MANAGEMENT SCIENCE
Vol. 55, No. 8, August 2009, pp. 1353–1367
ISSN 0025-1909 | EISSN 1526-5501 | 09 | 5508 | 1353

DOI 10.1287/mnsc.1090.1025
© 2009 INFORMS

Responding to Unexpected Overloads in Large-Scale Service Systems

Ohad Perry, Ward Whitt
Department of Industrial Engineering and Operations Research, Columbia University,
New York, New York 10027 {op2105@columbia.edu, ww2040@columbia.edu}

We consider how two networked large-scale service systems that normally operate separately, such as call centers, can help each other when one encounters an unexpected overload and is unable to immediately increase its own staffing. Our proposed control activates serving some customers from the other system when a ratio of the two queue lengths (numbers of waiting customers) exceeds a threshold. Two thresholds, one for each direction of sharing, automatically detect the overload condition and prevent undesired sharing under normal loads. After a threshold has been exceeded, the control aims to keep the ratio of the two queue lengths at a specified value. To gain insight, we introduce an idealized stochastic model with two customer classes and two associated service pools containing large numbers of agents. To set the important queue-ratio parameters, we consider an approximating deterministic fluid model. We determine queue-ratio parameters that minimize convex costs for this fluid model. We perform simulation experiments to show that the control is effective for the original stochastic model. Indeed, the simulations show that the proposed queue-ratio control with thresholds outperforms the optimal fixed partition of the servers given known fixed arrival rates during the overload, even though the proposed control does not use information about the arrival rates.

Key words: service systems; call centers; overload controls; queue-ratio routing, many-server queues; deterministic fluid models
History: Received August 27, 2008; accepted March 15, 2009, by Paul H. Zipkin, operations and supply chain management. Published online in *Articles in Advance* June 18, 2009.

1. Introduction

In a large-scale service system, such as a call center, under normal circumstances the arrival rates vary by time of day in a predictable way, and the staffing responds to that anticipated pattern, typically with fixed staffing levels over specified time intervals; see Aksin et al. (2007) and Gans et al. (2003) for background. However, occasionally, for various reasons, there may be unforeseen surges in demand, going significantly beyond the usual fluctuations, and lasting for a significant period of time. A demand surge might occur because of a catastrophic event in emergency response, a system failure experienced by an alternative service provider, or an unanticipated intense television advertising campaign in retail. Such unexpected demand surges typically cause congestion that cannot be eliminated entirely. Because the demand surge is sudden and unexpected, it may not be possible to immediately change the staffing level.

Fortunately, there may be an opportunity to alleviate the congestion caused by the overload by getting help from another service system, which ordinarily operates independently. For example, with the reduction of telecommunication costs, it is more and more common to have networked call centers, often geographically dispersed, even on different continents. Such sharing is typically possible among different hospitals in a metropolitan area. It is often desirable to operate these service systems separately, but their connection provides opportunities, in particular, to provide assistance under overloads. In this paper we consider how that might be done and how to assess the costs and benefits.

An important consideration is that we typically do not want sharing under normal loads. One reason is that it is easier to manage the different facilities separately, e.g., by maintaining clear accountability. Another reason is that the agents in each service facility may be less effective and/or less efficient serving the customers from the other system, because each requires specialized skills not required for the other. We want to consider the case in which serving the other class is possible, but that there are penalties for doing so. We will assume that the service rates are slower for nondesignated agents.

The proposed overload control applies directly to separate service systems run by a single organization, but could also be adopted by two different organizations by mutual agreement. Our analysis provides useful information about the likely consequences of any agreement, which should facilitate making the agreement. Current practice for call centers (that we are aware of) is limited to sharing within a single organization, and then only manually or on a regular basis under normal loading. Load-balancing schemes used in practice are described in §5.3 of Gans et al. (2003).

Thus, our goal is to develop a control to automatically detect when an overload has occurred (in either system, or in both) and, then, before the staffing levels can be changed, reduce the resulting congestion by activating appropriate sharing from agents in the other system. We also want to prevent undesired sharing under normal loads. By focusing on this overload problem, we aim to contribute new insight into the longstanding question about the costs and benefits of resource pooling; see §4.2 of Aksin et al. (2007) and references therein. Here we focus on a situation where we want to turn on and off the pooling.

1.1. Organization of the Paper

We start in §2 with a literature review. Next, in §3, we introduce our proposed modelling approach. As an idealized model of two large-scale service systems, which ordinarily operate separately, but have the capability of serving customers from the other system, we consider the Markovian X call-center model having two homogeneous customer classes and two homogeneous agent pools, where all the agents are cross trained but serve the other class inefficiently. For clarity, we provide a concrete example. We then introduce a cost framework to evaluate alternative controls. We indicate how we specify an overload incident and how we evaluate the performance consequence.

In §4, we introduce the proposed control, which is a variant of the queue-ratio controls introduced by Gurvich and Whitt (2009a, b). After reviewing that control (without thresholds), we show that it can perform very poorly for this unintended application, because it can induce inefficient sharing simultaneously in both directions. We then introduce our proposed alternative, which includes two thresholds, one for each direction of sharing.

In §5, we introduce a deterministic fluid model to approximate the overloaded system after the overload incident has occurred. We then introduce a convex cost structure and show how to select queue-ratio functions to minimize the long-run average cost in the overload incident for the fluid model. We then develop a numerical algorithm to compute the optimal queue-ratio functions for arbitrary convex cost functions. We exhibit explicit formulas for the optimal queue-ratio functions for special structured separable cost functions in §EC.4 in the e-companion.[1]

In §6, we discuss how to set the threshold parameters. In §7, we conduct simulation experiments to show that the optimal control for the fluid model is effective for the stochastic X model. Finally, we state our conclusions in §8. Supporting material appears in the e-companion.

2. Literature Review

In this paper, we contribute to the literature on overload (or congestion) control in queueing systems. There is a substantial literature studying controls that route (or assign) customers (or jobs) to servers, possibly exploiting thresholds. Many of these papers, such as Bell and Williams (2005) and references therein, focus on single-server systems without customer abandonment, whereas we focus on many-server systems with customer abandonment; we only discuss the many-server literature. (The distinction is between routing to one of several servers, as opposed to routing to one of several pools of servers.) It is now understood that the presence of many servers changes the problem; e.g., see Gurvich and Whitt (2009a). One feature of many-server systems with customer abandonment we will exploit is the rate at which the transient distribution approaches its steady-state limit: it tends to be much faster for many-server queues. In particular, the systems we consider tend to reach steady state in a few mean service times; we elaborate in §EC.1. Hence, in our analysis of performance during an overload incident, we approximate using the new steady state, determined by the new arrival rates (assumed constant). Customer abandonments ensure that the system remains stable.

Our paper can also be viewed as a contribution to the call-routing problem for multiclass and multisite call centers with skill-based routing; see §5 of Gans et al. (2003) and §§2.3.3, 4.1, and 4.2 of Aksin et al. (2007). Others have proposed responding to stochastic fluctuations and unexpected overloads by modulating demand in different ways: (i) admission control; (ii) making delay announcements that may induce customers to leave, use a different service channel (e.g., e-mail instead of voice), or call back later; and (iii) acting to reduce service times, e.g., by curtailing cross-selling activities; see §3 of Aksin et al. (2007) and Armony and Gurvich (2006).

In contrast, our paper relates to the larger literature exploiting server flexibility (supply-side management). One approach is to have extra temporary servers available on short notice; see Bhandari et al. (2008) and references therein. Instead, we propose using servers that are already working; i.e., we propose a form of resource pooling, which exploits cross training; see §4.2 of Aksin et al. (2007) and §5.1 of Gans et al. (2003). As should be anticipated, though, our

[1] An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

control tends to be more effective in alleviating congestion (rather than just balancing the service degradation) when the less-loaded system actually has some slack. Our work draws on the queue-ratio control proposed in Gurvich and Whitt (2009a, b), which applies to very general network topologies. Here we consider the relatively difficult X model, allowing sharing in both directions (see Figure 1), but our approach makes the model behave more like the N model; see Tezcan and Dai (2009).

However, we make significant departures from the previous literature. First, we want resource sharing only in the presence of the unanticipated overload, and only in the proper direction, which depends on the nature of the overload. Hence, we turn on and off the sharing. Second, we regard the overload as a rare, exceptional, unanticipated event, rather than a stochastic fluctuation in demand. Thus, we think that it is inappropriate to perform a long-run steady-state analysis of system performance with alternating normal and overload periods (although that could be done). Instead, we focus on a single overload in isolation.

Because the system tends to be overloaded, even after sharing has been activated, system performance tends to be well approximated by deterministic fluid approximations, as in Whitt (2004). From a heavy-traffic perspective, the system operates in the so-called efficiency driven (ED) many-server heavy-traffic regime, instead of the quality-and-efficiency-driven (QED) regime; see Garnett et al. (2002) and Gurvich and Whitt (2009a, b). Our paper also relates to the literature on arrival-rate uncertainty; see §4.4 of Gans et al. (2003) and §2.4 of Aksin et al. (2007). Arrival-rate uncertainty also tends to make deterministic fluid approximations remarkably accurate; e.g., see Whitt (2006), Bassamboo and Zeevi (2009), and references therein.

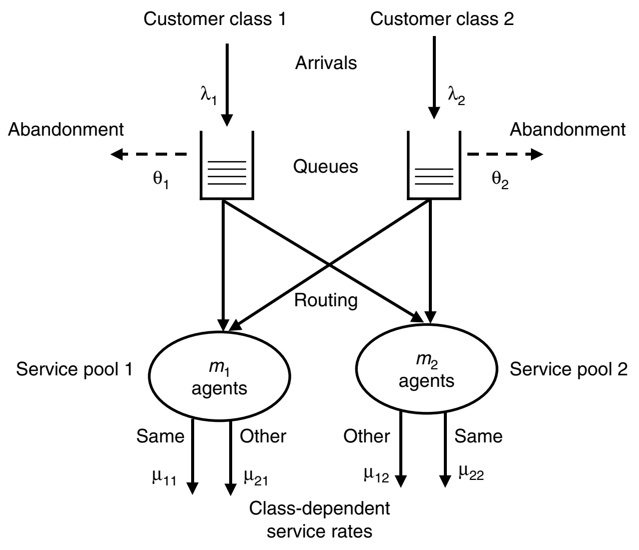
In closing, we mention the large literature on detection outside queueing, such as control charts in statistical quality control, sequential analysis, and change point problems, but we make no direct contact with it. However, detecting the new arrival rate is not the only issue: in simulations, our proposed control outperforms the optimal fixed partition of the servers given known arrival rates during the overload, even though our proposed control does not use direct information about the arrival rates; see §7.2.

