INDUSTRY STUDIES ASSOCIATION

INDUSTRY STUDIES ASSOCATION
WORKING PAPER SERIES

Call-Routing Schemes for Call-Center Sourcing

By

Noah Gans
Wharton Financial Institutions Center
OPIM Department
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

Yong-Pin Zhou
Department of Management Science
School of Business Administration
University of Washington
Seattle, WA 98195

2005
Industry Studies Association
Working Papers

WP-2005-05
http://isapapers.pitt.edu/

Call-Routing Schemes for Call-Center Outsourcing*

Noah Gans

OPIM Department
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

Yong-Pin Zhou

Management Science Department
University of Washington Business School
Seattle, WA 98195-3200

Companies may choose to outsource parts, but not all, of their call-center operations. In the course of studying contact centers in the telecommunications and financial services industries, we have observed the following (apparently) common scheme. A company classifies its customers as high or low-value, serving the former with their "in house" operations and routing the latter to an outsourcer. Typically, the company imposes 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, we provide a direct characterization of system performance. For the fourth, most complex, scheme we provide performance bounds for the important special case in which the service requirements of high and low-value callers are the same. These results allow us to systematically compare the performance of the various routing schemes. Our results suggest that, for clients with large outsourcing requirements, the simpler schemes that require little client-outsourcer coordination can perform very well.

---

*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.

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. Recently, this type of outsourcing has grown to become a global industry. For example, Convergys Corporation, a US-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) [16, 17]. 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 [50].

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 [49].

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 to be 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 waiting on hold, as well as 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 a higher-than-necessary staffing levels. (See §3.2 in Gans et al. [20].) Even when the number of in-house CSRs is the absolute minimum required to meet the high-value customers' service-level constraint, they 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

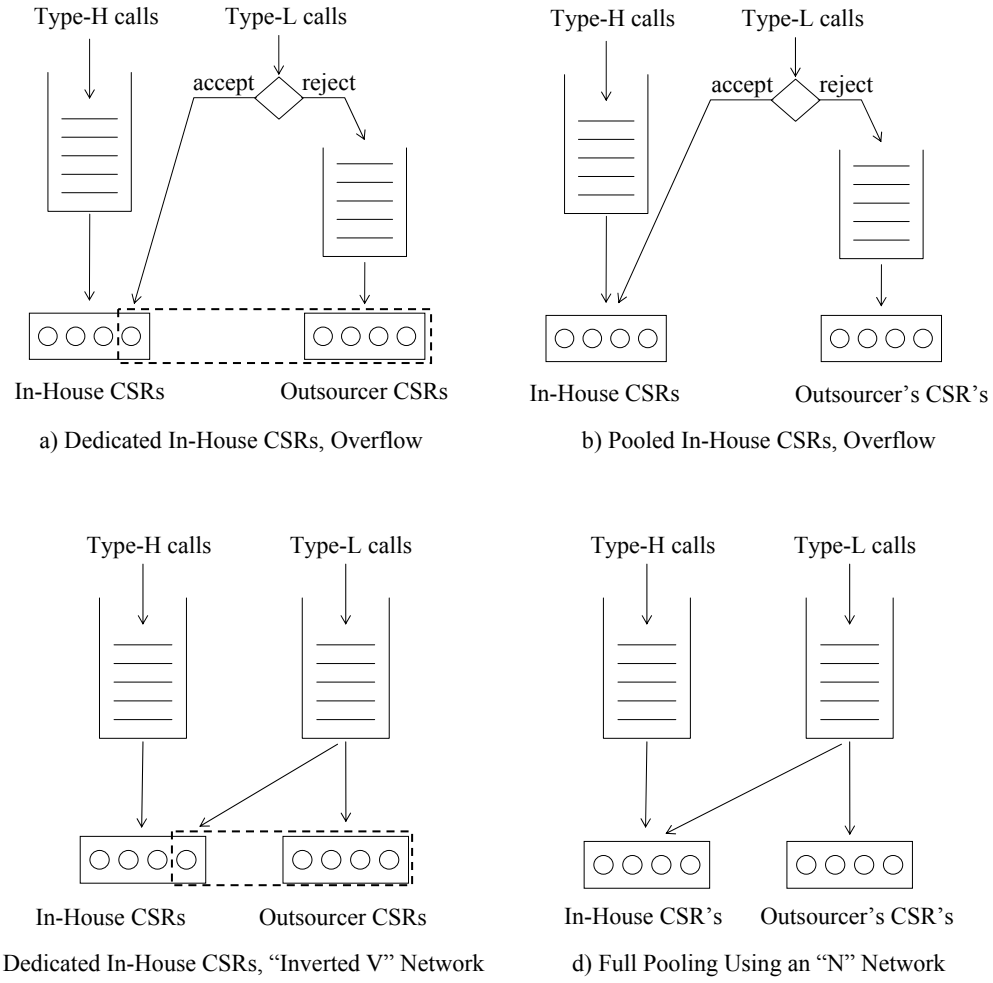

Figure 1: Routing Schemes for Type-L Calls

contracts commonly include volume-based (per-call) or capacity-based (per-agent-per-hour) fees. (For example, see [28, 29].) 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) 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 which 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 just the availability of (at least) one of the in-house CSRs that 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) 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[22], Gans et al. [20].) 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 are 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[22], Gans et al. [20]).

We note that the four schemes represent choices concerning the pooling of capacity across each of two dimensions. The left two panels' schemes 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 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 (PBX). 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 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 (VOIP) systems, the role of the PSTN can be played by a so-called "hosted service" provider. See Dawson [18].) 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 there exists any in 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. That 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 decision 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 λ_H and λ_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 Section 3 we focus on the dedicated-overflow and inverted-V schemes. For both of these system,

the queues for type-H and type-L calls are effectively decoupled, and most performance statistics can be calculated straightforwardly.

The analysis of the pooled-overflow scheme is more complex, and in Section 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 which 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 [21], though the proof approach for the current system differs from, and is much more direct than, that used in the previous paper. 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 Section 5 we use the pooled-overflow scheme's results to construct a lower bound on 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/∞ queue. This bound is in the spirit of a long line of pooling results, going back (at least) to Smith and Whitt [42].

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 5 or 6 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 which 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 $\lambda_L \ll \lambda_H$, 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 inter-overflow 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, due to the dedicated-overflow scheme's inability to queue type-L calls in house, are low.

Our results are interesting at 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 [45], 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 Section 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 μ_H may not equal μ_L. They can also be extended to cases in which the outsourcer-CSRs' service rate for type-L calls may differ from that of in-house CSRs. The appendices includes the formal development and all the proofs of the performance analysis results presented in the body of the paper, as well as some extensions for these more general systems.

The remainder of the paper is organized as follows. In Section 2 we review the related literature. Sections 3–5 develop the performance analysis of the various systems, and Section 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. Akşin et al [1] and Ren and Zhou [40] 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 Cachon and Harker [12], Benjaafar et al. [9], and Chevalier et al. [14], 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 [45] 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 [45] 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 Section 4 is most closely related to papers by Bhulai and Koole [10] and Gans and Zhou [21], 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 [10] and [21] 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 [24], Stanford and Grassmann [43], and Shumsky [41] model call-routing systems whose structures are closely related to ours. Tekin et al. [44] analyzes the impact of pooling, without dynamic routing, in multi-skilled centers. There also exists a growing body of work which 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 [3], Armony et al. [25], Armony and Maglaras [4, 5], Harrison and Zeevi [27], and Atar et al. [6]. A more general discussion of the application of these types of asymptotic results can be found in Gans et al. [20].

A long stream of work in telecommunications and, more recently, in call centers analyzes the behavior of overflow systems. For example, Matsumoto and Watanabe [35] and Meier-Hellstern [36] 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 [38], Koole and Talim [31, 32] and Chevalier and Tabordon [15] 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. 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, as well as 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 which 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/∞ queue. We can then use the well-known Erlang-C formula, together with the "Poisson arrivals see time average" (PASTA) property, to determine common measures of customer delay upon arrival. (See Kleinrock [30] and Wolff [51].) 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 meets 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 $\lambda_L B(R_L, m_L)$, where $R_L = \lambda_L/\mu_L$ and $B(R_L, m_L)$ is the Erlang-B quantity. (See Kleinrock [30].) 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, $1 - B(R_L, m_L)$, 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$^O = \frac{\text{ASA}^*}{B(R_L, m_L)}$ is required at the outsourcer. More difficultly, 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, one can use one of several approaches – such as that of Meier-Hellstern [36] or simulation – to calculate the service-level obtained at the outsourcer. In turn,

one can search for $m_O = \min\{m \,|\, \mathbf{E}\{\text{delay}\} \leq \text{ASA}^O\}$, the required outsourcer staffing level. In Section 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 $R_L = \lambda_L/\mu_L$ and $m_L + m_O$ servers. Thus, to determine required outsourcer staffing, we first calculate $m^* = \min\{m \,|\, \mathbf{E}\{\text{delay}\} \leq \text{ASA}^*\}$ 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. Appendix A contains the state transition diagram of a small inverted-V system.

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 $\xi_{i,j,0}$s and, in turn, the throughput of type-L calls in house:

$$m_L \cdot \mu_L \cdot C(R_L, m_L + m_O) \; + \; \sum_{i=0}^{m_L} \sum_{j=0}^{m_O} i \cdot \mu_L \cdot 1\{i + j < m_L + m_O\} \xi_{i,j,0} \,, \tag{1}$$

where $1\{\cdot\}$ is the indicator function. The overflow rate to the outsourcer is then λ_L, 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 Section 6 we will return to numerically investigate how this approach affects outsourcer staffing levels.

The remainder of this section is dedicated to the analysis of in-house routing policies for type-L calls. In Section 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 Section 4.2 we show that, as $\lambda_L \to \infty$, the routing policy is globally optimal, among non-priority 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 [21]. 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 overflow to the outsourcer if not put into service immediately. In [21], however, the behavior is less complex: there exists an infinite backlog of type-L calls, and one can be 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 [21] 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 it 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 $\mathrm{O}(m_I^2)$ variables and $\mathrm{O}(m_I^2)$ constraints. This characterization holds for the more general case, in which μ_H may or may not equal μ_L, as well as a broad range of service-level constraints. Given the similarity of the problem to [21], we save a formal discussion of this analysis for Appendix F.1.

Here, we further characterize this class of optimal policies for the special case in which $\mu_H = \mu_L \equiv \mu$ and the type-H service-level constraint is stated in terms of ASA, the average delay in queue. 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 no idle CSRs and $(s - m_I)$ 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.

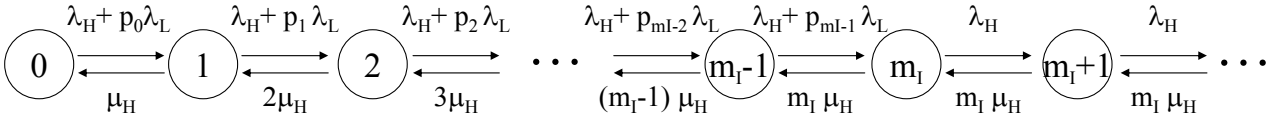

Figure 2: After-Action State Transition Diagram of CTMC

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 M/M/1/∞ system with arrival rate λ_H and service rate $m_I\mu$. For $\rho = \frac{\lambda_H}{m_I\mu} < 1$, 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 ξ_s, the steady-state probability that the CTMC is in state s, for all s, using the birth-and-death process's local balance equations:

$$\xi_{m_I} = \left[\left(\sum_{s=0}^{m_I-1} \prod_{i=s}^{m_I-1} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} \right) + \frac{1}{1-\rho} \right]^{-1}, \quad \xi_s = \begin{cases} \xi_{m_I} \prod_{i=s}^{m_I-1} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} & \forall 0 \leq s < m_I, \\ \xi_{m_I} \rho^{s-m_I} & \forall s \geq m_I. \end{cases} \tag{2}$$

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: $\sum_{s=0}^{m_I-1} (m_I - s)\, \xi_s$. In addition, for stationary policies, the upper bound on the average number in queue becomes $D^* \geq \sum_{q=0}^{\infty} q\, \xi_{m_I+q} = \xi_{m_I} \sum_{q=0}^{\infty} q\, \rho^q = \xi_{m_I} \tilde{d}(\rho)$, where $\tilde{d}(\rho) = \frac{\rho}{(1-\rho)^2}$.

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

$$\min \left\{ \sum_{s=0}^{m_I-1} \left[(m_I - s)\, \xi_{m_I} \prod_{i=s}^{m_I-1} \frac{(i+1)\,\mu}{\lambda_H \,+\, \lambda_L\, p_i} \right] \right\} \tag{3}$$

s. t.

$$\xi_{m_I} = \left[\left(\sum_{s=0}^{m_I-1} \prod_{i=s}^{m_I-1} \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i} \right) \,+ \frac{1}{1-\rho} \right]^{-1} \tag{4}$$

$$\tilde{d}(\rho)\, \xi_{m_I} \;\leq\; D^* \tag{5}$$

$$0 \;\leq\; p_i \;\leq\; 1, \qquad \forall i \in \{0, \ldots, m_I - 1\}. \tag{6}$$

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 $L \in \{0, \ldots, m_I - 1\}$ such that i) if $L > 0$ then $p_i = 1$ for all $i \in \{0, \ldots, L-1\}$; ii) if $L < m_I - 1$ then $p_i = 0$ for all $i \in \{L+1, \ldots, m_I - 1\}$; and iii) $p_L \in [0, 1]$.

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 them as reserving $m_I - (L + p_L)$ 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 $p'_i > p_i$ and $p'_{i+1} < p_{i+1}$, such that: 1) both p and p' obtain the same ξ_{m_I}; 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(m_I^2) time.

Proposition 1

System idleness (3) is strictly decreasing in p_i and ξ_{m_I} (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 λ_L is large relative to other event rates, system performance is quite close to the upper bound.

More specifically, in [21] 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 $\frac{\lambda_L}{\lambda_L+\lambda_H+m_I\mu}$ times the throughput achievable when $\lambda_L = \infty$.*

Thus, for λ_L that is "large" with respect to λ_H, μ, and m_I, 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 Appendix B.2.) 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 the 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 in order 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 optimally manage the N-network scheme 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 [41] provides an approximate evaluation of the N-network for a fixed routing policy.)

Therefore, rather than attempting to directly determine the optimal performance of the N-network, we develop bounds on its performance. In Section 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/∞ 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:

Proposition 3
Consider an N-network with arrival rates λ_H and λ_L, service rates $\mu_H = \mu_L \equiv \mu$, m_I in-house CSRs, m_O outsourcer CSRs, and a common service-level constraint, ASA*, on the average delay of both type-H and type-L calls.

i) *If there exists a call-routing policy which meets the service-level constraint in this N-network, then an Erlang-C system with arrival rate $\lambda_H + \lambda_L$, service rate μ, and $m_I + m_O$ servers will*

satisfy the service-level constraint as well.

ii) *The maximum in-house throughput of type-L calls in the N-network is bounded above by the optimal in-house throughput of type-L calls in an analogous pooled-overflow system with* $\lambda_L = \infty$ *and* m_I *in-house CSRs.*

Thus, given uniform service rates and ASA limits, a fully-pooled system will require no more outsourcer CSRs than its N-network analogue. Similarly, the performance of the pooled-overflow system with $\lambda_L = \infty$ provides an upper bound on in-house type-L throughput – equivalently, a lower bound on the outsourcer's offered load of type-L calls – in the N-network scheme. We note that part (ii) of Proposition 3 does not require that the call types' service-levels constraints be the same.

The type of pooling result stated in part (i) has a long history, going back at least to Smith and Whitt [42], and it is the assumption behind the recent performance analysis of policies for skills-based routing in Wallace and Whitt [45]. At the same time we note that, when assumptions concerning the uniformity of service requirements are violated, pooling need not be advantageous, at least in FCFS systems. For example, see Mandelbaum and Reiman [34] and Whitt [46].

Given the proposition's assumptions, we can use the associated fully-pooled system to calculate a simple lower bound on required outsourcer staffing. First, we let $R = (\lambda_H + \lambda_L)/\mu$ and find $m^* = \min\{m > R \,|\, \mathbf{E}\{\text{delay}\} \leq \text{ASA}^*\}$ in the Erlang-C system. Then we let $m_O = m^* - m_I$ be the lower bound.

Finally, we note that, together, the results of Section 4.2 and Proposition 3 suggest that, for systems with large λ_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 cause 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.

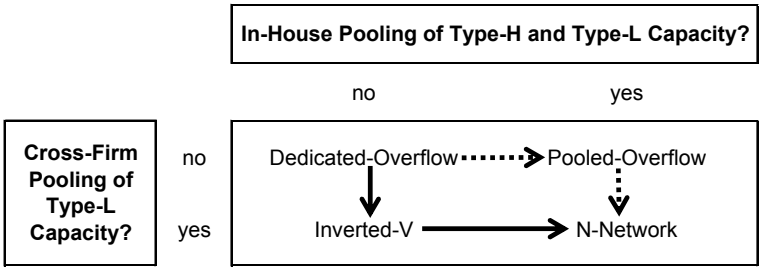

Figure 3: Potential Performance Orderings Among the Four Outsourcing Schemes

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 to 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 $\text{ASA}^* = 0.5$ 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_I, are roughly 5%, 25%, or 50% above m_H, the minimum required to meet the type-H service-level constraint of a thirty-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 outsourcer staffing.

