently by reformulating it as a maximum flow problem and solving it using for example the Even and Tarjan [11] algorithm in $O(n^{1/2}m)$ time. The maximum matching yields a subgraph with k customers and k servers with each customer connected to one server by an edge. Without loss of generality, we rename this set of customers and servers so that a customer that has been renamed P_i is connected to a server that has been renamed R_i . Then, we associate with this customer-server pair a state variable n_i , where $n_i = s_i + q_i$. The maximum matching also yields l unmatched vertices ($l = m + n - 2k$). These l vertices consist of m' (≥ 0) servers and n' (≥ 0) customers. If $n > k$, we rename these customer classes $P_{k+1}, P_{k+2}, \dots, P_n$ and associate with each a state variable n_i where $n_i = q_i$. Similarly, if $m > k$, the unmatched servers are renamed $R_{n+1}, R_{n+2}, \dots, R_{n+m}$, and associate with each a state variable $n_i = s_i$. This process results in a state vector $\mathbf{N} = (n_1, n_2, \dots, n_q)$, where $q = m + n - k$ and

$$n_i = \begin{cases} q_i + s_i & \text{if } 1 \leq i \leq k, \\ q_i & \text{if } k \leq i \leq n \text{ and } n > k, \\ s_i & \text{if } n \leq i \leq m \text{ and } m > k. \end{cases} \quad (3)$$

The process of generating the state vector \mathbf{N} is illustrated in Figure 2 for an example system with 3 customers and 4 servers.

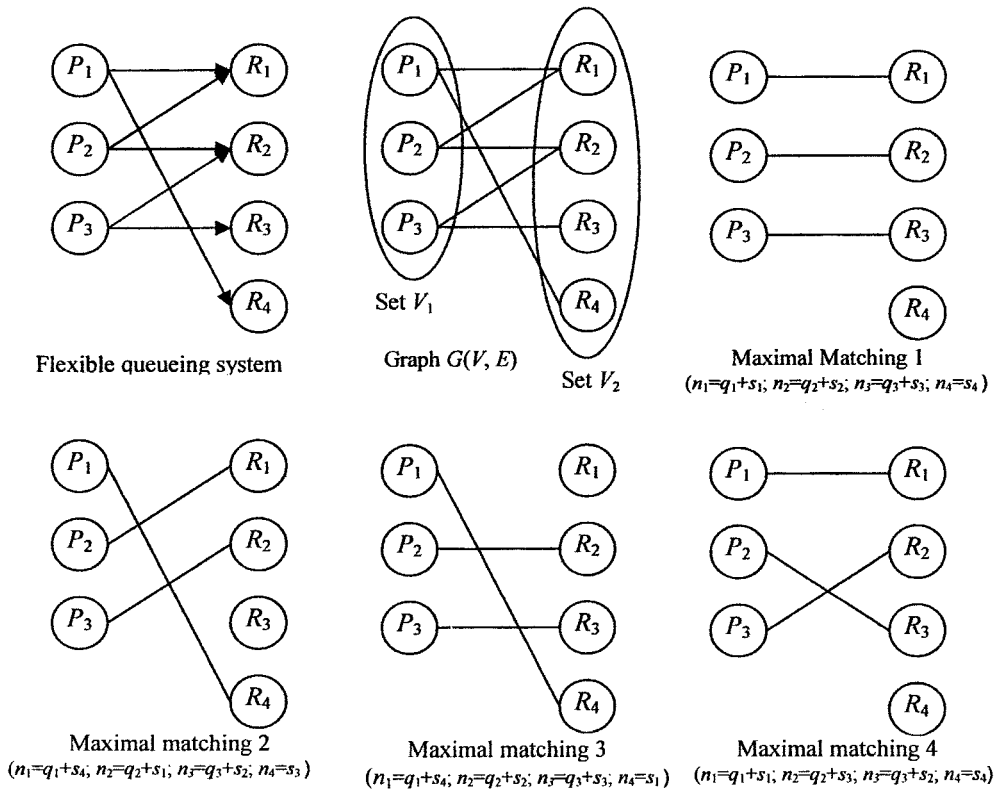


Figure 2. Examples of maximal matchings.

Our representation scheme which reduces the number of state variables from $(m + n)$ to $(m + n - k)$ can have a significant impact on the number of states that need to be evaluated. For example, consider a system with total flexibility so that every customer type can be routed to any server and every server can process every customer. Then, it is not difficult to show that the number of states for $n \leq m$, using a representation with $m + n$ state variables is given by

$$|S_1| = 2^m - 1 + \prod_{i=1}^n (w_i + 1). \tag{4}$$

In contrast, using our representation scheme the number of states is given by

$$|S_2| = 2^m - 1 + \prod_{i=1}^n w_i. \tag{5}$$

Tables 1(a) and 1(b) illustrate the magnitude of the state space reduction achieved using our scheme. Note that the savings are increasing in the size of the problem (i.e., n , m , and w_i). In

Table 1(a). Percentage reduction in the size of the state space ($w_i = 4$ for $i = 1, 2, \dots, n$).

n	m	$ S_1 $	$ S_2 $	Percentage Difference
3	3	132	71	46.21
4	4	640	271	57.66
5	5	3156	1055	66.57
6	6	15688	4159	73.49
7	7	78252	16511	78.90
8	8	390880	65791	83.17
9	9	1953636	262655	86.56
10	10	9766648	1049599	89.25

general, the savings are largest when flexibility is reasonably well distributed among servers and customers and k is near its maximum value of $\min(m, n)$.

3.2. Performance Evaluation

In this section, we develop models for the performance evaluation of flexible queueing systems. Our approach begins by determining the probability of occurrence of each system state for the different control policies under consideration. From these probabilities, we show how to obtain various performance measures of interest. We model our system as a continuous time Markov chain (CTMC) with state vectors $\mathbf{N} = (n_1, n_2, \dots, n_q)$, and n_i is as defined in subsection 3.1. Our Markov chain is a birth–death process since the system transitions from its current state through either a single customer arrival or a single customer departure. The limiting probabilities of system states can be obtained from the balance equations of the Markov chain by equating the rate at which the system enters a state with the rate at which it leaves it. This relationship results in a set of linear equations that can be solved using a general-purpose linear equation solver.

Following the approach in Sheikhzadeh et al., we define for each state vector $\mathbf{N} = (n_1, n_2, \dots, n_q)$ two sets of states, type a and type d , depending on whether an arrival or a departure causes the system to move to state \mathbf{N} . We denote these sets of states as N^a and N^d , respectively. Elements of N^a (N^d) are state vectors \mathbf{N}_i^a (\mathbf{N}_i^d), such that $n_i^a = n_i - 1$ ($n_i^d = n_i + 1$) and all other state variables having the same value as in \mathbf{N} . We define $\delta(x)$ as a function that returns 1 if $x \geq 1$, and 0 otherwise. We also define

$$v_i = \begin{cases} 1, & \text{if } 1 \leq i \leq n, \\ 0, & \text{otherwise,} \end{cases}$$

Table 1(b). Percentage reduction in the size of the state space ($n = m = 6$).

w_i	$ S_1 $	$ S_2 $	Percentage Difference
2	792	127	83.96
3	4159	792	80.96
4	15688	4159	73.49
5	46719	15688	66.42
6	117712	46719	60.31
7	262207	117712	55.11
8	531504	262207	50.67
9	1000063	531504	46.85
10	1771624	1000063	43.55

and

$$\omega_i = \begin{cases} 1, & \text{if } (1 \leq i \leq k) \text{ or } (n < i \leq m \text{ and } m > k), \\ 0, & \text{otherwise.} \end{cases}$$

The variable v_i is used to indicate if a state variable n_i includes the queue size of a customer i . Similarly, the variable ω_i is used to indicate if a state variable n_i includes the state of a server i . Finally, we define the parameter b_i , the upper bound on the state variable n_i , as follows:

$$b_i = \begin{cases} w_i + 1, & \text{if } \omega_i = v_i = 1, \\ w_i, & \text{if } v_i = 1 \text{ and } \omega_i = 0, \\ 1, & \text{otherwise.} \end{cases}$$

Whenever the system is in state \mathbf{N} , it is straightforward to show that it leaves this state as a result of a service completion with rate $\sum_{i=1}^q \delta(n_i)\omega_i\mu_i$ and as a result of an arrival with rate $\sum_{i=1}^q \delta(b_i - n_i)v_i\lambda_i$. If we use r_i^a (r_i^d) to denote the rates at which the system enters state \mathbf{N} from \mathbf{N}_i^a (\mathbf{N}_i^d), and if we use notation $p(\mathbf{N})$, $p(\mathbf{N}_i^a)$ and $p(\mathbf{N}_i^d)$ to denote the steady state probabilities of those states, we can then write the Markov chain balance equation as follows:

$$\left(\sum_{i=1}^q \delta(n_i)\omega_i\mu_i + \sum_{i=1}^q \delta(b_i - n_i)v_i\lambda_i \right) p(\mathbf{N}) = \sum_{\mathbf{N}_i^a \in N^a} r_i^a p(\mathbf{N}_i^a) + \sum_{\mathbf{N}_i^d \in N^d} r_i^d p(\mathbf{N}_i^d), \quad \forall \mathbf{N} \in S_2. \tag{6}$$

The set of linear equations in (6) along with the normalizing equation $\sum_{S_2} p(\mathbf{N}) = 1$ forms a set of $|S_2|$ simultaneous equations, which can be solved to determine the steady state probabilities, $p(\mathbf{N})$, $\mathbf{N} \in S_2$. However, in order to solve for $p(\mathbf{N})$, we need to first define the sets of entering states \mathbf{N}_i^a and \mathbf{N}_i^d and determine the associated rates r_i^a and r_i^d , respectively.