3. Modelling Approach

3.1. X Model

As an idealized model of two separate service systems with the capability of sharing, we consider the X model, depicted in Figure 1. The X model has two homogeneous customer classes and two homogeneous agent pools. We assume that each customer class has a service pool primarily dedicated to it, but all agents are cross trained, so that they can handle calls from the other class, even though they may do so inefficiently or ineffectively. Under normal loading (at or near forecasted arrival rates), we want each class to be served only by its designated agents, without any help from cross-trained agents in the other service pool. We assume that staffing has been performed in standard ways, so that the number of agents in each pool is adequate to meet performance targets at forecasted arrival rates. However, we also want to automatically activate sharing when there are unexpected unbalanced overloads, either when only one class is overloaded or when both classes are overloaded but one is much more overloaded than the other.

Figure 1 X **Call-Center Model**



More specifically, in this paper, we consider a fully Markovian model. Customers from the two classes arrive according to independent Poisson processes with arrival rates λ_1 and λ_2. There is a queue for each customer class, with customers from each class entering service in order of arrival. We assume that waiting customers have limited patience. A class i customer will abandon if he does not start service before a random time that is exponentially distributed with mean $1/\theta_i$. There are two service pools, with pool j having m_j homogeneous servers working in parallel. The service times are mutually independent exponential random variables, but the mean may depend on both the customer class and the service pool. The mean service time for a class i customer served by a type j agent is $1/\mu_{i,j}$. Let the service times, abandonment times, and arrival processes be mutually independent. Let $Q_i(t)$ be the number of class i customers in queue and let $Z_{i,j}(t)$ be the number of type j agents busy serving class i customers, at time t. With the assumptions above, the stochastic process $(Q_i(t), Z_{i,j}(t); i = 1, 2; j = 1, 2)$ becomes a six-dimensional continuous-time Markov chain, given any routing policy that depends on this six-dimensional state.

In this context, under normal loading we want each class served only by agents from its own designated service pool; i.e., we want $Z_{1,2}(t) \approx Z_{2,1}(t) \approx 0$ for all t. One possible reason is that the value of service by agents from the other pool might be less, perhaps because they lack specialized skills. Another possible reason is that service by the cross-trained agents is less efficient; we might have the *strong inefficient-sharing condition*

$$\mu_{1,1} > \mu_{1,2} \quad \text{and} \quad \mu_{2,2} > \mu_{2,1}. \tag{1}$$

We examine the inefficient-sharing case. Throughout this paper, we assume the *basic inefficient-sharing condition*

$$\mu_{1,1}\mu_{2,2} \geq \mu_{1,2}\mu_{2,1}. \tag{2}$$

Clearly, condition (1) implies condition (2). These conditions play a role in §5.2.

In this X-model setting with inefficient sharing, we suppose that an unexpected overload occurs at some unanticipated time that changes the arrival rates. We assume that we are unable to immediately change the staffing levels in response to that unexpected overload. However, we do have the option of allowing some of the cross-trained agents from the less-loaded service pool serve customers from the more overloaded customer class. In addition, we do not know the new arrival rates when the overload occurs. Thus, we need to develop a control that depends on the system history; in some way we must discover that the arrival rates have indeed changed. That is challenging, because stochastic fluctuations under normal loading may make us think that the arrival rates have changed when in fact they have not. We illustrate with the following example.

Example 1. To illustrate, consider a symmetric model with forecasted arrival rates $\lambda_1 = \lambda_2 = 90$ per unit of time, where the mean service time for customers served by designated agents is $\mu_{1,1}^{-1} = \mu_{2,2}^{-1} = 1.0$, whereas the mean service time for customers served by agents from the other pool is $\mu_{1,2}^{-1} = \mu_{2,1}^{-1} = 1.25$. We measure time in units of mean service times by designated agents, which for discussion we take to be five minutes. Notice that condition (1) holds here: For all agents, the mean time required to serve the other class is 25% greater than the mean time required to serve an agent's own class. Let customers abandon at rate $\theta_1 = \theta_2 = 0.4$.

Because serving the other class is less efficient, with these parameters it makes sense to operate the system as two separate systems. Following standard staffing methods for a single-class single-pool $M/M/m + M$ model, we may assign $m_1 = m_2 = 100$ agents to the two service pools. That makes the traffic intensities $\rho_1 \equiv \lambda_1/m_1\mu_{1,1} = \rho_2 = 0.90$, which we regard as normal loading. With this staffing, standard algorithms show that steady-state performance is quite good: 82% of the arrivals enter service immediately upon arrival without joining the queue, only 0.5% of the arrivals abandon, the average size of each queue is 1.1, and the expected conditional waiting time, given that the customer is served, is only 0.012 (about 3.6 seconds with a mean service time of five minutes).

Now suppose that, at some unanticipated time, the arrival rate for class 1 jumps to $\lambda_1 = 130$, whereas the arrival rate for class 2 remains at $\lambda_2 = 90$. If class 1 receives no help from pool 2, then class 1 experiences severe congestion. Assuming that the system reaches steady state after this shift in arrival rate (which does not take very long, approximately a few mean service times, as confirmed by simulations; see §EC.1), almost all class 1 customers must wait before starting service, 23% of the class 1 customers abandon, the average size of the class 1 queue becomes 75, the expected conditional waiting time given that a class 1 customer is served is 0.65 (3.25 minutes).

If, as system managers, we were able to recognize that the class 1 arrival rate had shifted to 130, then we might elect to reassign some of the class 2 agents. For example, we might let 25 of the pool 2 agents be devoted to serving class 1. That increases the total service rate responding to the class 1 arrival rate of 130 from 100 to $100 + (1/1.25)25 = 120$, and leaves a total service rate of $100 - 25 = 75$ to respond to the class 2 arrival rate of 90. Because sharing is inefficient, we must sacrifice 25 units of service rate for class 2 to gain 20 units of service rate for class 1.

Assuming that the two classes can be modelled as $M/M/m + M$ queues (which is only approximately correct for class 1 because its servers have become heterogenous), we can analyze the performance, e.g., by Whitt (2005). The pair of abandonment probabilities for the two classes changes from $(0.23, 0.005)$ to $(0.08, 0.17)$; the pair of mean queue lengths for the two classes changes from $(75, 1.1)$ to $(26, 38)$; and the pair of conditional expected waiting times given that the customer is served changes from $(0.65, 0.012)$ to $(0.205, 0.450)$ (1.03 minutes and 2.25 minutes, respectively). In this paper we develop a control that responds in a similar way, but does so automatically without having to know that the arrival rates made that specific shift, and without making a fixed partition of the agents.

3.2. Analysis with a Cost Function

The advantage of such sharing, or any other control that produces similar sharing by the inefficient cross-trained agents, depends on the cost of the congestion experienced. To assess that cost, we will assume that there is a cost function C, with $C(Q_1(t), Q_2(t))$ representing the expected cost rate incurred at time t if the vector of queue lengths at time t is $(Q_1(t), Q_2(t))$. If the overload incident takes place over the time interval $[a, b]$, then the expected total cost would be

$$C_T \equiv E\left[\int_a^b C(Q_1(t), Q_2(t))\,dt\right] = \int_a^b E[C(Q_1(t), Q_2(t))]\,dt. \tag{3}$$

We assume that the cost function C is convex and strictly increasing. The convexity explains why we might want to share when one class is much more overloaded than the other, no matter which class is overloaded.

In this context, our goal is to choose a routing policy, which may allow assignments to cross-trained agents, to achieve low (near-minimum) expected total cost for all possible overload incidents and resulting stochastic processes $(Q_1(t), Q_2(t))$, although producing only a negligible amount of sharing under normal loading. To define what we mean by an "overload incident," we can first specify an interval $[a, c]$ over which the arrival-rate vector $(\lambda_1(t), \lambda_2(t))$ differs from the nominal vector. (We assume that the arrival process is a nonhomogeneous Poisson process with these new arrival rates.) However, we should also include an additional interval $[c, b]$ after time c to allow the vector queue length $(Q_1(t), Q_2(t))$ to return to its nominal steady-state value. (Engineering judgement is required.) In our analysis, we simplify by restricting attention to scenarios, as in the example above, in which the pair of arrival rates (λ_1, λ_2) makes a sudden unexpected shift at some time, and remains at the new vector for a significant duration, so that the system reaches a new steady state at the new arrival-rate vector. (Customer abandonment ensures that the system reaches steady state for any arrival-rate vector.) Our control applies more generally.

For such scenarios, we simplify by reexpressing our goal as minimizing the expected steady-state cost; i.e., we aim to minimize $C_T \equiv E[C(Q_1, Q_2)]$, where (Q_1, Q_2) is the vector of steady-state queue lengths associated with the new arrival-rate vector associated with the overload. We will use this steady-state overload framework to set the control parameters and demonstrate effectiveness, but the control applies to other overload scenarios. For this steady-state analysis to be effective, it is important that the system approaches the new steady state associated with the overload relatively quickly. As illustrated in the concrete example above, this tends to happen in a few mean service times. We discuss this important point further in §EC.1.

In the context of Example 1, we might have a shift in arrival rates lasting five hours. It might not be possible to change the staffing in response, because it is in the middle of the same day. The initial transient period might last three mean service times or 15 minutes, which is 5% of the total overload incident. There might then be a recovery period lasting about five mean service times or 25 minutes, after which the system returns to steady state. For such overloads, the steady state is evidently reasonable, and it is essential for tractability. Even with this simplifying approximation, the control problem for the stochastic system is very difficult. We will get an approximate solution only after exploiting a fluid approximation in addition to this steady-state analysis; see §5.2. Even with that approximation, the analysis with a general increasing convex cost function gets complicated; see §5.2. However, as a byproduct, there is a very nice, simple story (explicit formulas for everything), provided that we assume a separable quadratic power cost function; see Proposition 5.

