Call-Routing Schemes for Call-Center Outsourcing

Noah Gans
OPIM Department, The Wharton School, University of Pennsylvania, Philadelphia, Pennsylvania 19104, gans@wharton.upenn.edu

Yong-Pin Zhou
Management Science Department, University of Washington Business School, Seattle, Washington 98195-3200, yongpin@u.washington.edu

Companies may choose to outsource parts, but not all, of their call-center operations. In some cases, they classify customers as high or low value, serving the former with their in-house operations and routing the latter to an outsourcer. Typically, they impose service-level constraints on the time each type of customer waits on hold.

This paper considers four schemes for routing low-value calls between the client company and the outsourcer. These schemes vary in the complexity of their routing algorithms, as well as the sophistication of the telephone and information technology infrastructure they require of the two operations. For three of these schemes, this paper provides a direct characterization of system performance. For the fourth, most complex, scheme the paper provides performance bounds for the important special case in which the service requirements of high- and low-value callers are the same. These results allow the systematic comparison of the performance of the various routing schemes. The results suggest that, for clients with large outsourcing requirements, the simpler schemes that require little client-outsourcer coordination can perform very well.

Key words: call routing; outsourcing; priority routing; call-center staffing; bursty call arrival

History: Received: May 24, 2004; accepted: June 2, 2006.

1. Introduction

Many companies choose to outsource parts of their call-center operations. That is, rather than serving their own customers, they subcontract part or all of their capacity to a company that specializes in call-center operations. This type of outsourcing has recently grown to become a global industry. For example, Convergys Corporation, a U.S.-based outsourcer with worldwide operations, reported 2003 revenues of $1.5 billion for its Customer Management Group, a division with 25,000 customer service representatives (CSRs) (2003, 2004). Wipro Spectramind, a large India-based outsourcer, reports that it has 9,300 employees who handle more than 4 million calls and 500,000 emails each month (Wipro Technologies 2004b).

A common reason for outsourcing is to lower costs. Human resources expenses associated with CSRs typically account for 60%-70% of call-center operating expenses. Outsourcers often have lower wage structures, which allow them to operate with a lower cost per call. For example, Wipro Spectramind reports that it typically provides 75% savings in labor costs to its clients (Wipro Technologies 2004a).

At the same time, companies that use outsourcers may continue to serve a significant fraction of their incoming calls. In one scheme, they classify their customers as belonging to one of two types: high-value customers, who are currently or potentially profitable; and low-value customers, who will never be highly profitable. The high-value customers are considered an important corporate asset, and they are served by in-house operations that the service provider trains and manages itself. The low-value customers are considered to be a “nuisance” and are routed to an outsourcer that provides a lower cost to serve but has less highly trained CSRs. Modern telephone infrastructure enables the identification of arriving calls as coming from high- or low-value customers, as well as the subsequent routing of calls.

Typically, client companies impose service-level constraints on the time each type of call spends waiting on hold. Common constraint forms include upper bounds on the average speed of answer (ASA), the average time calls spend on hold, and upper
bounds on fractiles of the waiting-time distribution. An example of the latter is, “80% of the calls must be handled in 20 seconds or less.” These types of constraints are applied to both low-value and high-value customers, although the specific level of service may differ. Together with projections concerning the arrival rates and service times of incoming calls, these service-level constraints drive lower bounds on the numbers of CSRs that must work during a given period.

Given adequate capacity to serve high-value customers, the service provider’s pool of CSRs will generally have additional capacity available to handle some low-value calls as well. This may be due to shift scheduling constraints, which force some planning periods to have higher-than-necessary staffing levels (see §3.2 in Gans et al. 2003). Even when the number of in-house CSRs is the absolute minimum required to meet the high-value customers’ service-level constraint, the organization may still have some capacity to take the low-value calls opportunistically, rather than on-demand.

Service providers have economic incentives to make use of this excess capacity. Outsourcing contracts commonly include volume-based (per call) or capacity-based (per agent–per hour) fees (for example, see Jackson 1999, K’Djah Worldwide). By taking some low-value calls in-house, the service provider can reduce some of these variable outsourcing costs. In addition, a company may consider its in-house CSRs to be of higher quality than its outsourcer’s—with better training, a more neutral accent, etc.—and prefer to use in-house CSRs as a first choice, when they are available.

The use of an outsourcer to handle some, but not all, low-value customers requires coordination between the two companies, both in the staffing of agents and in the routing of calls. Figure 1 depicts four schemes that vary in both the degree of coordination and resulting system complexity. In all of the figure’s panels, type-H calls come from high-value customers and type-L calls from low-value ones.

Panel (a) in the figure displays the simplest scheme. Here, type-H calls are served by a dedicated group of in-house CSRs, and any excess in-house capacity is dedicated to serving low-value calls. An “overflow”
scheme is used for routing low-value calls: If an arriving type-L call finds a type-L, in-house CSR idle, then the call is taken in house; otherwise it overflows to the outsourcer, where it queues and is served first-come, first-served (FCFS). We call this an overflow scheme with dedicated agents, or a dedicated-overflow scheme.

Note that, because capacity for high- and low-value customers is not pooled under this dedicated scheme, there will be instances at which both type-H calls are queued and in-house CSRs, dedicated to type-L calls, are idle, and vice versa. At these moments, a more complex routing scheme might be able to take fuller advantage of idle in-house capacity, thereby reducing outsourcing costs.

Panel (b) depicts a scheme that more fully uses idle in-house CSRs by pooling high- and low-value customer calls, but it does not pool capacity between the client company and the outsourcer (i.e., it uses an overflow scheme). We call this a pooled-overflow system.

Routing decisions in the pooled-overflow system can be much more complex than in the dedicated scheme. In particular, the decision to process an arriving type-L call in house now depends on the state of all in-house CSRs, rather than on just the availability of (at least) one of the in-house CSRs who is dedicated to type-L calls.

At the same time, under both of the overflow schemes there will still be instances at which there exist idle in-house CSRs and type-L calls queued at the outsourcer. Panel (c) of the figure displays an alternative scheme in which in-house CSRs dedicated to handling type-L calls are pooled with the outsourcer’s agents, and they can all access a common type-L queue. In this system, the routing options for type-L calls take the form of an “inverted-V” network (see Garnett and Mandelbaum 2000, Gans et al. 2003). Accordingly, we call this scheme inverted V. To minimize the number of type-L calls handled by the outsourcer, the client company can specify that type-L calls be preferentially routed to its CSRs whenever one of the agents it has dedicated to type-L calls is available.

Panel (d) depicts a routing scheme that pools across all in-house CSRs, as well as between the client company and the outsourcer. As in the inverted-V scheme, type-L calls are held in a common queue, and as in the pooled-overflow scheme, all in-house CSRs may take both type-H and type-L calls. Given the form of the routing options, this scheme is commonly called an N-network scheme (see Garnett and Mandelbaum 2000, Gans et al. 2003).

We note that the four schemes represent choices concerning the pooling of capacity across each of two dimensions. The left two panels’ schemes in the figure partition in-house capacity between type-H and type-L calls, while the right two panels’ systems pool in-house capacity across both types of calls. Similarly, the top two panels show schemes in which type-L calls overflow from the in-house group of CSRs to the outsourcer, while the bottom two panels depict schemes in which a common queue allows the pooling of type-L capacity across the two organizations. Thus, the N-network shown in Panel (d) has the most ability to pool, but this benefit comes at the cost of more difficult routing controls and more stringent requirements for the telecommunications infrastructure.

In the overflow schemes, information concerning the occupancy of in-house CSRs is available from the client company’s local automatic call distributor (ACD), and routing decisions are executed by the client company’s private automatic branch exchange. Thus, the decision of whether to serve an arriving type-L call in house or to have it overflow can be made and executed locally at the client, without having to coordinate with the outsourcer.