3.2.1. The Sets of Entering States

There are two types of constraints that define whether a state can be included in either set N^a or N^d . The first constraint deals with feasibility. Depending on the routing matrix and the control policies, there are certain states that can never occur. The second restriction stems from the requirement that it should be possible to go from a member of N^a (or N^d) to \mathbf{N} by a one-step transition as result of a single arrival or a single departure.

We first consider the set N^a . The system can move into state \mathbf{N} from \mathbf{N}_i^a , only if $n_i^a = n_i - 1$ and all other state variables have exactly the same values as in \mathbf{N} . That is,

$$n_i^a = n_i - 1 \quad \text{and} \quad n_k^a = n_k \quad \forall k \neq i. \tag{7}$$

The state \mathbf{N}_i^a exists only when one of the following mutually exclusive conditions holds:

$$1 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, \quad l \in Q_i, \quad \omega_i = v_i = 1; \tag{8}$$

$$0 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, \quad l \in Q_i, \quad \omega_i = 0, \quad v_i = 1; \text{ or} \tag{9}$$

$$n_i^a = 0 \text{ and } n_r^a \leq 1, \quad \forall r \neq i, \quad r \in T_i, \quad \omega_i = 1. \quad (10)$$

Condition 8 states that for n_i^a ($n_i = q_i + s_i$ since $w_i = v_i = 1$) to increase by 1, all the servers capable of processing customers of type i must be busy. This follows from the fact that we do not allow a queue to form while a feasible server is idle. Note that we do not require that the server represented by n_i be occupied by a customer of type i . Condition 9 states that for n_i ($n_i = q_i$ since $\omega_i = 0$) to increase by 1, all the servers capable of processing customer i must be busy. In this case n_i can be zero since it represents customer queue size. Condition 10 considers cases when n_i ($n_i = q_i + s_i$ or $n_i = s_i$) increases from 0 to 1. This is possible only if server i is idle which occurs only if there are no customers in the queue that can be processed on server i . From the above we can now define set N^a as follows:

$$N^a = \{N_i^a : 1 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 1, v_i = 1; \\ 0 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 0, v_i = 1; \text{ or} \\ n_i^a = 0 \text{ and } n_r^a \leq 1, \quad \forall r \neq i, r \in T_i, \omega_i = 1; \text{ and } i = 1, \dots, q\}.$$

Similarly, the system moves into a state N from a state N_i^d only if $n_i^d = n_i + 1$ and all other state variables maintain the same values as in N . That is,

$$n_i^d = n_i + 1 \quad \text{and} \quad n_k^d = n_k, \quad \forall k \neq i. \quad (11)$$

It is possible to show that the above holds only if one of the following conditions is true:

$$n_i^d = 1 \text{ and } n_r^d \leq 1, \quad \forall r \neq i, \quad r \in T_i, \quad \omega_i = 1; \quad (12)$$

$$2 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, \quad l \in Q_i, \quad \omega_i = 1, \quad v_i = 1; \text{ or} \quad (13)$$

$$1 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, \quad l \in Q_i, \quad \omega_i = 0. \quad (14)$$

Condition 12 states that a departure would cause n_i^d to decrease from 1 to 0 if there are no other customers present in queue that can be processed on server i . Condition 13 considers the case where $n_i^d \geq 2$ and follows from the fact that a queue of customer type i cannot form unless all its feasible servers are busy. Condition (14) is similar to (13) when n_i describes only the queue size of a customer type i ($n_i = q_i$). Set N^d is thus given by

$$N^d = \{N_i^d : n_i^d = 1 \text{ and } n_r^d \leq 1, \quad \forall r \neq i, r \in T_i, \omega_i = 1; \\ 2 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 1, v_i = 1; \text{ or} \\ 1 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 0, v_i = 1; i = 1, \dots, q\}.$$

3.2.2. Transition Rates r_i^a and r_i^d

In this section, we show how the transition rates for the control policies we consider can be determined. Recall that we define control policies in terms of a server and customer selection rule combination.

The SP-LQF Policy. Under the SP-LQF policy, servers are selected based on a strict priority scheme and customers are selected from the class with the longest queue. When either condition (8) or (9) holds, a transition from \mathbf{N}_i^a to \mathbf{N} clearly occurs with rate

$$r_i^a = \lambda_i. \tag{15}$$

However, when condition (10) holds, the transition rate depends on the routing priorities. First note that for n_i^a to increase from 0 to 1, we need an arrival from a customer type that belongs to the set T_i . Although necessary, this condition is not sufficient since an arrival of a customer of type P_k from the set T_i may not be routed to server i unless server i has the highest priority among those available to process customer P_k . This means that the following condition,

$$\alpha_{ki} < \alpha_{kj} \quad \text{for all } j \in Q_k \text{ that satisfy } n_j = 0 \text{ and } w_j = 1,$$

must be satisfied. Since the arrival rate of customers of type k is λ_k , the transition rate r_i^a is given by

$$r_i^a = \sum_{k \in T_i} \lambda_k \gamma(\alpha_{ki}),$$

where

$$\gamma(\alpha_{ki}) = \begin{cases} 1, & \text{if } \alpha_{ki}(1 - \delta(n_i)) \leq \alpha_{kt}, \quad \forall k \in T_i, \quad \forall t \in Q_k; \text{ and} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Putting it all together, we have

$$r_i^a = \begin{cases} r_i^a = \lambda_i, & \text{if } 0 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, v_i = 1; \\ \sum_{k \in T_i} \lambda_k \gamma(\alpha_{ki}), & \text{if } n_i^a = 0 \text{ and } n_r^a \leq 1, \quad \forall r \neq i, r \in T_i, \omega_i = 1. \end{cases}$$

Similarly, we can derive the transition rates \mathbf{N}_i^d to \mathbf{N} . If condition (12) holds, then we clearly have $r_i^d = \mu_i$. When condition (13) or (14) holds, the transition rate depends on the relative size of the queues. There can be a transition from \mathbf{N}_i^d to \mathbf{N} if the queue for customer P_i is one of the longest queues for any of the servers in the set Q_i . In other words, for a server k in the set Q_i to select queue i , queue i must be one of the longest queues in the set T_k (the set of feasible customers for server k). We denote by B_k the number of customers that have the longest queue in the set T_k . When $B_k > 1$, queue i is selected by server k with probability $1/B_k$. Thus, the transition rate can be written as

$$r_i^d = \begin{cases} \mu_i & \text{if } n_i^d = 1 \text{ and } n_q^d \leq 1, \quad \forall q \neq i, q \in T_i, \omega_i = 1; \\ \sum_{k \in Q_i} \frac{\mu_k \varepsilon_k(P_i)}{B_k} & \text{if } 2 \leq n_i^d \leq b_i, \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 1; \text{ and} \\ \sum_{k \in Q_i} \frac{\mu_k \varepsilon_k(P_i)}{B_k} & \text{if } 1 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, v_i = 0, \end{cases} \tag{17}$$

where B_k is the number of customer types that have the longest queue in the set T_k , and $\varepsilon_k(P_i) = 1$ if the queue of customer type i is one of the B_k longest queues in the set T_k , and 0 otherwise.

The RR-LQF Policy. The sets of entering states are the same for this queue selection policy as in the SP-LQF control policy. Only the transition rates differ. If an arrival of customer type P_k occurs, where P_k is of one of the customer types in set T_i , this arrival has an equal chance of being routed to any of the idle servers in the set Q_k . Thus,

$$r_i^a = \begin{cases} r_i^a = \lambda_i, & \text{if } 1 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 1, \nu_i = 1; \text{ or} \\ & 0 \leq n_i^a \leq b_i - 1 \text{ and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 0, \nu_i = 1; \text{ and} \\ \sum_{k \in T_i} \frac{\nu_k \lambda_k}{\sum_{r \in Q_k} (1 - \delta(n_r))}, & \text{if } n_i^a = 0 \text{ and } n_r^a \leq 1, \quad \forall r \neq i, r \in T_i, \omega_i = 1. \end{cases} \quad (18)$$

The rate r_i^d is the same as in the SP-LQF case.