4. Proposed Control

We start by briefly reviewing the *fixed-queue-ratio* (FQR) *routing rule* from Gurvich and Whitt (2009b) and then we show that the FQR rule without thresholds can perform poorly with inefficient sharing, where the conditions in the theorems of Gurvich and Whitt (2009b) are violated. Then we introduce our proposed modification of FQR to treat unexpected overloads. It involves general queue-ratio functions, as in Gurvich and Whitt (2009a), and thresholds, one of each for each direction of sharing.

4.1. FQR and Its Difficulties with Inefficient Sharing

With two queues, FQR can be implemented by considering a (weighted) *queue-difference stochastic process* $D(t) \equiv Q_1(t) - rQ_2(t)$, $t \geq 0$, where r is a single target-ratio parameter that management can set. With FQR for the X model, a newly available agent in either service pool serves the customer at the head of the class 1 (class 2) queue if $D(t) > 0$ ($D(t) < 0$), and serves the customer at the head of its own queue if $D(t) = 0$. The goal of FQR is to maintain a nearly constant queue ratio: $Q_1(t)/Q_2(t) \approx r$ throughout time. When $r = 1$, FQR coincides with serving the longer queue.

Under regularity conditions, the FQR control has two very desirable features for large-scale service systems, which makes it possible to reduce the multiclass multipool staffing-and-routing problem to the well-understood, single-class single-pool staffing problem. First, if the required conditions are satisfied, then FQR tends to produce *state-space collapse* (SSC); i.e., for the X model, the two-dimensional queue-length vector

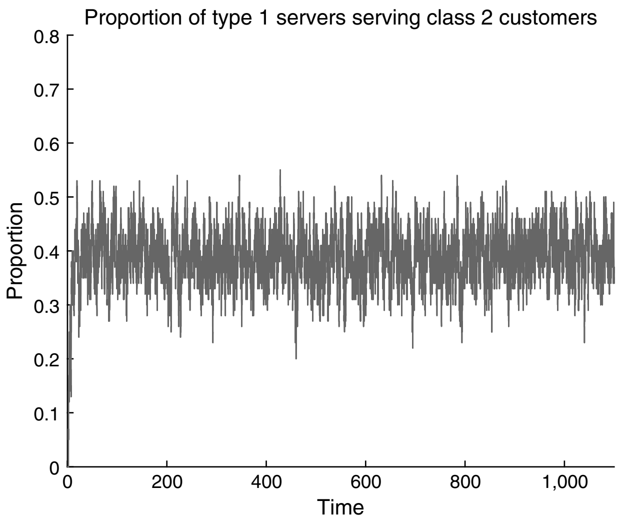
$(Q_1(t), Q_2(t))$ tends to evolve approximately as a one-dimensional process determined by the total queue length $Q_\Sigma(t) \equiv Q_1(t) + Q_2(t)$. In particular, $Q_i(t) \approx p_i Q_\Sigma(t)$ for $i = 1, 2$, where $p_1 = r/(1 + r) = 1 - p_2$; e.g., see Figure EC.8 in §EC.2. Moreover, it does so in a way such that all three stochastic processes—$Q_\Sigma(t)$, $Q_1(t)$, and $Q_2(t)$—remain appropriately stable as $t \to \infty$. Indeed, Gurvich and Whitt (2009b) show that, under regularity conditions, FQR achieves SSC asymptotically in the QED many-server heavy-traffic limiting regime. Second, with FQR, it is possible to choose the ratio parameter r (or, equivalently, the queue proportions p_i) to determine the optimal level of staffing to achieve desired service-level differentiation; i.e., staffing costs are minimized subject to meeting class-dependent delay targets $P(W_i > T_i) = \alpha$; see §EC.2 and Gurvich and Whitt (2009b). Gurvich and Whitt (2009a) also showed how to staff to minimize convex costs under normal loading. In that case, the asymptotically optimal control in the QED regime is not FQR, but a state-dependent generalization: the *queue-and-idleness-ratio* (QIR) control. Our optimal queue ratios for the fluid model under overloading with convex costs are of the same state-dependent form.

However, in our setting, where service provided by nondesignated agents is inefficient, neither FQR nor QIR, without the extra thresholds, is appropriate in normal loading, because they induce undesired sharing. Because of the inefficient sharing, the system is *not* work conserving; sharing causes the required workload to increase. Indeed, the conditions in the key theorems of Gurvich and Whitt (2009a, b) are violated. In fact, those conditions are actually needed to maintain stability. (However, for FQR without the thresholds, SSC is still achieved; the two queues explode together.)

Example 2. To illustrate, consider the X model with parameters $m_1 = m_2 = 100$, $\mu_{1,1} = \mu_{2,2} = 1.0$, $\mu_{1,2} = \mu_{2,1} = 0.8$, $\lambda_1 = \lambda_2 = 0.99$, and $\theta_1 = \theta_2 = 0.0$ (no abandonment). Because the traffic intensities are $\rho_i = \lambda_i / m_i \mu_{i,i} = 0.99$, the two separate systems without sharing are stable (with mean queue length 85 and mean waiting time 0.85). However, if we use FQR with $r = 1$, then inefficient sharing is generated, so that a significant proportion of each agent pool is busy serving the other class. As a consequence, the arrival rate actually exceeds the service rate and the queue lengths diverge to infinity. Here, there still is SSC, but the two queue lengths diverge together.

This difficulty when FQR is applied inappropriately is illustrated by Figures 2 and 3. They show the sample paths of $Q_1(t)$ and $Z_{2,1}(t)$, starting empty, in one simulation run. After an initial transient period, the number of agents serving the other class fluctuates around $E[Z_{1,2}] = E[Z_{2,1}] \approx 39$, whereas the queue grows in an approximately linear rate; the simulation estimate is $E[Q_i(t)] \approx 6.8t$, $t \geq 0$. (These numerical values are estimated from multiple simulation runs. The confidence intervals are less than 1%. We develop analytical approximations to describe this behavior in a subsequent paper.)

Figure 2 **Sample Path of $Z_{2,1}(t)$ for FQR**

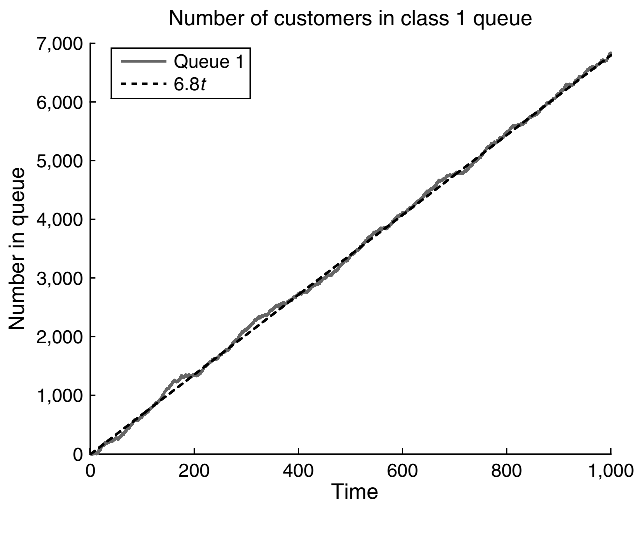


Customer abandonment necessarily prevents the queues from exploding. Even in the worst case, when all agents are dedicated to the wrong class, the system would be stable. However, there still is performance degradation, e.g., with $\theta_1 = \theta_2 = 0.2$ and $r = 1$ about 39% of the agents in each pool are busy serving customers from the other class which causes the queues to grow from 10, if there is no sharing, to 34. More details appear in §EC.2.

4.2. Proposed Control: FQR-T

Here is the lesson from the previous subsection: If we are going to use a queue-ratio control, then we need to

Figure 3 **Sample Path of $Q_1(t)$ for FQR**



take extra measures to prevent sharing under normal loading. First, we want to prevent simultaneous inefficient sharing in both directions. Hence, we restrict the routing to *one-way sharing* at any time: We do not allow a newly available type 2 agent to serve a waiting class 1 customer if there are any type 1 agents busy serving class 2 customers. And similarly in the other direction. (However, over time, the direction of one-way sharing may change; we are not considering the so-called N model, which only allows one-way sharing in one fixed direction.)

From cost considerations, discussed in §5, we want to allow different ratio parameters $r_{1,2}$ and $r_{2,1}$ for the different ways we may share. (In general, we may need more complicated ratio functions or, equivalently, sharing regions; see §5, especially Figure 4.) To permit sharing only in the presence of unbalanced overloads, we suggest *fixed-queue-ratio routing with thresholds* (FQR-T). In addition to the two ratio parameters $r_{1,2}$ and $r_{2,1}$, we introduce two positive thresholds $\kappa_{1,2}$ and $\kappa_{2,1}$. We then define two queue-difference stochastic processes:

$$D_{1,2}(t) \equiv Q_1(t) - r_{1,2}Q_2(t) \quad \text{and} \\ D_{2,1}(t) \equiv r_{2,1}Q_2(t) - Q_1(t). \tag{4}$$

As long as $D_{1,2}(t) < \kappa_{1,2}$ and $D_{2,1}(t) < \kappa_{2,1}$, we do not allow any sharing, i.e., we only let agents serve customers from their designated class.

However, available pool 2 agents are assigned to class 1 customers when $D_{1,2}(t) \geq \kappa_{1,2}$, provided that no pool 1 agents are still serving a class 2 customer. As soon as the first pool 2 agent is assigned to serve a class 1 customer, we drop the threshold $\kappa_{1,2}$, but keep the other threshold $\kappa_{2,1}$. (We could elect to add another threshold for the sharing; see §EC.4.) Once one-way sharing has been activated with pool 2 helping class 1, we use ordinary FQR with ratio parameter $r_{1,2}$. Upon service completion, a newly available type 2 agent serves the customer at the head of the class 1 queue (the class 1 customer who has waited the longest) if $D_{1,2}(t) > 0$; otherwise the agent serves a customer from his own class. In this phase, pool 1 agents only serve class 1 customers. Only one-way sharing in this direction will be allowed until either the class 1 queue becomes empty or the other difference process crosses the other threshold, i.e., when $D_{2,1}(t) \geq \kappa_{2,1}$. As soon as either of these events occurs, newly available pool 2 agents are only assigned to class 2 and the threshold $\kappa_{1,2}$ is reinstated.