The inverted-V network requires that type-L calls be held in a common queue and routed directly either to the client’s or the outsourcer’s CSRs. This can be accomplished by having the client’s and outsourcer’s common long-distance carrier, the so-called public service telephone network (PSTN), hold all type-L calls in queue. (In voice-over internet protocol systems, the role of the PSTN can be played by a “hosted service” provider; see Dawson 2005.) Whenever a type-L call arrives, the PSTN polls first the client then the outsourcer, to see if an agent is available. Similarly, whenever an agent becomes available at either the client or the outsourcer, that call center’s ACD polls the PSTN to deliver a call, if any exist in the queue.

While the inverted-V scheme requires that calls be queued at the PSTN, it does not require complex coordination among the client, the outsourcer, and the
long-distance carrier. Furthermore, the extra costs of holding calls at the PSTN may be offset by reductions in line charges associated with the overflow schemes. Specifically, each time a call overflows from the client to the outsourcer, an extra telecommunications link is established. Such a link may be owned by the PSTN, in which case a charge is incurred per call, or it may be privately owned by the client and outsourcer.

Finally, full operation of the N-network requires that detailed occupancy information be collected from the client’s and outsourcer’s ACDs and fed back to a PSTN, which then uses the agent and queue occupancies at both locations to make complex routing decisions. Occupancy updates and routing decisions must be made in real time, each time a call arrives or a CSR becomes free. Thus, while the N network shares the line-charge advantage of the inverted-V scheme, the telecommunications and information technology infrastructure required to support its remote coordination is significantly more sophisticated.

In this paper, we investigate the relative effectiveness of these schemes, analyzing Markovian versions of the systems. The arrival processes of type-H and type-L calls are stationary and Poisson, with respective rates of \( \lambda_H \) and \( \lambda_L \). Service times of type-H calls are independent and identically distributed (i.i.d.), exponentially distributed random variables with mean \( 1/\mu_H \), and type-L service times have i.i.d., exponential distribution with mean \( 1/\mu_L \). The arrival processes and service times are assumed to be independent of each other, as well as of the system state.

Our main focus is the important special case in which service-time distributions and service-level standards are uniform across calls. When the contents of high- and low-value customers’ calls are roughly the same, we can directly characterize effective routing and staffing policies for the inverted-V and both overflow systems. When, in addition, service-level standards are the same across types, we can construct simple bounds on the behavior of the N network that are useful in comparing its performance with those of the other three schemes.

In §3 we focus on the dedicated-overflow and inverted-V schemes. For both of these systems, the queues for type-H and type-L calls are effectively decoupled, and most performance statistics can be calculated in a straightforward manner.

The analysis of the pooled-overflow scheme is more complex, and in §4 we determine an effective class of routing policies, along with a related performance characterization. Among all possible routing controls, we consider a class of policies that give priority to type-H calls. We show that, among type-H priority policies, easily computable “randomized threshold reservation policies” minimize the overflow of type-L calls to the outsourcer, subject to the service-level constraint on type-H calls. This result is an analogue of that of Gans and Zhou (2003), though the proof approach for the current system differs from, and is much more direct than, that used in Gans and Zhou (2003). In addition we show that, as the arrival rate of type-L calls grows large, the performance of these priority policies quickly converges to the global optimum, which does not assume priority routing of type-H calls.

In §5 we use the pooled-overflow scheme’s results to construct a lower bound on the outsourcer’s workload under the N-network scheme. For the case in which service rates and delay guarantees for type-H and type-L calls are equivalent, we also show that the number of CSRs required at the outsourcer can be bounded below by the number derived from an analogous \( M/M/m/\infty \) queue. This bound is in the spirit of a long line of pooling results, going back (at least) to Smith and Whitt (1981).

Section 6 then uses the results of §§3–5 to construct a numerical comparison of the three simpler schemes, relative to the N-network’s lower bounds, along two performance dimensions: required outsourcer workload and number of outsourcer CSRs. The results are positive and, in some cases, surprising:

- For large systems the pooled-overflow scheme performs quite well in both dimensions. Here, outsourcer workloads are nearly equal to the lower bound on that for the N-network system. Similarly, the number of extra outsourcer agents, above and beyond the lower bound, never exceeds more than one or two, a less-than 1% difference.
- In contrast, neither the dedicated-overflow nor the inverted-V systems achieves this level of performance in large systems. Even in very large examples, with roughly 5,000 agents required at the outsourcer, both of these schemes required five or six CSRs more than the associated lower bounds. This gap appears to
reflect these systems’ inability to pool type-L service across the many in-house CSRs dedicated to type-H calls.

• At the same time, there are cases in which both the dedicated-overflow and inverted-V systems outperform the pooled-overflow scheme. In particular, in examples in which the total stream of type-L calls is not as large—requiring less than 250 CSRs—and the fraction of type-L calls sent to the outsourcer is low—less than 50%—the pooled-overflow scheme may not minimize either the number of outsourcer CSRs or outsourcer’s offered load.

These last results provide a numerical counterexample that indicates that the type-H priority policies we consider for the pooled-overflow system are not necessarily optimal for smaller problem instances. It is also interesting to note that two distinct effects appear to degrade the pooled-overflow scheme’s performance.

One is a first-order effect concerning the relative scales of the type-H and type-L arrival processes. Specifically, when \( \lambda_L < \lambda_H \), type-H arrivals appear to crowd out in-house type-L service in the pooled-overflow system, and both the type-L load offered to the outsourcer as well as the required number outsourcer CSRs can be strictly greater than those for other routing schemes.

The other is a second-order effect, driven by the regularity of the type-L overflow process. As the outsourcer’s workload drops to less than half of the raw type-L load, the burstiness of overflows in the pooled-overflow system increases and appears to drive the need for extra outsourcer staffing, even when the pooled-overflow system minimizes the outsourcer’s offered load. Furthermore, analysis of the overflow process suggests that it is an increase in the coefficient of variation of the interoverflow time, rather than serial correlation, that drives the extra staffing.

Finally, in many of the examples, the performance of the simple dedicated-overflow system is identical to or nearly the same as that of the more complex inverted-V scheme. This occurs when type-L arrival rates are high, a result that is not unexpected: High arrival rates imply that the performance losses from the dedicated-overflow scheme’s inability to queue type-L calls in house are low.

These results are interesting on a number of levels. First, they show that the simple, suboptimal routing schemes we describe can perform nearly optimally in large systems. This result is positive in that these less-complex schemes can be both less expensive and less difficult to coordinate, when compared with more sophisticated routing schemes. Our results also echo those in Wallace and Whitt (2005), which shows that, given uniform service requirements across call types, adequate staffing levels, rather than sophisticated routing policies, can drive effective performance in skills-based routing systems. Lastly they suggest that, for large systems, variants of the pooled-overflow scheme, in particular, can be useful in determining outsourcer staffing, as well as for making call-routing decisions. In §7 we discuss these findings, as well as the limitations of our results.

We note that, while the numerical comparisons among the routing scheme are necessarily limited to systems with uniform service rates, the simpler routing schemes can be generalized to consider cases in which \( \mu_H \) may not equal \( \mu_L \). They can also be extended to cases in which the outsourcer-CSR’s service rate for type-L calls may differ from that of in-house CSRs. The appendices include the formal development and all the proofs of the performance analysis results presented in the paper, as well as some extensions for these more general systems.

The remainder of the paper is organized as follows. In §2 we review the related literature. Sections 3–5 develop the performance analysis of the various systems, and §6 describes the results of numerical tests that compare the various routing schemes. Section 7 concludes with a discussion of the results.