	Example Parameters				Required Outsourcer CSRs				Outsourcer Type-L Load (CSRs)			
	R_H	R_L	m_H	m_I	D-O	P-O	V	N	D-O	P-O	V	N
1	20	2	23	24	3	3	3	1	1.3	0.8	1.3	0.0
2	20	2	23	29	1	1	0	0	0.0	0.2	0.0	0.0
3	20	2	23	35	0*	1	0	0	0.0	0.0	0.0	0.0
4	20	10	23	24	12	11	12	10	9.1	7.2	9.1	6.8
5	20	10	23	29	7	7	7	5	4.8	3.7	4.7	1.0
6	20	10	23	35	3	3	1	0	1.2	1.2	0.5	0.0
7	20	20	23	24	22	21	22	20	19.0	17.0	19.0	16.8
8	20	20	23	29	18	16	17	15	14.4	12.1	14.2	11.0
9	20	20	23	35	12	11	11	9	9.0	7.4	8.7	5.0
10	20	50	23	24	53	51	53	50	49.0	46.9	49.0	46.8
11	20	50	23	29	48	46	48	45	44.1	41.3	44.1	41.0
12	20	50	23	35	42	40	42	39	38.3	35.5	38.2	35.0
13	20	200	23	24	204	202	204	201	199.0	196.8	199.0	196.8
14	20	200	23	29	199	196	199	196	194.0	191.1	194.0	191.0
15	20	200	23	35	193	190	193	190	188.1	185.1	188.0	185.0
16	100	10	104	109	8	10	8	6	5.6	5.1	5.5	1.0
17	100	10	104	130	0*	1	0	0	0.0	0.3	0.0	0.0
18	100	10	104	156	0*	0*	0	0	0.0	0.0	0.0	0.0
19	100	50	104	109	49	48	49	46	45.1	41.5	45.1	41.0
20	100	50	104	130	29	30	28	25	24.9	22.9	24.5	20.0
21	100	50	104	156	6	9	2	0	4.1	5.2	1.4	0.0
22	100	100	104	109	100	98	99	96	95.1	91.2	95.0	91.0
23	100	100	104	130	79	78	78	75	74.3	70.8	74.1	70.0
24	100	100	104	156	53	53	52	49	49.0	46.1	48.4	44.0
25	100	200	104	109	200	197	200	196	195.0	191.1	195.0	191.0
26	100	200	104	130	179	177	179	175	174.1	170.3	174.1	170.0
27	100	200	104	156	153	151	153	149	148.3	144.6	148.1	144.0
28	100	1000	104	109	1001	998	1001	997	995.0	991.0	995.0	991.0
29	100	1000	104	130	980	977	980	976	974.0	970.0	974.0	970.0
30	100	1000	104	156	954	951	954	950	948.1	944.1	948.0	944.0
31	500	50	506	531	30	38	29	25	25.9	25.6	25.5	19.0
32	500	50	506	633	0*	0*	0	0	0.0	0.0	0.0	0.0
33	500	50	506	759	0*	0*	0	0	0.0	0.0	0.0	0.0
34	500	250	506	531	230	230	230	225	225.1	219.4	225.0	219.0
35	500	250	506	633	129	131	128	123	124.0	120.1	123.4	117.0
36	500	250	506	759	13	21	2	0	10.4	15.1	1.7	0.0
37	500	500	506	531	481	478	481	475	475.1	469.1	475.0	469.0
38	500	500	506	633	379	377	379	373	373.3	367.7	373.1	367.0
39	500	500	506	759	254	252	253	247	248.0	243.1	247.3	241.0
40	500	1000	506	531	981	977	981	975	975.0	969.0	975.0	969.0
41	500	1000	506	633	879	875	879	873	873.1	867.2	873.0	867.0
42	500	1000	506	759	753	749	753	747	747.3	741.5	747.1	741.0
43	500	5000	506	531	4982	4976	4981	4976	4975.0	4969.0	4975.0	4969.0
44	500	5000	506	633	4880	4875	4879	4874	4873.0	4867.0	4873.0	4867.0
45	500	5000	506	759	4754	4749	4753	4748	4747.1	4741.1	4747.1	4741.0

*In these cases overflow is so infrequent that the client need not outsource.

Table 1: Required Outsourcer Staffing and Offered Load for 45 Numerical Examples

Table 1 summarizes the experiments and results. Columns 2 though 5 describe input parameters: R_H, R_L, m_H, and m_I. Columns 6 though 9 show the minimum number of CSRs required by each of the routing schemes – dedicated-overflow (D-O), pooled-overflow (P-O), inverted-V (V), as well as the lower bound on the N-network (N) – so that the system meets the type-L call service-level constraint. Columns 10 through 13 display the type-L work-loads (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 graphical 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.

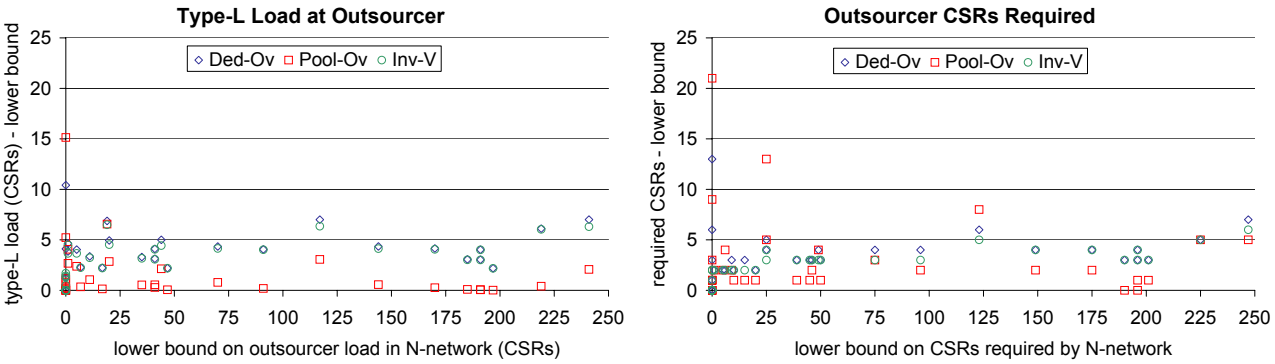

Figure 4: Performance vs. Lower Bound – Examples with Small Outsourcing Operations

The figure shows that, in most cases, the type-L load offered to the outsourcer, as well as the required number of outsourcer CSRs, falls within 5 or 6 CSRs of the lower bound. For small outsourcing operations this scale of excess is large in percentage terms, but as the size of the required outsourcing operation increases to 150 CSRs and above, this represents a premium of less than 3%.

The figure also shows that, even though in most cases the pooled-overflow scheme tends to outperform both the dedicated-overflow and inverted-V schemes, there are cases in which the pooled overflow system is outperformed by other two schemes.

In the small cases, in which the lower bound on the required number of CSRs is no more than 25, when the pooled-overflow scheme is outperformed, it is usually outperformed along both performance dimensions. In these cases, it appears that type-H arrivals unnecessarily "crowd out" type-L service in the pooled in-house systems, and it is better to protect type-L capacity by eliminating, rather than promoting, the pooling of type-L and type-H service.

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 λ_L. They only become optimal (asymptotically) as λ_L grows large.

In contrast, in the medium-sized 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 pooled-overflow system that drives up the required number of outsourcer CSRs. We will further investigate this phenomenon in Section 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.

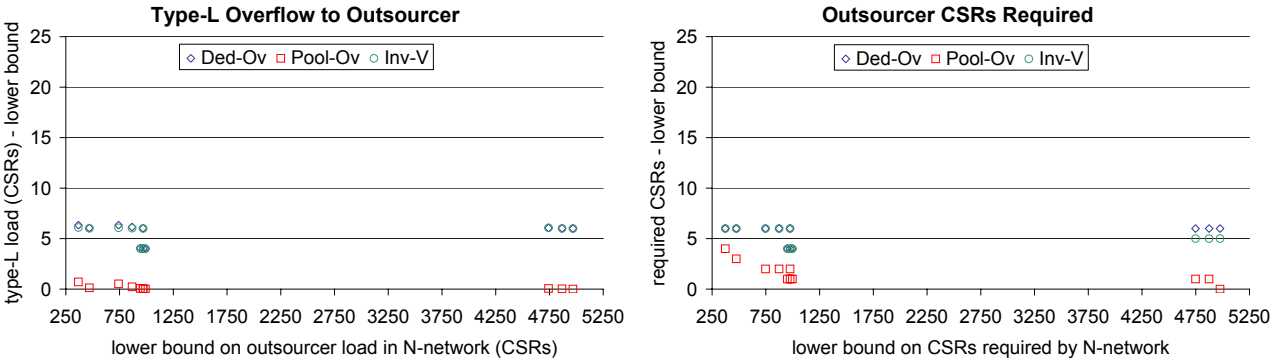

Figure 5: Performance vs. Lower Bound – Examples with Large Outsourcing Operations

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 of 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 Section 4.2. In cases which 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 to 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 sized 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 pooled-overflow system that drives up the CSR requirement in the medium-sized examples. Here, we further investigate the phenomenon by considering two manifestations of bursty behavior: large variations among individual inter-overflow times, and serial correlation among successive inter-overflow times.

More precisely, consider the sequence of inter-overflow times of type-L calls to the outsourcer, $\{T_i | \, i = 1, 2, \ldots\}$. Given a stationary (version of such a) sequence of T_is, we consider the coefficient of variation (CV) as well as the 1-step correlation coefficient (CC) between adjacent inter-overflow 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 non-zero 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 inter-overflow time can be found in Appendix D, and its (non-closed form) generalization, to systems in which μ_H may or may not equal μ_L, can be found in Appendix F.2. Analysis for the 1-step serial correlation appears in Appendix E. These last results also include (a non-closed 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) inter-overflow 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 inter-overflow time can be quite large – in several examples more than 3 – and is negatively associated with the fraction of calls overflowing to

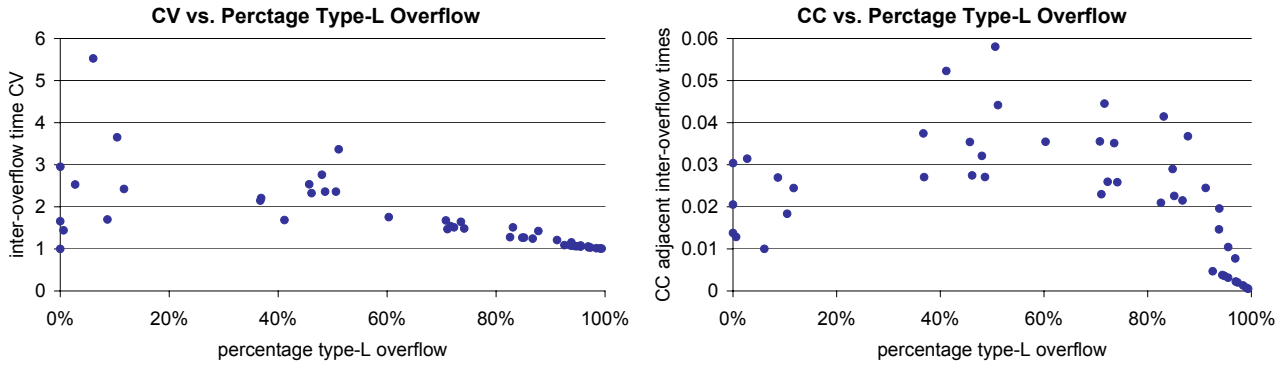

Figure 6: Extra CV and CC of the Inter-Overflow Time vs. Percent of Type-L Calls Overflowing

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 1-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 inter-overflow times are independent (CC=0). (See Wolff [52] and Fischer and Meier-Hellstern [19].) 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 inter-overflow 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 λ_L in our system), and an "off" state, during which there are no arrivals. We can use IPP arrivals to model inter-overflow 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 [19] and Kukzura [33].) In turn, we can use a spectral-expansion approach, similar to that described in Chakka and Harrison [13], to determine the performance of an IPP/M/m/∞ 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 which depends only on the CV of the inter-overflow time, and not the CC, a number that is directly comparable to the required number of outsourcers in the pooled-overflow system. Let m_O^{P-O} be required number of outsourcer CSRs in the pooled-overflow scheme, and let m_O^{IPP} be the IPP analogue.

Figure 7 plots $\left[m_O^{P-O} - m_O^{IPP}\right]$ against the pooled-overflow system's inter-overflow 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.

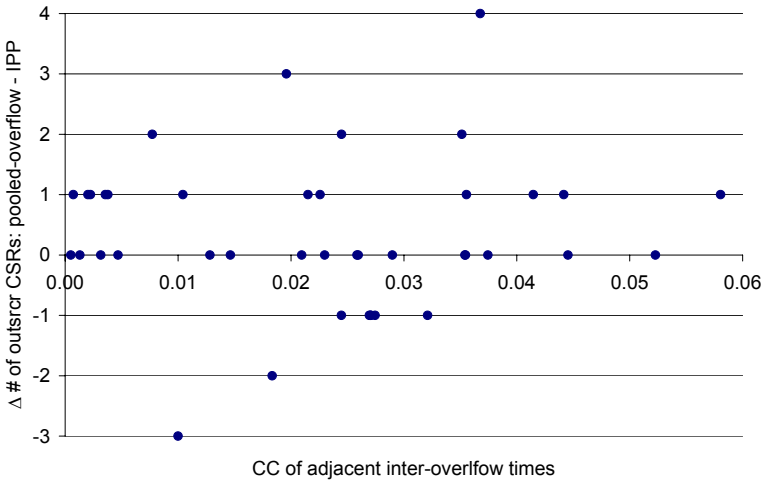

Figure 7: $m_O^{P-O} - m_O^{IPP}$ versus CC in the Pooled-Overflow System

First, the IPP-derived staffing numbers are extremely close to those required by pooled-overflow (P-O) system. Recall that required outsourcer staffing numbers range from near-zero to nearly 5,000. In all but 3 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 $\log_{10}(m_O^{P-O})$ on $\log_{10}(m_O^{IPP})$. As a benchmark, we note that perfect agreement among the various $\log_{10}(m_O^{P-O})$'s and $\log_{10}(m_O^{IPP})$'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 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 inter-overflow 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 recom-

mended by the more general 2nd-moment asymptotic results of Halfin and Whitt (see Halfin and Whitt [26], as well as a correction in Whitt [47]), 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 inter-overflow 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 inter-overflow 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, they 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, results from Mandelbaum and Reiman [34] and Whitt [46] 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, since 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 Appendix F 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 difference 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, since the analysis of the in-house system remains unchanged.

Here, however, the inverted-V system becomes more difficult to evaluate, since the pool of type-L CSRs now has two different service rates, one for in-house CSRs and the other for the outsourcer CSRs'. (For example, see Armony [3].)

In contrast, in all of these settings (as in the original case of uniform service standards) neither optimal controls and 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 which 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. [11] and Avramidis et al. [7]). 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 [27], Whitt[48], and Bassamboo et al. [8].) 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 queue before being served (Zohar et al. [54] and Brown et al. [11]). 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 straightforwardly extended to account for abandonment behavior. (See Garnet et al. [23] and Zeltyn and Mandelbaum [53].) 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 two-fold: 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 inter-overflow 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.

References

[1] O. Z. Akşin, F. de Véricourt, and F. Karaesmen. Call Center Outsourcing Contract Design and Choice. Working Paper, Koç University, 2004.

[2] E. Altman and A. Schwartz. Markov Decision Problems and State-Action Frequencies. *SIAM Journal of Control and Optimization*, 29:786–809, 1991.

[3] M. Armony. Dynamic Routing in Large-Scale Service Systems with Heterogenous Servers. *Queueing Systems*, 51(3-4):287 - 329, 2005.

[4] M. Armony and C. Maglaras. On customer Contact Centers with a Call-Back Option: Customer Decisions, Sequencing Rules, and System Design. *Operations Research*, 52(2), 2004.

[5] M. Armony and C. Maglaras Contact Centers with a Call-Back Option and Real-Time Delay Information. *Operations Research*, 52(4):527-545, 2004.

[6] R. Atar, A. Mandelbaum, and M. Reiman. Scheduling a Multi-Class Queue with Many Exponential Servers: Asymptotic Optimality in Heavy Traffic. *Annals of Applied Probability*, 14(3):1084-1134, 2004.

[7] A. N. Avramidis, A. Deslauriers, and P. L'Ecuyer. Modeling Daily Arrivals to a Telephone Call Center. *Management Science*, 5:896–908, 2004.

[8] A. Bassamboo, J. M. Harrison and A. Zeevi. Design and Control of a Large Call Center: Asymptotic Analysis of an LP-Based Method. Forthcoming, *Operations Research*, 2006.

[9] S. Benjaafar, E. Elahi, and K. L. Donohue. Outsourcing via Service Quality Competition. Working Paper, University of Minnesota, 2004.

[10] S. Bhulai and G. Koole. A Queueing Model for Call Blending in Call Centers. *IEEE Transactions on Automatic Control*, 48:1434-1438, 2003.

[11] L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. Statistical Analysis of a Telephone Call Center: a Queueing Science Perspective. To appear in *Journal of the American Statistical Association*.

[12] G. P. Cachon and P. T. Harker. Competition and Outsourcing with Scale Economies. *Management Science*, 48:1314–1334, 2003.

[13] R. Chakka and P. G. Harrison. The MMCPP/GE/c Queue. *QUESTA*, 38:307–326, 2001.

[14] P. Chevalier, Y. De Rongé, N. Tabordon, and L. Talbot. Services Subcontracting: An Economic Analysis. Working Paper, Université Catholique de Louvain, 1998.

[15] P. Chevalier and N. Tabordon. Overflow Analysis and Cross-Trained Servers. *International Journal of Production Economics*, 85:47-60, 2003.

[16] Convergys Corporation. 2003 Annual Report. http://www.convergys.com/annual_report.html.

[17] Convergys Corporation. Customer Care Brochure. http://www.convergys.com/pdf/brochures/ccbrochure.pdf.

[18] K. Dawson. Call Centers in the Cloud. *Call Center Magazine*, (February 2005), 28–32, 2005.

[19] W. Fischer and K. Meier-Hellstern. The Markov-Modulated Poisson Process (MMPP) Cookbook. *Performance Evaluation*, 18:149–171, 1989.

[20] N. Gans, G. Koole, and A. Mandelbaum. Telephone Call Centers: Tutorial, Review, and Research Prospects. *Manufacturing & Service Operations Management*, 5:79–141, 2003.