We can initiate sharing in the opposite direction when first $D_{2,1}(t) \geq \kappa_{2,1}$ and there are no class 2 agents serving class 1 customers. At the first time both conditions are satisfied, we start sharing with a pool 2 agent serving a class 1 customer. When that first assignment takes place, we remove the threshold $\kappa_{2,1}$ and again use FQR with one-way sharing, but now with the ratio parameter $r_{2,1}$.

Upon arrival, a class i customer is routed to pool i if there are idle servers; otherwise the arrival goes to the end of the class i queue. An arrival might increase the queue to a point that sharing is activated. Then the first customer in queue is served by the other class (presumably the agent that has been idle the longest, but we do not focus on individual agents).

The queue-difference stochastic processes in (4) will never provide any instantaneous motivation to have agents of both types simultaneously inefficiently serving the other class if $r_{1,2} \geq r_{2,1}$. That property will be satisfied when we apply a cost function to specify the ratio parameters in §5.2.

To illustrate how FQR-T performs in normal loading (heavy load, but not overloaded), we again consider Example 2 with abandonments at rate $\theta_i = 0.2$. We let $r_{1,2} = r_{2,1} = 1$, so that there is no change from FQR above, but now we add thresholds $\kappa_{1,2} = \kappa_{2,1} = 10$. The performance is greatly improved with FQR-T compared to FQR without thresholds: $E[Z_{1,2}] = E[Z_{2,1}] \approx 2.0$ for FQR-T, whereas $E[Z_{1,2}] = E[Z_{2,1}] \approx 39$ for FQR. As a consequence, the performance for FQR-T is almost the same as without sharing. In particular, with FQR-T, the abandonment rate is slightly higher than without sharing (2.5% compared to 2.0%), but the average queue length is actually less (9.4 compared to 10.0). In fact, FQR-T can outperform no sharing with larger threshold values, because of the resource-pooling effect. For more details, see §EC.2.

5. Fluid Approximation for Steady State of the X Model

To obtain a tractable characterization of performance for FQR-T and find good queue-ratio parameters, we now introduce a deterministic fluid approximation. To describe the steady-state behavior of our model when there is no sharing, we first discuss the case of a single customer class served by a single service pool—the classical $M/M/m+M$ model, with arrival rate λ, individual service rate μ, and abandonment rate θ. Afterward we treat the more general X model.

5.1. One Class and One Pool

For the $M/M/m+M$ model, the approximating deterministic fluid model has been studied in Whitt (2004) via many-server heavy-traffic limits. Here we will derive the simple steady-state formulas directly. We assume that input and output (which we call fluid) occurs deterministically at the specified rates. We think of the system as large and thus regard the number of customers and servers as continuous quantities as well. Thus, fluid arrives deterministically and continuously at constant rate λ. Fluid also is served and abandons deterministically and continuously at rates that are directly proportional to the number of busy servers and the queue length, respectively. If the "number" of busy servers is x, then fluid is served at rate $x\mu$; if the queue length is q, then fluid abandons at rate $q\theta$.

We say that the system is overloaded if the input rate exceeds the maximum possible total service rate. Given m servers, each working at rate μ, the maximum possible total service rate is $m\mu$. Thus, the system is overloaded if $\lambda > m\mu$, and not overloaded otherwise. If the system is overloaded, then in steady state all servers will be busy and there will be a queue of waiting fluid, with content q, which can be determined simply be equating the rate in to the rate out, including customer abandonment: *rate in* $\equiv \lambda = m\mu + q\theta \equiv$ *rate out*. As an immediate consequence, we get $q = (\lambda - m\mu)/\theta$. If the system is not overloaded, i.e., if $\lambda \le m\mu$, then there will be no queue. Then we can describe the steady state via the amount of spare service capacity (number of idle servers), s, which again can be determined by equating the rate in to the rate out: *rate in* $\equiv \lambda = (m - s)\mu \equiv$ *rate out*. As an immediate consequence, we get $s = m - (\lambda/\mu)$. Without directly specifying whether or not the system is overloaded, we can write

$$q = \frac{(\lambda - m\mu)^+}{\theta} \quad \text{and} \quad s = \left(m - \frac{\lambda}{\mu}\right)^+, \tag{5}$$

where $(x)^+ \equiv \max\{x, 0\}$. We always have the *complementarity relation* $qs = 0$.

From the point of view of our analysis, we regard λ as an unknown parameter, but we consider the remaining parameters m, μ, and θ as fixed and known. For any given λ, we can compute q and s as indicated above. With our overload control problem in mind, it is significant that we can recover λ from the pair (q, s), because we want to learn about λ by observing (q, s). If $q > 0$ and $s = 0$, then necessarily we are *overloaded*, and $\lambda = \theta q + m\mu$; if $q = 0$ and $s > 0$, then necessarily we are *underloaded* (which includes normally loaded), and $\lambda = (m - s)\mu$; if $q = 0$ and $s = 0$, then necessarily we are *critically loaded*, and $\lambda = m\mu$; we cannot have $q > 0$ and $s > 0$. For an overloaded fluid queue, λ is an increasing linear function of q; for an underloaded queue, λ is a decreasing linear function of s.

As discussed in Whitt (2004), we can also describe the transient behavior of the fluid model and determine other performance measures. For example, if the fluid model is overloaded, then the associated approximate potential steady-state waiting time (virtual waiting time for a customer with infinite patience) is $w = \log(\lambda/m\mu)/\theta_1 = \log(\rho)/\theta_1$, where $\rho \equiv \lambda/m\mu$ is the traffic intensity, satisfying $\rho > 1$; see (2.26) of Whitt (2004).

Note that an increasing convex function of w is an increasing convex function of λ for $\lambda \ge m\mu$. Because λ is a positive linear function of q under overloads, we see that an increasing convex function of w itself is a convex increasing function of q, as we have assumed in our optimization formulation. Similarly, the abandonment rate in the overloaded fluid model is $\theta q = \lambda - m\mu$, so the abandonment rate is an increasing linear function of q under overloads.

5.2. Optimal Solution for the X Fluid Model

The X fluid model is a natural generalization of the single-class single-pool fluid model above. Now we have two deterministic arrival rates λ_1 and λ_2, one for each class, with the additional parameters $\{m_j, \theta_i, \mu_{i,j};\ i = 1, 2;\ j = 1, 2\}$. Closely paralleling the discussion above, we will be characterizing the steady-state performance in terms of the quantities (Q_1, Q_2, S_1, S_2), where Q_i is the fluid content at the class i queue, whereas S_j is the amount of spare capacity at pool j.

The steady-state behavior of the X fluid model depends on the number of agents from each pool assigned to (and actually busy serving customers from) each customer class, i.e., the deterministic vector $(Z_{1,1}, Z_{1,2}, Z_{2,1}, Z_{2,2})$, where $Z_{i,j}$ is the number of pool j agents assigned to serve class i customers, which is regarded as a continuous variable. To be legitimate assignments, we must have $Z_{i,j} \ge 0$ for all i and j with $Z_{1,1} + Z_{2,1} \le m_1$ and $Z_{1,2} + Z_{2,2} \le m_2$. Because these agents are actually busy serving customers, we must also have $\lambda_1 \ge Z_{1,1}\mu_{1,1} + Z_{1,2}\mu_{1,2}$ and $\lambda_2 \ge Z_{2,1}\mu_{2,1} + Z_{2,2}\mu_{2,2}$. Once we assign values to these variables $Z_{i,j}$, we reduce the X model to two single-class single-pool models. The arrival rate for class i is λ_i, whereas the service rate for class i is $Z_{i,1}\mu_{i,1} + Z_{i,2}\mu_{i,2}$. Class i is then overloaded if and only if $\lambda_i > Z_{i,1}\mu_{i,1} + Z_{i,2}\mu_{i,2}$, in which case the steady-state fluid content in the class i is

$$Q_i = \frac{\lambda_i - Z_{i,1}\mu_{i,1} - Z_{i,2}\mu_{i,2}}{\theta_i}. \tag{6}$$

If class i is not overloaded, then $Q_i = 0$. The spare capacity in pool j in steady state is $S_j = m_j - Z_{1,j} - Z_{2,j} \ge 0$, $j = 1, 2$.

In this X fluid model setting, for known arrival rates, our initial goal is to determine the minimum cost $C^*(\lambda_1, \lambda_2)$, which is the minimum of $C(Q_1(Z_{1,1}, Z_{1,2}), Q_2(Z_{2,1}, Z_{2,2}))$ for specified arrival-rate vector (λ_1, λ_2), which we denote simply by $C(Z_{1,1}, Z_{1,2}, Z_{2,1}, Z_{2,2})$, over all feasible fixed assignment vectors $(Z_{1,1}, Z_{1,2}, Z_{2,1}, Z_{2,2})$ in $\mathbb{R}^4$ with $Q_i \equiv Q_i(Z_{i,1}, Z_{i,2})$ defined in (6). We let the asterisk denote the optimal solution. (We do not consider more general controls.) We will apply the optimal solution to find the optimal state-dependent queue-ratio functions.

Let q_i be the queue length of class i and let s_i be the spare capacity in pool i when there is no sharing. They can be expressed as in (5), with formulas depending on i. In the fluid model, we regard the system as being in normal loading if neither queue is overloaded without sharing, i.e., if $q_1 = q_2 = 0$, but the amount of spare capacity is not too large. Because the cost function is increasing and convex, under normal loading we achieve the minimum cost by letting $Z_{1,2} = Z_{2,1} = 0$ (no sharing) to obtain $Q_i = 0$ for $i = 1, 2$. The unexpected overload means that either $q_1 > 0$ or $q_2 > 0$, or both. Henceforth we assume that to be the case.