The RR-SP Policy. The rate r_i^a for the RR-SP policy is the same as in the RR-LQF policy, while the rate r_i^d is given as follows. When condition (12) holds, a transition from N_i^d to N clearly occurs with rate:

$$r_i^d = \mu_i. \quad (20)$$

However, when either condition (13) or (14) holds, the transition rate depends on the strict priority scheme. First note that for n_i^d to decrease by 1, we need a departure from a server that belongs to the set Q_i . Although necessary, this condition is not sufficient since a departure from a server R_k from the set Q_i may not decrease n_i^d by 1, unless the customer type i has the highest priority among all the customer types in queue that can be routed to R_k . This means that the following conditions must be satisfied:

$$\gamma_{ik} < \gamma_{jk} \quad \text{for all } j \in T_k \text{ that satisfy } n_j \geq 2 \text{ and } \nu_j = 1, \omega_j = 1 \text{ or}$$

$$\gamma_{ik} < \gamma_{jk} \quad \text{for all } j \in T_k \text{ that satisfy } n_j \geq 1 \text{ and } \nu_j = 1, \omega_j = 0.$$

Since the departure rate from server R_k is μ_k , the transition rate r_i^d is given by

$$r_i^d = \sum_{k \in Q_i} \mu_k \theta(\gamma_{ik})$$

where

$$\theta(\gamma_{ik}) = \begin{cases} 1, & \text{if } \gamma_{ik}(1 - \delta(n_i - 1)) \leq \gamma_{tk}, \quad \forall k \in Q_i, \quad \forall t \in T_k, \nu_t = \omega_t = 1; \\ & \text{or } \gamma_{ik}(1 - \delta(n_i)) \leq \gamma_{tk}, \quad \forall k \in Q_i, \quad \forall t \in T_k, \omega_t = 0; \text{ and} \\ 0, & \text{otherwise.} \end{cases} \quad (21)$$

Putting it all together, we have:

$$r_i^a = \begin{cases} \mu_i, & \text{if } n_i^d = 1 \text{ and } n_r^d \leq 1, \quad \forall r \neq i, r \in T_i, \omega_i = 1; \\ \sum_{k \in Q_i} \mu_k \theta(\gamma_{ik}), & \text{if } 2 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, v_i = \omega_i = 1; \\ & \text{or } 1 \leq n_i^d \leq b_i \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, v_i = 1, \omega_i = 0; \text{ and} \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

The SP-SP Policy. The rate r_i^a for the SP-SP is the same as in the SP-LQF policy, while the rate r_i^d is the same as in the RR-SP policy. The transition rates for the remaining policies, namely SP-RS and RR-RS, can be determined in a similar fashion. The details are omitted for brevity.

3.3. Systems with a Bound on the Total Number of Customers in the System

Our analysis can be extended to system where there is a global, instead of customer type-specific, bound on the number of customers in the system. This means that the maximum number of customers that can be allowed in the system (regardless of type) is b . With this requirement, it is not difficult to show that a transition from a state \mathbf{N}_i^a occurs only if one of the following conditions holds:

$$n_i^a \geq 1, \quad \sum_{i=1}^q n_i < b, \quad \text{and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, \quad \omega_i = 1, \quad v_i = 1; \tag{23}$$

$$n_i^a \geq 0, \quad \sum_{i=1}^q n_i < b, \quad \text{and } n_l \neq 0, \quad \forall l \neq i, l \in Q_i, \quad \omega_i = 0, \quad v_i = 1; \text{ or} \tag{24}$$

$$n_i^a = 0 \text{ and } n_r^a \leq 1, \quad \forall r \neq i, r \in T_i, \omega_i = 1. \tag{25}$$

Similarly, a transition from \mathbf{N}_i^d to \mathbf{N} occurs only if one of the following conditions is true:

$$n_i^d = 1 \text{ and } n_r^d \leq 1, \quad \forall r \neq i, r \in T_i, v_i = 1; \tag{26}$$

$$2 \leq n_i^d \leq b \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 1; \text{ or} \tag{27}$$

$$1 \leq n_i^d \leq b \text{ and } n_l^d \neq 0, \quad \forall l \neq i, l \in Q_i, \omega_i = 0. \tag{28}$$

The transition rates r_i^a (r_i^d) associated with conditions (23)–(25) [(26)–(28)] are the same as those associated with conditions (8)–(10) [(12)–(14)] of the previous section for each pair of server and customer selection rules we consider.

3.4. Performance Measures

From $p(\mathbf{N})$, we can obtain the marginal probability $p(n_i)$ associated with the state variable n_i . Our primary measure of performance is throughput for each customer type i which can be obtained as follows:

$$\tau_i^p = \lambda_i(1 - p(n_i = b_i)), \quad (29)$$

from which, we can then obtain total system throughput as

$$\tau_s = \sum_{i=1}^n \tau_i^p. \quad (30)$$

We can also derive expressions for throughput due to each server j as

$$\tau_j^R = \mu_j(1 - p(n_j = 0)). \quad (31)$$

Several other measures of performance can be obtained as well, including expected queue size for each customer type, average utilization of each server, expected total WIP in the system, and expected flow time.

4. NUMERICAL EXAMPLES AND INSIGHTS

In this section, we use numerical examples to generate some insights into the behavior of flexible queueing systems. We focus on the role flexibility plays and how it interacts with other system parameters. We also examine the effect of control policies and examine the interactions between control policies, flexibility and asymmetry in system parameters. The treatment is not meant to be comprehensive but simply to illustrate how the model can be useful in developing better understanding of the behavior of flexible queueing systems. We make five main observations. First, we show that additional flexibility without a well-designed control policy can reduce throughput. Second, for symmetric systems, we show that a *chained* configuration of server/customer flexibilities provides most of the throughput benefits of total flexibility. We also show that increased chaining exhibits diminishing returns. Third, for symmetric systems, we show that throughput is maximum when flexibility is balanced among the various customers and the servers. However, we show that this is not necessarily the case for asymmetric systems with heterogeneous servers or asymmetric demand rates. For these systems, we find that chaining is not always superior to a nonchained configuration. Fourth, we examine the impact of control policies under different conditions of asymmetry. We show that there is a range of asymmetry in which the difference in throughput due to different control policies is maximum. Fifth, we illustrate how strict priorities can be used to model a wide range of policies, including those that take customer and server flexibility into account. We show that flexibility-based policies can improve throughput in asymmetric systems with partial flexibility.

OBSERVATION 1: Increasing flexibility for one or more customers can reduce system throughput.

Consider the five scenarios shown in Figure 3. The scenarios correspond to systems with varying customer routing flexibility. Values of system parameters are as follows: $\mu_j = 1$ for $j = 1, 2, \dots, 5$; $\lambda_1 = 5.0$, $\lambda_2 = 1.25$, $\lambda_3 = 0.75$, $\lambda_4 = 0.5$, $\lambda_5 = 0.25$; and $b_i = 2$ for $i = 1, \dots, 5$. The queue selection rule is SP with the customer with the higher arrival rate assigned a lower priority. Server selection is also of the SP type with servers ordered per the priority scheme $\alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 < \alpha_5$ for any customer–server assignment. System throughput

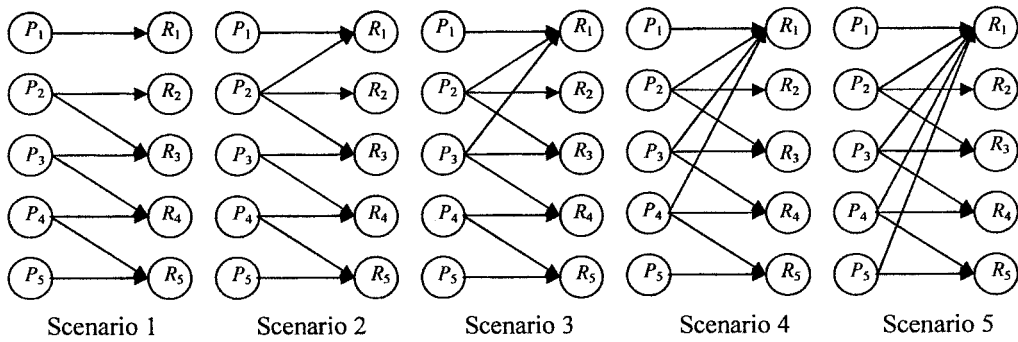


Figure 3. Flexibility scenarios for Observation 1.

corresponding to each scenario is shown in Table 2(a). In each scenario, we increase the flexibility of one or more customers. As we can see, any increase in flexibility leads to a reduction in system throughput. The amount of this decrease is sensitive to whose flexibility, among the customers, is increased. These effects can be explained as follows. In each scenario, greater flexibility is achieved by allowing more customers to be routed to server R_1 . This increases the overall loading of R_1 and increases the probability of balking for all classes that were initially assigned to it (e.g., P_1). In turn, this leads to lower throughput for these customers, which may not always be matched by increased throughput for customers whose flexibility has been increased. This would be particularly the case when the demand rates for the original customers are high or when the flexibility of the newly assigned customer is already high (see Figs. 14 and 15 for additional examples). Clearly, using a better control policy could mitigate these effects. For example, as shown in Table 2(b), using the LQF instead of the SP rule in

Table 2(a). The effect of higher flexibility on throughput.