[21] N. Gans and Y-P. Zhou. A Call-Routing Problem with Service-Level Constraints. *Operations Research*, 51:255-271, 2003.

[22] O. Garnett and A. Mandelbaum. An Introduction to Skills-Based Routing and Its Operational Complexities. Teaching Note, Technion, 2000.

[23] O. Garnett, A. Mandelbaum, M. Reiman. Designing a Call Center with Impatient Customers. *Manufacturing & Service Operations Management*, 4:208–227, 2002.

[24] L. Green. A Queueing System with General-Use and Limited-Use Servers. *Operations Research*, 33(1):168–182, 1985.

[25] I. Gurvich, M. Armony, and A. Mandelbaum. Staffing and Control of Large-Scale Service Systems with Multiple Customer Classes and Fully Flexible Servers. Working Paper, NYU, 2004.

[26] S. Halfin and W. Whitt. Heavy-Traffic Limits for Queues with Many Exponential Servers. *Operations Research*, 29:567–587, 1981.

[27] J. M. Harrison and A. Zeevi. Dynamic Scheduling of a Multiclass Queue in the Halfin and Whitt Heavy Traffic Regime. *Operations Research*, 52(4):243-257, 2004.

[28] K. E. Jackson. Snared by the Web of Outsourcing Opportunities? *Call Center Magazine*, web edition, November 11, 1999. http://www.callcentermagazine.com/article/CCM20000503S0023.

[29] K'Djah Worldwide. Call Center Outsourcing: Pricing Models. http://www.kdjah.com.

[30] L. Kleinrock. *Queueing Systems. Volume I: Theory*, John Wiley & Sons, 1975.

[31] G. Franx, G. Koole, and A. Pot. Approximating Multi-Skill Blocking Systems by HyperExponential Decomposition. Forthcoming, *Performance Evaluation*, 2006.

[32] G. Koole, A. Pot, and J. Talim. Routing Heuristics for Multi-Skill Call Centers. *Proceedings of the 2003 Winter Simulation Conference*, 1813–1816, 2003.

[33] A. Kuczura. The Interrupted Poisson Process as an Overflow Process. *The Bell System Technical Journal*, 52(3):437–448, 1973.

[34] A. Mandelbaum and M. I. Reiman. On Pooling in Queueing Networks. *Management Science*, 44:971–981, 1998.

[35] J. Matsumoto and Y. Watanabe. Individual Traffic Characteristics of Queueing Systems with Multiple Poisson and Overflow Inputs. *IEEE Trans. on Comm.*, COM-33(1):1–9, 1985.

[36] K. S. Meier-Hellstern. The Analysis of a Queue Arising in Overflow Models. *IEEE Transactions on Communications*, 37(4):367–372, April 1989.

[37] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models : An Algorithmic Approach.* Number 2 in Johns Hopkins Series in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 1981.

[38] E. Pinker and R. Shumsky. The Efficiency-Quality Tradeoff of Crosstrained Workers. *Manufacturing & Service Operations Management* 2:32–48, 2000.

[39] M. L. Puterman. *Markov Decision Processes.* John Wiley & Sons, New York, 1994.

[40] Z. J. Ren and Y-P. Zhou. Call Center Outsourcing: Coordinating Staffing Level and Service Quality. Working Paper, University of Washington, 2004.

[41] R. Shumsky. Approximation and Analysis of a Queueing System with Flexible and Specialized Servers. *OR Spectrum*, 26:307–330, 2004.

[42] D. R. Smith and W. Whitt. Resource Sharing for Efficiency in Traffic Systems. *Bell System Technical Journal*, 60(13): 39–55, 1981.

[43] D. A. Stanford and W. K. Grassmann. Bilingual Server Call Centers. In *Analysis of Communication Networks: Call Centres, Traffic and Performance*, D.R. McDonald and S.R.E. Turner (eds.). Fields Institute Communications 28:31–47, 2000.

[44] E. Tekin, W. J. Hopp, and M. P. Van Oyen. Pooling Strategies for Call Center Agent Cross-Training. Working Paper, University of North Carolina at Chapel Hill, 2004.

[45] R. Wallace and W. Whitt. A Staffing Algorithm for Call Centers with Skill-Based Routing. *Manufacturing & Service Operations Management*, 7:276-294, 2005.

[46] W. Whitt. Partitioning Customers into Service Groups. *Management Science*, 45, 1999.

[47] W. Whitt. A Diffusion Approximation for the G/GI/n/m Queue. *Operations Research*, 52:922–941, 2004.

[48] W. Whitt. Staffing a Call Center with Uncertain Arrival Rate and Absenteeism. Forthcoming, *Production and Operations Management*.

[49] Wipro Technologies. Why Outsourcing from India? http://www.wipro.com/spectramind/why_bpo.htm. May 8, 2004.

[50] Wipro Technologies. Wipro Spectramind Fact Sheet. http://www.wipro.com/spectramind/factsheet.htm. May 8, 2004.

[51] R. W. Wolff. Poisson Arrivals See Time Averages. *Operations Research* 30:223–231, 1982.

[52] R. W. Wolff. *Stochastic Modeling and The Theory of Queues*. Englewood Cliffs, New Jersey: Prentice-Hall Prentice Hall, 1989.

[53] S. Zeltyn and A. Mandelbaum. Call Centers with Impatient Customers: Many-Server Asymptotics of the M/M/n+G Queue. Working Paper, Technion, 2004.

[54] E. Zohar, A. Mandelbaum, and N. Shimkin. Adaptive Behavior of Impatient Customers in Tele-Queues: Theory and Empirical Support. *Management Science*, 48:566–583, 2002.

Appendices for

Call-Routing Schemes for Call-Center Outsourcing

Appendix

There are six appendices. Appendix A gives the transition diagram of a small example of the inverted-V system described in Section 3.2. Appendix B includes analysis related to Section 4 and Appendix C analysis related to Section 5. Appendices D and E cover the analysis of the coefficient of variation and correlation coefficient of the inter-overflow time in the pooled overflow system. These results are used in Section 6. Finally, Appendix F generalizes the results of Appendices B and D for systems in which μ_H may be different from μ_L.

A Transition Diagram of a Small Inverted-V System in Section 3.2

Figure 8 shows the transition diagram for a small inverted-V system, with $m_L = 1$ and $m_O = 2$ CSRs, in which calls are preferentially routed to the client's in-house CSR.

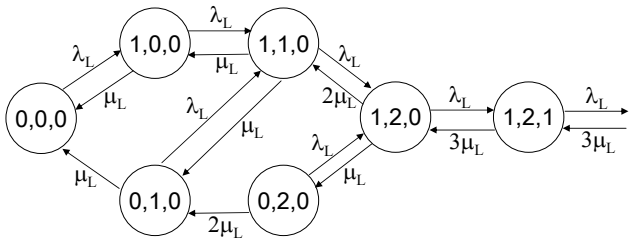

Figure 8: Transition Diagram for Inverted-V with $m_L = 1$ and $m_O = 2$ CSRs

B Analysis of Pooled-Overflow Systems in Section 4

B.1 Optimality of Randomized Threshold-Reservation Policies in Section 4.1

We defer to Appendix F.1 the proof that, among type-H priority policies, there exist stationary, type-H work-conserving policies that maximize the processing throughput of type-L calls in house. (Appendix F covers the general case in which μ_H may be different from μ_L, of which $\mu_H = \mu_L$ – studied here – is a special case.) Here, we make use of Appendix F.1's general result, and we consider only stationary, type-H work-conserving, type-H priority policies. We then prove Theorem 1's result: among these policies, there exist so-called "randomized threshold reservation" policies that are optimal.

Before we begin, we note that, in Section 4.1, we use Little's Law to state the service-level constraint in the pooled-overflow system in terms of the average number of calls in queue, rather than average delay in queue. In fact, the NLP formulation (3)–(6) can actually accommodate a more general class of occupancy-based constraint. In particular, in the service-level constraint (5), the delay-cost function associated with queue-length q, $d(q)$, need only satisfy the following assumptions:

i) $d(0) = 0$ and $d(\bar{q})$ is nondecreasing in $\bar{q}$; ii) $\sup_{\bar{q}} d(\bar{q}) > D^*$; and iii) $\tilde{d}(\alpha) \stackrel{\text{def}}{=} \sum_{\bar{q}=0}^{\infty} \alpha^{\bar{q}}\, d(\bar{q}) < \infty$ for all $\alpha \in (0,1)$. For more on these conditions, please see the statement of Assumption 1 in Appendix F.1.

We now proceed to prove Theorem 1. We begin by noting that, in principal, we could use substitution to eliminate ξ_{m_I} and constraint (4), which defines it, from the NLP formulation (3)–(6). We choose to keep it because the extra variable facilitates our analysis of the problem.

In particular, the inclusion of ξ_{m_I} in the service-level constraint (5) highlights the fact that, for fixed ρ, it is the value of ξ_{m_I} that uniquely determines the constraint's left-hand-side. The following lemma shows that there may be many sets of p_ss from which a given ξ_{m_I} may be obtained:

Lemma 1

Suppose there is a $p \in [0,1]^{m_I}$ for which $0 < p_i, p_{i+1} < 1$ and $0 \le i \le m_I - 2$. Then there also exists a $p' \in [0,1]^{m_I} \neq p$ for which $p'_j = p_j$ for all $j \notin \{i, i+1\}$ and $\xi_{m_I}(p') = \xi_{m_I}(p)$. Furthermore, either $p'_i > p_i$ and $p'_{i+1} < p_{i+1}$ or $p'_i < p_i$ and $p'_{i+1} > p_{i+1}$.

Proof

Let $h_i = \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i}$ and $l_{j,k} = \prod_{i=j}^{k} h_i$. Whenever the range is empty, we define the product to be 1 and the summation to be 0. Then (4) can be written as

$$\xi_{m_I} \;=\; \left[\sum_{s=0}^{m_I-1} l_{s,m_I-1} \;+ \frac{1}{1-\rho} \right]^{-1} ,$$

Note that if $X \stackrel{\text{def}}{=} \sum_{s=0}^{m_I-1} l_{s,m_I-1}$ remains constant, so will ξ_{m_I}. Substituting, in turn, for h_i, we have

$$X \;=\; \sum_{s=0}^{i} l_{s,i-1} h_i h_{i+1} l_{i+2,m_I-1} \;+\; h_{i+1} l_{i+2,m_I-1} \;+\; \sum_{s=i+2}^{m_I-1} l_{s,m_I-1} \,.$$

Solving for h_{i+1} yields

$$h_{i+1} \;= \frac{X \;-\; \sum_{s=i+2}^{m_I-1} l_{s,m_I-1}}{h_i \sum_{s=0}^{i} l_{s,i-1} l_{i+2,m_I-1} \;+\; l_{i+2,m_I-1}} \,.$$

For a given X, we can implicitly differentiate h_{i+1} with respect to h_i to yield

$$\frac{\partial\, h_{i+1}}{\partial\, h_i} \;= -\, \frac{\left(X \;-\; \sum_{s=i+2}^{m_I-1} l_{s,m_I-1}\right) \left(\sum_{s=0}^{i} l_{s,i-1} l_{i+2,m_I-1}\right)}{\left(h_i \sum_{s=0}^{i} l_{s,i-1} l_{i+2,m_I-1} \;+\; l_{i+2,m_I-1}\right)^2} \le 0 \,.$$

Thus, as we increase (decrease) h_i by an infinitesimal amount, it is always possible for us to simultaneously decrease (increases) h_{i+1} to maintain a constant X.

Since both $\partial h_i / \partial p_i < 0$ and $\partial h_{i+1} / \partial p_{i+1} < 0$ we conclude that, equivalently, for a given $0 < p_i, p_{i+1} < 1$ and X, a decrease (increase) in p_i can be compensated for with a simultaneous increase (decrease) in p_{i+1}. □

Therefore, given a set of routing probabilities p, with elements p_i and p_{i+1} that are interior, we can construct an alternative set p', with different p'_i and p'_{i+1}, that obtains the same ξ_{m_I}. Furthermore, p'_i and p'_{i+1} move away from p_i and p_{i+1} in opposite directions.

Now suppose two solutions, p and p', yield the same ξ_{m_I}. Which has the higher objective function value? To answer the question, we again consider two solutions, p and p', as defined in Lemma 1.

Lemma 2

Suppose there are p and p' defined as in Lemma 1. If $p'_i > p_i$, or equivalently $p'_{i+1} < p_{i+1}$, then system idleness (3) *under p' is strictly less then that under p .*

Proof

Again, let $h_i = \frac{(i+1)\mu}{\lambda_H + \lambda_L p_i}$ and $l_{j,k} = \prod_{i=j}^{k} h_i$. Whenever the range is empty, we define the product to be 1 and the summation to be 0.

In the proof of Lemma 1, we showed that a decrease in h_i could be compensated by a simultaneous increase in h_{i+1} to maintain a constant $X = \sum_{s=0}^{m_I-1} l_{s,m_I-1}$.

Now consider such a change and its effect on each of the terms l_{s,m_I-1} within the summation. In particular, note

i) $\sum_{s=i+2}^{m_I-1} l_{s,m_I-1} = \sum_{s=i+2}^{m_I-1} \prod_{j=s}^{m_I-1} h_j$ does not change.

ii) From (i), we see that $\sum_{s=0}^{i+1} l_{s,m_I-1}$ remains constant, since $X - \sum_{s=i+2}^{m_I-1} l_{s,m_I-1}$ does not change.

iii) The single term, $l_{i+1,m_I-1} = \prod_{j=i+1}^{m_I-1} h_j$, increases.

iv) From (ii) and (iii) we see that

- l_{i+1,m_I-1} increases and $\sum_{s=0}^{i+1} l_{s,m_I-1}$ remains constant $\Longrightarrow \sum_{s=0}^{i} l_{s,m_I-1}$ decreases;
- $l_{0,m_I-1}, \ldots, l_{i,m_I-1}$ all move in the same direction (as $h_i h_{i+1}$) $\Longrightarrow l_{0,m_I-1}, \ldots, l_{i,m_I-1}$ all decrease.

Similarly, consider the change in the objective function. Recall that ξ_{m_I} remains constant. Using the definition of $l_{j,k}$ and grouping terms, as above, we see that (3) can be rewritten as

$$\xi_{m_I} \left[\sum_{s=0}^{i} (m_I - s)\, l_{s,m_I-1} \;+\; (m_I - (i+1))\, l_{i+1,m_I-1} \;+\; \sum_{s=i+2}^{m_I-1} (m_I - s)\, l_{s,m_I-1} \right].$$

Furthermore, noting that $m_I - s > m_I - (i+1)$ for $s < i+1$ we can rearrange terms to restate (3) as

$$\xi_{m_I} \left[\sum_{s=0}^{i} ((i+1) - s)\, l_{s,m_I-1} \;+\; \sum_{s=0}^{i+1} (m_I - (i+1))\, l_{s,m_I-1} \;+\; \sum_{s=i+2}^{m_I-1} (m_I - s)\, l_{s,m_I-1} \right].$$

Because the first term strictly decreases, and the second and third terms do not change, an appropriate, simultaneous increase in p_i and decrease in p_{i+1} strictly decreases system idleness. The same argument can be used to show that an opposite change strictly increases (3). □

Thus, by appropriately increasing the probability of putting a type-L call into service in state i and decreasing the probability of putting a type-L call into service in state $i+1$, the in-house pool can maintain the same type-H service level *and* strictly improve its type-L throughput.

Together, Lemmas 1 and 2 show that, whenever there exists a feasible p with $p_i < 1$ and $p_{i+1} > 0$, we can improve the NLP's objective function by simultaneously increasing p_i and decreasing p_{i+1} until one of them hits a boundary (either $p_i = 1$ or $p_{i+1} = 0$). In turn, this allows us to demonstrate that a randomized threshold reservation policy is optimal:

Proof of Theorem 1

Note that a stationary, type-H priority, type-H work-conserving policy is an (L, p_L) policy if and only if $p_i = 0 \implies p_j = 0$ for all $j > i$ and $p_i = 1 \implies p_j = 1$ for all $j < i$.

Then we consider each of two contradictory cases. First, suppose that there exists an optimal policy for which $p_i = 0$ and $p_j > 0$ for some $j > i$. Then there exists a k such that $p_k = 0$ and $p_{k+1} > 0$, and we can simultaneously maintain the feasibility of the service-level constraint (5) and decrease the objective function value (3) by increasing p_k and decreasing p_{k+1} as in Lemmas 1 and 2. Thus, the original policy could not have been optimal. Similarly, suppose there exists an optimal policy for which $p_i = 1$ and $p_j < 1$ for some $j < i$. Then there exists $p_{k-1} < 1$ and $p_k = 1$, and the argument used in the previous case leads to a contradiction here as well. □

Proof of Proposition 1

As in Lemma 2, define $X = \sum_{s=0}^{m_I-1} \prod_{i=s}^{m_I-1} \frac{(i+1)\mu}{\lambda_H+\lambda_L p_i}$. Then differentiation shows that $\frac{\partial X}{\partial p_i} < 0$ for all p_i, so $\xi_{m_I} = [X + 1/(1-\rho)]^{-1}$ is increasing in each p_i.