2. Literature Review

There is little in the academic literature that is specifically devoted to call-routing problems related to outsourcing. Aşkin et al. (2004) and Ren and Zhou (2004) address the related issue of contract design for call-center outsourcing, their models suppressing the queueing-control aspects of the problem on which we focus. Papers such as those by Cachon and Harker (2003), Benjaafar et al. (2004), and Chevalier et al. (1998) address contract issues related to outsourcing in a more stylized setting.
In contrast, there are many papers related to call-routing in call centers. We review the various groups in turn.

A recent paper by Wallace and Whitt (2005) develops and analyzes the performance of a family of heuristics for staffing and call routing in call centers that use skills-based routing. The policies analyzed in Wallace and Whitt (2005) differ from those we analyze; for example, they do not provide the performance guarantees that we seek. At the same time, both the paper’s assumptions—that all call-types share the same service requirements—and its broader conclusion—that elaborate routing policies need not be necessary to obtain near-optimal system performance—are echoed in our assumptions and results.

The pooled-overflow routing scheme analyzed in §4 is most closely related to papers by Bhulai and Koole (2003) and Gans and Zhou (2003), which model the mixing of high-priority, inbound calls with that of lower-priority work, such as “callbacks” or emails. The papers assume, however, that there exists an infinite backlog of low priority work, and therefore their system dynamics differ somewhat from those in the current paper. Nevertheless, both Bhulai and Koole (2003) and Gans and Zhou (2003) determine effective policies that are analogues of the policies identified in this paper.

More generally there exists a growing literature on work routing and capacity pooling in customer contact centers. Papers by Green (1985), Stanford and Grassmann (2000), and Shumsky (2004) model call-routing systems whose structures are closely related to ours. Tekin et al. (2004) analyzes the impact of pooling, without dynamic routing, in multiskilled centers. There also exists a growing body of work that uses asymptotic (rather than exact) analysis of routing policies, as both the offered load and the number of CSRs become large. Recent examples include Armony (2005), Gurvich et al. (2004), Armony and Maglaras (2004a, b), Harrison and Zeevi (2004), and Atar et al. (2004). A more general discussion of the application of these types of asymptotic results can be found in Gans et al. (2003).

A long stream of work in telecommunications and, more recently, in call centers analyzes the behavior of overflow systems. For example, Matsumoto and Watanabe (1985) and Meier-Hellstern (1989) represent multiple stages of overflow and model overflow from one stage to the next as a Markov Modulated Poisson Process (MMPP). Papers by Pinker and Shumsky (2000), Franx et al. (2006), Koole et al. (2003), and Chevalier and Tabordon (2003) perform approximate analysis of overflow schemes for skills-based routing in call centers. All of these papers model overflows from one group to another as occurring in only one state, when all servers in the group are busy. Because overflows in the pooled-overflow system can take place in an infinite set of states, our analysis is somewhat different and more complex. We use generating functions to solve the infinite sets of equations.

3. Dedicated-Overflow and Inverted-V Network Systems

In this section we describe our analysis of the two routing schemes in which the client maintains separate groups of CSRs, one dedicated to handling type-H calls and the other to type-L calls. The analysis assumes that the number of in-house CSRs, $m_I$, is known, as are the arrival and service rates. It then characterizes two measures of system performance: the outsourcer’s offered load of type-L calls and the minimum number of outsourcer CSRs it requires to meet the service-level constraint for type-L calls.

3.1. Dedicated-Overflow System

Consider a client company that elects to use the dedicated-overflow scheme shown in Panel (a) of Figure 1. The company partitions its $m_I$ in-house CSRs into two groups: $m_H$ CSRs are dedicated to the service of type-H calls and $m_L = m_I - m_H$ CSRs to the type-L calls.

The queueing system for type-H calls becomes a simple $M/M/m/\infty$ queue. We can then use the well-known Erlang-C formula, together with the “Poisson arrivals see time average” property, to determine common measures of customer delay on arrival (see Kleinrock 1975 and Wolff 1982). Because delay measures are decreasing in the number of servers, the client company can choose $m_H$ to be the minimum number of servers that meet its service-level target.

The type-L arrivals to these in-house CSRs behave like an $M/M/m/m$ system, the Markovian version of an Erlang-loss system. Therefore, the rate at which
calls arrive to the outsourcer is simply \( \lambda_i B(R_{L_i}, m_i) \), where \( R_{L_i} = \lambda_i / \mu_L \) and \( B(R_{L_i}, m_i) \) is the Erlang-B quantity (see Kleinrock 1975). In turn, per call costs associated with outsourcing are straightforward to calculate.

The calculation of the number of CSRs required at the outsourcer requires more work, however. First, the service-level requirement at the outsourcer must be adjusted to account for the fraction of calls, \( 1 - B(R_{L_i}, m_i) \), that were handled in house with no delay. For example, if the original upper bound on average delay is \( ASA^* \), then an average delay of at most \( ASA^O = ASA^* / B(R_{L_i}, m_i) \) is required at the outsourcer. What is more difficult is that the arrival process is not Poisson. Rather, it can be a bursty process, which implies that the number of CSRs required at the outsourcer may be higher than that required for a Poisson arrival with equivalent rate.

Given a fixed number of agents, \( m_i \), one can use one of several approaches—such as that of Meier-Hellstern (1989) or simulation—to calculate the service-level obtained at the outsourcer. In turn, one can search for \( m_O = \min \{ m \mid E[\text{delay}] \leq ASA^O \} \), the required outsourcer staffing level. In §6 we use simulation to search for \( m_O \).

### 3.2. Inverted-V Network for Type-L Calls

In the inverted-V network, shown in Panel (c) of Figure 1, the client company again partitions its \( m_i \) in-house CSRs into two groups: \( m_H \) CSRs are dedicated to the service of type-H calls and \( m_L = m_i - m_H \) CSRs to type-L calls. As in the dedicated-overflow scheme, the queue for type-H calls behaves as an Erlang-C system.

The inverted-V system differs from the dedicated-overflow system in the treatment of type-L calls. In the dedicated-overflow scheme, in-house throughput of type-L calls was straightforward to calculate, while the determination of the number of CSRs required at the outsourcer required more work. In the inverted-V system, the reverse is true.

More specifically, given \( m_O \) outsourcer CSRs, type-L occupancy in the inverted-V scheme is that of an Erlang-C system with offered load \( R_L = \lambda_i / \mu_L \) and \( m_L + m_O \) servers. Thus, to determine required outsourcer staffing, we first calculate \( m^* = \min \{ m \mid E[\text{delay}] \leq ASA^* \} \) and then set \( m_O = (m^* - m_L)^* \).

In contrast, the determination of the in-house throughput of type-L calls requires that we keep separate track of in-house and outsourcer CSRs. Section A of the online appendix contains the state transition diagram of a small inverted-V system.\(^1\)

Let \( \xi_{i,j,k} \) be the steady-state probability that there are \( i \) in-house and \( j \) outsourcer CSRs busy and \( k \) type-L calls waiting in queue. Then \( \sum_{k=0}^{\infty} \xi_{m_L, m_O, k} \) is the probability that all \( m_L + m_O \) servers are busy, which is the same as that for a simple Erlang-C system: \( C(R_L, m_L + m_O) \). Given \( C(R_L, m_L + m_O) \), one can invert the remaining \( (m_L \times m_O) \) matrix of balance constraints to determine the remaining \( \xi_{i, j, k} \) and, in turn, the throughput of type-L calls in house:

\[
m_L \cdot \mu_L \cdot C(R_L, m_L + m_O) \\
\sum_{i=0}^{m_L} \sum_{j=0}^{m_O} i \cdot \mu_L \cdot 1[i+j < m_L+m_O] \xi_{i, j, k},
\]

where \( 1[\cdot] \) is the indicator function. The overflow rate to the outsourcer is then \( \lambda_{L_i} \), less the quantity calculated in (1).