	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
Scenario 1	0.967741	0.99253	0.654998	0.468458	0.228868	3.3126
	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.967742	0.6471	0.698342	0.57678	0.422632	3.3126
Scenario 2	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.978557	0.610363	0.67052	0.56896	0.420293	3.24869
	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.813833	1.0776	0.659329	0.468943	0.228985	3.24869
Scenario 3	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.982466	0.613041	0.654076	0.547927	0.414268	3.21178
	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.749	1.07363	0.689526	0.470339	0.229287	3.21178
Scenario 4	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.98429	0.614561	0.655705	0.535872	0.400271	3.1907
	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.719609	1.07099	0.689018	0.481092	0.229986	3.1907
Scenario 5	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.985452	0.615856	0.65734	0.53666	0.373837	3.16914
	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.691541	1.06837	0.688001	0.481508	0.239724	3.16914

Table 2(b). The effect of higher flexibility on throughput (LQF Policy).

Scenario 1	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.967742	0.524806	0.721104	0.590107	0.524955	3.32871
Scenario 2	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.967742	1.02965	0.648367	0.46144	0.221512	3.32872
Scenario 3	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.974386	0.503267	0.708753	0.588273	0.524766	3.29945
Scenario 4	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.88841	1.07656	0.651316	0.46162	0.221533	3.29944
Scenario 5	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.972746	0.518508	0.710992	0.575389	0.523418	3.30105
Scenario 6	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.908242	1.03606	0.67215	0.462915	0.221684	3.30105
Scenario 7	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.970527	0.523728	0.719368	0.581548	0.514851	3.31002
Scenario 8	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.934724	1.03055	0.650816	0.471252	0.222678	3.31002
Scenario 9	τ_1^R	τ_2^R	τ_3^R	τ_4^R	τ_5^R	τ_S
	0.970812	0.524269	0.720242	0.585485	0.502968	3.30378
Scenario 10	τ_1^P	τ_2^P	τ_3^P	τ_4^P	τ_5^P	τ_S
	0.928066	1.03008	0.649418	0.464136	0.232079	3.30378

selecting customers tends to dampen the above effects. These effects would altogether disappear if an optimal control policy were used.

Several recent studies (Hopp, Tekin, and van Oyen [17], Jordan and Graves [18], and Sheikhzadeh, Benjaafar, and Gupta [25]) have shown in a variety of contexts that a chained configuration in which each customer can be routed to one of two neighboring servers and each server can process customers from two neighboring classes achieves most of the benefits of total flexibility. In the following observation, we confirm that this is true for symmetric systems (i.e., systems with identical servers, customer classes, and queue sizes). We also show that higher order chaining exhibits diminishing returns. A system with chaining of order K ($K \geq 2$) refers to systems where each customer can be routed to K neighboring servers and each server can process K neighboring customers. Systems with varying orders of chaining are shown in Figure 4.

OBSERVATION 2: In a symmetric system, increased chaining exhibits diminishing returns with most of the value of total flexibility achieved with the initial chain.

To illustrate the above result, consider the system scenarios shown in Figure 4. The scenarios correspond to systems with progressively higher orders of chaining. Scenario 1 corresponds to a system with no flexibility and scenario 6 to a system with total flexibility (chaining of order $K = 6$). For each scenario, we obtain the corresponding throughput. Representative results are shown in Figure 5 (system parameters for the data we show are: $\mu_j = 1$ for $j = 1, \dots, 6$; $\lambda_i = \lambda$ and $b_i = 2$ for $i = 1, \dots, 6$, and the control policy is SP-LQF). As we can see, most of the increase in throughput is achieved by forming the initial chain. Higher orders of chaining increase throughput only marginally and in progressively diminishing amounts. These results suggest that total flexibility, which corresponds to chaining of maximal order, is generally unjustified if increases in flexibility require significant investments.

Although increases in chaining exhibit diminishing returns, this is not true with increases in flexibility in general. In Figure 7, we illustrate the effect of gradually increasing flexibility one

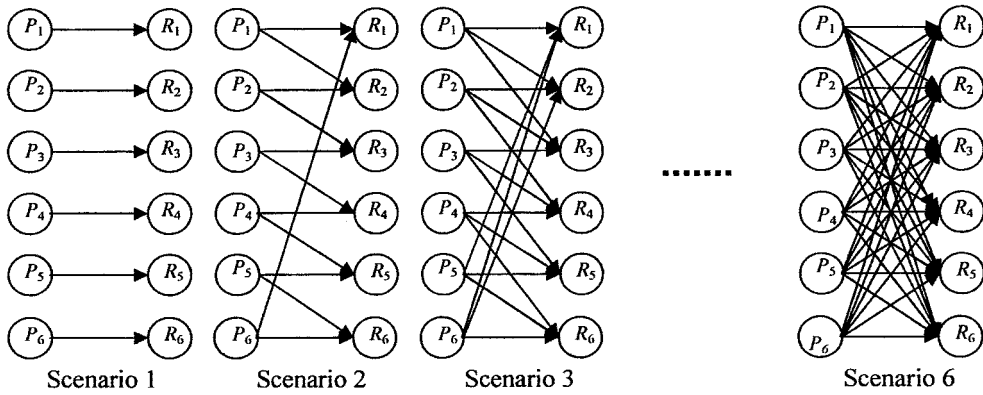


Figure 4. Flexibility scenarios for Observation 2.

customer at a time (i.e., adding a single arc at a time to the graph). In this case, each level of chaining is attained via a series of increases to the number of arcs (see Fig. 6). From Figure 7, we see that, surprisingly, the marginal increase in throughput is not always decreasing in the number of arcs. In fact, within each level of chaining, it exhibits increasing returns, with the last link in each chain realizing the largest marginal gain in throughput. This result further underscores the importance of forming fully connected chains. This is different from the observation made in Hopp, Tekin, and van Oyen [17] for serial systems, where only the last link in the chain is found to exhibit increasing returns.

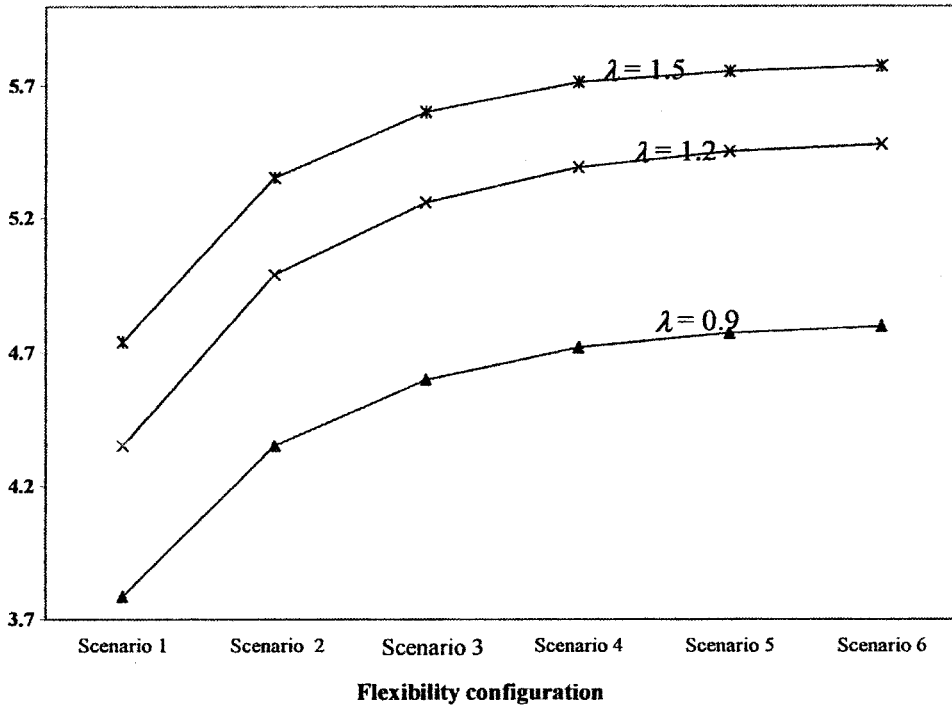


Figure 5. The effect of chaining on system throughput.