Now consider two sets of routing probabilities, $p, p' \in [0,1]^{m_I}$, with $p'_k > p_k$ and $p'_i = p_i$ for all $i \neq k$, and define $\Delta_i = \xi_i(p') - \xi_i(p)$. Then we have the following.

i) As demonstrated above, $\xi_{m_I}(p') > \xi_{m_I}(p)$, or $\Delta_{m_I} > 0$. The fact that $\xi_i = \xi_{m_I}\rho^{i-m_I}$ for all $i > m_I$ implies that $\xi_i(p') > \xi_i(p)$, or $\Delta_i > 0$ for all $i \geq m_I$.

ii) Together (i) and the fact that $\sum_{i=0}^{\infty} \xi(p') = \sum_{i=0}^{\infty} \xi(p) = 1$ imply that $\sum_{i=0}^{m_I-1} \xi(p') < \sum_{i=0}^{m_I-1} \xi(p)$, or $\sum_{i=0}^{m_I-1} \Delta_i < 0$.

iii) Together (i) and the fact that $\xi_i = \xi_{m_I} \prod_{j=i}^{m_I-1} \frac{(j+1)\mu}{\lambda_H+\lambda_L p_j}$ imply that, for all $i \in \{k+1, \ldots, m_I-1\}$, we have $\xi_i(p') > \xi_i(p)$, or $\Delta_i > 0$.

iv) Together (ii) and (iii) imply that $\sum_{i=0}^{k} \xi(p') < \sum_{i=0}^{k} \xi(p)$, or $\sum_{i=0}^{k} \Delta_i < 0$.

v) Together (iv) and the fact that $\xi_i = \xi_k \prod_{j=i}^{k} \frac{(j+1)\mu}{\lambda_H+\lambda_L p_j}$, or equivalently $\frac{\xi_i(p)}{\xi_k(p)} = \frac{\xi_i(p')}{\xi_k(p')}$, imply that, for all $i \leq p$, we have $\xi_i(p') < \xi_i(p)$, or $\Delta_i < 0$.

Finally, recall that the NLP's objective (3) is to minimize system idleness, $\sum_{i=0}^{m_I-1}(m_I - i)\xi_i$.

Then we have $\xi_i(p') = \xi_i(p) + \Delta_i$, and

$$\begin{aligned}\sum_{i=0}^{m_I-1}(m_I-i)\xi_i(p') &= \sum_{i=0}^{m_I-1}(m_I-i)\xi_i(p) + \sum_{i=0}^{k}(m_I-i)\Delta_i + \sum_{i=k+1}^{m_I-1}(m_I-i)\Delta_i \\ &\leq \sum_{i=0}^{m_I-1}(m_I-i)\xi_i(p) + \sum_{i=0}^{k}(m_I-(k+1))\Delta_i + \sum_{i=k+1}^{m_I-1}(m_I-(k+1))\Delta_i\end{aligned}$$

since $\Delta_i < 0$ for $i \leq k$ and $\Delta_i > 0$ for $i > k$. But

$$\sum_{i=0}^{k}(m_I-(k+1))\Delta_i + \sum_{i=k+1}^{m_I-1}(m_I-(k+1))\Delta_i = \sum_{i=0}^{m_I-1}(m_I-(k+1))\Delta_i < 0$$

since $\sum_{i=0}^{m_I-1}\Delta_i < 0$. Therefore, $\sum_{i=0}^{m_I-1}(m_I-i)\xi_i(p') < \sum_{i=0}^{m_I-1}(m_I-i)\xi_i(p)$, so the objective function (3) is decreasing in p_k. □

B.2 Efficiency of Type-H Priority Policies

In this section we prove Proposition 2, which states that the expected throughput of the best type-H priority policy is within $\frac{\lambda_L}{\lambda_L+\lambda_H+m_I\mu}$ of an upper bound on the globally optimal throughput for the pooled-overflow system.

Proof of Proposition 2

Part (i).

We use a sample-path argument to prove that the optimal throughput when $\lambda_L = \infty$ is an upper bound on the globally optimal throughput when $\lambda_L < \infty$. Let system 1 have "$\lambda_L = \infty$" – so that there is always a type-L call waiting to be served – and let system 2 have $\lambda_L < \infty$. Then a simple sample-path argument shows that any policy used in system 2 – including the optimal policy – can be matched exactly in system 1, so that the type-L throughput and type-H delay performance of the two systems are identical. Since this policy is feasible, though not necessarily optimal, for system 1, optimal type-L throughput in system 1 must be at least as large as that in system 2.

The sample-paths of the two systems are constructed to have the same type-H inter-arrival times and the same service times, once calls are put into service. The only difference between the two systems is that, rather than having explicit type-L inter-arrival times, system 1 always has a type-L call waiting to be served.

The sample-path argument, itself, is then trivial. At every instant that a call of either type arrives to system 2, there also exists a call of the same type in system 1 that is waiting to be served, and every call that is put into service in system 2 can be feasibly put into service in system 1. Thus system 1 can feasibly match the performance of system 2.

Part (ii)

Next, we use a coupling argument to show that, for a fixed $\lambda_L < \infty$, there exists a type-H priority

policy whose type-L throughput is at least as great as $\frac{\lambda_L}{\lambda_L+\lambda_H+m_I\mu}$ of the globally optimal throughput when $\lambda_L = \infty$.

Let system 1 have "$\lambda_L = \infty$" – so that there always exists a type-L call waiting to be served – and let system 2 have $\lambda_L < \infty$. We couple the two systems using a single set of i.i.d., exponentially distributed inter-event times of rate $\lambda_H + \lambda_L + m_I\mu$. Without loss of generality, we assume that the time scale is set so that $\lambda_H + \lambda_L + m_I\mu = 1$.

The event generator is driven off of system 1 and works as follows. Given inter-event times that are exponentially distributed with mean $1/(\lambda_H+\lambda_L+m_I\mu) = 1$, the conditional probability that the next event is a service completion by CSR i equals μ. If CSR i is busy, then the service is real, and if it is idle, then the service is a "dummy" completion. Similarly, the conditional probability the event is a type-H arrival equals λ_H, and the probability that it is a "dummy" type-L arrival equals λ_L. Note that, since system 1 always has a type-L call waiting to be served, the arrival of this additional call does not change the system state.

System 2 is coupled to system 1 in a straightforward fashion. Type-H arrivals are the same as in system 1. Type-L arrivals in this system are at the same time as in system 1; in this case they are real, however. Service completions occur at the same epochs as in system 1. Furthermore, the CSRs in the two systems are matched and labelled $i = 1, \ldots, m_I$.

Two related points concerning this coupling mechanism are important to observe. First, we need only consider policies which put calls into service at event epochs, and in this proof we only consider policies within this broad class. In addition, for system 1 we only consider policies which put type-L calls into service at event epochs that correspond to type-H arrivals or at service completions, since these are the event epochs associated with system 1. That is, in system 1 there always exists a type-L call waiting to be served, type-L "arrivals" are dummy, and there exist optimal policies which ignore these dummy type-L arrival epochs.

Then given an arbitrary call-routing policy (within the class described above) that is used in system 1, we construct a policy for system 2 as follows. Whenever system 1 puts a type-H job into service with CSR i, system 2 does so as well. Whenever system 1 puts a type-L job into service with CSR i, system 2 waits for the next event: if it is a type-L arrival, then system 2 puts the type-L call into service with CSR i as well; if it is a service completion by CSR i, then system 2 does nothing; in all the other cases, system 2 puts a "dummy" type-L job into service with CSR i so that, for accounting purposes, the occupancies of the two systems remains the same.

One last important element of the coupling mechanism is worth noting here. When system 2 puts a type-L call into service – either dummy or real – the processing time of the call equals the residual service time of the type-L call that had previously been put into service in system 1. Because type-L service times are exponentially distributed with rate μ, however, the conditional distribution of the residual service-time in system 1 is exponential with rate μ as well. Therefore, by assigning system 1's residual service times to type-L calls in system 2, we generate a sequence of service times for

type-L calls in system 2 that are exponentially distributed with rate μ as well.

Thus, the only times at which the evolutions of the two systems differ are those moments in which system 1 has put a type-L call into service and system 2 is still waiting for the next event to occur. At these times, the number of busy servers in system 2 is one less than in system 1; and after the next event, the two "accounting" occupancies are, again, the same. Most importantly, there are always at least as many idle servers in system 2 as in system 1.

We claim that system 2 feasibly handles a fraction $\frac{\lambda_L}{\lambda_H+\lambda_L+m_I\mu}$ of the type-L throughput of system 1. For feasibility, note that system 2 can and does take type-H calls at exactly the same epochs as system 1. Therefore the length of the type-H queue is always equal in the two systems, and system 2 is feasible whenever system 1 is. For throughput, note that every time system 1 takes a type-L call, there is a probability of $\frac{\lambda_L}{\lambda_H+\lambda_L+m_I\mu}$ that the next event will be a type-L arrival and that system 2 will take a real, rather than dummy, type-L job as well. So the type-L throughput of system 2 is $\frac{\lambda_L}{\lambda_H+\lambda_L+m_I\mu}$ times that of system 1.

We also note that if system 1 uses a type-H priority policy, then the policy induced in system 2 will be of type-H priority as well. Furthermore, from [21] we know that there exist type-H priority policies that are optimal in system 1. Thus, there exists a type-H priority policy in system 2 that achieves $\frac{\lambda_L}{\lambda_H+\lambda_L+m_I\mu}$ of the type-L throughput obtained using the optimal policy in system 1. □

Proposition 2 states that, for λ_L that is "large" with respect to λ_H, μ, and m_I, the performance of type-H priority policies should be excellent. A natural next question is "how large is 'large'?" The results of numerical tests, shown below, indicate that the numbers are quite reasonable.

We set the time scale so that average service times equal $\mu^{-1} = 3.33$ minutes, and we impose a service-level constraint that the ASA of type-H calls must be 0.5 minutes or less. We then systematically vary λ_H and λ_L. The three panels of Figure 9 display the results for three sizes of in-house pools: a small system, driven by $\lambda_H = 6$, or equivalently $R_H = \lambda_H/\mu = 20$; a medium system, with $\lambda_H = 30$ and $R_H = 100$; and a larger system, with $\lambda_H = 150$ and $R_H = 500$.

In each panel the horizontal axis shows the number of CSRs used in the pool, as it climbs above the minimum needed to meet the type-H service-level constraint. The vertical axis marks the load of type-L calls that is processed in house. (We plot load, in CSRs, rather than throughput rates, so that the scales of the two axes are comparable.) Each curve within a panel shows, for a given $R_L = \lambda_L/\mu$, the load of type-L calls processed in-house by the optimal type-H priority policy.

Figure 9 shows that, in fact, type-L throughput is nearly optimal for relatively large R_L. In each panel, the curve for $R_L = \infty$ represents an upper bound on optimal performance, and the curves for the finite R_L's systematically approach the upper bound. For small in-house systems, with $R_H = 20$, the performance when $R_L = 100$ is nearly optimal. This represents a rate of low-value calls that is five times the rate of the high-value calls. Given the "80–20" maxim, that 20% of the customers provide 80% of the value to a company, this appears to be a quite reasonable balance. Furthermore, as the

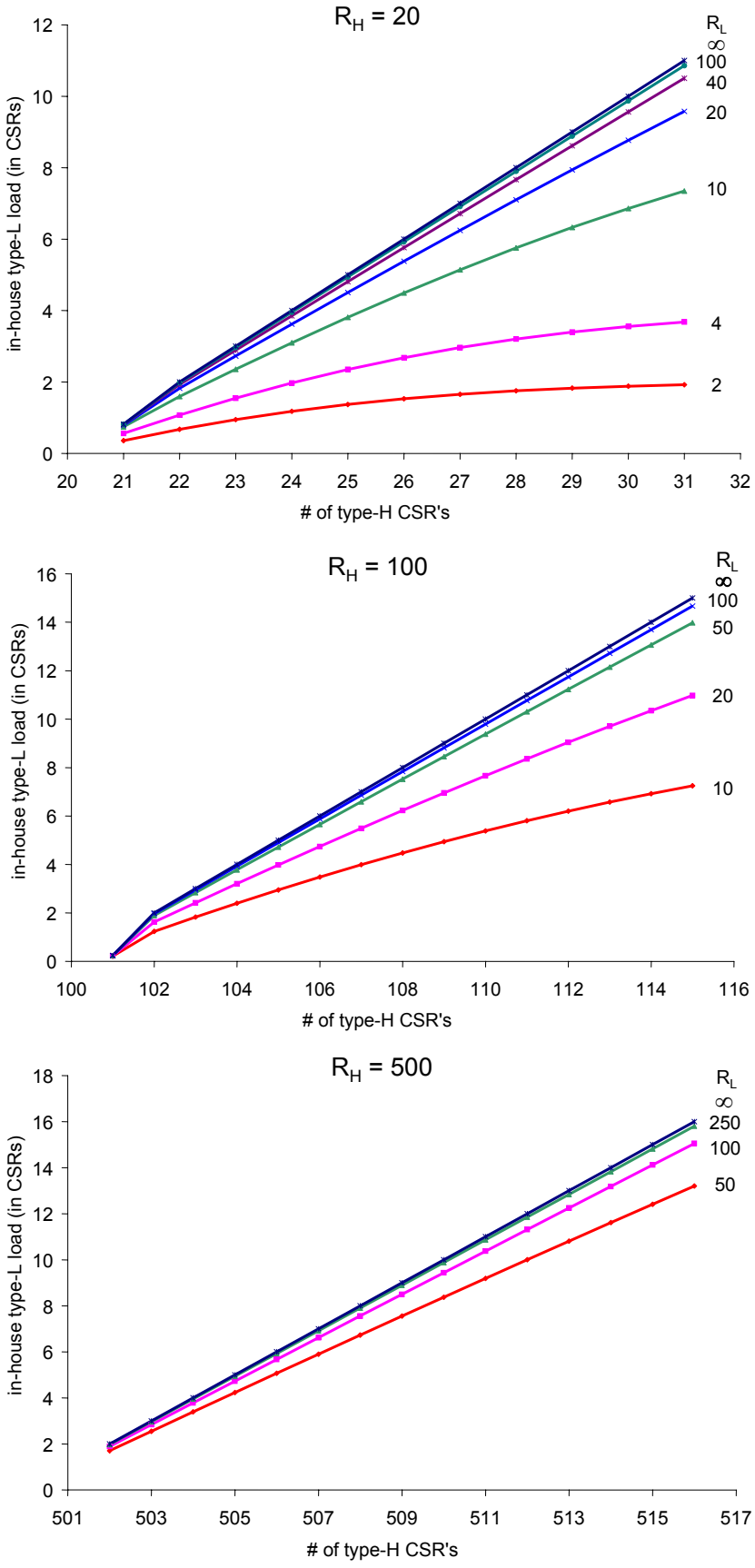

Figure 9: In-House Type-L Load for Type-H Priority Policies

scale of type-H traffic grows, the relative level of type-L traffic required to obtain nearly-optimal performance systematically declines. With $R_H = 500$, low-value calls need only have $R_L = 250$ – *half* the rate of high-value traffic – in order 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.

C Performance Bounds for the N-Network System in Section 5

In this appendix we prove Proposition 3's performance bounds for the N-network system.

Proof of Proposition 3

Part (i)

Step 1. Given an N-network with m_I in-house and m_O outsourcer CSRs and routing policy π, consider a pooled system with $m = m_I + m_O$ CSRs, all capable of handling both types of calls. Label the first m_I of the pool "in-house" and the last m_O of them "outsourcer" and use routing policy π. The performance of the two systems will be exactly the same.

Step 2. Consider the pooled system in which routing policy π is used. Let ASA_H^π and ASA_L^π be the ASA achieved by policy π in the N-network for type-H and type-L calls respectively. Then because $ASA_H^\pi \leq ASA^*$ and $ASA_L^\pi \leq ASA^*$,

$$ASA^\pi \;=\; \frac{\lambda_H}{\lambda_H+\lambda_L}\,ASA_H^\pi \;+\; \frac{\lambda_L}{\lambda_H+\lambda_L}\,ASA_L^\pi \;\leq\; ASA^*.$$

From Little's law, we know that there exists a long-run average number in queue, L^π, such that $L^\pi = (\lambda_H + \lambda_L)ASA^\pi$.

Step 3. We use a coupling argument to show we can construct a not necessarily work-conserving first-come-first-served service discipline, π', that maintains the same number-in-queue process as there is under π. More specifically, let $\{T_1, T_2, \ldots\}$ be an i.i.d. sequence of exponentially-distributed inter-arrival times of mean $(\lambda_H + \lambda_L)^{-1}$. Let $\{p_1, p_2, \ldots\}$ be an i.i.d. sequence of Bernoulli random variables that equal one – corresponding to a type-H, rather than type-L, arrival – with probability $\frac{\lambda_H}{\lambda_H+\lambda_L}$. Finally, let $\{S_1, S_2, \ldots\}$ be an i.i.d. sequence of exponentially-distributed service times of mean μ^{-1}. These service-time realizations are sampled in the order in which calls are put into service, no matter whether the call is of type-H or type-L.

Then every time π puts a call into service, π' does as well. Since the inter-arrival-time, call-type, and service-time samples are the same in the two systems, the number-in-system process is identical in the two systems. In turn, $L^\pi = L^{\pi'}$.

Step 4. Define FCFS to be a work-conserving first-come-first-served service discipline. Then a FCFS policy will have number-in-system process that is no more than that of π'. Therefore,

$L^{FCFS} \leq L^{\pi}$, and by Little's Law

$$ASA^{FCFS} \;=\; \frac{L^{FCFS}}{\lambda_H + \lambda_L} \;\leq\; \frac{L^{\pi}}{\lambda_H + \lambda_L} \;=\; ASA^{\pi} \;\leq\; ASA^{*}.$$

Finally, by PASTA, we know that expected delay in queue of every type-H and every type-L call is

$$ASA_H^{FCFS} \;=\; ASA_L^{FCFS} \;=\; ASA^{FCFS} \;\leq\; ASA^{*}.$$

Thus, if policy π is feasible for a system with m_I in house and m_O outsourcer CSRs, then a FCFS policy with $m = m_I + m_O$ pooled CSRs is feasible as well. This proves part (i).

Part (ii)

The result follows from the fact that the globally optimal type-L throughput when $\lambda_L = \infty$ is also an upper bound on the throughput achievable in the N-network scheme. The proof is analogous to that of Proposition 2. □

D Inter-Overflow Time CV in the Pooled-Overflow System

The analysis in this appendix characterizes the moments of the inter-overflow times of type-L calls from the client company to the outsourcer in the pooled-overflow scheme. This information is used to numerically evaluate the coefficient of variation (CV) of the inter-overflow time in the examples in Section 6.4.