### 4. Pooled-Overflow System

The pooled-overflow scheme shown in Figure 1’s Panel (b) is more complex for the client company to implement. In the dedicated-overflow and inverted-V schemes, type-H service levels were maintained by simply segmenting the in-house CSRs, at the cost of a loss of capacity pooling. In the pooled-overflow scheme, the system can regain the benefit of pooled in-house capacity, but incoming type-L calls must be overflowed to the outsourcer—in sufficient quantities and at the right times—so that the type-H service level constraint continues to be met.

In this section, we consider policies that minimize the overflow of type-L calls, subject to a service-level constraint on type-H calls. We note that this objective minimizes per-call outsourcing costs, rather than the number of outsourcer CSRs. In §6 we will return to investigate numerically how this approach affects outsourcer staffing levels.

\(^1\) An online appendix to this paper is available at http://www.informs.org/Pubs/Supplements/MSOM/1526-5498-2007-09-01-0033-ecompanion.pdf.
The remainder of this section is dedicated to the analysis of in-house routing policies for type-L calls. In §4.1 we demonstrate that, among type-H priority policies, a simple class of threshold policies is optimal, and we indicate how the thresholds can be calculated. In §4.2 we show that, as \( \lambda_1 \to \infty \), the routing policy is globally optimal among nonpriority policies as well.

4.1. Randomized Threshold-Reservation Policies

As noted in the introduction, this routing problem is an analogue of a system analyzed in Gans and Zhou (2003). The essential difference between the two is the behavior of type-L calls. In the current model, type-L calls arrive according to a Poisson process and flow over to the outsourcer if not put into service immediately. In Gans and Zhou (2003), however, the behavior is less complex: There exists an infinite backlog of type-L calls, and one can put into service at any time. In fact, the infinite-backlog system can be viewed as a special case of the current problem, one in which \( \lambda_L \to \infty \). (See Proposition 2, below.) Despite this difference, we can apply the approach of Gans and Zhou (2003) to show that a simple class of threshold-based policies is optimal in the current problem.

We begin with some definitions. A routing policy is type-H priority if it puts arriving type-L calls into service only when there are no type-H calls waiting to be served. It is type-H work-conserving if type-H calls never wait in queue when there are idle CSRs, and the policy is stationary if its actions at a given epoch are history-independent.

It can be shown that, among type-H priority policies, there exist stationary, type-H work-conserving policies that are optimal. Furthermore, such an optimal policy can be determined using a linear program with \( O(m_H^2) \) variables and \( O(m_H^2) \) constraints. This characteristic holds for the more general case, in which \( \mu_H \) may or may not equal \( \mu_L \), as well as a broad range of service-level constraints. Given the similarity of the problem to Gans and Zhou (2003), we save a formal discussion of this analysis for §E.1 of the online appendix.

Here we further characterize this class of optimal policies for the special case in which \( \mu_H = \mu_L = \mu \) and the type-H service-level constraint is stated in terms of ASA. The use of type-H priority, type-H work-conserving policies implies that there are never both idle CSRs and type-H calls in queue. Thus, we can represent the state space (after actions are taken) using one dimension: states \( s \in [0, \ldots, m_I - 1] \) have \( (m_I - s) \) idle servers and no type-H calls in queue, while states \( s \in \{m_I, m_I + 1, \ldots \} \) have \( s \) type-H calls in queue. The use of stationary policies implies that routing decisions are based only on the current system state, \( s \), and it allows for the randomization between the two actions available in each state: accepting or rejecting the incoming type-L call. We let \( p_s \) (\( s \leq m_I - 1 \)) be the stationary probability that an arriving type-L call that finds the system in state \( s \) is routed to an in-house CSR.

Figure 2 shows the transition diagram of the resulting continuous time Markov chain (CTMC). This is a simple birth and death process whose tail states, \( s \geq m_I \), behave like those of an \( \text{M/M/1}/\infty \) system with arrival rate \( \lambda_H \) and service rate \( m_I \mu \). For \( \rho = \lambda_H / m_I \mu < 1 \), the system is stable, no matter what the choice of the routing probabilities, \( (p_0, \ldots, p_{m_I - 1}) \).

For any fixed set of routing probabilities, \( (p_0, \ldots, p_{m_I - 1}) \), we can determine \( \xi_s \), the steady-state probability that the CTMC is in state \( s \), for all \( s \), using the birth-and-death process’ local balance equations:

\[
\xi_{m_I} = \left[ \sum_{i=0}^{m_I-1} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} + \frac{1}{1-\rho} \right]^{-1},
\]

\[
\xi_s = \begin{cases} 
\xi_{m_I} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} & \forall 0 \leq s < m_I, \\
\xi_{m_I} \rho^{s-m_I} & \forall s \geq m_I.
\end{cases}
\]  

(2)

Because the state space is defined in terms of system occupancy, it is convenient for us to account for type-H calls’ service-level constraint in terms of occupancy as well, and we can use Little’s Law to find...
$D^* = \lambda_H \cdot \text{ASA}^*$, the average number in queue that corresponds to the target ASA.

We formulate a simple nonlinear program to find an optimal policy. Our original objective—to minimize the overflow rate of type-L calls—is equivalent to the minimization of system idleness: $\sum_{s=0}^{m_l-1} (m_l - s)\xi_s$. In addition, for stationary policies, the upper bound on the average number in the queue becomes $D^* \geq \sum_{q=0}^{\infty} q\xi_{m_l+q} = \xi_{m_l} \sum_{q=0}^{\infty} q\rho^q = \xi_{m_l} \hat{d}(\rho)$, where $\hat{d}(\rho) = \rho/(1-\rho)^3$.

Therefore, the constrained optimization problem can be specified as the following nonlinear program (NLP):

$$\begin{align*}
\min & \quad \sum_{s=0}^{m_l-1} \left[ (m_l - s)\xi_{m_l} \prod_{i=s}^{m_l-1} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} \right] \\
\text{s.t.} & \quad \xi_{m_l} = \left[ \left( \sum_{s=0}^{m_l-1} \prod_{i=s}^{m_l-1} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} + \frac{1}{1-\rho} \right)^{-1} \\
& \quad \hat{d}(\rho)\xi_{m_l} \leq D^* \\
& \quad 0 \leq p_i \leq 1, \quad \forall i \in \{0, \ldots, m_l-1\}. 
\end{align*}$$

**Theorem 1.** Suppose that $\rho < 1$. If the NLP (3)–(6) is feasible, then there exists a policy with the following form that is optimal: There exists a single $L \in \{0, \ldots, m_l-1\}$ such that (i) if $L > 0$, then $p_i = 1$ for all $i \in \{0, \ldots, L-1\}$; (ii) if $L < m_l - 1$, then $p_i = 0$ for all $i \in \{L+1, \ldots, m_l - 1\}$; and (iii) $p_i \in [0, 1]$.

We call the class of policies defined in Theorem 1 randomized threshold reservation policies. They are completely defined by the threshold, $L$, and the associated probability, $p_L$, and we will also refer to them as $(L, p_L)$ policies. One may think of them as reserving $m_l - (L + p_L)$ CSRs to handle type-H calls.