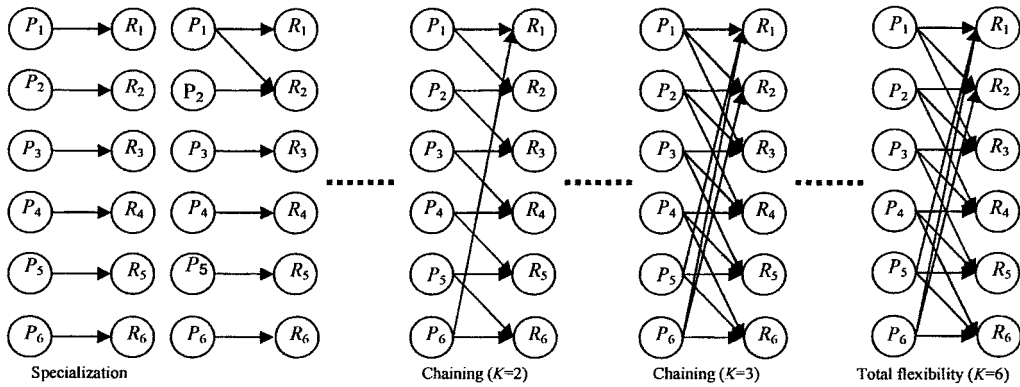


Figure 6. Increasing flexibility one arc at a time.

OBSERVATION 3: In a symmetric system, a balanced allocation of flexibility among the customers leads to higher throughput.

Consider the four system scenarios shown in Figure 8. The scenarios correspond to systems where the total amount of customer flexibility (i.e., the number of edges connecting customers to servers) remains fixed but the allocation of this flexibility (edges) is varied. System throughput for varying levels of flexibility asymmetry is shown in Figure 9 (values of system parameters

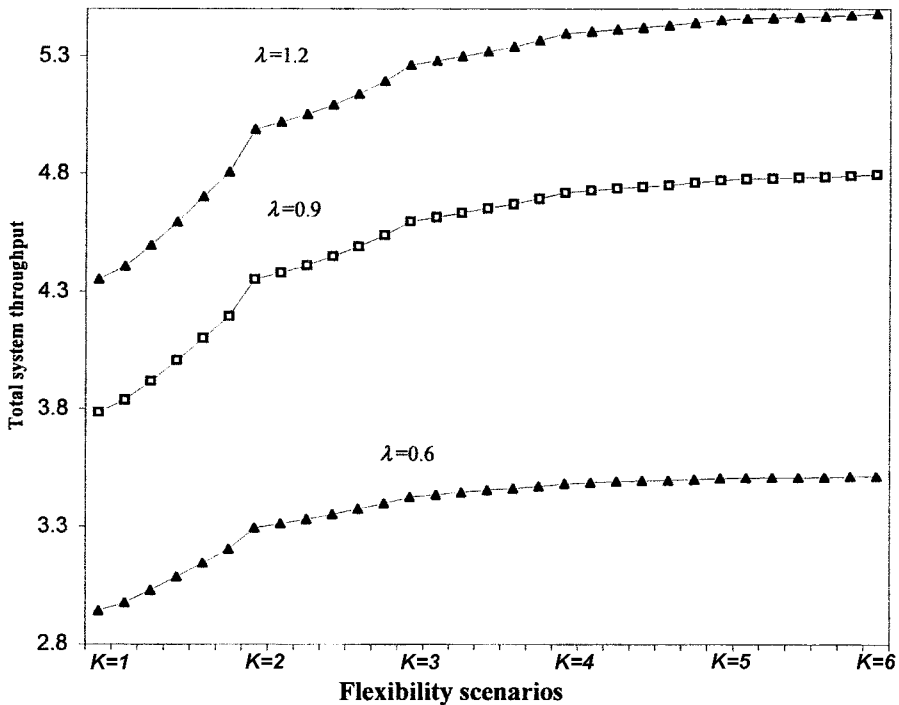


Figure 7. The effect of flexibility on system throughput.

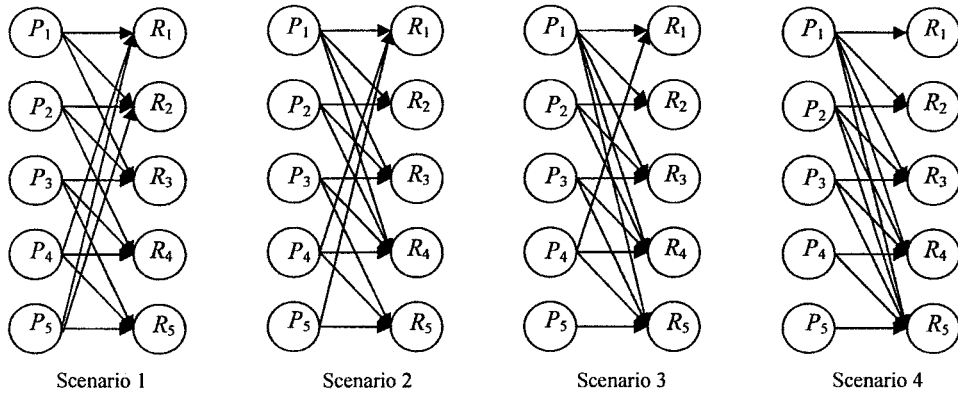


Figure 8. Flexibility allocation scenarios for Observation 3.

here and in subsequent observations, unless noted, are the same as in Observation 2; the results are shown for the SP-LQF policy with similar effects observed for other policies), from which it is clear that increased balancing in customer flexibility leads to higher throughput.

A balanced allocation is however not always optimal. In fact, as we note in the following observation, an asymmetric allocation of flexibility can be more beneficial when there is asymmetry in customer demand rates.

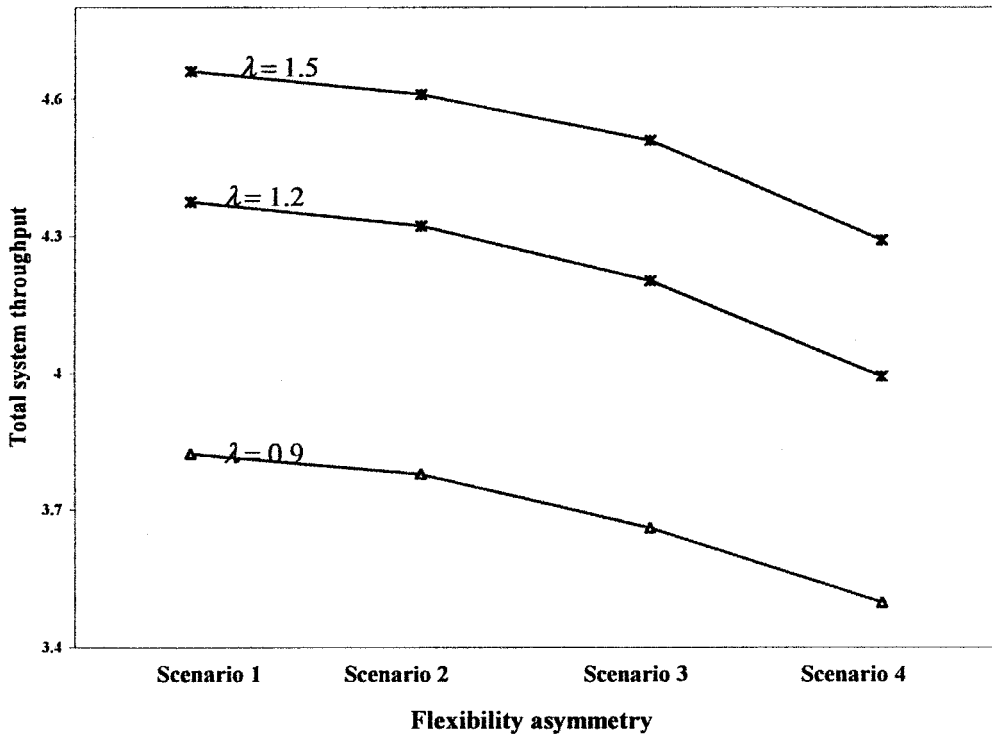


Figure 9. The effect of flexibility asymmetry on system throughput.