For notational convenience, we drop the subscript, I, from the number of in-house CSRs, m_I. Thus, the total number of in-house CSRs is referred to as "m" below.

When the client company uses the (L, p_L) policy to for type-L call admission, the overflow of type-L calls to the outsourcer follows a Markov Modulated Poisson Process (MMPP): it is Poisson with rate λ_L when $s > L$, Poisson with rate $(1 - p_L)\lambda_L$ when $s = L$, and rate 0 when $s < L$. For such a policy, $p_0 = \cdots = p_{L-1} = 1$ and $p_{L+1} = \cdots = p_{m-1} = 0$.

We now characterize the inter-overflow time. Let $\tilde{T}$ denote the random inter-overflow time, and let $\tilde{T}_s$, $\forall\ s$, denote the random time to next overflow starting in after-action state s. Because the overall arrival process of type-L calls is Poisson, we can apply PASTA to obtain

$$\mathbf{P}\left\{\tilde{T} \leq t\right\} \;=\; \sum_s \left(\frac{\xi_s(1-p_s)}{\sum_s \xi_s(1-p_s)}\right) \mathbf{P}\left\{\tilde{T}_s \leq t\right\}, \tag{7}$$

where ξ_s can be computed from (2) with $p_0 = \cdots = p_{L-1} = 1$ and $p_{L+1} = \cdots = p_{m-1} = 0$.

Because $\mathbf{P}\left\{\tilde{T} \leq t\right\}$ is a convex combination of $\mathbf{P}\left\{\tilde{T}_s \leq t\right\}$ for all $s \geq L$, the k^{th} moment of $\tilde{T}$ will be the same convex combination of the k^{th} moment of $\tilde{T}_s$. Therefore, it suffices to find all the moments of $\tilde{T}_s$ for any $s \geq L$.

Let $\mu_s = \begin{cases} s\mu, & \forall s \le m \\ m\mu, & \forall s > m \end{cases}$ be the total service rate in state s and $\lambda_s = \lambda_H + \lambda_L$ be the total arrival rate in state s. Then $\omega_s = \lambda_s + \mu_s$ is the total rate at which next event takes place in state s. In particular, $\omega_m = \lambda_H + \lambda_L + m\mu$. Denoting by $\tilde{e}_\omega$ an exponential random variable with rate ω, we obtain the following relationship among all $\tilde{T}_s$s:

$$\tilde{T}_s \;=\; \tilde{e}_{\omega_s} + \begin{cases} \tilde{T}_{s-1} & \text{w.p.} \quad \mu_s/\omega_s \\ 0 & \text{w.p.} \quad (1-p_s)\lambda_L/\omega_s \\ \tilde{T}_{s+1} & \text{w.p.} \quad (\lambda_H + p_s\lambda_L)/\omega_s \end{cases} \qquad \forall\ s \ge 1. \tag{8}$$

Let $T_s(\theta)$ denote the Laplace transform of $\tilde{T}_s$. Then (8) implies

$$T_s(\theta) \;=\; \frac{\mu_s}{\omega_s+\theta} T_{s-1}(\theta) + \frac{(1-p_s)\lambda_L}{\omega_s+\theta} + \frac{\lambda_H + p_s\lambda_L}{\omega_s+\theta} T_{s+1}(\theta), \qquad \forall s \ge 1. \tag{9}$$

For all $s \ge m$, $p_s = 0$, $\mu_s = m\mu$, and $\omega_s = \omega_m$. Therefore when $s \ge m$, from (9) we obtain

$$T_{m+q}(\theta) = \frac{m\mu}{\omega_m+\theta} T_{m+q-1}(\theta) + \frac{\lambda_L}{\omega_m+\theta} + \frac{\lambda_H}{\omega_m+\theta} T_{m+q+1}(\theta), \qquad \forall q \ge 1, \tag{10}$$

where $q = s - m$ and q is the type-H queue length.

The solution to (10) has a geometric form, determined completely by the boundary point, $T_m(\theta)$:

Proposition 4
The solution to (10) *is:*

$$T_{m+q}(\theta) = T_m(\theta) z_1^q(\theta) + \frac{\lambda_L}{\lambda_L+\theta}\left[1 - z_1^q(\theta)\right], \tag{11}$$

where $z_1(\theta) = \frac{(\omega_m+\theta) - \sqrt{(\omega_m+\theta)^2 - 4m\mu\lambda_H}}{2\lambda_H}$.

Proof
The proof will follow these three steps:

1. Show that the solution to (10) is of the form of $T_{m+q}(\theta) = J_1(\theta) z_1(\theta)^q + J_2(\theta) z_2(\theta)^q + \frac{\lambda_L}{\lambda_L+\theta}$, for some easily calculated $J_1(\theta), J_2(\theta), z_1(\theta), z_2(\theta)$, where $0 \le z_1(\theta) < 1 < z_2(\theta)$.
2. Show that $\lim_{q\to\infty} T_{m+q}(\theta) = \frac{\lambda_L}{\lambda_L+\theta}$.
3. From the first two steps, we must have $J_2(\theta) = 0$ and $J_1(\theta) = T_m(\theta) - \frac{\lambda_L}{\lambda_L+\theta}$.

Step 1 To simplify notation, we will suppress the θ in all of the Laplace transforms, so that $T_{m+q}(\theta)$ will become T_{m+q}, and so on. We define $\gamma_1 = \frac{m\mu}{\omega_m+\theta}, \gamma_2 = \frac{\lambda_H}{\omega_m+\theta}, \gamma_3 = \frac{\lambda_L}{\omega_m+\theta}$. Then (10) becomes:

$$T_{m+q} = \gamma_1 T_{m+q-1} + \gamma_3 + \gamma_2 T_{m+q+1} \qquad \forall q \ge 1. \tag{12}$$

Multiplying both sides by z^{q+1} and summing from 1 to ∞, we obtain

$$\sum_{q=1}^{\infty} T_{m+q} z^{q+1} = \gamma_1 \sum_{q=1}^{\infty} T_{m+q-1} z^{q+1} + \gamma_3 \sum_{q=1}^{\infty} z^{q+1} + \gamma_2 \sum_{q=1}^{\infty} T_{m+q+1} z^{q+1}. \tag{13}$$

Denoting $f(z) = \sum_{q=0}^{\infty} T_{m+q} z^q$, we have

$$\begin{aligned} z[f(z) - T_m] &= \gamma_1 z^2 f(z) + \gamma_3 z^2/(1-z) + \gamma_2[f(z) - T_m - T_{m+1} z] && (14) \\ \Longrightarrow f(z) &= \frac{\gamma_2 T_{m+1} z - T_m z - \gamma_3 z^2/(1-z) + \gamma_2 T_m}{\gamma_1 z^2 - z + \gamma_2} && (15) \\ &= \frac{(1-z)(\gamma_2 T_{m+1} z - T_m z + \gamma_2 T_m) - \gamma_3 z^2}{\gamma_2 (1 - z_1 z)(1 - z_2 z)(1-z)}, && (16) \end{aligned}$$

where $z_1 \le z_2$ are the two roots of $\gamma_2 z^2 - z + \gamma_1 = 0$ (because $\gamma_2 \neq 0$):

$$z_1 = \frac{1 - \sqrt{1 - 4\gamma_1\gamma_2}}{2\gamma_2} \qquad \text{and} \qquad z_2 = \frac{1 + \sqrt{1 - 4\gamma_1\gamma_2}}{2\gamma_2}. \tag{17}$$

Since $z_1 + z_2 > 0$ and $z_1 z_2 = \gamma_1/\gamma_2 > 1$, we have $0 < z_1 < 1 < z_2$. Carrying out partial-fraction expansion, we obtain

$$f(z) = \left[\frac{J_1}{1 - z_1 z} + \frac{J_2}{1 - z_2 z} + \frac{J_3}{1 - z} \right], \tag{18}$$

where

$$\begin{aligned} J_1 &= [1 - z_1 z] f(z) \mid_{z = 1/z_1} = \frac{\gamma_2 T_{m+1} + (\gamma_2 z_1 - 1) T_m + \gamma_3/(1 - z_1)}{\gamma_2 (z_1 - z_2)}, \\ J_2 &= [1 - z_2 z] f(z) \mid_{z = 1/z_2} = \frac{\gamma_2 T_{m+1} + (\gamma_2 z_2 - 1) T_m + \gamma_3/(1 - z_2)}{\gamma_2 (z_2 - z_1)}, \\ \text{and } J_3 &= (1 - z) f(z) \mid_{z=1} = \frac{-\gamma_3}{\gamma_1 - 1 + \gamma_2} = \frac{\lambda_L}{\lambda_L + \theta}. \end{aligned} \tag{19}$$

Then from (18) we have

$$T_{m+q} = J_1 z_1^q + J_2 z_2^q + \frac{\lambda_L}{\lambda_L + \theta}. \tag{20}$$

Because $\gamma_2(z_2 - z_1) \neq 0$, J_1 and J_2 always exist. Thus (18) and (20) are always valid. And it is clear that the series T_{m+q}, as defined in (20), satisfies (12).

Step 2 When an overflow occurs in state s, the probability that the next type-L arrival will also be overflowed approaches 1 as $s \to \infty$. Therefore, the time until the next overflow approaches an exponential random variable with rate λ_L as $s \to \infty$. We now formalize this.

Suppose we start in state $m + q$. Denote by $\tilde{R}_q$ the random amount of time it takes to reach the first state, L, in which it is possible to accept a new type-L call. Also, we denote by $\tilde{E}$ the random amount of time it takes for the next type-L call to arrive. Clearly $\tilde{E}$ is exponentially distributed with rate λ_L.

It is easy to show (by induction, for example) that $\tilde{R}_q$ is stochastically larger than the sum of $q + 1$ $\exp(\omega_m)$ random variables, one $\exp(\omega_{m-1})$ random variable, one $\exp(\omega_{m-2})$ random variable,

..., and one $\exp(\omega_{L+1})$ random variable. This sum, in turn, is stochastically larger than the sum of $q+m-L$ $\exp(\omega_m)$ random variables.

So if we denote a $\Gamma(q+m-L,\omega_m)$ random variable by $\tilde{X}_q$ and its *p.d.f.* by $g_q(t) = \frac{\omega_m e^{-\omega_m t}(\omega_m t)^{q+m-L-1}}{(q+m-L-1)!}$, we have

$$\begin{aligned}\mathbf{P}\left\{\tilde{R}_q \le \tilde{E}\right\} &= \int_0^\infty \mathbf{P}\left\{\tilde{R}_q < t\right\} \lambda_L e^{-\lambda_L t} dt \le \int_0^\infty \mathbf{P}\left\{X < t\right\} \lambda_L e^{-\lambda_L t} dt \\ &= \mathbf{P}\left\{X \le \tilde{E}\right\} = \int_0^\infty \mathbf{P}\left\{\tilde{E} \ge t\right\} g_q(t) dt = \int_0^\infty e^{-\lambda_L t} g_q(t) dt.\end{aligned}$$

Now $\forall\ \epsilon > 0, \exists\ \bar{t}$, s.t. $e^{-\lambda_L \bar{t}} < \epsilon$. Moreover, $g_q(t) = \frac{\omega_m e^{-\omega_m t}(\omega_m t)^{q+m-L-1}}{(q+m-L-1)!} \to 0$ uniformly on $t \in [0,\bar{t}]$ as $q \to \infty$. Hence $\exists Q_\epsilon$ s.t. $g_q(t) \le \epsilon$ for all $q > Q_\epsilon$ and $t \in [0,\bar{t}]$, and

$$\begin{aligned}\int_0^\infty e^{-\lambda_L t} g_q(t) dt &= \int_{\bar{t}}^\infty e^{-\lambda_L t} g_q(t) dt + \int_0^{\bar{t}} e^{-\lambda_L t} g_q(t) dt \\ &\le \epsilon \int_{\bar{t}}^\infty g_q(t) dt + \epsilon \int_0^{\bar{t}} e^{-\lambda_L t} dt \le \epsilon \left(1 + \frac{1}{\lambda_L}\right), \qquad \forall\ q > Q_\epsilon.\end{aligned}$$

Therefore, $\lim_{q\to\infty} \mathbf{P}\left\{\tilde{R}_q \le \tilde{E}\right\} = 0$.

If we denote the *c.d.f.* of $\tilde{T}_{m+q}$ by $F_q(t)$, then

$$F_q(t) = \mathbf{P}\left\{\tilde{T}_{m+q} \le t\right\} = \mathbf{P}\left\{\tilde{R}_q + \tilde{T}_L \le t\right\} \mathbf{P}\left\{\tilde{R}_q \le \tilde{E}\right\} + (1 - e^{-\lambda_L t}) \mathbf{P}\left\{\tilde{R}_q > \tilde{E}\right\}.$$

Therefore,

$$\begin{aligned}\mid F_q(t) - (1 - e^{-\lambda_L t}) \mid &\le \mathbf{P}\left\{\tilde{R}_q \le \tilde{E}\right\} \left[(1 - e^{-\lambda_L t}) + \left|\mathbf{P}\left\{\tilde{R}_q + \tilde{T}_{\bar{x}_L} \le t\right\}\right|\right] \\ &\le 2\mathbf{P}\left\{\tilde{R}_q \le \tilde{E}\right\}. \qquad (21)\end{aligned}$$

Because (21) holds for all t and $\lim_{q\to\infty} \mathbf{P}\left\{\tilde{R}_q \le \tilde{E}\right\} = 0$, we have $\lim_{q\to\infty} F_q(t) = 1 - e^{-\lambda_L t}$ uniformly. Moreover,

$$\begin{aligned}\lim_{q\to\infty} T_{m+q}(\theta) = \lim_{q\to\infty} \int_0^\infty e^{-st} dF_q(t) &\overset{*}{=} \lim_{q\to\infty} s \int_0^\infty e^{-st} F_q(t) dt \\ &\overset{\triangle}{=} s \int_0^\infty \lim_{q\to\infty} e^{-st} F_q(t) dt = s \int_0^\infty e^{-st}(1 - e^{-\lambda_L t}) dt = \frac{\lambda_L}{\lambda_L + \theta}.\end{aligned}$$

Here, (*) is in Wolff [52, p. 533, eq. (21)]; and ($\triangle$) follows from the uniform convergence of $F_q(t)$.

Step 3 Because $z_2(\theta) > 1$, the convergence of $T_{m+q}(\theta)$ implies that $J_2(\theta) = 0$. Consequently, $T_{m+q}(\theta) = J_1(\theta) z_1^q(\theta) + \frac{\lambda_L}{\lambda_L+\theta}$. Letting $q = 0$, we obtain $J_1(\theta) = T_m(\theta) - \frac{\lambda_L}{\lambda_L+\theta}$. Plugging this and $J_2(\theta) = 0$ into (20), we obtain

$$T_{m+q}(\theta) = \left(T_m(\theta) - \frac{\lambda_L}{\lambda_L + \theta}\right) z_1^q(\theta) + \frac{\lambda_L}{\lambda_L + \theta} = T_m(\theta) z_1^q(\theta) + \frac{\lambda_L}{\lambda_L + \theta}\left[1 - z_1^q(\theta)\right]. \qquad (22)$$

□

Proposition 4 solves the (infinitely many) tail equations (9) for $s > m$. To completely determine the system, we need (9) for $s \leq m$, (11), and one other equation. The following corollary provides this equation:

Corollary 1

$$\lambda_H T_{m+1}(\theta) + [\lambda_H z_2(\theta) - (\omega_m + \theta)]\, T_m(\theta) + \frac{\lambda_L}{1 - z_2(\theta)} = 0, \tag{23}$$

where $z_2(\theta) = \frac{(\omega_m+\theta)+\sqrt{(\omega_m+\theta)^2-4m\mu\lambda_H}}{2\lambda_H}$.

Proof

From the proof of Proposition 4 we know that $J_2(\theta) = 0$ for all θ. Therefore, from (19), we get

$$\frac{\gamma_2(\theta)T_{m+1}(\theta) + (\gamma_2(\theta)z_2(\theta) - 1)T_m(\theta) + \gamma_3(\theta)/(1 - z_2(\theta))}{\gamma_2(\theta)(z_2(\theta) - z_1(\theta))} = 0,$$

which simplifies to

$$\lambda_H T_{m+1}(\theta) + [\lambda_H z_2(\theta) - (\omega_m + \theta)]\, T_m(\theta) + \frac{\lambda_L}{1 - z_2(\theta)} = 0.$$

□

In what follows, we use the fact that the kth derivative of $-T_s(\theta)$ evaluated at 0 gives the kth moment of $\tilde{T}_s$ to derive expressions for the first two moments of the inter-overflow time, which is then used to calculate the CV.

We first derive a complete solution of (9). Equation (11) solves (9) for $s > m$. The remaining equations in (9), together with (23), are linear, and they can be easily and quickly solved. In matrix-vector notation, (9) for states $s \leq m$, together with (23), can be expressed as:

$$Q(\theta)X(\theta) = y(\theta), \tag{24}$$

where $X(\theta) = \Big(T_0(\theta), \ldots, T_m(\theta), T_{m+1}(\theta)\Big)^{\mathsf{T}}$, $y(\theta) = \Big(0, \ldots, \frac{(1-p_L)\lambda_L}{\omega_L+\theta}, \frac{\lambda_L}{\omega_{L+1}+\theta}, \ldots, \frac{\lambda_L}{\omega_m+\theta}, -\frac{\lambda_L}{\lambda_H(1-z_2(\theta))}\Big)^{\mathsf{T}}$ and

$$Q(\theta) = \begin{pmatrix}
1, & -\frac{\omega_0}{\omega_0+\theta}, & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & -\frac{i\mu}{\omega_i+\theta}, & 1, & -\frac{\lambda_H+\lambda_L}{\omega_i+\theta}, & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & -\frac{L\mu}{\omega_i+\theta}, & 1, & -\frac{\lambda_H+p_L\lambda_L}{\omega_L+\theta}, & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & -\frac{i\mu}{\omega_i+\theta}, & 1, & -\frac{\lambda_H}{\omega_i+\theta}, & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & -\frac{m\mu}{\omega_m+\theta}, & 1, & -\frac{\lambda_H}{\omega_m+\theta} & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & z_2(\theta) - \frac{\omega_m+\theta}{\lambda_H}, & 1 & . & . & . & . & . & . & . & .
\end{pmatrix}.$$

Here the superscript T indicates transposition.