A sketch of the proof of the optimality of $(L, p_L)$ policies is as follows. Given any feasible set of routing probabilities, $p$, with elements $p_i < 1$ and $p_{i+1} > 0$ for some $i$, we can construct an alternative set, $p'$, with $p' > p$, and $p'_{i+1} < p_i$, such that (1) both $p$ and $p'$ obtain the same $\xi_{m_l}$ and (2) $p'$ obtains a strictly better objective function value than $p$. This implies that $p$ cannot be optimal. Therefore, any optimal set of routing probabilities must fall within the class described in Theorem 1.

While the NLP formulation (3)–(6) is useful in identifying the class of $(L, p_L)$ policies as optimal, we need not solve it explicitly to find the optimal $L$ and $p_L$. In particular, the following monotonicity property implies that the optimal $(L, p_L)$ policy can be found via line search in pseudo-O($n_l^2$) time.

**Proposition 1.** System idleness (3) is strictly decreasing in $p_i$, and $\xi_{m_l}$ (4) is strictly increasing in $p_i$ for all $i$.

### 4.2. Effectiveness of Type-H Priority Policies

So far we have assumed that priority is given to type-H calls. While this priority occurs naturally in many practical situations, it is also worthwhile asking how well type-H priority policies perform relative to some global standard.

The globally optimal throughput performance of these systems is not generally known, but we can construct an upper bound on the global optimum. In this subsection we present this bound and use it to test how well optimal type-H priority policies perform. In fact, our tests show that, when $\lambda_l$ is large relative to other event rates, system performance is quite close to the upper bound.

More specifically, in Gans and Zhou (2003) we show that there exist type-H priority policies that are globally optimal for the extreme case in which $\lambda_l = \infty$. We can show that the performance of this system provides an upper bound on the optimal system performance for any $\lambda_l < \infty$. Moreover, the upper bound becomes tight as $\lambda_l \to \infty$.

**Proposition 2.** Let $\lambda_l = \infty$ denote a system in which there always exists a type-L call waiting to be served.

(i) The globally optimal type-L throughput when $\lambda_l = \infty$ is at least as great as the globally optimal throughput for fixed $\lambda_l < \infty$.

(ii) The optimal type-H priority throughput for fixed $\lambda_l < \infty$ is at least as great as $\lambda_l/((\lambda_l + \lambda_H + m_l\mu)$ times the throughput achievable when $\lambda_l = \infty$.

Thus, for $\lambda_l$ that is “large” with respect to $\lambda_H$, $\mu$, and $m_l$, the performance of type-H priority policies should be excellent. A natural next question is, “How large is ‘large’?” The results of numerical tests indicate that the numbers are quite reasonable (see §B.2 of the online appendix). For small in-house systems requiring 20 type-H CSRs, a load of low-value calls that is five times that of the high-value calls is nearly optimal. Given the “80–20” maxim, that 20% of customers provide 80% of the value to a company, this...
appears to be a quite reasonable balance. Furthermore, as the scale of type-H traffic grows, the relative level of type-L traffic required to obtain nearly optimal performance systematically declines. In large systems requiring hundreds of type-H CSRs, low-value arrivals need only be half the rate of high-value traffic to provide excellent performance. Thus, when \( \mu_H = \mu_L \), type-H priority policies should have excellent, if not globally optimal, performance for relatively large systems.

### 5. N-Network Scheme

The controls needed to manage the N-network scheme optimally are more difficult to determine than those of the previous three schemes. While our analysis of the pooled-overflow system tracked the occupancy of in-house CSRs, that for the N network requires that we track server occupancy both in house and at the outsourcer, as well as queue lengths for type-H and type-L calls (Shumsky 2004 provides an approximate evaluation of the N network for a fixed-routing policy).

Therefore, rather than attempting to determine the optimal performance of the N network directly, we develop bounds on its performance. In §6 we then use these bounds to assess the relative effectiveness of the three other outsourcing schemes.

A simple bound on the minimum number of agents required at the outsourcer can be constructed using the performance of an \( M/M/m/\infty \) system. This bound requires that \( \mu_H = \mu_L \equiv \mu \) and that ASA targets are the same for both type-H and type-L calls:

\[
\text{PROPOSITION 3. Consider an N network with arrival rates } \lambda_H \text{ and } \lambda_L \text{, service rates } \mu_H = \mu_L \equiv \mu, \text{ in-house CSRs, } m_O \text{ outsourcer CSRs, and a common service-level constraint, } \text{ASA}^*, \text{ on the average delay of both type-H and type-L calls.}
\]

- (i) If there exists a call-routing policy that meets the service-level constraint in this N network, then an Erlang-C system with arrival rate \( \lambda_H + \lambda_L \)/\( \mu \) and find \( \max = \min \{m > R \mid \text{E[delay]} \leq \text{ASA}^* \} \) in the Erlang-C system. Then we let \( m_O = m^* - m \) be the lower bound.

Finally, we note that, together, the results of §4.2 and Proposition 3 suggest that, for systems with large \( \lambda_L \), the throughput performance of the pooled-overflow system should be very good. In terms of the outsourcer staffing level, however, it is not clear how well the pooled-overflow scheme compares with the N network and its lower bound. It is possible, for example, that in maximizing the throughput, the threshold reservation routing policy used in the pooled-overflow system creates very bursty overflow to the outsourcer and causes the staffing requirement to be much higher than the lower bound on the N network.

### 6. Performance Comparison of the Four Outsourcing Schemes

In this section, we compare the performance of the four outsourcing schemes along two dimensions: the number of CSRs required at the outsourcer and the type-L throughput that the outsourcer must serve. We begin by noting that there exist two orderings
of the performance achievable by the systems, represented by the two paths depicted in Figure 3. Because, in general, pooling provides the better usage of existing capacity, performance should improve along each path.

The first set of orderings is indicated by the solid arrows. Because the dedicated-overflow system is obtained by using a specific routing policy within an inverted-V system, and an N-network can be feasibly operated as an inverted-V scheme, the performance improvement along the path is obvious. Note also that the ordering holds on a sample-path basis, without regard to the specifics of the arrival processes or service-time distributions.

The second potential set of orderings is indicated by the dashed arrows. The N network may be operated as a pooled-overflow system, so it outperforms the pooled-overflow system. In theory, a similar ordering should also exist between the pooled- and dedicated-overflow schemes, but the particular pooled-overflow scheme studied in this paper, which restrictively assumes type-H priority, is not guaranteed to outperform the dedicated-overflow scheme on either performance dimension.

Thus, there exist a number of questions regarding the performance of the various routing schemes, including the following: (1) How do the simpler routing schemes compare with the lower bound for the N network? (2) How does the pooled-overflow scheme compare with the dedicated-overflow and inverted-V systems? (3) How much worse does the dedicated-overflow scheme perform when compared with the inverted-V? (4) In which cases does one scheme perform better or worse? Why? In this section we use numerical examples to provide insights into these questions.

6.1. Numerical Tests and Results

We report the results of a suite of 45 test cases. The examples assume that type-H and type-L calls have a common mean service time and service-level standard: $\mu^{-1} = 3.33$ minutes per call, and $ASA^* = 0.5$ minutes. We then systematically vary the type-H and type-L arrival rates, as well as the in-house staffing level, $m_I$, relative to the type-H offered load. Type-H arrival rates of $\lambda_H \in \{6, 30, 150\}$ per minute cover small, medium, and large in-house systems with offered loads $R_H = \lambda_H / \mu = \{20, 100, 500\}$. Type-L loads, $R_L = \lambda_L / \mu$, are scaled so that they range from one order of magnitude below to one order of magnitude above that for type-H arrivals: $R_L / R_H \in \{0.1, 0.5, 1.0, 2.0, 10.0\}$. Numbers of in-house CSRs, $m_H$, are roughly 5%, 25%, or 50% above $m_H$, the minimum required to meet the type-H service-level constraint of a 30-second ASA.