The natural model state is (λ_1, λ_2), but an equivalent representation is (q_1, s_1, q_2, s_2), where we always have the complementarity relation $q_1 s_1 = q_2 s_2 = 0$. If $q_i > 0$, then $\lambda_i = m_i\mu_{i,i} + q_i\theta_i$; if $s_i > 0$, then $\lambda_i = (m_i - s_i)\mu_{i,i}$. This alternative representation implies that, for the X fluid model, we can determine the arrival rates by observing the queue lengths and spare capacities.

Let $Z^*_{i,j}$ be the optimal value of the variable $Z_{i,j}$. We start by stating some basic propositions, which serve to simplify our X-fluid-model optimization problem. We first reduce the number of variables from four to two. The following is immediate.

Proposition 1 (No Idle Agents). *If we do not have $Q^*_1 = Q^*_2 = 0$, then there should be no idle agents, i.e., $S^*_j = 0$ or, equivalently, $Z^*_{1,j} + Z^*_{2,j} = m_j$ for $j = 1, 2$.*

As a consequence of Proposition 1, if $q_1 > 0$, $q_2 = 0$, and $s_2 > 0$, then necessarily $Z^*_{1,2} > 0$. Moreover, either $Z^*_{1,2} \geq s_2$ or $Q^*_1 = Q^*_2 = 0$.

We next show that inefficient sharing implies no two-way sharing.

Proposition 2 (One-Way Sharing). *Because the service rates satisfy the inefficient-sharing condition $\mu_{1,1}\mu_{2,2} \geq \mu_{1,2}\mu_{2,1}$ in* (2), *it suffices to consider one-way sharing; i.e., $Z^*_{1,2}Z^*_{2,1} = 0$.*

Proof. Suppose that $Z_{1,2} > 0$ and $Z_{2,1} > 0$, so that we have sharing in both directions. It suffices to assume that $Q_1 > 0$ and $Q_2 > 0$. We will show that, for appropriate positive variables $x_{1,2}$ and $x_{2,1}$, if we replace $(Z_{1,2}, Z_{2,1})$ by $(Z_{1,2} - x_{1,2}, Z_{2,1} - x_{2,1})$, then both queue lengths will decrease until one of the variables $Z_{1,2} - x_{1,2}$ or $Z_{2,1} - x_{2,1}$ becomes zero or both queues become empty. We define $x_{2,1}$ as an appropriate constant multiple of $x_{1,2}$, so that we have a single real variable. To do so, let $\gamma_i \equiv \lambda_i - Z_{i,1}\mu_{i,1} - Z_{i,2}\mu_{i,2} > 0$ for $i = 1, 2$. Then let $x_{2,1} \equiv \beta x_{1,2}$, where $\beta \equiv (\gamma_2\mu_{1,2} + \gamma_1\mu_{2,2})/(\gamma_2\mu_{1,1} + \gamma_1\mu_{2,1})$. Then we consider what happens as we increase $x_{1,2}$, assuming that β remains constant. Let $\Delta_i \equiv \theta_i(Q_i(0) - Q_i(x_{1,2}))$, where $Q_i(x_{1,2})$ denotes Q_i with the initial vector of sharing levels $(Z_{1,2}, Z_{2,1})$ replaced by $(Z_{1,2} - x_{1,2}, Z_{2,1} - \beta x_{1,2})$. Then

$$\Delta_1 = x_{1,2}\gamma_1\left(\frac{\mu_{1,1}\mu_{2,2} - \mu_{1,2}\mu_{2,1}}{\gamma_2\mu_{1,1} + \gamma_1\mu_{2,1}}\right) \quad \text{and}$$
$$\Delta_2 = x_{1,2}\gamma_2\left(\frac{\mu_{1,1}\mu_{2,2} - \mu_{1,2}\mu_{2,1}}{\gamma_2\mu_{1,1} + \gamma_1\mu_{2,1}}\right). \tag{7}$$

Clearly, $\Delta_i \geq 0$ for both i if and only if inequality (2) holds. Moreover, from (6) and (7), we see that both queues become empty at the same level of $x_{1,2}$. Hence, we can decrease both variables $Z_{1,2}$ and $Z_{2,1}$ by increasing $x_{1,2}$ until one of these variables becomes zero or both queue lengths simultaneously become zero. □

As a consequence of Proposition 2, we can reexpress the basic optimization problem, first, in terms of two convex real-valued functions of a single real variable, $C_{1,2}$ and $C_{2,1}$, and second, in terms of a single combined convex function of a single real variable, C_c. Let 1_A be the indicator function of the set A; i.e., $1_A(x) = 1$ if $x \in A$ and $1_A(x) = 0$ otherwise. We put the short proof required in §EC.3.1.

Proposition 3 (Single-Variable Functions). *Because the inefficient-sharing condition* (2) *holds, the optimal cost can be expressed as*

$$\begin{aligned} C^*(\lambda_1, \lambda_2) &= C^*(q_1, s_1, q_2, s_2) \\ &= \min\{C_{1,2}(Z_{1,2}), C_{2,1}(Z_{2,1})\} \\ &= \min\{C_c(Z_{1,2} - Z_{2,1})\} \end{aligned} \tag{8}$$

over $Z_{1,2}$ and $Z_{2,1}$ such that $0 \leq Z_{1,2} \leq m_2$, $0 \leq Z_{2,1} \leq m_1$, and $Z_{1,2}Z_{2,1} = 0$, where

$$\begin{aligned} &C_{1,2}(Z_{1,2}) \\ &\equiv C_{1,2}(Z_{1,2}; \lambda_1, \lambda_2) \\ &\equiv C\left(\frac{(\lambda_1 - m_1\mu_{1,1} - Z_{1,2}\mu_{1,2})^+}{\theta_1}, \frac{(\lambda_2 - (m_2 - Z_{1,2})\mu_{2,2})^+}{\theta_2}\right) \\ &\equiv C_{1,2}(Z_{1,2}; q_1, s_1 = 0, q_2, s_2) \\ &\equiv C\left(\frac{(q_1 - \mu_{1,2}Z_{1,2})^+}{\theta_1}, \frac{(q_2 - s_2\mu_{2,2} + \mu_{2,2}Z_{1,2})^+}{\theta_2}\right), \end{aligned} \tag{9}$$

$$\begin{aligned} C_c(Z_{1,2} - Z_{2,1}) &\equiv C_{1,2}(Z_{1,2} - Z_{2,1})1_{\{Z_{1,2} - Z_{2,1} \geq 0\}} \\ &\quad + C_{2,1}(-(Z_{1,2} - Z_{2,1}))1_{\{Z_{1,2} - Z_{2,1} < 0\}} \\ &= C_{1,2}(Z_{1,2})1_{\{Z_{1,2} > 0\}} + C_{2,1}(Z_{2,1})1_{\{Z_{2,1} > 0\}} \\ &\quad + C(q_1, q_2)1_{\{Z_{1,2} = Z_{2,1} = 0\}}, \end{aligned} \tag{10}$$

with q_i and s_i defined in (5), *satisfying $q_1 s_2 = q_2 s_2 = 0$, and $C_{2,1}(Z_{2,1})$ defined analogously to $C_{1,2}(Z_{1,2})$ in* (9). *The functions $C_{1,2}$ and $C_{2,1}$ are continuous strictly convex functions of the single real variables $Z_{1,2}$ and $Z_{2,1}$ over their domain. If, in addition, the stronger inefficient-sharing condition $\mu_{1,1} > \mu_{1,2}$ and $\mu_{2,2} > \mu_{2,1}$ in* (1) *holds,* *then C_c is also a continuous strictly convex function of the single real variable $Z_{1,2} - Z_{2,1}$ over the domain specified in this proposition.*

Corollary 1 (Three Intervals). *If the stronger inefficient-sharing condition (1) holds, then for each pair of arrival rates (λ_1, λ_2) or initial state (q_1, s_1, q_2, s_2) (without sharing), there are two thresholds $\zeta_{1,2} \geq \zeta_{2,1}$ such that exactly one of the following occurs:*

$$\begin{aligned} &\text{(i)}\ \ Z^*_{1,2} > 0 \quad \text{and} \quad Z^*_{2,1} = 0 \quad \text{for } Z_{1,2} - Z_{2,1} > \zeta_{1,2}, \\ &\text{(ii)}\ \ Z^*_{2,1} > 0 \quad \text{and} \quad Z^*_{1,2} = 0 \quad \text{for } Z_{1,2} - Z_{2,1} < \zeta_{2,1}, \\ &\text{(iii)}\ \ Z^*_{1,2} = Z^*_{2,1} = 0 \quad \text{for } \zeta_{2,1} \leq Z_{1,2} - Z_{2,1} \leq \zeta_{1,2}. \end{aligned} \tag{11}$$

The value of Corollary 1 will be clear when we turn our attention to the queue ratio r below. We can apply Proposition 2 to get further simplification if there is initially spare capacity. Then, from the beginning, we know that we can only have sharing with help provided by the pool with spare capacity; i.e., if $q_1 > 0 > s_2$, then $Z^*_{1,2} > 0$ and $Z^*_{2,1} = 0$, so that it suffices to minimize $C_{1,2}(Z_{1,2})$.

It is natural to have the cost function C be smooth, in which case the optimal solution can be found by simple calculus. Proposition EC.1 concludes that, if the optimal solution found by calculus falls outside the feasible set, then the actual optimum value is obtained at the nearest boundary point.

It is easy to see that there is a one-to-one correspondence between the queue ratio $r \equiv Q_1/Q_2$ and the real variable $Z_{1,2} - Z_{2,1}$ used to specify the optimization problem in Proposition 3. That implies that there is a one-to-one correspondence between the fixed-agent-allocation optimization problem (choosing $Z_{1,2}$ and $Z_{2,1}$) and the (fixed) queue-ratio control problem (choosing state-dependent queue-ratio functions $r_{1,2}$ and $r_{2,1}$) in the fluid-model context. We establish it formally in §EC.3.3.