To derive the moments of the inter-overflow time, we repeatedly differentiate (24) and set $\theta = 0$:

Theorem 2

Let $\mathbf{1} \in R^{m+2}$ *be the vector whose components are all 1. Then the first two moments of the inter-overflow time are*

$$E(\tilde{T}) = \frac{\sum_{s=L}^{m-1} \xi_s(1-p_s)E(\tilde{T}_s) + \xi_m \left[\frac{E(\tilde{T}_m)-\frac{1}{\lambda_L}}{1-z_1(0)\frac{\lambda_H}{m\mu}} + \frac{1}{\lambda_L\left(1-\frac{\lambda_H}{m\mu}\right)}\right]}{\sum_{s\geq L}\xi_s(1-p_s)}, \tag{25}$$

where

$$\left(E(\tilde{T}_0),\ldots,E(\tilde{T}_{m+1})\right)^{\mathsf{T}} = Q^{-1}(0)\left[Q'(0)\mathbf{1} - y'(0)\right], \tag{26}$$

and

$$E(\tilde{T}^2) = \frac{1}{\sum_{s\geq L}\xi_s(1-p_s)}\left\{\sum_{s=L}^{m-1}\xi_s(1-p_s)E(\tilde{T}_s^2) + \xi_m\left[\frac{E(\tilde{T}_m^2)-\frac{2}{\lambda_L^2}}{1-z_1(0)\left(\frac{\lambda_H}{m\mu}\right)} + \frac{2}{\lambda_L^2\left(1-\frac{\lambda_H}{m\mu}\right)} + \frac{2\left(\frac{1}{\lambda_L}-E(\tilde{T}_m)\right)z_1'(0)\left(\frac{\lambda_H}{m\mu}\right)}{\left(1-z_1(0)\frac{\lambda_H}{m\mu}\right)^2}\right]\right\}, \tag{27}$$

where

$$\left(E(\tilde{T}_0^2),\ldots,E(\tilde{T}_{m+1}^2)\right)^{\mathsf{T}} = Q^{-1}(0)[2Q'(0)Q(0)^{-1}Q'(0)\mathbf{1} - 2Q'(0)Q(0)^{-1}y'(0) - Q''(0)\mathbf{1} + y''(0)]. \tag{28}$$

Proof

Differentiating (24) once, we obtain $Q'(\theta)X(\theta)|_{\theta=0} + Q(\theta)X'(\theta)|_{\theta=0} = y'(\theta)|_{\theta=0}$. Since $X(0) = \mathbf{1}$, yields (26). This gives us $E(\tilde{T}_s)$ for states $s \leq m+1$. For all the other states, we use (11):

$$\begin{aligned} E(\tilde{T}_{m+q}) &= -T'_{m+q}(0) \\ &= -\left[T'_m(0)z_1^q(0) + T_m(0)qz_1^{q-1}(0)z_1'(0) - \frac{1}{\lambda_L}\left(1-z_1^q(0)\right) - qz_1^{q-1}(0)z_1'(0)\right] \\ &= \left(E(\tilde{T}_m) - \frac{1}{\lambda_L}\right)z_1^q(0) + \frac{1}{\lambda_L}, \end{aligned} \tag{29}$$

where $E(\tilde{T}_m)$ is already determined in (26). Note that (29) is derived for $q \geq 1$, but clearly the result applies to $q = 0$ as well.

So the overall first moment of overflow is:

$$\begin{aligned} E(\tilde{T}) &= \frac{\sum_{i=L}^{m-1}\xi_i(1-p_i)E(\tilde{T}_i) + \sum_{q=0}^{\infty}\xi_{m+q}E(\tilde{T}_{m+q})}{\sum_{s\geq L}\xi_s(1-p_s)} \\ &= \frac{\sum_{i=L}^{m-1}\xi_i(1-p_i)E(\tilde{T}_i) + \xi_m\sum_{q=0}^{\infty}\left(\frac{\lambda_H}{m\mu}\right)^q\left[\left(E(\tilde{T}_m)-\frac{1}{\lambda_L}\right)z_1^q(0)+\frac{1}{\lambda_L}\right]}{\sum_{s\geq L}\xi_s(1-p_s)} \\ &= \frac{\sum_{i=L}^{m-1}\xi_i(1-p_i)E(\tilde{T}_i) + \xi_m\left[\frac{E(\tilde{T}_m)-\frac{1}{\lambda_L}}{1-z_1(0)\frac{\lambda_H}{m\mu}} + \frac{1}{\lambda_L\left(1-\frac{\lambda_H}{m\mu}\right)}\right]}{\sum_{s\geq L}\xi_s(1-p_s)}. \end{aligned} \tag{30}$$

Thus we have shown (25).

Now for the second moments. For states $s \leq m+1$ we will differentiate (24) twice and set $\theta = 0$: $y''(0) = Q''(0)\mathbf{1} + 2Q'(0)X'(0) + Q(0)X''(0)$ and

$$\begin{aligned}
&\left(E(\tilde{T}_0^2), \ldots, E(\tilde{T}_{m+1}^2)\right)^{\mathsf{T}} \\
= \;& X''(0) = Q^{-1}(0)[-2Q'(0)X'(0) - Q''(0)\mathbf{1} + y''(0)] \\
= \;& Q^{-1}(0)[2Q'(0)Q(0)^{-1}\left(Q'(0)\mathbf{1} - y'(0)\right) - Q''(0)\mathbf{1} + y''(0)] \\
= \;& Q^{-1}(0)[2Q'(0)Q(0)^{-1}Q'(0)\mathbf{1} - 2Q'(0)Q(0)^{-1}y'(0) - Q''(0)\mathbf{1} + y''(0)].
\end{aligned}$$

This proves (28). The second moments of $\tilde{T}$ corresponding to other states can be calculated similarly to (29). For $q \geq 2$:

$$\begin{aligned}
E(\tilde{T}_{m+q}^2) &= T''_{m+q}(0) \\
&= T''_m(0)z_1^q(0) + 2T'_m(0)qz_1^{q-1}(0)z'_1(0) + T_m(0)q(q-1)z_1^{q-2}(0)[z'_1(0)]^2 + T_m(0)qz_1^{q-1}(0)z''_1(0) \\
&\quad + \frac{2}{\lambda_L^2}\left(1 - z_1^q(0)\right) - \frac{2}{\lambda_L}\left[-qz_1^{q-1}(0)z'_1(0)\right] - [q(q-1)z_1^{q-2}(0)[z'_1(0)]^2 + qz_1^{q-1}(0)z''_1(0)] \\
&= T''_m(0)z_1^q(0) + 2T'_m(0)qz_1^{q-1}(0)z'_1(0) + \frac{2}{\lambda_L^2}\left(1 - z_1^q(0)\right) - \frac{2}{\lambda_L}\left[-qz_1^{q-1}(0)z'_1(0)\right] \\
&= E(\tilde{T}_m^2)z_1^q(0) - 2qE(\tilde{T}_m)z_1^{q-1}(0)z'_1(0) + \frac{2}{\lambda_L^2}\left(1 - z_1^q(0)\right) + \frac{2}{\lambda_L}qz_1^{q-1}(0)z'_1(0), \qquad (31)
\end{aligned}$$

where $E(\tilde{T}_m)$ and $E(\tilde{T}_m^2)$ are determined in (26) and (28) respectively.

Note that even though (31) is derived for $q \geq 2$, it applies to $q = 0, 1$ as well: for $q = 0$, it is trivial; for $q = 1$ we have

$$\begin{aligned}
E(\tilde{T}_{m+1}^2) &= T''_{m+1}(0) \\
&= T''_m(0)z_1(0) + 2T'_m(0)z'_1(0) + \frac{2}{\lambda_L^2}\left(1 - z_1(0)\right) + \frac{2}{\lambda_L}z'_1(0) \\
&= E(\tilde{T}_m^2)z_1(0) - 2E(\tilde{T}_m)z'_1(0) + \frac{2}{\lambda_L^2}\left(1 - z_1(0)\right) + \frac{2}{\lambda_L}z'_1(0). \qquad (32)
\end{aligned}$$

Thus, the overall second moment of overflow is:

$$\begin{aligned}
E(\tilde{T}^2) &= \frac{1}{\sum_{s\geq L}\xi_s(1-p_s)}\left\{\sum_{s=L}^{m-1}\xi_s(1-p_s)E(\tilde{T}_s^2)\right. \\
&\quad \left. + \xi_m\sum_{q=0}^{\infty}\left(\frac{\lambda_H}{m\mu}\right)^q\left[E(\tilde{T}_m^2)z_1^q(0) - 2qE(\tilde{T}_m)z_1^{q-1}(0)z'_1(0) + \frac{2}{\lambda_L^2}(1 - z_1^q(0)) + \frac{2}{\lambda_L}qz_1^{q-1}(0)z'_1(0)\right]\right\} \\
&= \frac{1}{\sum_{s\geq L}\xi_s(1-p_s)}\left\{\sum_{s=L}^{m-1}\xi_s(1-p_s)E(\tilde{T}_s^2)\right. \\
&\quad \left. + \xi_m\left[\frac{E(\tilde{T}_m^2) - \frac{2}{\lambda_L^2}}{1 - z_1(0)\left(\frac{\lambda_H}{m\mu}\right)} + \frac{2}{\lambda_L^2\left(1 - \frac{\lambda_H}{m\mu}\right)} + \frac{2\left(\frac{1}{\lambda_L} - E(\tilde{T}_m)\right)z'_1(0)\left(\frac{\lambda_H}{m\mu}\right)}{\left(1 - z_1(0)\frac{\lambda_H}{m\mu}\right)^2}\right]\right\}.
\end{aligned}$$

Thus we have shown (27). □

The first two moments of the inter-overflow time, (25) and (27), allow us to easily calculate the CV of the inter-overflow time, $CV = \sqrt{E(\tilde{T}^2) - E^2(\tilde{T})} \,/\, E(\tilde{T})$. We use this approach for the numerical analysis in Section 6.4.

E Inter-Overflow-Time CC in the Pooled-Overflow System

The analysis in this appendix characterizes the serial correlation between successive inter-overflow times of type-L calls from the client company to the outsourcer in the pooled-overflow scheme. This information is used to numerically evaluate the correlation coefficient (CC) of the inter-overflow time in the examples in Section 6.4

For notational convenience, we drop the subscript, I, from the number of in-house CSRs, m_I. Thus, the total number of in-house CSRs is referred to as "m" below.

E.1 1-step correlation

The 1-step serial correlation of the inter-overflow time is defined as

$$CC = \frac{E(\tilde{T}_i\tilde{T}_{i+1}) - E(\tilde{T}_i)E(\tilde{T}_{i+1})}{\sigma_{\tilde{T}_i}\sigma_{\tilde{T}_{i+1}}}, \tag{33}$$

where $\tilde{T}_i$ and $\tilde{T}_{i+1}$ are two consecutive inter-overflow times.

Given a stationary system, $E(\tilde{T}_i) = E(\tilde{T}_{i+1}) = E(\tilde{T})$ and $\sigma_{\tilde{T}_i} = \sigma_{\tilde{T}_{i+1}} = \sqrt{E(\tilde{T}^2) - E^2(\tilde{T})}$, where $E(\tilde{T})$ and $E(\tilde{T}^2)$ are given in (25) and (27). So it remains to derive $E(\tilde{T}_i\tilde{T}_{i+1})$.

Some additional notation is introduced in Table 2.

$F_{ij}(t)$:	the probability that starting in state i the next overflow will occur in state j and it will take no more than t for that to occur
$f_{ij}(t)$:	$= F'_{ij}(t)$
$f_i(t)$:	the PDF of the time to next overflow if the current state is i
$F^k_{ij}(t)$:	the probability that starting in state i the subsequent k-th overflow will occur in state j and it will take no more than t for that to occur
$f^k_{ij}(t)$:	the corresponding PDF
E_i:	the expected time to next overflow if the current state is i, shorthand for $E(\tilde{T}_i)$

Table 2: Notation

When an (L, p_L) policy is used, an overflow can only occur in states L and above. Let π be the steady-state distribution of the beginning state of an overflow. Recall that $p_0 = \cdots = p_{L-1} = 1$ and

$p_{L+1} = \cdots = p_{m-1} = 0$. Thus, $\pi_i = \frac{(1-p_i)\xi_i}{\sum_j (1-p_j)\xi_j}, \forall i$, and

$$\begin{aligned} E(\tilde{T}_i \tilde{T}_{i+1}) &= \int_0^\infty \int_0^\infty xy \left(\sum_{i,j} \pi_i f_{ij}(x) f_j(y) \right) dxdy = \sum_{i,j} \pi_i E_j \int_0^\infty x f_{ij}(x) dx \\ &= -\sum_{i,j} \pi_i E_j \hat{f}'_{ij}(s)|_{s=0} = -(\pi_0, \pi_1, \dots) \cdot \hat{f}'(0) \cdot \begin{pmatrix} E_0 \\ E_1 \\ \dots \end{pmatrix}, \end{aligned} \tag{34}$$

where E_j is the shorthand for $E(\tilde{T}_j)$, and $\hat{f}(s)$ is the matrix containing $\hat{f}_{ij}(s)$, the Laplace transform of $f_{ij}(x)$ for all i, j. So it remains for us to solve for the matrix $\hat{f}'(0) = \{\hat{f}'_{ij}(s)|_{s=0}\}_{i,j}$.

1) For $j \neq i$,

$$\begin{aligned} F_{ij}(t) &= \Pr(\text{starting in } i, \text{ it takes no more than } t \text{ for the next overflow to occur in } j) \\ &= \lim_{\delta t \to 0} \Big\{ \Pr(0 \text{ event in } \delta t) F_{ij}(t-\delta t) + \Pr(\text{one event}) \Big[\frac{\mu_i}{\omega_i} F_{i-1,j}(t-\delta t) + \frac{(1-p_i)\lambda_L}{\omega_i} 0 \\ &\qquad + \frac{\lambda_H + p_i \lambda_L}{\omega_i} F_{i+1,j}(t-\delta t) \Big] + o(\delta t) \Big\} \\ &= \lim_{\delta t \to 0} \Big\{ e^{-\omega_i \delta t} F_{ij}(t-\delta t) + e^{-\omega_i \delta t} \delta t \left[\mu_i F_{i-1,j}(t-\delta t) + (\lambda_H + p_i \lambda_L) F_{i+1,j}(t-\delta t) \right] + o(\delta t) \Big\}. \end{aligned}$$

In turn,

$$\begin{aligned} \lim_{\delta t \to 0} \frac{F_{ij}(t) - F_{ij}(t-\delta t)}{\delta t} &= \lim_{\delta t \to 0} \left\{ \frac{e^{-\omega_i \delta t} - 1}{\delta t} F_{ij}(t-\delta t) + e^{-\omega_i \delta t} \left[\mu_i F_{i-1,j}(t-\delta t) + (\lambda_H + p_i \lambda_L) F_{i+1,j}(t-\delta t) \right] \right\} \\ f_{ij}(t) &= -\omega_i F_{ij}(t) + \mu_i F_{i-1,j}(t) + (\lambda_H + p_i \lambda_L) F_{i+1,j}(t). \end{aligned} \tag{35}$$

2) For $j = i$,

$$\begin{aligned} F_{ii}(t) &= \Pr(\text{starting in } i, \text{ it takes no more than } t \text{ for the next overflow to occur in } i) \\ &= \lim_{\delta t \to 0} \Big\{ \Pr(0 \text{ event}) F_{ii}(t-\delta t) + \Pr(\text{one event}) \Big[\frac{(1-p_i)\lambda_L}{\omega_i} 1 + \frac{\mu_i}{\omega_i} F_{i-1,i}(t-\delta t) \\ &\qquad + \frac{\lambda_H + p_i \lambda_L}{\omega_i} F_{i+1,i}(t-\delta t) \Big] + o(\delta t) \Big\} \\ &= \lim_{\delta t \to 0} \Big\{ e^{-\omega_i \delta t} F_{ii}(t-\delta t) + e^{-\omega_i \delta t} \delta t \left[(1-p_i)\lambda_L + \mu_i F_{i-1,i}(t-\delta t) + (\lambda_H + p_i \lambda_L) F_{i+1,i}(t-\delta t) \right] + o(\delta t) \Big\}. \end{aligned}$$

In turn,

$$\begin{aligned} \lim_{\delta t \to 0} \frac{F_{ii}(t) - F_{ii}(t-\delta t)}{\delta t} &= \lim_{\delta t \to 0} \Big\{ \frac{e^{-\omega_i \delta t} - 1}{\delta t} F_{ii}(t-\delta t) + e^{-\gamma_i \delta t} \left[(1-p_i)\lambda_L + \mu_i F_{i-1,i}(t-\delta t) \right. \\ &\qquad \left. + (\lambda_H + p_i \lambda_L) F_{i+1,i}(t-\delta t) \right] \Big\} \\ f_{ii}(t) &= -\omega_i F_{ii}(t) + (1-p_i)\lambda_L + \mu_i F_{i-1,i}(t) + (\lambda_H + p_i \lambda_L) F_{i+1,i}(t). \end{aligned}$$

So, if we let

$$\Gamma = \begin{pmatrix} 0 & 0 & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & (1-p_L)\lambda_L & 0 & \dots \\ \dots & \dots & 0 & \lambda_L & 0 \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \leftarrow \text{row corresponding to state } L,$$

$$A = \begin{pmatrix} -\omega_0 & \lambda_H+\lambda_L & & & & & & & \\ & \cdots & & & & & & & \\ & \mu_i & -\omega_i & \lambda_H+\lambda_L & & & & & \\ & & & \cdots & & & & & \\ & & & \mu_L & -\omega_L & \lambda_H+p_L\mu_L & & & \\ & & & & & \cdots & \lambda_H & & \\ & & & & & \mu_m & -\omega_m & \lambda_H & \\ & & & & & & \mu_{m+1} & -\omega_{m+1} & \lambda_H \\ & & & & & & & \mu_{m+2} & \cdots \end{pmatrix}, \text{ and } F = \begin{pmatrix} F_{00} & F_{01} & \cdots \\ F_{10} & F_{11} & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix},$$

then

$$F' = \Gamma + AF.^1 \tag{36}$$

Since $\hat{f}(s) = s\hat{F}(s)$, we have:

$$\hat{f}(s) = \frac{\Gamma}{s} + \frac{A}{s}\hat{f}(s) \quad \Rightarrow \quad (sI - A)\hat{f}(s) = \Gamma. \tag{37}$$

Differentiating this we obtain $\hat{f}(s) + (sI - A)\hat{f}'(s) = 0$, and $\hat{f}(s)|_{s=0} = A\hat{f}'(s)|_{s=0}$. From (37) we know that $\hat{f}(s)|_{s=0} = -A^{-1}\Gamma$. Thus $\hat{f}'(s)|_{s=0} = -A^{-2}\Gamma$.