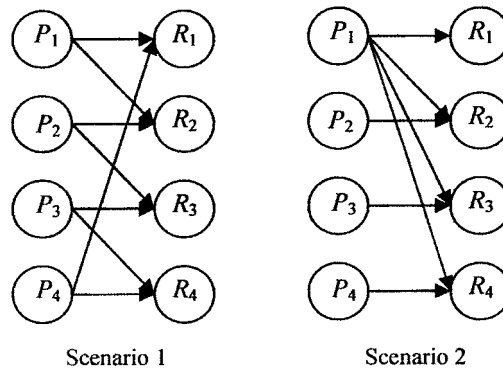


Figure 10. Flexibility allocation scenarios for Observation 4.

OBSERVATION 4: In systems with asymmetric demand rates, an asymmetric allocation of flexibility can (but not always) lead to higher throughput.

We illustrate the above by comparing the performance of the two flexibility scenarios shown in Figure 10. In order to examine the effect of demand asymmetry, we vary the fraction of the demand due to customer class P_1 while keeping the total demand on the system constant. We assume the demand rate for customers 2, 3, and 4 are maintained equal. In this fashion, a simple measure of demand asymmetry is the ratio of the demand rate for customer 1 to the demand rate of any of the other customers. We denote this ratio by V and vary it from 1 to 5. The results are shown in Figure 11. We can clearly see that when demand is sufficiently asymmetric ($V > 2$), scenario 2 leads to higher throughput. This effect is more significant in the more asymmetric systems. Note also that while higher demand asymmetry decreases throughput in a balanced system, throughput actually increases with demand asymmetry in the unbalanced one.

An important implication of the above is that chaining is not always superior to nonchaining. In fact, in asymmetric systems, we find that there is always a nonchained configuration that performs better than chaining. It is tempting to argue that, in asymmetric systems, customers with the higher demand rates should be assigned greater flexibility. However, we find that this is not always true. The optimal allocation of flexibility generally depends on both customer demand rates as well as the capacity of the servers to which customers are routed. To see this, consider the extreme case where the customer with the highest demand rate can be routed to only one server but this server has nearly zero mean processing time. In this case, clearly, providing more flexibility to this customer will do little to improve throughput. In general, we find that the optimal allocation of flexibility involves a complex relationship between customer demand rates and server capacities.

Asymmetry also affects the difference in performance between different control policies. In particular, there is a range of asymmetry, in which this difference is maximum.

OBSERVATION 5: The difference in performance between different queue selection rules is affected by demand asymmetry. There is a level of asymmetry under which this difference is maximum.

We consider a system with three servers and three customer types. We vary the asymmetry in the demand rates of the different customers by increasing the relative contribution of different customer types to total demand while maintaining total demand constant. We use

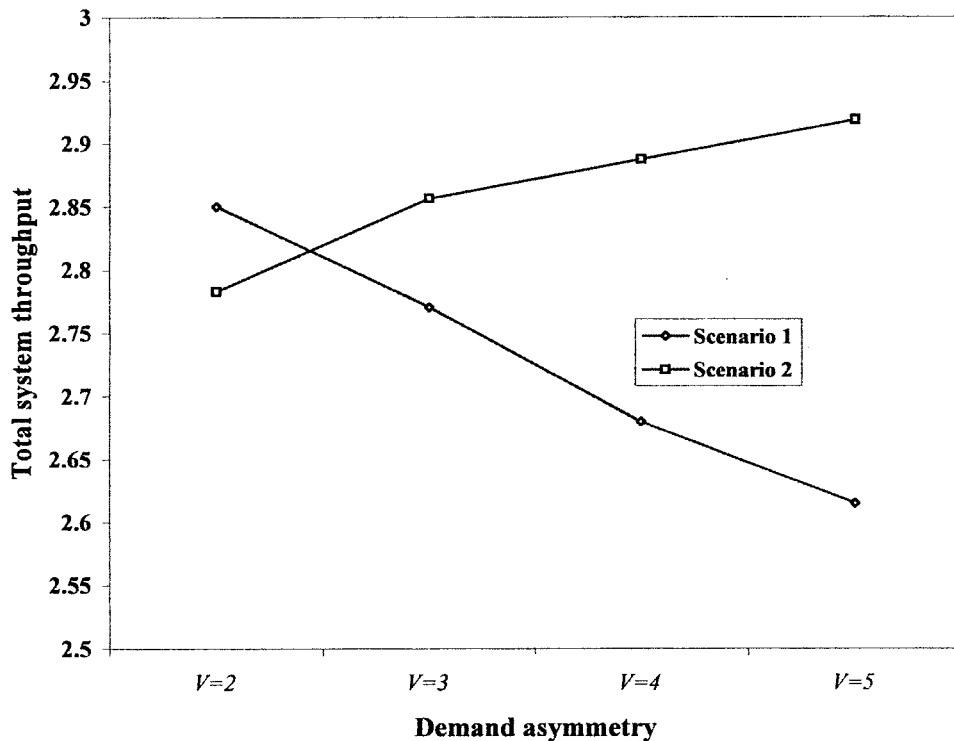


Figure 11. The effect of demand asymmetry.

$$V = \frac{\lambda_1}{\lambda_2} = \frac{\lambda_2}{\lambda_3}$$

as our measure of demand asymmetry. We compare two customer selection rules, SP_1 and SP_2 . Under policy SP_1 , we give highest priority to the customer class with the lowest demand rate (class 3). Under SP_2 , we give highest priority to the customer class with the highest demand rate (class 1). The difference in throughput between the two rules is shown in Figure 12 (the results are shown for $\mu_i = \mu = 1$ for $i = 1, \dots, 3$; the notation $\rho = (\lambda_1 + \lambda_2 + \lambda_3)/3$ is used to denote overall system loading). As we can see, the difference is maximum when asymmetry is in the mid-range.

A similar effect is observed with respect to server selection rules and asymmetries in processing rate. In that case, we find that the impact of server selection rules is insignificant where processing rate asymmetry is either very high or very low. It is however, significant when the asymmetry is in the mid-range.

OBSERVATION 6: In asymmetric systems, control policies that take advantage of customer and server flexibility, as well as the available capacity to these customers and the customers demand rates, can lead to higher throughput.

We compare the following four policies. Under the first policy, policy H_1 , servers and customer classes are assigned priorities based on their flexibility. The server with the least

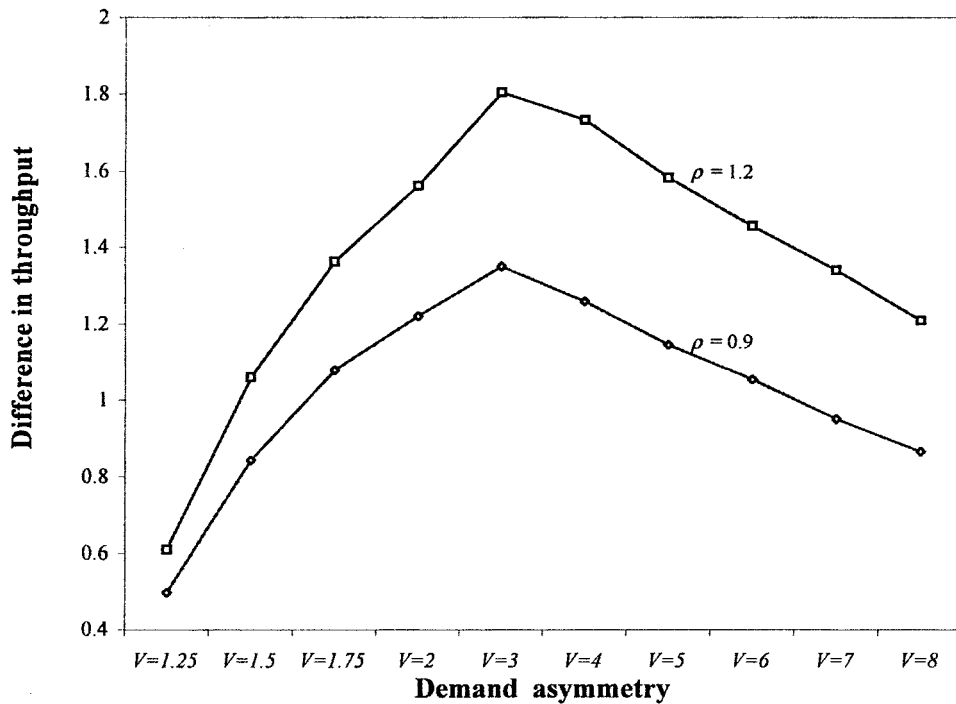


Figure 12. The effect of demand on performance of queue selection policies.

flexibility is assigned highest priority and the same for customers. Flexibility is measured by the number of customers a server can process or the number of servers to which a customer can be assigned. Under the second policy, policy H_2 , servers are assigned priorities based on the ratio of the sum of arrival rates of all customers that can be assigned to a server to the server's processing rate. That is, each server j ($j = 1, \dots, m$) is assigned a priority based on the ratio $\sum_{i=1}^n a_{ij}\lambda_i/\mu_j$ with lower ratios corresponding to higher priorities. Similarly, customers are assigned priorities based on the ratio of a customer's arrival rate to the sum of the processing rates of servers to which the customer can be assigned. That is, each customer i ($i = 1, \dots, n$) is assigned a priority based on the ratio $\lambda_i/\sum_{j=1}^m a_{ij}\mu_j$, with higher ratios corresponding to higher priorities. Hence, policy H_2 gives priority to servers with the least potential load and to customers with the least potential available capacity. Under the third policy, policy H_3 , servers (customers) are assigned priorities based on their service (arrival) rates with higher rates corresponding to higher priorities. The fourth policy, H_4 , is included for benchmarking and corresponds to the RR-RS policy.