Finally, we provide a basis for an efficient algorithm to determine the equivalent optimal controls. To do so, we effectively reduce the dimension from two to one by observing that special weighted sums of the queue lengths (and corresponding weighted sums of the arrival rates) are independent of the agent-assignment variables $Z_{1,2}$ and $Z_{2,1}$. We only state the result for $Z_{1,2}$; the corresponding result for $Z_{2,1}$ is stated in §EC.3.4. The proof is verification by direct computation, so we omit it. For understanding, it may be helpful to refer to Figure 4 in §5.3.

Proposition 4 (Constant Weighted Queue Lengths). *Let*

$$a_{1,2} \equiv \frac{\mu_{2,2}\theta_1}{\mu_{1,2}\theta_2} \quad \text{and} \quad \tilde{a}_{1,2} \equiv \frac{\mu_{1,2}}{\mu_{2,2}}. \tag{12}$$

Consider any initial state (λ_1, λ_2), or equivalently (q_1, s_1, q_2, s_2), with $s_1 = 0$. Then

$$\begin{aligned} w_{1,2} &\equiv a_{1,2}\left(\frac{\lambda_1 - m_1\mu_{1,1}}{\theta_1}\right) + \left(\frac{\lambda_2 - m_2\mu_{2,2}}{\theta_2}\right) \\ &= a_{1,2}q_1 + q_2 - \frac{s_2\mu_{2,2}}{\theta_2} \\ &= a_{1,2}Q_1(Z_{1,2}) + Q_2(Z_{1,2}) - \frac{S_2(Z_{1,2})\mu_{2,2}}{\theta_2} \end{aligned} \tag{13}$$

for all $Z_{1,2}$ with $0 \leq Z_{1,2} \leq m_2$.

Proposition 4 implies that the locus of all non-negative queue-length vectors $(Q_1, Q_2) \equiv (Q_1(Z_{1,2}), Q_2(Z_{1,2}))$ associated with initial state (λ_1, λ_2), or equivalently (q_1, s_1, q_2, s_2), with $s_1 = 0$, is on the line $\{(Q_1, Q_2)\colon a_{1,2}Q_1 + Q_2 = w_{1,2}\}$ in the nonnegative quadrant. Thus, for any nonnegative constant $w_{1,2}$, the optimal queue-length vector (Q^*_1, Q^*_2) and the optimal queue-ratio $r^*_{1,2} \equiv Q^*_1/Q^*_2$ restricted to one-way sharing $(Z_{2,1} = 0)$ are the same for all initial states (q_1, s_1, q_2, s_2) with $s_1 = 0$ satisfying (13) provided that $q_1 \geq Q^*_1$. In that case, $a_{1,2}Q^*_1 + Q^*_2 = w_{1,2}$. That same optimal queue-length vector and optimal queue ratio holds for all arrival pairs (λ_1, λ_2) where $s_1 = 0$, $Z_{2,1} = 0$ and

$$\begin{aligned} \lambda_1 + \tilde{a}_{1,2}\lambda_2 &= \widetilde{w}_{1,2} \\ &\equiv \frac{\theta_1\theta_2 w_{1,2} + a_{1,2}\theta_2 m_1\mu_{1,1} + \theta_1 m_2\mu_{2,2}}{a_{1,2}\theta_2}. \end{aligned} \tag{14}$$

And similarly for sharing in the other direction; see §EC.3.4.

5.3. Computing the Optimal Queue-Ratio Functions

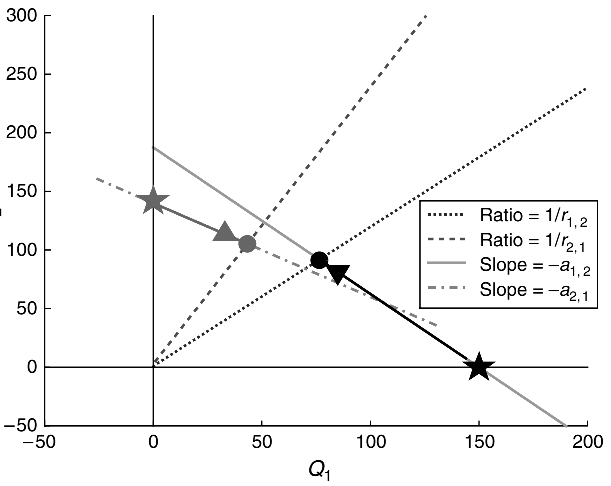
We now demonstrate how to numerically find the optimal state-dependent queue ratios $r^*_{1,2}$ and $r^*_{2,1}$ as functions of the fluid state (Q_1, S_1, Q_2, S_2). With the thresholds, this gives us a state-dependent *queue-ratio control with thresholds* (QR-T). To illustrate, we consider a (nonseparable) quadratic cost function of the form

$$C(Q_1, Q_2) = 3Q_1^2 + 2Q_2^2 + Q_1Q_2 + 10Q_1 + 5Q_2. \tag{15}$$

For any vector of arrival rates (λ_1, λ_2) we can assign one, and only one, point in the (Q_1, Q_2) plane, which represents the queue lengths associated with these arrival rates, when there is no sharing. To represent spare capacity, we allow negative values; i.e., $-Q_i$ is shorthand for $-S_i\mu_{i,i}/\theta_i$. (We actually plot $(Q_1 - S_1\mu_{1,1}/\theta_1, Q_2 - S_2\mu_{2,2}/\theta_2)$ even though the axes are simply labelled Q_i.)

We apply Proposition 4 to find the optimal queue ratios. We first consider when pool 2 helps class 1.

Figure 4 **Curves of Optimal Queue Ratios for an X Model with Parameters $\mu_{i,i} = 1$, $\mu_{1,2} = \mu_{2,1} = 0.8$, $\theta_i = 0.3$, $m_i = 100$, and Quadratic Cost Function $C(Q_1, Q_2)$ from (15)**



Notes. Also shown are two possible initial states, depicted by stars, and the lines of constant weighted queue lengths that relate the initial states to (reciprocals of) the optimal queue ratios, depicted by circles. Negative queues stand for spare capacity.

To treat that case, we let $\lambda_2 = m_2\mu_{2,2}$, so that class 2 has no queue before pool 2 helps class 1. We then assume that $\lambda_1 > m_1\mu_{1,1}$ so that class 1 is overloaded. We then choose a large set of positive weighted arrival sums $\{\tilde{w}^1_{1,2}, \ldots, \tilde{w}^n_{1,2}\}$ and find the optimal queue ratio for each. In the first step, we let $\lambda_1 \equiv \tilde{w}_{1,2} - \tilde{a}_{1,2}\lambda_2$, using (14). We then write (15) as a function of $Z_{1,2}$, take its derivative and find the optimal $Z^*_{1,2}$. Plugging $Z^*_{1,2}$ in the queue equations gives us the optimal queue lengths (for the specific arrival rates), and the optimal queue ratio $r^*_{1,2}$. We repeat this for every $\tilde{w}^i_{1,2}$ to get the curve $1/r^*_{1,2}$ depicted in Figure 4. To find the curve $1/r^*_{2,1}$ we go through essentially the same procedure for $Z^*_{2,1}$.

Figure 4 simultaneously depicts the three optimal sharing regions in the two-dimensional state space and the two curves of optimal queue ratios. It was generated using Matlab on a system with the following parameters: $m_1 = m_2 = 100$, $\mu_{1,1} = \mu_{2,2} = 1$, $\mu_{1,2} = \mu_{2,1} = 0.8$, and $\theta_1 = \theta_2 = 0.3$. In addition, Figure 4 shows how to find the optimal queue ratio for two possible initial queue lengths denoted by stars. When the initial queue-length vector is $(Q_1, Q_2) = (150, 0)$ (equivalently, $\lambda_1 = 145$ and $\lambda_2 = 100$), then the optimal queue-length vector is $(Q^*_1, Q^*_2) = (76.5, 91.8)$ and the optimal queue ratio is $r^*_{1,2} \equiv Q^*_1/Q^*_2 = 0.83$. This optimal queue ratio is the intersection of the curve $1/r_{1,2}$ with the line with slope $-a_{1,2}$ that passes through $(150, 0)$ (the circle on the $1/r_{1,2}$ curve). When the initial queue-length vector is $(Q_1, Q_2) = (0, 150)$ (equivalently, $\lambda_1 = 100$ and $\lambda_2 = 145$), we get $r^*_{2,1} = 0.41$ and $(Q^*_1, Q^*_2) = (46.6, 112.8)$. The optimal queue ratio is also the intersection of the curve $1/r_{2,1}$ with the line with slope $-a_{2,1}$ that passes through $(0, 150)$ (the circle on the $1/r_{2,1}$ curve).

Both the $1/r^*_{1,2}$ and $1/r^*_{2,1}$ curves seem to be linear, although that is actually not quite the case; the $r^*_{i,j}$'s are not constants for this cost function. For example, we already noted that, for $(\lambda_1, \lambda_2) = (145, 100)$ the optimal queue ratio is $r^*_{1,2} = 0.83$. If we change λ_1 to 110 then the optimal ratio becomes 0.80. For the other sharing direction, if $(\lambda_1, \lambda_2) = (100, 145)$, then $r^*_{2,1} = 0.41$, but if we change λ_2 to 110, then the optimal ratio changes to 0.38.

The fact that the two optimal-ratio curves are nearly linear in Figure 4 suggests that we can approximate the optimal queue-ratio function by fixed queue ratios, depending only on the direction of sharing; i.e., we can use FQR-T with only two values: one for $r_{1,2}$ and the other for $r_{2,1}$. In our example we may choose to use $r_{1,2} = 0.8$ and $r_{2,1} = 0.4$. The cost for using a nearly optimal ratio is very small in the fluid approximation, and even smaller in the stochastic system.

To understand when the optimal queue-ratio functions are nearly linear, as in the example above, and what the structure should be more generally, we investigate structured cost functions in §EC.4. We obtain explicit analytical expressions in special cases. We focus on separable cost functions: $C(Q_1, Q_2) = C_1(Q_1) + C_2(Q_2)$, where each component cost function C_i is strictly convex, strictly increasing and twice differentiable. For example, we find that QR-T reduces to FQR-T exactly when $C_i(Q_i) = c_i Q_i^{n_i}$ with $n_1 = n_2$; the case $n_1 = n_2 = 2$ is close to (15).