So (34) becomes:

$$\begin{aligned} E(\tilde{T}_i\tilde{T}_{i+1}) &= (\pi_0, \pi_1, \dots, \pi_i, \dots) \cdot A^{-2} \cdot \Gamma \cdot \begin{pmatrix} E_0 \\ E_1 \\ \cdots \end{pmatrix}. \\ &= (\pi_0, \pi_1, \dots, \pi_i, \dots) \cdot A^{-2} \cdot \begin{pmatrix} 0 \\ \cdots \\ (1-p_L)\lambda_L E_L \\ \lambda_L E_{L+1} \\ \cdots \\ \lambda_L E_j \\ \cdots \end{pmatrix}. \end{aligned}$$

Clearly it would be hard to invert the infinite matrix A. So we will let U and V be such that

$$U \cdot A = (\pi_0, \pi_1, \dots, \pi_i, \dots), \tag{38}$$

$$A \cdot V = \begin{pmatrix} 0 \\ \cdots \\ (1-p_L)\lambda_L E_L \\ \lambda_L E_{L+1} \\ \cdots \\ \lambda_L E_j \\ \cdots \end{pmatrix}, \tag{39}$$

and derive U and V directly. $E(\tilde{T}_i\tilde{T}_{i+1}) = U \cdot V$ follows.

[1] The solution of (36) is straightforward: $F(t) = \left(e^{At} - I\right) A^{-1}\Gamma$ and $f(t) = e^{At}\Gamma$. This is consistent with equation (5) in Fischer and Meier-Hellstern [19]. The matrices are related as follows: $\Gamma = \Lambda$, $A = Q - \Lambda$. However, the Laplace transform approach we adopt here can be extended to calculate any k-lag correlation.

Our approach is to solve the tails of U and V. The rest of the probabilities can be obtained by inverting a finite portion of A.

Vector V: We know from (39) that, for states $i \geq m$,

$$m\mu v_{m+q} - (m\mu + \lambda_H + \lambda_L)v_{m+q+1} + \lambda_H v_{m+q+2} = \lambda_L E_{m+q+1}.$$

Let $f_v(s) = \sum_{q=0}^{\infty} v_{m+q}s^q$, then

$$f_v(s) = \frac{g_v(s)s^2 - (m\mu + \lambda_H + \lambda_L)v_m s + \lambda_H \left[v_m + v_{m+1}s\right]}{m\mu s^2 - (m\mu + \lambda_H + \lambda_L)s + \lambda_H}, \tag{40}$$

where due to (29),

$$g_v(s) = \sum_{q=0}^{\infty} \lambda_L E_{m+q+1}s^q = \frac{\lambda_L E_m - 1}{1/z_1(0) - s} + \frac{1}{1-s}. \tag{41}$$

Let s_1 and s_2 be the two roots of the denominator of (40):

$$s_1 = \frac{(m\mu + \lambda_H + \lambda_L) + \sqrt{(m\mu + \lambda_H + \lambda_L)^2 - 4m\mu\lambda_H}}{2m\mu} > 1 \tag{42}$$

$$s_2 = \frac{(m\mu + \lambda_H + \lambda_L) - \sqrt{(m\mu + \lambda_H + \lambda_L)^2 - 4m\mu\lambda_H}}{2m\mu} < 1. \tag{43}$$

It's important to note that $s_1 = 1/z_1(0)$, where $z_1(0)$ is previously defined in Proposition 4 and equation (17). Then (40) and (41) combine to yield:

$$f_v(s) = \frac{a_v}{s - s_1} + \frac{b_v}{s - s_2} + \frac{c_v}{s - 1} + \frac{d_v}{(s - s_1)^2}, \tag{44}$$

where a_v, b_v, c_v, and d_v are constants that can be determined as

$$a_v = \frac{df_v(s)\,(s - s_1)^2}{ds}\,|_{s=s_1}, b_v = f_v(s)\,(s - s_2)\,|_{s=s_2}, c_v = f_v(s)\,(s - 1)\,|_{s=1}, d_v = f_v(s)\,(s - s_1)^2\,|_{s=s_1}\,. \tag{45}$$

Expanding (44), we obtain the tail of v:

$$v_{m+q} = -a_v s_1^{-1-q} - b_v s_2^{-1-q} - c_v + d_v(q+1)s_1^{-2-q}. \tag{46}$$

We note that because $0 < s_2 < 1$, we must have $b_v = 0$, which yields

$$\left[(m\mu + \lambda_H + \lambda_L)s_2 - \lambda_H\right] v_m - \lambda_H s_2\; v_{m+1} = \left(\frac{\lambda_L E_m - 1}{s_1 - s_2} + \frac{1}{1 - s_2}\right) s_2^2. \tag{47}$$

(46) solves the tail for V using the equations $i > m$ in (39). The rest of the equations in (39), along with (47), give us the rest of the vector V:

$$\begin{pmatrix} v_0 \\ \cdots \\ v_i \\ \cdots \\ v_m \\ v_{m+1} \end{pmatrix} = \begin{pmatrix} -\omega_0 & \bar{\lambda} & & & & \\ & \cdots & & & & \\ & \mu_i & -\omega_i & \lambda_H + p_i\lambda_L & & \\ & & & \cdots & & \\ & & & \mu_m & -\omega_m & \lambda_H \\ & & & & \omega_m s_2 - \lambda_H & -\lambda_H s_2 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \cdots \\ (1 - p_i)\lambda_L E_i \\ \cdots \\ \lambda_m E_m \\ \left(\frac{\lambda_L E_m - 1}{s_1 - s_2} + \frac{1}{1 - s_2}\right) s_2^2 \end{pmatrix}. \tag{48}$$

Vector U: Similarly we can find the tail of U. We know from (38) that, for states $i \geq m$,

$$\lambda_H u_{m+q} - (m\mu + \lambda_H + \lambda_L)u_{m+q+1} + m\mu u_{m+q+2} = \pi_{m+q+1}.$$

Define $f_u(s) = \sum_{q=0}^{\infty} u_{m+q}s^q$, then

$$f_u(s) = \frac{\frac{\pi_m}{m\mu/\lambda_H - s}s^2 - (m\mu + \lambda_H + \lambda_L)u_m s + m\mu\left[u_m + u_{m+1}s\right]}{\lambda_H s^2 - (m\mu + \lambda_H + \lambda_L)s + m\mu} \tag{49}$$

$$= \frac{h_u(s)}{(s - 1/s_1)(s - 1/s_2)(s - m\mu/\lambda_H)}. \tag{50}$$

Because $h_u(s)$ is a quadratic polynomial of s, it can be further simplified to (it is easy to check that $m\mu/\lambda_H \neq 1/s_1$ and $m\mu/\lambda_H \neq 1/s_2$):

$$f_u(s) = \frac{a_u}{s - 1/s_1} + \frac{b_u}{s - 1/s_2} + \frac{c_u}{s - m\mu/\lambda_H}, \tag{51}$$

where a_u, b_u, and c_u are constants that can be determined as

$$a_u = f(s)\left(s - 1/s_1\right)|_{s=1/s_1}, \quad b_u = f(s)\left(s - 1/s_2\right)|_{s=1/s_2}, \quad c_u = f(s)\left(s - m\mu/\lambda_H\right)|_{s=m\mu/\lambda_H}. \tag{52}$$

Because $s_1 > 1$, we must have $a_u = 0$, which yields:

$$\left[(m\mu + \lambda_H + \lambda_L)/s_1 - m\mu\right]u_m - (m\mu/s_1)u_{m+1} = \frac{\pi_m}{m\mu/\lambda_H - 1/s_1}/s_1^2. \tag{53}$$

Expanding (51), and using (53), we obtain the tail of u:

$$u_{m+q} = -b_u s_2^{1+q} - c_u\left(\lambda_H/(m\mu)\right)^{1+q}. \tag{54}$$

And the rest of the vector U can be solved as follows:

$$\begin{pmatrix} u_0 \\ \cdots \\ u_i \\ \cdots \\ u_m \\ u_{m+1} \end{pmatrix} = \left[\begin{pmatrix} -\omega_0 & \bar{\lambda} & & & & \\ & \cdots & & & & \\ & \mu_i & -\omega_i & \lambda_H + p_i\lambda_L & & \\ & & & \cdots & \lambda_H & \\ & & & \mu_m & -\omega_m & \frac{\omega_m}{s_1} - \mu_m \\ & & & & \mu_{m+1} & -\frac{\mu_{m+1}}{s_1} \end{pmatrix}^T\right]^{-1} \begin{pmatrix} \pi_0 \\ \cdots \\ \pi_i \\ \cdots \\ \pi_m \\ \frac{\pi_m/s_1^2}{\frac{\mu_{m+1}}{\lambda_H} - \frac{1}{s_1}} \end{pmatrix}. \tag{55}$$

$U \cdot V$: (46) and (54) allow us to express $U \cdot V$ as

$$U \cdot V = \sum_{i=0}^{m-1} u_i v_i + b_u a_v \frac{s_2}{s_1 - s_2} + c_u a_v \frac{\lambda_H}{m\mu\, s_1 - \lambda_H} + b_u c_v \frac{s_2}{1 - s_2}$$
$$+ c_u c_v \frac{\lambda_H}{m\mu - \lambda_H} - b_u d_v \frac{s_2}{(s_1 - s_2)^2} - c_u d_v \frac{m\mu\lambda_H}{(m\mu s_1 - \lambda_H)^2}. \tag{56}$$

E.2 k-step correlation

Our approach to the 1-step correlation can be extended to the k-step correlation for any k:

$$f^k = \Gamma F^{k-1} + AF^k. \tag{57}$$

The Laplace transform of this gives us:

$$\hat{f}^k(s) = \frac{\Gamma}{s}\hat{f}^{k-1}(s) + \frac{A}{s}\hat{f}^k(s) \quad \Rightarrow (sI - A)\hat{f}^k(s) = \Gamma\hat{f}^{k-1}(s). \tag{58}$$

This easily gives us that

$$\hat{f}^k(s)|_{s=0} = -A^{-1}\Gamma\hat{f}^{k-1}(s)|_{s=0} = \left(-A^{-1}\Gamma\right)^k.$$

To avoid double superscripts, we will use the overhead dot to represent derivatives. Then differentiating (58) we also obtain:

$$\begin{aligned}
\hat{f}^k(s)|_{s=0} + (sI - A)\dot{\hat{f}}^k(s)|_{s=0} &= \Gamma\dot{\hat{f}}^{k-1}(s)|_{s=0} && (59)\\
\left(-A^{-1}\Gamma\right)^k - A\dot{\hat{f}}^k(0) &= \Gamma\dot{\hat{f}}^{k-1}(0) && (60)\\
\dot{\hat{f}}^k(0) &= A^{-1}\left(-A^{-1}\Gamma\right)^k - A^{-1}\Gamma\dot{\hat{f}}^{k-1}(0). && (61)
\end{aligned}$$

Given that $\dot{\hat{f}}^1(0) = \hat{f}'(0) = -A^{-2}\Gamma$, this recursive definition can be used to find the k-step serial correlations. The details are complex and are omitted here.

F The Pooled-Overflow System when $\mu_{\mathbf{H}}$ May Differ From $\mu_{\mathbf{L}}$

In this appendix, we consider a more general class of systems in which μ_H may differ from μ_L. (Systems in which $\mu_H = \mu_L$ represent a special case.) We note that, when $\mu_H \neq \mu_L$, our analysis of the dedicated-overflow and inverted-V schemes is not affected, since capacity for type-H and type-L calls is partitioned.

Because the pooled-overflow scheme shares in-house capacity across both types of calls, its analysis does become more complex, however. Appendix F.1 proves that, among type-H priority policies, there are stationary, type-H work-conserving policies that maximize the in-house throughput of type-L calls. It then formulates a linear program with $O(m_I^2)$ variables and $O(m_I^2)$ constraints that identifies such an optimal policy. Appendix F.2 extends Appendix D's analysis of the moments of the inter-overflow time CV to include cases in which μ_H may not equal μ_L.

For notational convenience, we drop the subscript, I, from the number of in-house CSRs, m_I. Thus, the total number of in-house CSRs is referred to as "m" below.

F.1 An LP for Finding Optimal Type-H Priority Policies

In Section 4.1, we derived the optimality of (L, p_L) policies using the NLP (3)–(6). For the more general case in which μ_H may not equal μ_L, we use a different, linear programming (LP) approach to solve the constrained optimization problem.

As before, we define the state and action spaces, S and A, in terms of numbers of calls. The dimensionality of the state space in the LP will be greater than that in the NLP of Section 4.1, however. Instead of simply tracking the total number of calls in the system, we now must mark how many CSRs are busy with type-H and type-L calls, respectively. Furthermore, we will track system performance before (rather than after) action, and we need to record the presence or absence of an arriving type-L call.

Formally, we now define each state of the system, $s \in S$, at discrete event epoch t as follows. Let

$$S = \{(i^H, i^L, q^H, q^L) : i^H, i^L \geq 0;\ i^H + i^L \leq m;\ q^H \geq 0;\ q^L \in \{0, 1\}\} \tag{62}$$

and $s_t \in S$ be the number of calls in service or in queue at t: i^H represents the number of type-H calls in service; i^L, the number of type-L calls in service; and q^H, the number of type-H calls in queue. If the event at t is an arrival of a type-L call, then $q^L = 1$; otherwise $q^L = 0$.

Again, $s_t = (i_t^H, i_t^L, q_t^H, q_t^L)$ represents the state of the system at transition t, *before* any action is taken. In contrast, the analysis of Section 4.1 focused on the states after actions are taken, the *after-action states*. Nevertheless, in the following analysis, it will sometimes be useful for us to refer to after-action states. We do so by denoting the after-action states with an overbar above the state descriptor: that is, after-action states are $\bar{s} \in \bar{S}$.

Because arriving type-L calls are either immediately put into service or routed to the outsourcer, there never exists a type-L call in queue after action, and we drop element $\bar{q}^L$ from the descriptor. Thus the after-action state space becomes

$$\bar{S} = \{(\bar{\imath}^H, \bar{\imath}^L, \bar{q}^H) : \bar{\imath}^H, \bar{\imath}^L \geq 0;\ \bar{\imath}^H + \bar{\imath}^L \leq m;\ \bar{q}^H \geq 0\}\,, \tag{63}$$

and $\bar{s}_t = (\bar{\imath}_t^H, \bar{\imath}_t^L, \bar{q}_t^H)$ denotes the after-action state at transition t.

In any state, a system controller may put one or more calls into service, or it may do nothing. Accordingly, let c^H and c^L be the numbers of type-H and type-L calls put into service at an arbitrary event epoch. We define the set of feasible actions in state $s \in S$ to be

$$A_s = \{(c^H, c^L) :\ c^H, c^L \geq 0;\ c^H + c^L \leq m - (i^H + i^L);\ c^H \leq q^H;\ c^L \leq q^L\}, \tag{64}$$

and the action taken at time t to be $a_t \in A_{s_t}$. We denote the superset of all feasible actions as $A = \{(c^H, c^L) : 0 \leq c^H \leq m;\ c^L \in \{0, 1\};\ (c^H + c^L) \leq m\} \supseteq A_s$ for all $s \in S$. Observe that A is finite.

A *policy* is a rule that the system controller uses to choose an action to take at each event epoch t. Let $\mathcal{H}_t = \{(s_0, a_0), \ldots, (s_{t-1}, a_{t-1}) \cup s_t\}$, be the *history* of the system up to event epoch t. Then

a *non-anticipating* policy is a rule which, given $\mathcal{H}_t$, chooses an action a_t, possibly at random, among the actions of A_{s_t}. A type-H *priority* policy never puts type-L calls into service when there is a type-H call in queue. We define Π to be the class of all non-anticipating, type-H priority policies.

The objective is to find a policy, $\pi \in \Pi$, that maximizes the rate at which type-L calls are served in house. For a given before-action state, $s \in S$, define the *reward* associated with action a to be $R(s,a)$. We let $R(s,a)$ equal the number of type-L calls put into service. In turn, we define

$$\bar{R}_\pi(s) \stackrel{\text{def}}{=} \liminf_{n\to\infty} \frac{1}{n} \mathsf{E}_\pi \left[\sum_{t=0}^{n-1} R(s_t, a_t) | s_0 = s \right] \tag{65}$$

to be the long run average rate at which a policy $\pi \in \Pi$ serves type-L calls.