We consider a system with five customer classes and five servers. We consider 21 flexibility configuration scenarios. We start with a dedicated scenario in which each customer can be routed to only one server and each server can process only one customer (scenario 1). In scenarios 2–6, we increase flexibility by adding one link at a time between customers and servers until we reach a chained configuration (i.e., in scenario 2 customer 1 can be assigned to servers 1 or 2, in scenario 3, customer 1 can be assigned to servers 1 or 2 and customer 2 can be assigned to servers 2 or 3, etc.). In scenarios 7–21, we further increase the flexibility one customer at a time and one link at a time starting with customer 1 until each customer has full flexibility (e.g., in scenario 7, customer 1 can be assigned to either servers 1, 2, or 3, in scenario

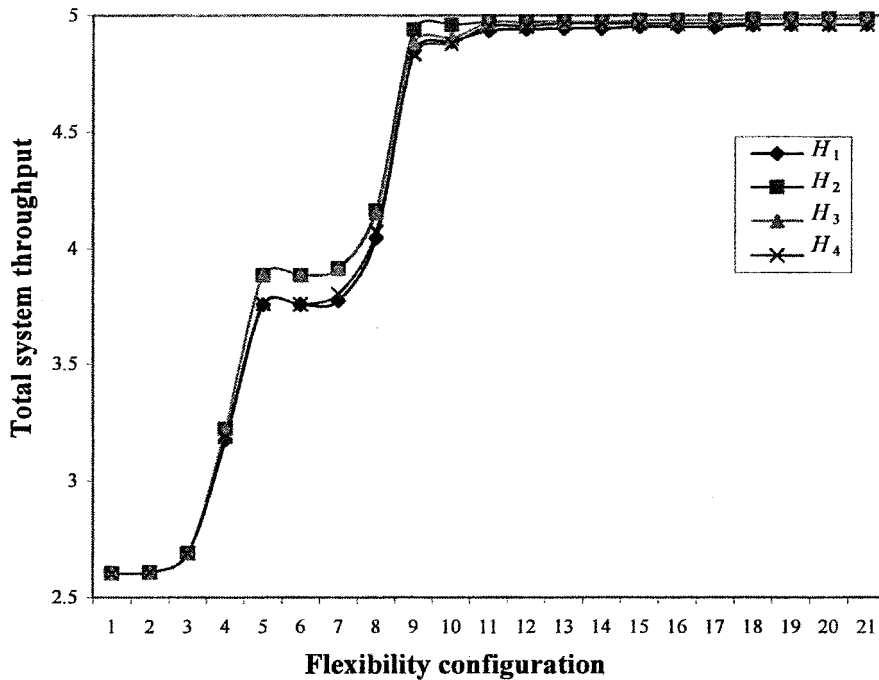


Figure 13. The joint effect of control policies and flexibility (scenario A_1 - L_5).

8, it can be assigned to either servers 1, 2, 3, or 4, etc.). Scenario 21 corresponds to a system with full flexibility where any customer can be routed to any server and any server can process any customer.

We consider five levels of system loading, $L_1 = 0.6$, $L_2 = 0.9$, $L_3 = 1.2$, $L_4 = 1.5$, and $L_5 = 1.8$, where $L_i = \sum_{j=1}^n \lambda_j / \sum_{j=1}^m \mu_j$ for $i = 1, \dots, 5$. We also consider four levels of demand and service rate asymmetry, A_1 , A_2 , A_3 , and A_4 . For asymmetry level A_1 , $\mu_1 = 0.1$ and $\mu_j = \mu_{j-1} + 0.45$ for $j = 2, \dots, 5$, and $\lambda_5 = 0.1 \times L_1$, and $\lambda_i = \lambda_{i+1} + 0.45 \times L_i$ for $i = 2, \dots, 5$. For asymmetry level A_2 , we assign equal processing rates and services rates to all the customers and all the servers. For level A_3 , we invert the assignments in level A_1 , by setting $\mu_5 = 0.1$ and $\mu_j = \mu_{j+1} + 0.45$; $j = 1, \dots, 4$, and by letting $\lambda_1 = 0.1 \times L_1$; $\lambda_i = \lambda_{i-1} + 0.45 \times L_i$ for $i = 2, \dots, 5$. For level A_4 , the demand rates are the same as in A_1 and the service rates are the same as in A_3 .

The effect of the different control policies for the different flexibility configurations and asymmetry levels is illustrated in Figures 13–15 (the results are shown for systems with buffer levels $b_i = 3$ for $i = 1, \dots, 5$, but are qualitatively similar for other buffer values). As we can see, from Figures 13 and 14, policy H_2 dominates the other policies, suggesting that an optimal policy would take into account both the flexibility of servers and customers as well as the capacity available to the customers relative to their demand rates. The results also suggest that when there are significant asymmetries in demand and service rates, assigning priorities based on these rates could be more helpful than assigning priorities based on flexibility (see Fig. 13). Figure 14 illustrates how higher flexibility under a suboptimal policy, such as H_3 or H_4 , could lead to lower throughput (this effect is observed even when there is symmetry in demand and service rates).

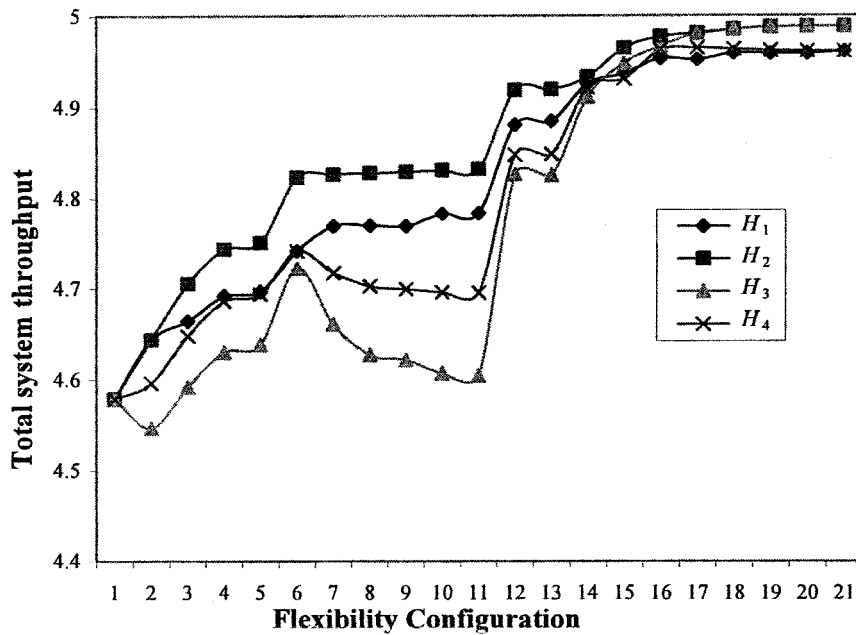


Figure 14. The joint effect of control policies and flexibility (scenario A_4-L_5).