In all four schemes, we analytically determine the in-house throughput and overflow rate of type-L calls. For the inverted-V system, as well as the lower bound on the N-network scheme, numbers of outsourcer CSRs are analytically determined; for the dedicated-overflow and pooled-overflow schemes, simulation is used to determine minimal outsourcing staffing.

Table 1 summarizes the experiments and results. Columns 2–5 describe input parameters: $R_H$, $R_L$, $m_H$, and $m_I$. Columns 6–9 show the minimum number of CSRs required by each of the routing schemes—dedicated-overflow, pooled-overflow, inverted-V, and the lower bound on the N network—so that the system meets the type-L call service-level constraint. Columns 10–13 display the type-L workloads (in numbers of CSRs) handled by the outsourcer under each of the schemes.

The table’s results reveal a number of interesting phenomena. To facilitate our discussion, we also present graphic versions of the results in the figures below.

6.2. Results for Examples with Smaller Outsourcing Operations

Figure 4 shows the results for examples with smaller outsourcing operations, requiring at most 250 CSRs. In both panels, the horizontal axis plots the performance measure under the N-network scheme, and the vertical axis displays the difference between the other three schemes and the N-network scheme.
Table 1  Required Outsourcer Staffing and Offered Load for 45 Numerical Examples

<table>
<thead>
<tr>
<th>Example parameters</th>
<th>Required outsourcer CSRs</th>
<th>Outsourcer type-L load (CSRs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_N$ $R_L$ $m_N$ $m_L$</td>
<td>D-O P-O V N</td>
<td>D-O P-O V N</td>
</tr>
<tr>
<td>1 20 2 23 24</td>
<td>3 3 3 1</td>
<td>1.3 0.8 1.3 0.0</td>
</tr>
<tr>
<td>2 20 2 23 29</td>
<td>1 1 0 0</td>
<td>0.0 0.2 0.0 0.0</td>
</tr>
<tr>
<td>3 20 2 23 35</td>
<td>0* 1 0 0</td>
<td>0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>4 20 10 23 24</td>
<td>12 11 12 10</td>
<td>9.1 7.2 9.1 6.8</td>
</tr>
<tr>
<td>5 20 10 23 29</td>
<td>7 7 7 5</td>
<td>4.8 3.7 4.7 1.0</td>
</tr>
<tr>
<td>6 20 10 23 35</td>
<td>3 3 3 1</td>
<td>1.2 1.2 0.5 0.0</td>
</tr>
<tr>
<td>7 20 20 23 24</td>
<td>22 21 22 20</td>
<td>19.0 17.0 19.0 16.8</td>
</tr>
<tr>
<td>8 20 20 23 29</td>
<td>18 16 17 15</td>
<td>14.4 12.1 14.2 11.0</td>
</tr>
<tr>
<td>9 20 20 23 35</td>
<td>12 11 11 9</td>
<td>9.0 7.4 8.7 5.0</td>
</tr>
<tr>
<td>10 20 50 23 24</td>
<td>53 51 53 50</td>
<td>49.0 46.9 49.0 46.8</td>
</tr>
<tr>
<td>11 20 50 23 29</td>
<td>48 46 48 45</td>
<td>44.1 41.3 44.1 41.0</td>
</tr>
<tr>
<td>12 20 50 23 35</td>
<td>42 40 42 39</td>
<td>38.3 35.5 38.2 35.0</td>
</tr>
<tr>
<td>13 20 200 23 24</td>
<td>204 202 204 201</td>
<td>199.0 196.8 199.0 196.8</td>
</tr>
<tr>
<td>14 20 200 23 29</td>
<td>199 196 199 196</td>
<td>194.0 191.1 194.0 191.0</td>
</tr>
<tr>
<td>15 20 200 23 35</td>
<td>193 190 193 190</td>
<td>188.1 185.1 188.0 185.0</td>
</tr>
<tr>
<td>16 100 10 104 109</td>
<td>8 10 8 6</td>
<td>5.6 5.1 5.5 1.0</td>
</tr>
<tr>
<td>17 100 10 104 130</td>
<td>0* 0 0 0</td>
<td>0.0 0.3 0.0 0.0</td>
</tr>
<tr>
<td>18 100 10 104 156</td>
<td>0* 0* 0* 0*</td>
<td>0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>19 100 50 104 109</td>
<td>49 48 49 46</td>
<td>45.1 41.5 45.1 41.0</td>
</tr>
<tr>
<td>20 100 50 104 130</td>
<td>29 30 28 25</td>
<td>24.9 22.9 24.5 20.0</td>
</tr>
<tr>
<td>21 100 50 104 156</td>
<td>6 9 2 0</td>
<td>4.1 5.2 1.4 0.0</td>
</tr>
<tr>
<td>22 100 100 104 109</td>
<td>100 98 99 96</td>
<td>95.1 91.2 95.0 91.0</td>
</tr>
<tr>
<td>23 100 100 104 130</td>
<td>79 78 78 75</td>
<td>74.3 70.8 74.1 70.0</td>
</tr>
<tr>
<td>24 100 100 104 156</td>
<td>53 53 52 49</td>
<td>49.0 46.1 48.4 44.0</td>
</tr>
<tr>
<td>25 100 200 104 109</td>
<td>200 197 200 196</td>
<td>195.0 191.1 195.0 191.0</td>
</tr>
<tr>
<td>26 100 200 104 130</td>
<td>179 177 179 175</td>
<td>174.1 170.3 174.1 170.0</td>
</tr>
<tr>
<td>27 100 200 104 156</td>
<td>153 151 153 149</td>
<td>148.3 144.6 148.1 144.0</td>
</tr>
<tr>
<td>28 100 1,000 104 109</td>
<td>1,001 998 1,001 997</td>
<td>995.0 991.0 995.0 991.0</td>
</tr>
<tr>
<td>29 100 1,000 104 130</td>
<td>980 977 980 976</td>
<td>974.0 970.0 974.0 970.0</td>
</tr>
<tr>
<td>30 100 1,000 104 156</td>
<td>954 951 954 950</td>
<td>948.1 944.1 948.0 944.0</td>
</tr>
<tr>
<td>31 500 50 506 531</td>
<td>30 38 29 25</td>
<td>25.9 25.6 25.5 19.0</td>
</tr>
<tr>
<td>32 500 50 506 633</td>
<td>0* 0* 0* 0*</td>
<td>0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>33 500 50 506 759</td>
<td>0* 0* 0* 0*</td>
<td>0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>34 500 250 506 531</td>
<td>230 230 230 225</td>
<td>225.1 219.4 225.0 219.0</td>
</tr>
<tr>
<td>35 500 250 506 633</td>
<td>129 131 128 123</td>
<td>124.0 120.1 123.4 117.0</td>
</tr>
<tr>
<td>36 500 250 506 759</td>
<td>13 21 2 0</td>
<td>10.4 15.1 1.7 0.0</td>
</tr>
<tr>
<td>37 500 500 506 531</td>
<td>481 478 481 475</td>
<td>475.1 469.1 475.0 469.0</td>
</tr>
<tr>
<td>38 500 500 506 633</td>
<td>379 377 379 373</td>
<td>373.3 367.7 373.1 367.0</td>
</tr>
<tr>
<td>39 500 500 506 759</td>
<td>254 252 253 247</td>
<td>248.0 243.1 247.3 241.0</td>
</tr>
<tr>
<td>40 500 1,000 506 531</td>
<td>981 977 981 975</td>
<td>975.0 969.0 975.0 969.0</td>
</tr>
<tr>
<td>41 500 1,000 506 633</td>
<td>879 875 879 873</td>
<td>873.1 867.2 873.0 867.0</td>
</tr>
<tr>
<td>42 500 1,000 506 759</td>
<td>753 749 753 747</td>
<td>747.3 741.5 747.1 741.0</td>
</tr>
<tr>
<td>43 500 5,000 506 531</td>
<td>4,982 4,976 4,981 4,976</td>
<td>4,975.0 4,969.0 4,975.0 4,969.0</td>
</tr>
<tr>
<td>44 500 5,000 506 633</td>
<td>4,880 4,875 4,879 4,874</td>
<td>4,873.0 4,867.0 4,873.0 4,867.0</td>
</tr>
<tr>
<td>45 500 5,000 506 759</td>
<td>4,754 4,749 4,753 4,748</td>
<td>4,747.1 4,741.1 4,747.1 4,741.0</td>
</tr>
</tbody>
</table>

Note: D-O = dedicated-overflow; P-O = pooled-overflow; V = inverted-V; N = N-network.