Proposition 5 (Explicit Solution). *When* $C(Q_1, Q_2) = c_1Q_1^2 + c_2Q_2^2$, *FQR-T is optimal for the X fluid model with*

$$r^*_{1,2} \equiv \frac{a_{1,2}c_2}{c_1} = \frac{c_2\mu_{2,2}\theta_1}{c_1\mu_{1,2}\theta_2}, \qquad r^*_{2,1} \equiv \frac{a_{2,1}c_2}{c_1} = \frac{c_2\mu_{2,1}\theta_1}{c_1\mu_{1,1}\theta_2},$$

$$Z^*_{1,2} = \frac{(c_1\mu_{1,2}\theta_1)(q_1 - (s_1\mu_{1,1}/\theta_1)) - (c_2\mu_{2,2}\theta_2)(q_2 - (s_2\mu_{2,2}/\theta_2))}{c_1\mu^2_{1,2}/\theta_1 + c_2\mu^2_{2,2}/\theta_2},$$

$$Z^*_{2,1} = \frac{c_2\mu_{2,1}\theta_2(q_2 - (s_2\mu_{2,2})/\theta_2) - c_1\mu_{1,1}\theta_1(q_1 - (s_1\mu_{1,1})/\theta_1)}{c_1\mu^2_{1,1} + c_2\mu^2_{2,1}}. \tag{16}$$

In Proposition 5, the cost is specified by a single parameter: the ratio c_1/c_2 specifies the relative importance of the two queues. (The remaining parameter is equivalent to choosing the monetary units.) Finally, we caution that other cases (e.g., linear costs) can be quite different; see §EC.4.

5.4. Application to the Stochastic Model

We can directly apply the QR-T control derived above to the stochastic model. Figure 4 identifies three sharing regions to apply to the stochastic process $(Q_1(t), S_1(t), Q_2(t), S_2(t))$ once sharing has been activated. There are two regions for each direction of sharing; e.g., if sharing has been activated with pool 2 helping class 1, available pool 2 agents serve class 1 customers when the queue-length vector falls in the lower right region, whereas there is no sharing in the other two regions. The way to share is described in §4.2.

6. Choosing the Thresholds

We now consider how to choose the thresholds $\kappa_{1,2}$ and $\kappa_{2,1}$. These thresholds have two important roles: first, they automatically detect when the system becomes overloaded and, second, they prevent unwanted sharing in normal loading. If the thresholds are too large, then the queues may not reach them during the overload. (Abandonments necessarily keep the queues from increasing without bound, even under overloads.) On the other hand, as discussed in §4.1, if the thresholds are too small, then sharing may be activated too often, so that we may get inefficient sharing.

Unfortunately, the fluid analysis cannot reveal the "right" size of the thresholds, because the fluid queues are empty under normal loading. We need to understand the extent of the stochastic fluctuations, something that is not captured by the fluid approximation. At this point, it is convenient to apply many-server heavy-traffic limits to gain additional insight. To understand the general idea, it suffices to refer to established limits for the basic $M/M/n+M$ model, as in Garnett et al. (2002) and Whitt (2004). There, both fluid models and refined diffusion process models are obtained as limits as the scale increases, where scale is measured by the number n of servers. What is unusual here, though, is that we are simultaneously interested in the QED regime and the ED or overloaded regime.

The QED regime is appropriate to describe normal loading, which is what prevails before the overload occurs, whereas the ED regime is appropriate to describe the overloaded system. In both cases, the arrival rate is allowed to grow as $n \to \infty$, whereas the service rate and abandonment rate are held fixed. The important insight is that the queue lengths tend to be of order $O(\sqrt{n})$ in the QED regime, as depicted by the diffusion limit in the QED regime, whereas the queue lengths tend to be of order $O(n)$ in the ED regime, as depicted by the fluid limit in the ED regime.

Thus, to prevent unwanted sharing when the system is normally loaded, we should choose the thresholds to be of size bigger than $O(\sqrt{n})$. That ensures that the weighted-queue-difference processes, $D_{1,2}$ and $D_{2,1}$, will not move above the thresholds by random fluctuations. On the other hand, we should choose the thresholds to be $o(n)$, so that the thresholds will be asymptotically negligible compared to the $O(n)$ fluid content. Then, asymptotically, they will be exceeded instantaneously when the overload occurs and they will not significantly alter the queue ratios. From this simple reasoning, we see that it suffices to have $\kappa_{i,j}^{(n)} = O(n^p)$ as $n \to \infty$ for $1/2 < p < 1$. (Incidentally, that scaling also makes the thresholds out of reach in normal loading in the law-of-the-iterated logarithm scaling of $(n \log\log n)^{1/2}$.)

This asymptotic analysis shows that the thresholds chosen in this way are asymptotically optimal, both during normal loading and during overload incidents. Asymptotically, the thresholds will be exceeded negligibly often during normal loading; asymptotically, the thresholds do not alter the optimal average cost in the overload incidents. For the case of normal loading, we can apply the QED results in Garnett et al. (2002); for the overload incidents, the ED results in Whitt (2004) provide only heuristic support, because they apply only to the $M/M/n+M$ model. We intend to prove the ED fluid model for the X model with FQR-T in a subsequent paper.

Of course, we actually have a system with one fixed n. When we want to apply the theory to a real system, with a finite number of agents, it becomes hard to distinguish between $O(n)$ and $O(\sqrt{n})$. For example, if $n = 100$, then both $10 = 0.1n$ and $10 = \sqrt{n}$. Thus, as in any application of asymptotic results, we should numerically verify that the values chosen are appropriate, and refine them if necessary, for which we can use simulation. For example, for a system with 100 agents in each pool (and abandonment rates less than service rates), we found that $\kappa_{i,j} = 10$ is effective. We found that the performance is not too sensitive to the choice of the thresholds, provided that they are neither too small nor too large. We present simulation results for FQR-T under normal loading in §EC.5.2, including a sensitivity analysis for the thresholds.

7. Simulation Experiments

Our analysis has been based on a fluid approximation of a stochastic system. It remains to show that the fluid approximation is suitably accurate for the stochastic system and that the optimal control for the fluid model works well in the stochastic system. For those purposes, we conduct simulation experiments.

7.1. Accuracy of Fluid Approximation

In this subsection, we investigate the accuracy of the approximation. To show how the accuracy increases as the system becomes larger, we simulated three cases, each case represents an element in a sequence of queueing systems indexed by n, scaled to satisfy a many-server heavy-traffic limit in the ED regime as $n \to \infty$. We use the same fixed service and abandonment rates as before ($\mu_{i,i} = 1$, $\mu_{1,2} = \mu_{2,1} = 0.8$ and $\theta_i = 0.3$). We consider a fixed queue ratio $r = 1$. We

Table 1 Comparison of Steady-State Performance Measures in Fluid Approximation with Corresponding Simulation Results for the Markovian X Model

	$n = 25$		$n = 100$		$n = 400$	
Perf. meas.	Approx.	Sim.	Approx.	Sim.	Approx.	Sim.
Q_1	13.9	13.5 ±0.4	55.6	52.8 ±1.2	222.2	216.7 ±7.0
Q_1/n	0.56	0.54 ±0.02	0.56	0.53 ±0.01	0.56	0.54 ±0.02
Q_2	13.9	15.7 ±0.5	55.6	58.4 ±1.2	222.2	223.1 ±7.0
Q_2/n	0.56	0.63 ±0.02	0.56	0.58 ±0.01	0.56	0.56 ±0.02
Ratio	1.0	0.98 ±0.02	1.0	0.90 0.00	1.0	0.96 0.00
$Z_{1,2}$	4.2	4.8 ±0.2	16.7	17.7 ±0.3	66.7	66.4 ±2.2
$Z_{1,2}/n$	0.17	0.19 ±0.01	0.17	0.18 ±0.00	0.17	0.17 ±0.01
Cost (thousands)	1.37	1.79 ±0.01	19.35	20.28 ±0.81	299.6	299.8 ±19.2

Notes. For each n, there are n agents in each pool, with $\lambda_1 = 1.3n$, $\lambda_2 = n$, $\mu_{i,i} = 1$, $\mu_{1,2} = \mu_{2,1} = 0.8$, and $\theta_i = 0.3$. The thresholds $\kappa_{1,2} = \kappa_{2,1}$ are 3, 10, 30 for $n = 25, 100, 400$, respectively.

let the arrival rates be $\lambda_1 = 1.3n$ and $\lambda_2 = n$, when there are n agents in each service pool. The three cases we consider are $n = 25, 100, 400$. We let the thresholds for these three values of n be $\kappa_{1,2} = \kappa_{2,1} = 3, 10, 30$, respectively. (The thresholds were dropped when exceeded.)

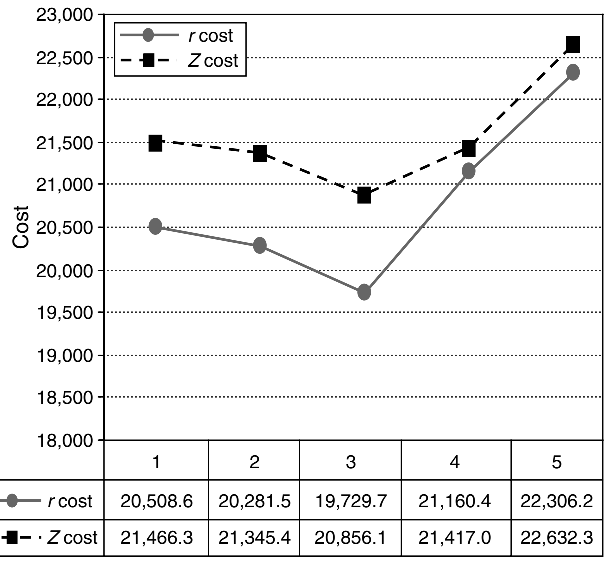
Table 1 shows the results. Each result is the average of five independent simulation runs having 300,000 arrivals in each run. The half-width of 95% confidence intervals, calculated using the t random variable with four degrees of freedom, are also given.

To show both the actual performance and the convergence to the fluid limit as n increases, we display both the direct values and the scaled values, dividing by n. Because the scaled values tend to be nearly independent of n, we witness the heavy-traffic fluid limit. We see that the approximations get better as n increases, but they are already not too bad when $n = 25$.