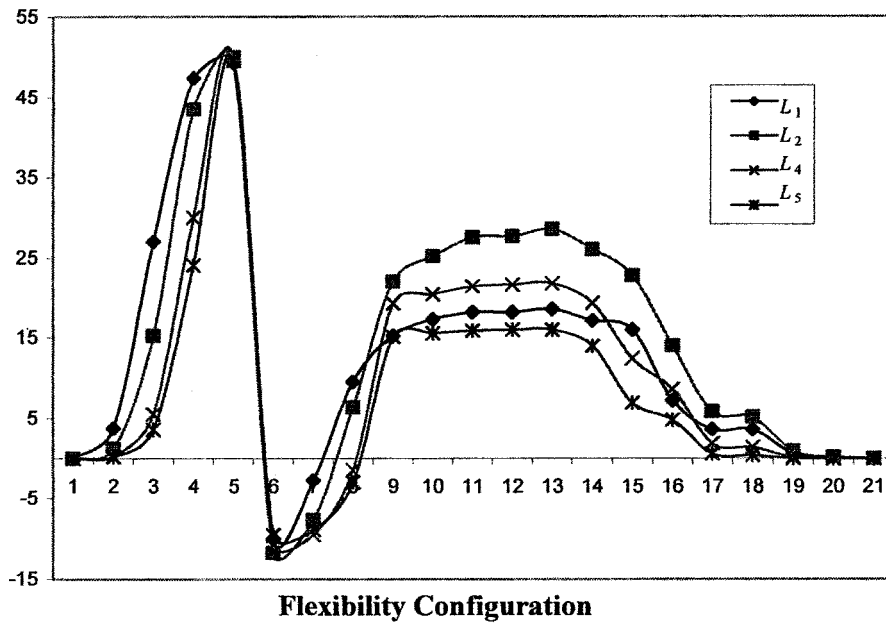


Figure 15. The effect of flexibility on the percentage difference in throughput between systems A_1 and A_3 (policy H_2).

Figures 13 and 14 illustrate how the effect of increasing flexibility could be different depending on the asymmetry in demand and service rates. In particular, flexibility is most valuable when it is associated with either the fastest servers or the customers with the largest demand (this explains the observed large jumps in throughput with one step increases in flexibility). More importantly, the results show that although there is value to choosing a good control policy, the effect of control policies is less significant than that of flexibility. An increase in flexibility, if it is carefully designed, can lead to significantly larger improvements in throughput than can be achieved by improving control alone.

Finally, we note that the effect of flexibility can vary widely depending on the asymmetry in demand and service rates. For the same number of *links* between customers and servers (and for the same aggregate capacity and demand), throughput can vary widely depending on how capacity (demand rates) is distributed among the servers (customers). This is dramatically illustrated in Figure 15 where the percentage difference in throughput between systems with asymmetry A_1 and systems with asymmetry A_3 is shown for different levels of flexibility and different levels of loading. Note that an increase in flexibility can switch the ordering of the two systems in either direction, so that an increase in flexibility can make a particular distribution of capacity and demand rates more or less desirable. This points to the need to carry out, whenever possible, capacity allocation jointly with flexibility design.

6. CONCLUDING COMMENTS

In this paper, we presented a framework for the representation, modeling and analysis of flexible queueing systems. The analytical model allows for the analysis of general system configurations with an arbitrary number of customers and servers, an arbitrary flexibility matrix, asymmetric demand and processing rates, asymmetric bounds on customer queue sizes, and a wide range of control policies. The models are generic and can be used to analyze flexible queueing systems in a variety of applications. They can also serve as a decision support tool for the planning and design of these systems. Furthermore, our characterization of the probability distribution of system states and the transition probability between these states offers the opportunity to formulate optimal control problems (e.g., using the framework of a Markov decision process).

Our model can be extended in a variety of ways. This includes relaxing the assumptions of Poisson demand and exponential processing times and allowing service times to vary by customer and server. It would then be useful to examine the impact of demand and service variability on different system configurations and different control policies. In many applications, such as manufacturing, the processing of multiple customers on the same servers is accompanied by losses in efficiencies due to switchover times or costs. It would be worthwhile to extend the models to account for these inefficiencies. In other applications, such as telecommunication networks, the processing of a customer requires the simultaneous contribution of more than one server. For these applications, there is a need to extend the analysis to systems where customers consume varying amounts of capacity.

ACKNOWLEDGMENTS

The authors are grateful to William Cooper and Mike Taaffe for many useful comments on an earlier draft. The research of the second author is supported by NSF through Grants DMII-9908437 and 9988721. We are also grateful to an anonymous reviewer for many helpful comments.

REFERENCES

- [1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Networks flows*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [2] O.Z. Aksin and F. Karaesmen, *Designing flexibility: Characterizing the value of cross-training practices*, Working Paper, INSEAD, Cedex, France, 2002, www.ku.edu.tr/~fkaraesmen/pdfs/flex2102.pdf.
- [3] M. Armony, *Queueing networks with interacting service resources*, Ph.D. thesis, Stanford University, Stanford, CA, 1999.
- [4] S. Benjaafar, Performance bounds for the effectiveness of pooling in multi-processing systems, *European J Oper Res* 87 (1995), 375–388.
- [5] S. Benjaafar, Demand allocation in multi-product/multi-facility make-to-stock systems, *Management Sci* (2004), in review.
- [6] S. Benjaafar and D. Gupta, Workload allocation in multi-product/multi-facility production systems with setup times, *IIE Trans* 31 (1998), 339–352.
- [7] J.A. Buzacott, Commonalities in reengineered business processes: Models and issues, *Management Sci* 42 (1996), 768–782.
- [8] J.A. Buzacott and J.G. Shanthikumar, *Stochastic models of manufacturing systems*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [9] J.M. Calabrese, Optimal workload allocation in open networks of multi-server queues, *Management Sci* 38 (1992), 1792–1802.
- [10] R. Cooper, *Introduction to queueing theory*, 2/E, North-Holland, Amsterdam, 1981.
- [11] S. Even and R.E. Tarjan, Network flow and testing graph connectivity, *SIAM J Comput* 4 (1975), 507–518.
- [12] N. Gans, G. Koole, and A. Mandelbaum, Telephone call centers: Tutorial, review and research prospects, *Manufacturing Serv Oper Management* 5 (2003), 79–141.
- [13] O. Garnett and A. Mandelbaum, *An introduction to skills routing and its operational complexities*, Teaching Note, Technion, Haifa, Israel, 2001, www.ie.technion.ac.il/serveng/Homeworks/HW9.pdf.
- [14] B. Hajek, Optimal control of two interacting service stations, *IEEE Trans Automat Control* AC-29 (1984).
- [15] R.W. Hall, *Queueing methods: For services and manufacturing*, Prentice Hall, Englewood Cliffs, NJ, 1991.
- [16] J.M. Harrison and M.J. Lopez, Heavy traffic resource pooling in parallel server systems, *Queueing Syst* 33 (1999), 339–368.
- [17] W. Hopp, E. Tekin, and M.P. Van Oyen, Benefits of skill chaining in production lines with cross-trained workers, Working Paper, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 2001.
- [18] W.J. Jordan and S.C. Graves, Principles on the benefits of manufacturing process flexibility, *Management Sci* 41 (1995), 577–594.
- [19] L. Kleinrock, *Queueing systems: Computer applications*, Wiley, New York, 1976, Vol. 2.
- [20] G. Koole and A. Mandelbaum, Queuing models of call centers, an introduction, *Ann Oper Res* 113 (2002), 41–59.
- [21] C.N. Laws, Resource pooling in queueing networks with dynamic routing, *Adv Appl Probab* 24 (1992), 699–726.
- [22] A. Mandelbaum and M. Reiman, On pooling in queueing networks, *Management Sci* 44 (1998), 971–981.
- [23] K.W. Ross, *Multiservice loss models for broadband telecommunication networks*, Springer-Verlag, London, 1995.
- [24] L.I. Sennot, *Stochastic Dynamic Programming and the Control of Queueing Systems*, Wiley, New York, 1999.
- [25] M. Sheikhzadeh, S. Benjaafar, and D. Gupta, Machine sharing in manufacturing systems: flexibility versus chaining, *Int J Flexible Manuf Syst* 10 (1998), 351–378.
- [26] R.A. Shumsky, Approximation and analysis of a call center with flexible and specialized servers, Working Paper, University of Rochester, Rochester, 2003, http://omg.simon.rochester.edu/omg-HOME/shumsky/flex_serv.PDF.
- [27] D.R. Smith and W. Whitt, Resource sharing for efficiency in traffic systems, *Bell Syst Tech J* 60 (1981), 39–55.

- [28] K. Stecke and J.J. Solberg, The optimality of unbalancing both workloads and machine group sizes in closed queueing networks of multiserver queues, *Oper Res* 45 (1985), 882–910.
- [29] S. Stidham, Jr. and R. Weber, A survey of Markov decision models for control of networks of queues, *Queueing Syst* 13 (1993), 291–314.
- [30] Y.T. Wang and R.J.T. Morris, Load sharing in distributed systems, *IEEE Trans Comput* C-34 (1985), 204–217.
- [31] W. Winston, Optimality of shortest line discipline, *J Appl Probab* 14 (1977), 181–189.
- [32] R. Weber, On the optimal assignment of customers to parallel queues, *J Appl Probab* 15 (1978), 406–413.
- [33] W. Whitt, Stochastic models for the design and management of customer contact centers: Some research directions, Working Paper, Columbia University, New York, 2002, www.columbia.edu/~ww2040/IEOR6707F02.html.