*In these cases overflow is so infrequent that the client need not outsource.
The superior performance of the dedicated-overflow scheme, in terms of outsourcer load, also provides numerical evidence that type-H priority policies are not optimal with respect to minimizing outsourcer load for small $\lambda_L$. They only become optimal (asymptotically) as $\lambda_L$ grows large.

In contrast, in the medium-size examples, in which the lower bound on the required outsourcer CSRs falls between 25 and 125, the pooled-overflow system performs well on measures of offered load but not on required CSRs. That is, the pooled-overflow system sends fewer low-value calls to the outsourcer yet requires more outsourcer CSRs to meet the type-L service-level constraint. This suggests that, in these cases, it is the burstiness of the stream of overflows from the pooled-overflow system that drives up the required number of outsourcer CSRs. We will further investigate this phenomenon in §6.4.

### 6.3. Results for Examples with Larger Outsourcing Operations

Figure 5 shows the results for examples with larger outsourcing operations, requiring at least 250 CSRs. Its two panels are analogues of those in the previous figure.

The figure shows that, in these examples, the pooled-overflow scheme performs consistently well in terms of both overflow rate and required outsourcer CSRs. In all the figure’s examples, the type-L load served by the outsourcer is nearly tight on the lower bound. This is consistent with the results of Proposition 2 and the numerical results of §4.2. In cases that require 750 or more outsourcer CSRs (examples 40 through 46), the number of CSRs required for the pooled-overflow scheme is also consistently within one or two of the N network’s lower bound. This is excellent on a percentage basis: within 1% of the lower bound for very large systems.

Furthermore, there is also reason to believe that, in these examples, the actual number of outsourcer CSRs required for the N network may not be tight on the lower bound. More specifically, recall that the lower bound for the N network assumes that outsourcer CSRs also handle type-H calls, and note that, in examples 40–46, the type-H load is quite large, with $R_{H} = 500$. It may be the case that, given an inability to pool the examples’ high volumes of type-H calls...
across a large number of outsourcer CSRs, the number of outsourcer CSRs required for the N network might slip one or two from the lower bound.

Similarly, the consistently weaker results for the dedicated-overflow and inverted-V systems suggests that lack of in-house pooling of type-H and type-L service limits the effectiveness of the schemes in large systems.

6.4. Performance Drivers for the Pooled-Overflow Scheme

In our 45 numerical examples, the performance of the pooled-overflow system varied with the overall level of capacity required at the outsourcer. In some small systems, it performed better than the alternatives, but in others it performed significantly worse. For medium-size systems, it consistently performed better in terms of outsourcer load, but in some of these examples, it nevertheless required more outsourcer CSRs. Finally, for large systems it consistently performed best in both dimensions.

We hypothesize that it is the burstiness of the stream of overflows from the pooled-overflow system that drives up the CSR requirement in the medium-size examples. Here we further investigate the phenomenon by considering two manifestations of bursty behavior: large variations among individual interoverflow times and serial correlation among successive interoverflow times.

More precisely, consider the sequence of interoverflow times of type-L calls to the outsourcer, \( \{T_i \mid i = 1, 2, \ldots \} \). Given a stationary (version of such a) sequence of \( T_i \)'s, we consider the coefficient of variation (CV), as well as the one-step correlation coefficient (CC) between adjacent interoverflow times \( T_i \) and \( T_{i+1} \), which is defined as \( \text{cov}(T_i, T_{i+1})/\sigma_T^2 \).

If the overflow process were Poisson, then, by definition, the CV would be one and the CC would be zero. But overflow processes are typically not Poisson, and they can have CVs greater than one, as well as nonzero CCs.

The fact that overflow only occurs in one state, when all servers are busy, facilitates the analysis of the Erlang-B/dedicated-overflow scheme. In contrast, the analysis of the pooled-overflow system is much more complex, since overflows depend on the particular routing policy and can occur in many states.

Nevertheless, with some effort we can use difference equations to develop closed-form characterizations of both of the desired statistics. Analysis and results for the CV of the interoverflow time can be found in §D of the online appendix, and its (non-closed form) generalization, to systems in which \( \mu_H \) may or may not equal \( \mu_L \), can be found in §F.2 of the online appendix. Analysis for the one-step serial correlation appears in §E of the online appendix. These last results also include a (nonclosed form) generalization to the \( k \)-step serial correlation.

For each of our 45 numerical examples, we calculate the CV and the CC of the (stationary) interoverflow time in the pooled-overflow system. Figure 6 plots the two statistics against the percentage of type-L calls that are handled by the outsourcer.

The figure’s first panel shows that the CV of the interoverflow time can be quite large—in several examples more than three—and is negatively associated with the fraction of calls overflowing to the outsourcer.
When 100% of type-L calls overflow, the arrival process to the outsourcer is Poisson with $CV = 1$, and as the fraction of overflowing calls decreases (i.e., overflow becomes more sporadic), the CV appears to systematically increase. Examples with very small percentage overflows, however, show wide variation among CVs.

The second panel shows the contrasting behavior of the one-step CC. In Erlang-B systems (e.g., dedicated-overflow schemes), overflow occurs in only one state of the underlying Markov chain—when all servers are busy—and this implies that interoverflow times are independent (CC = 0) (see Wolff 1989 and Fischer and Meier-Hellstern 1989). In the pooled-overflow scheme, however, calls can overflow in (infinitely) many states, and there can be serial correlation. But the results here suggest that the correlation is quite weak.

In addition, the relationship between correlation and percent overflow has an inverted-U shape. When almost all type-L calls overflow, the arrival to the outsourcer is close to Poisson and the CC is close to zero. Conversely, when very few calls overflow, the underlying Markov chain at the in-house call center is likely to have experienced many transitions between consecutive overflows, so the correlation between interoverflow times again becomes very weak.

Thus, differences among the examples’ CVs appear to be more significant that those among CCs. An important complementary question is whether the differences among CVs or CCs appear to have a stronger effect on outsourcer staffing levels in the pooled-overflow scheme. To address this second question, we compare the required outsourcer staffing levels in the pooled-overflow scheme with those with an outsourcer whose overflow process is an interrupted Poisson process (IPP).

More specifically, an IPP is an MMPP in which the modulating CTMC has only two states, an “on” state, during which arrivals are Poisson (at rate $\lambda_L$ in our system), and an “off” state, during which there are no arrivals. We can use IPP arrivals to model interoverflow times whose first two moments match those of the pooled-overflow system and, at the same time, have zero CC (see Fischer and Meier-Hellstern 1989 and Kuczura 1973). In turn, we can use a spectral-expansion approach, similar to that described in Chakka and Harrison (2001), to determine the performance of an IPP/M/$m$/\infty analogue of the pooled-overflow system’s outsourcer and then search for the smallest $m$ that satisfies the required service-level constraint.