7.2. Comparing the Two Controls

In the fluid analysis, choosing the number of agents in each pool that are helping customers from the other class is equivalent to choosing the queue ratio, $r_{i,j}$. However, that is not the case in the actual stochastic system. With specified numbers of agents serving customers from the other class, the queue ratio fluctuates randomly. With specified queue ratios, the numbers of agents helping the other class fluctuates randomly. Moreover, with specified numbers of agents serving customers from the other class, the two queue-length processes evolve independently. In sharp contrast, with specified queue ratios, the queue-length processes are strongly dependent, as in Figure EC.8. This suggests that there is a big difference between the two controls in a real, stochastic system. We thus expect the average cost under FQR-T to be different than the average cost when fixing $Z_{i,j}$. We conducted simulation experiments to compare the two controls.

Figure 5 Comparison of Cost of Using FQR-T with Different Ratios and Cost of Fixing the Equivalent $Z_{1,2}$s



	1	2	3	4	5
● r cost	20,508.6	20,281.5	19,729.7	21,160.4	22,306.2
■ Z cost	21,466.3	21,345.4	20,856.1	21,417.0	22,632.3

Notes. In order, the five ratios considered are $r = 1.2, 1, 0.83, 0.6, 0.4$, where $r = 0.83$ is the optimal queue ratio according to the fluid model. Based on the fluid model, the equivalent $Z_{1,2}$s are, respectively, 15, 17, 19, 22, 25, where each $Z_{1,2}$ has been rounded up to the smallest integer greater than its actual value. The average cost of five independent simulation runs is written beneath each point. The cost function is (15), and the system parameters are $m_i = 100$, $\lambda_1 = 130$, $\lambda_2 = 100$, $\mu_{i,i} = 1$, $\mu_{1,2} = \mu_{2,1} = 0.8$, and $\theta_i = 0.3$.

To compare the two controls—FQR-T and fixed $Z_{i,j}$—we simulated a system with $m_i = 100$ agents in each pool, arrival rates $\lambda_1 = 130$ and $\lambda_2 = 100$, service rates $\mu_{1,1} = \mu_{2,2} = 1$ and $\mu_{1,2} = \mu_{2,1} = 0.8$, and abandonment rates $\theta_1 = \theta_2 = 0.3$. Because class 1 is overloaded, we took $\kappa_{2,1} = \kappa_{1,2} = 10$, but once we go over the threshold $\kappa_{1,2}$, we drop it, so that it becomes $\kappa_{1,2} = 0$.

Figure 5 presents simulation results comparing the two average costs for five different cases: (1) $r_{1,2} = 1.2, Z_{1,2} = 15$; (2) $r_{1,2} = 1, Z_{1,2} = 17$; (3) $r_{1,2} = 0.83$, $Z_{1,2} = 19$ (optimal point); (4) $r_{1,2} = 0.6, Z_{1,2} = 22$; and (5) $r_{1,2} = 0.4, Z_{1,2} = 25$. For each point, we fixed the queue-ratio $r_{1,2}$, and used FQR-T with this ratio. For each such $r_{1,2}$, there is an equivalent $Z_{1,2}$ in the fluid equations. Because this $Z_{1,2}$ is not necessarily an integer, we rounded it up to the smallest integer larger than $Z_{1,2}$, i.e., we used $\lceil Z_{1,2} \rceil$ in each simulation of the fixed-$Z_{i,j}$ control. According to the fluid approximation, the optimal queue ratio is $r_{1,2} = 0.83$, and the respective optimal $Z_{1,2}$ is equal to 18.4, rounded up to 19 in the simulation experiments.

For each case, we conducted five independent simulation runs using FQR-T, and five independent simulation runs with a fixed $Z_{1,2}$, each run with 300,000 arrivals. The independent replications make it possible to reliably estimate confidence intervals using the t statistic with four degrees of freedom. The large number of arrivals ensures that the transient behavior in the beginning of the simulation, before reaching steady state, does not affect the final simulation estimates of the steady-state averages. Additional simulation results are given in Table EC.1 in §EC.5, including the half-width of 95% confidence intervals and a comparison of the simulation to the fluid approximation.

There are several interesting observations to be made: First, the r-cost curve lies significantly below the Z-cost curve, which shows that FQR-T is a superior control. At the optimal point for FQR-T, the average cost under FQR-T is about 5.4% smaller than the average cost under the fixed-$Z_{1,2}$ control.

Secondly, FQR-T tends to be a more robust control. Small changes in r do not produce large changes in the cost. Note that the largest r value here (1.2) is three times as large as the smallest r value (0.4), whereas the largest Z value here (25) is only 1.6 times as large as the smallest Z value (15). Moreover, the average costs when using FQR-T with $r_{1,2} = 1.2$ and $r_{1,2} = 1$ are still smaller than the cost of fixing $Z_{1,2}$ at its optimal value. For further discussion, see §EC.5.

8. Conclusions

In this paper, we studied ways to respond to unexpected overloads in large-scale service systems. We considered the Markovian X model with two customer classes and two service pools, assuming that agents are more effective serving customers from their own class than customers from the other class, as specified by the inefficient-sharing conditions in (1) and (2). Thus, we want negligible sharing under normal loads, but we want to activate sharing when there is an unexpected overload at an unanticipated time, without knowing what the new arrival rates will be.

The main ideas for analyzing the performance and determining appropriate queue-ratio functions for the QR-T and FQR-T controls we propose are (i) to use steady-state analysis and (ii) to apply an approximating deterministic fluid model. The QR-T and FQR-T controls proposed for the actual stochastic system are direct applications of the corresponding optimal controls derived for the fluid model in §5.2. We developed an algorithm to find the optimal queue-ratio curves for a general convex cost function in Proposition 4 and §5.3. The resulting QR-T control is easily understood as a partition of the state space into three sharing regions, as depicted in Figure 4, with two regions for each direction of sharing.

In Proposition 5 we also provided strong justification for FQR-T when the cost function has the form $C(Q_1, Q_2) = c_1 Q_1^2 + Q_2^2$ for some constants c_1 and c_2. In that case, we proved that FQR-T is optimal for the fluid model (i.e., the optimal QR-T control reduces to an instance of FQR-T) and exhibited the explicit optimal queue-ratio parameters. Then the optimal queue-ratio parameters depend on the cost function C only via the single parameter c_1/c_2, which succinctly captures the relative importance of the two queues. For other sharing regions, see §EC.4.

The main ideas for gaining insight into appropriate threshold values were to apply (i) many-server heavy-traffic asymptotics and (ii) simulation. Heuristically, we showed that the thresholds should be asymptotically optimal simultaneously for periods of normal loading and for periods of overload. Asymptotically, no trade-off need be made. The requirement is that the thresholds should be of order $O(n^p)$ as $n \to \infty$, where $1/2 < p < 1$ and n is the system scale factor. We used simulation to verify that the thresholds work well for given finite n.

Our FQR-T (or QR-T) control is appealing for several reasons. First, it is automatic and simple; we need not directly discover the arrival rates to find out when overloads occur, and then decide what amount of sharing should be done. Instead, FQR-T automatically detects the time the system becomes overloaded, and then automatically enforces the optimal ratio, by observing only the size of the two queues. It is easier to use the information about the queues, which is readily available, than to use information about the arrival rates, which is not readily available. Moreover, simulation experiments indicate that FQR-T performs better (produces lower expected costs) than fixing $Z_{i,j}$ at their optimal values, even with known arrival rates; see Figure 5.

It remains to mathematically prove that the fluid model with QR-T or FQR-T arises as the many-server heavy-traffic limit of scaled overloaded queueing systems as the scale increases. It also remains to show that FQR-T is asymptotically optimal among all possible controls in such a limit. It remains to develop similar controls for more complex multiclass multipool systems with nonexponential service-time and abandonment-time distributions.

9. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://mansci.journal.informs.org/.

Acknowledgments
This research was supported by National Science Foundation Grant DMI-0457095.

References

Aksin, Z., M. Armony, V. Mehrotra. 2007. The modern call center: A multi-disciplinary perspective on operations management research. *Production Oper. Management* **16**(6) 665–688.

Armony, M., I. Gurvich. 2006. When promotions meet operations: Cross-selling and its effect on call-center performance. Working paper, Stern School of Business, New York University, New York.

Bassamboo, A., A. Zeevi. 2009. On a data-driven method for staffing large call centers. *Oper. Res.* **57**(3) 714–726.

Bell, S. L., R. J. Williams. 2005. Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: Asymptotic optimality of a thrshold policy. *Electronic J. Prob.* **10** 1044–1115.

Bhandari, A., A. Scheller-Wolf, M. Harchol-Balter. 2008. An exact and efficient algorithm for the constrained dynamic operator staffing problem for call centers. *Management Sci.* **54**(2) 339–353.

Gans, N., G. Koole, A. Mandelbaum. 2003. Telephone call centers: Tutorial, review and research prospects. *Manufacturing Service Oper. Management* **5**(2) 79–141.

Garnett, O., A. Mandelbaum, M. I. Reiman. 2002. Designing a call center with impatient customers. *Manufacturing Service Oper. Management* **4**(3) 208–227.

Gurvich, I., W. Whitt. 2009a. Scheduling flexible servers with convex delay costs in many-server service systems. *Manufacturing Service Oper. Management*. **11**(2) 237–253.

Gurvich, I., W. Whitt. 2009b. Service-level differentiation in many-server service systems via queue-ratio routing. *Oper. Res.* Forthcoming.

Tezcan, T., J. G. Dai. 2009. Dynamic control of N-systems with many servers: Asymptotic optimality of a static priority policy in heavy traffic. *Oper. Res.* Forthcoming.

Whitt, W. 2004. Efficiency-driven heavy-traffic approximations for many-server queues with abandonments. *Management Sci.* **50**(10) 1449–1461.

Whitt, W. 2005. Engineering solution of a basic call-center model. *Management Sci.* **51**(2) 221–235.

Whitt, W. 2006. Staffing a call center with uncertain arrival rate and absenteeism. *Production Oper. Management* **15**(1) 88–102.