The result is an outsourcer staffing number that depends only on the CV of the interoverflow time, and not the CC, a number that is directly comparable to the required number of outsourcers in the pooled-overflow system. Let $m^p_{\sigma}$ be the required number of outsourcer CSRs in the pooled-overflow scheme, and let $m^\text{IPP}_{\sigma}$ be the IPP analogue.

![Figure 6](image-url)
Figure 7 plots \([m^{P-O}_{O} - m^{IPP}_{O}]\) against the pooled-overflow system’s interoverflow time CC for 41 of this section’s 45 examples. (Cases 17, 18, 31, and 32, which required no outsourcing, were excluded.) The figure shows two important relationships.

First, the IPP-derived staffing numbers are extremely close to those required by pooled-overflow system. Recall that required outsourcer staffing numbers range from near zero to nearly 5,000. In all but three cases, however, the IPP-driven number varies from the actual by as little as ±2.

The strength of the relationship can be quantified using simple linear regression. To account for large differences among the scales of the examples, we regress \(\log_{10}(m^{P-O}_{O})\) on \(\log_{10}(m^{IPP}_{O})\). As a benchmark, we note that perfect agreement among the various \(\log_{10}(m^{P-O}_{O})\)'s and \(\log_{10}(m^{IPP}_{O})\)'s would yield \(R^2 = 1.00\), an estimate of the slope of 1.00 (with \(p\)-value of zero) and an estimate of the intercept of zero. The actual results are not very different from the ideal: \(R^2 > 0.997\); and the estimate for the slope is 1.0225, with a vanishingly small \(p\)-value. But the estimate of the intercept of \(-0.0566\) is significantly different from zero, with \(p\)-value of 0.0038. Thus, the IPP model displays a tendency to understaff on examples with smaller numbers of CSRs. (An analogous regression with absolute, rather than log-log, scaling estimates the slope to be even closer to 1.0 and the intercept to be not significantly different from zero.)

Second, Figure 7 shows that the CCs of the interoverflow times do not appear to provide much additional information regarding required staffing.

It is worth noting that we have performed the same set of analyses using staffing levels recommended by the more general second-moment asymptotic results of Halfin and Whitt (1981; see a correction in Whitt 2006), rather than the IPP model. In these tests, we obtain qualitatively similar regression results.

Thus, the IPP model’s ability to track the pooled-overflow model’s required staffing levels appears to follow from the matching of the CV of the interoverflow time, rather than the specific “on-off” nature of the IPP model. More generally, our numerical results indicate that it is the CV, rather than the serial correlation, of the pooled-overflow system’s interoverflow times that strongly drives required staffing levels at the outsourcer.

7. Discussion

In this paper, we have considered four systems for managing the flow of low-value calls between a service provider and its outsourcer. These schemes vary in their ability to pool the service of low- and high-value calls in house, as well as their capacity to pool the service of low-value calls across the client and outsourcer.

We demonstrate that a relatively simple routing scheme, such as the pooled-overflow system, can perform nearly optimally in large systems. At the same time, we show that, in smaller systems, the reservation of adequate type-L capacity is necessary to effectively serve type-L calls, something that the pooled-overflow system does not do well.

It is important to note, however, that these insights hold only insofar as type-H and type-L service times and delay standards are similar; the relative performance of these schemes remains an open question when service statistics differ across customer classes. On the one hand, the additional flexibility afforded by the N-network system may allow it to significantly outperform the simpler schemes. On the other hand, results from Mandelbaum and Reiman (1998) and Whitt (1999) suggest that, in these cases, pooling across call types may not provide superior performance, at least in FCFS systems. Therefore, it may be the case that an inverted-V system, which partitions service classes but pools type-L capacity across organizations, performs well for systems with \(\mu_H \neq \mu_L\).

In addition to performance, it is also worth considering the schemes’ relative ease of implementation under more general conditions. When \(\mu_H \neq \mu_L\), or the calls’ delay standards differ, few changes are required to implement the dedicated-overflow or inverted-V systems, because they segregate the service of type-H and type-L calls. For the pooled-overflow system, however, both the optimal routing policy and the characteristics of the overflow process become more difficult to evaluate. Nevertheless, as §F of the online appendix shows, critical results for the pooled-overflow system do extend to these cases as well.

Alternatively, in some systems, type-L service times may vary between in-house CSRs and the outsourcer CSRs. For example, training or infrastructure differences may drive the two operations’ service times.
to differ. In these cases, the dedicated-overflow and pooled-overflow schemes are still relatively straightforward to implement, because the analysis of the in-house system remains unchanged. Here, however, the inverted-V system becomes more difficult to evaluate, as the pool of type-L CSRs now has two different service rates, one for in-house CSRs and the other for the outsourcer CSRs (see Armony 2005).

In contrast, in all these settings (as in the original case of uniform service standards), neither optimal controls nor performance characterizations are yet available for the N network. Indeed, performance analysis of the N network remains an open problem.

In addition to class-specific behavior and standards, there exist other potentially significant factors that warrant additional investigation. Three that bear mentioning here are arrival rate uncertainty, customer abandonment behavior, and the potential for the outsourcer to pool capacity across clients.

First, we note that call center staffing decisions are typically made using uncertain forecasts of arrival rates (Brown et al. 2005, Avramidis et al. 2004). If arrival-rate uncertainty is large enough, then supply-demand (CSR staffing versus arrival rate) mismatches can occur that dwarf lower-level queueing effects. In fact, even when arrival rates are known in advance, constraints on CSR schedules can cause similar gross capacity imbalances. In cases of significant arrival-rate uncertainty, the fluid-model approach used in recent papers may be effective in solving a combined staffing and routing problem (see Harrison and Zeevi 2004, Whitt 2006, and Bassamboo et al. 2006). At the same time, to the extent that arrival rates become known (after staffing but) before routing decisions are made, then the queueing and quality-of-service issues addressed in this paper may remain of interest.

A second important phenomenon is that impatience can lead customers to abandon the queue before being served (Zohar et al. 2002, Brown et al. 2005). In particular, customer abandonment tends to quickly stabilize overloaded systems, fundamentally changing waiting-time behavior and statistics. In these cases, models for the dedicated-overflow and inverted-V schemes, which partition type-H and type-L customers, can be extended in a straightforward manner to account for abandonment behavior (see Garnett et al. 2002 and Zeltyn and Mandelbaum 2004). Significant work is required, however, to incorporate abandonment behavior into the analysis of the pooled-overflow and N-network systems.

Finally, we note that it may be possible for an outsourcer to serve arrival streams of many clients with a single pool of CSRs. The benefit of real-time pooling in this manner is twofold: In addition to exploiting “square-root” laws of economies of scale, the pooling of uncorrelated streams of arrivals can potentially smooth out individual overflow processes that may be bursty, reducing the aggregate stream’s interoverflow time CV. Pooling across clients may also allow for smoothing of arrival rates over time. That is, if two clients’ overflow rates are projected to vary over the course of a day, but with different patterns, then the combined rates may be smoother and allow for more efficient scheduling of CSRs. For a scheme like this to work, however, the outsourcer’s CSRs must be (able to be) cross trained to handle many different clients’ calls, and its information systems must be robust and secure enough to integrate many clients’ databases on each CSR’s desktop. There may also be special difficulty in implementing this pooling on a call-by-call basis, as the mental “set-up” associated with switching among clients’ calls may be too costly.

Acknowledgments
Research supported by the Wharton Financial Institutions Center, the Wharton-INSEAD Alliance, the Fishman-Davidson Center for Service and Operations Management, the Center for International Business Education and Research at the University of Washington, and NSF Grant SBR-9733739.

References