Because the state space is defined in terms of system occupancy, it is convenient for us to account for type-H calls' service-level in terms of occupancy as well. We denote by $D(s,a)$ the "delay cost" associated with state-action combination (s,a), and we let $D(s_t, a_t)$ be some non-negative function of the number of type-H calls remaining in queue after the policy's action at t.

In particular, we let $d(\bar{q})$ be the delay-cost function associated with the after-action queue length q, and let the constraint on the type-H service level be that the long-run average delay cost be no more than D^*. For the delay-cost function $d(\bar{q})$, we assume:

Assumption 1

i) $d(0) = 0$ *and* $d(\bar{q})$ *is nondecreasing in* $\bar{q}$*;*

ii) $\sup_{\bar{q}} d(\bar{q}) > D^*$*; and*

iii) $\tilde{d}(\alpha) \stackrel{\text{def}}{=} \sum_{\bar{q}=0}^{\infty} \alpha^{\bar{q}}\, d(\bar{q}) < \infty$ *for all* $\alpha \in (0,1)$.

Item (i) ensures that the cost increases as the backlog grows. Let $H(n)$ be the number of type-H arrivals put into serve in the first n transitions. Then together items (i) and (ii) imply that any sample path for which $\lim_{n\to\infty} H(n)/n < \lambda_H$ is also one for which $\lim_{n\to\infty} d(\bar{q})/n > D^*$, and thus violates the service level constraint. Finally, item (iii) defines the generating function $\tilde{d}$ and implies that the service-level cost of occupancy grows sub-exponentially. All of these restrictions are satisfied by formulations of standard service-level constraints, such as bounds on expected occupancy and on the tail distribution of occupancy.

In turn, we define

$$\bar{D}_\pi(s) \stackrel{\text{def}}{=} \limsup_{n\to\infty} \frac{1}{n} \mathsf{E}_\pi \left[\sum_{t=0}^{n-1} D(s_t, a_t) | s_0 = s \right], \tag{66}$$

and we require that $\bar{D}_\pi(s) \le D^*$, where D^* is an exogenously defined upper bound on the average backlog cost.

Although $D(s,a)$ and $\bar{D}_\pi(s)$ are defined as functions of queue occupancy, rather than delay in queue, in many cases they are equivalent. In particular, given the use of a stationary policy, one can use Little's Law to translate between several common versions of occupancy and delay constraints. (See Gans and Zhou [21].)

Using these definitions of reward and cost, we can formally state the problem of maximizing the throughput of type-L calls, subject to the service level constraint on type-H queue, as follows:

$$\sup_{\pi\in\Pi} \quad \bar{R}_\pi(s) \qquad \text{s.t.} \quad \bar{D}_\pi(s) \quad \leq \quad D^* \,. \tag{67}$$

Any policy π that satisfies the constraint in (67) is called *feasible*. If it also achieves the supremum in (67), then it is *optimal* or solves the constrained optimization problem (COP).

In turn, we can characterize an easily computable class of policies that solves the COP. For the general case, in which μ_H may not equal μ_L, there exist optimal policies $\pi \in \Pi$ that are type-H work conserving. These *work-conserving* policies never allow a type-H call to queue when there is an idle CSR. Furthermore, among type-H priority, type-H work-conserving policies, there are, in turn, *stationary* (history-independent) policies that are optimal. We call the class of stationary, type-H priority, type-H work-conserving policies Π^*. Below, we will show that, among all policies in Π, there exist policies within Π^* that maximize the throughput rate of type-L calls.

The first step is to show that, by considering only type-H work conserving policies we do not unintentionally degrade system performance. The lemma can be proved using the argument that proves Lemma 7 in [21].

Lemma 3 *Suppose there exists a feasible type-H priority policy, π. Then there exists a feasible type-H priority, type-H work-conserving policy π with the same throughput as π.*

The lemma allows us to significantly simplify the problem. If a policy gives priority to and is work conserving with respect to type-H calls, then it must be the case that $\bar{\imath}^H < m \implies \bar{q}^H = 0$ and $\bar{q}^H > 0 \implies \bar{\imath}^H = m$. This allows us to reduce the state-space, any eliminating a separate identifier for the state of the type-H queue. We therefore let

$$S = \{(i^H, i^L, q^L) : i^H \geq 0;\ 0 \leq i^L \leq m;\ q^L \in \{0,1\}\}, \tag{68}$$

where i^L and q^L are defined as before. Similarly, the after-action state space becomes

$$\bar{S} = \{(\bar{\imath}^H, \bar{\imath}^L) : \bar{\imath}^H \geq 0;\ 0 \leq \bar{\imath}^L \leq m\}. \tag{69}$$

Actions also simplify. At time 0, a type-H work-conserving policy puts as many type-H calls into service as possible. Then at subsequent event epochs there will never be the opportunity to put more than one type-H call into service at a time – otherwise the policies will not have been work conserving. By the same argument, it will never be the case that a type-H and type-L call are put

into service at the same time. Therefore, at any event after which there exists a type-H call in queue, there is only one optimal action – put the type-H call into service. We can embed this action into the state transitions of the Markov chain.

Without loss of generality, we define the time scale so that $\lambda_H + \lambda_L + m(\mu_H + \mu_L) = 1$. Therefore, we may view transition rates as probabilities. For example, $\lambda_H = \frac{\lambda_H}{\lambda_H + \lambda_L + m(\mu_H + \mu_L)}$ equals the expected number of type-H arrivals per period, as well as the probability that the next event is a type-H arrival.

The state transitions of the Markov chain are as follows:

$$(i^H_{t+1}, i^L_{t+1}, q^L_{t+1}) \quad = \quad \begin{cases} (\bar{i}^H_t + 1, \bar{i}^L_t, 0), & \text{w. p. } \lambda_H; \\ (\bar{i}^H_t, \bar{i}^L_t, 1), & \text{w. p. } \lambda_L; \\ (\bar{i}^H_t - 1, \bar{i}^L_t, 0), & \text{w. p. } \min\{\bar{i}^H_t, m - \bar{i}^L_t\}\, \mu_H; \\ (\bar{i}^H_t, \bar{i}^L_t - 1, 0), & \text{w. p. } \bar{i}^L_t\, \mu_L; \text{ and} \\ (\bar{i}^H_t, \bar{i}^L_t, 0), & \text{w. p. } m(\mu_H + \mu_L) - \min\{\bar{i}^H_t, m - \bar{i}^L_t\}\, \mu_H - \bar{i}^L_t \mu_L. \end{cases} \tag{70}$$

Here, $\min\{\bar{i}^H_t, m - \bar{i}^L_t\}$ represents the number of type-H calls in service, after action, at epoch t. In turn, feasible actions reduce to

$$A_s = \{c^L : c^L \in \{0, 1\};\ c^L \leq (m - (i^H + i^L))^+\}, \tag{71}$$

where $(m - (i^H + i^L))^+$ denotes the number of idle servers. In turn, $A = \{0, 1\}$.

Thus, the complexity of the routing problem has been reduced substantially. There exist only $\frac{m(m+1)}{2}$ states in which a type-L call has arrived to a system with at least one CSR free, and in each of these states there are only two feasible actions: accept or reject the call. When all m CSRs become busy, there are no decisions to be made, and the evolution of the system states follows a Markov chain.

Furthermore, using arguments analogous to those used in [21], we can show that each type-H priority, type-H work conserving policy that is stationary and deterministic also: i) induces a single, positive recurrent class of states, with expected absorption time into that class that is finite; ii) has limiting state-action frequencies which correspond to the stationary distribution of the induced Markov chain; and iii) has uniformly integrable one-period revenues. Therefore, we can appeal to Theorem 7.1 in Altman and Schwartz [2] to show that there exist stationary, type-H priority, type-H work conserving policies that are optimal:

Lemma 4 *If there exists a policy $\pi \in \Pi$ that is feasible, then exists a policy $\pi \in \Pi^*$ that is optimal.*

The result in [2] also implies that we can formulate an LP whose optimal solution identifies the optimal policy.

Because the set of states in which all m CSRs are busy is infinite, the LP has an infinite set of balance equations. Nevertheless, we can use algebraic substitution to develop closed-form expressions for essential quantities related to these tail states and make the LP finite. More specifically, under

the following assumption, we can collapse the states of the Markov chain when all m CSRs are busy into a set of $O(m^2)$ linear equations.

Assumption 2

i) *The pool-H queue is stable:* $\rho < 1$.

ii) *Either* $\mu_H \leq \mu_L$ *or* $\lambda_H \neq m(\mu_H - \mu_L)$.

The first condition allows us to show that any policy $\pi \in \Pi^*$ achieves a steady state with uniformly bounded costs. The second condition, which occurs almost surely, allows us to use a set of generating functions to collapse the "tail" states of the Markov chain induced by a stable, stationary policy.

Remark 1 Part (ii) of Assumption 2 is a technical assumption that assures that the denominator of the generating function used in Lemma 5, below, has distinct roots. This simplifies the expression of the partial fraction expansion. In the case that part (ii) of Assumption 2 is violated, we can still use the generating-function approach to formulate an analogous, finite LP, albeit one with more complex expressions. For details, see [21]. Because the assumption is essentially never violated, however, we omit the treatment of this case, here.

Let $\xi_{i,j,k}(a)$ be the stationary probability of entering (before-action) state $(i^H = i, i^L = j, q^L = k)$ and taking action $a \equiv c^L$. Then the following lemma shows that the tail probabilities can be conveniently characterized:

Lemma 5 (Gans and Zhou [21])
Let

$$\gamma_1(j) \stackrel{\text{def}}{=} \frac{\lambda_H}{\lambda_H + (m-j)\mu_H + j\mu_L}, \tag{72}$$

$$\gamma_2(j) \stackrel{\text{def}}{=} \frac{(m-j)\mu_H}{\lambda_H + (m-j)\mu_H + j\mu_L}, \text{ and} \tag{73}$$

$$\gamma_3(j) \stackrel{\text{def}}{=} \frac{(j+1)\mu_L}{\lambda_H + (m-j)\mu_H + j\mu_L}. \tag{74}$$

If the conditions of Assumption 2 hold, then for each j the quadratic equation

$$g(j,z) \stackrel{\text{def}}{=} \gamma_2(j)z_j^2 - z_j + \gamma_1(j) = 0 \tag{75}$$

has roots $z_j \leq z_j'$ with $0 < z_j < 1$. In turn, for any policy $\pi \in \Pi^*$ there exists constants $a_{j,l}$ such that

$$a_{m,m} = \sum_{a=0}^{1} \xi_{0,m-a,a}(a) \tag{76}$$

$$a_{j,j} = \sum_{a=0}^{1} \xi_{m-j,j-a,a}(a) + \frac{\gamma_3(j)}{\gamma_2(j)} \sum_{a=0}^{1} \xi_{m-j-1,j+1-a,a}(a)$$

$$+ \frac{\gamma_3(j) z_j}{\gamma_2(j)(z'_j - z_j)} \sum_{k \geq j+1} \frac{a_{j+1,k}}{1 - z_k/z_j} - \frac{\gamma_3(j) z'_j}{\gamma_2(j)(z'_j - z_j)} \sum_{k \geq j+1} \frac{a_{j+1,k}}{1 - z_k/z'_j}$$

$$\forall\ 0 \leq j \leq m-1 \tag{77}$$

$$a_{j,k} = \frac{-\gamma_3(j)}{\gamma_2(j)(1 - z_j/z_k)(1 - z'_j/z_k)} a_{j+1,k} \qquad \forall\ 0 \leq j < k \leq m, \tag{78}$$

and

$$\sum_{l=0}^{1} \xi_{m-j+q,j,l}(0) = \sum_{k=j}^{m} a_{j,k} z_k^q, \qquad \forall 0 \leq j \leq m, \forall q \geq 1. \tag{79}$$

Thus, given Assumption 2, for any policy $\pi \in \Pi^*$, the tail states of the Markov chain can be represented as mixtures of geometric series, and we can use (79) to collapse the tail and formulate a finite-dimension LP.

We define the LP that finds a policy $\pi \in \Pi^*$ that maximizes the throughput of type-L calls. Specifically, let $s = (i', j', k')$ be shorthand for a given before-action state, and let $K(i,j)$ be the set of state-action pairs (s,a) that land the system in *after-action* state (i,j). If $K(i,j) = \emptyset$ then interpret the associated summation as equal to zero. Then, using (72)-(79) from Lemma 5, the following LP finds a constrained optimal policy:

$$\max \sum_{i+j<m} \xi_{i,j,1}(1) \tag{80}$$

s. t.

$$\xi_{i,j,0}(0) = \lambda_H \sum_{(s,a) \in K(i-1,j)} \xi_s(a) + (i+1)\mu_H \sum_{(s,a) \in K(i+1,j)} \xi_s(a) + (j+1)\mu_L \sum_{(s,a) \in K(i,j+1)} \xi_s(a)$$
$$+ ((m-i)\mu_H + (m-j)\mu_L) \sum_{(s,a) \in K(i,j)} \xi_s(a) \qquad 0 \leq i+j \leq m-1 \tag{81}$$

$$\sum_{a=0}^{1} \xi_{i,j,1}(a) = \lambda_L \sum_{(s,a) \in K(i,j)} \xi_s(a) \qquad 0 \leq i+j \leq m-1 \tag{82}$$

$$\xi_{m-j,j,0}(0) = \lambda_H \sum_{(s,a) \in K(m-j-1,j)} \xi_s(a) + (j+1)\mu_L \sum_{k=j+1}^{m} a_{j+1,k} z_k + (m-j)\mu_H \sum_{k=j}^{m} a_{j,k} z_k$$
$$+ (j\mu_H + (m-j)\mu_L) \sum_{(s,a) \in K(m-j,j)} \xi_s(a) \qquad 0 \leq j \leq m-1 \tag{83}$$

$$\xi_{0,m,0}(0) = m\mu_H \sum_{(s,a) \in K(0,m)} \xi_s(a) \tag{84}$$

$$\xi_{m-j,j,1}(0) = \lambda_L \sum_{(s,a) \in K(m-j,j)} \xi_s(a) \qquad 0 \leq j \leq m \tag{85}$$

$$a_{m,m} = \sum_{a=0}^{1} \xi_{0,m-a,a}(a) \tag{86}$$

$$a_{j,j} = \sum_{a=0}^{1} \xi_{m-j,j-a,a}(a) + \frac{\gamma_3(j)}{\gamma_2(j)} \sum_{a=0}^{1} \xi_{m-j-1,j+1-a,a}(a)$$

$$+ \frac{\gamma_3(j) z_j}{\gamma_2(j)(z'_j - z_j)} \sum_{k \geq j+1} \frac{a_{j+1,k}}{1 - z_k/z_j} - \frac{\gamma_3(j) z'_j}{\gamma_2(j)(z'_j - z_j)} \sum_{k \geq j+1} \frac{a_{j+1,k}}{1 - z_k/z'_j}$$

$$\forall\ 0 \leq j \leq m-1 \qquad (87)$$

$$a_{j,k} = \frac{-\gamma_3(j)}{\gamma_2(j)(1 - z_j/z_k)(1 - z'_j/z_k)} a_{j+1,k} \qquad \forall\ 0 \leq j < k \leq m \qquad (88)$$

$$\sum_{j=0}^{m} \sum_{k=j}^{m} a_{j,k} \tilde{d}(z_k) \leq D^* , \qquad (89)$$

$$\sum_{i+j \leq m} \sum_{k=0}^{1} \sum_{a \in A_s} \xi_{i,j,k}(a) + \sum_{j=0}^{m} \sum_{k=j}^{m} a_{j,k} \frac{z_k}{1 - z_k} = 1 \qquad (90)$$

$$\xi_s(a) \geq 0 \qquad \forall s \in S, a \in A_s . \qquad (91)$$

Here, the objective function (80) maximizes the rate at which type-L calls are put into service. The constraints (81)–(88) are the system's balance constraints. Of these, (86)–(88) define the $\xi_{i,j,k}$s associated with boundary states in terms of the geometric series of Lemma 5. Constraint (89) ensures that the service-level is met and (90) that probabilities sum to one. Constraints (91) ensure that the probabilities are non-negative. (Note that the LP formulation must drop one redundant balance equation. See §8.8 in Puterman [39].)

Thus, an optimal policy $\pi \in \Pi^* \subset \Pi$ can be found via the solution of the LP. The following theorem summarizes and formalizes all the results so far:

Theorem 3

Suppose Assumptions 1 and 2 hold. Then there exists an LP (80)–(91) with $O(m^2)$ variables and $O(m^2)$ constraints which is feasible if and only if there exists a policy $\pi \in \Pi$ that is feasible as well. The optimal solution of the LP determines a policy $\pi \in \Pi^*$ that solves the COP (67).

We note that, although the proof of Theorem 3 is a direct analogue to that used when $\lambda_L = \infty$, differences in the two systems' type-L call dynamics leads the LP in the current paper to differ from – and to have fewer decision variables than – the LP in [21]. In particular, the fact that $\lambda_L < \infty$ implies that at most one type-L call can be put into service at a time. This is not the case when $\lambda_H = \infty$, however. Therefore, the current LP has $\mathrm{O}(m^2)$, rather than the $\mathrm{O}(m^3)$, decision variables.

F.2 Inter-Overflow Time CV in the Pooled-Overflow System

Next, we extend the results regarding the moments of inter-overflow time, in particular Proposition 4, to the case in which μ_H may not equal μ_L. Rather than limiting the analysis to randomized threshold-reservation policies, (L, p_L), our analysis in this section holds for the overflow process that results from any stationary, type-H priority, type-H work conserving policy $\pi \in \Pi^*$. For example, the policies can be derived using Theorem 3 and the associated LP formulation (80)-(91).

For brevity, let $\tilde{T}_{\bar{\jmath},\bar{q}}$ denote the time to the next type-L call "overflow" starting in after-action state $(m-\bar{\jmath}+\bar{q},\bar{\jmath})$, and let $T_{\bar{\jmath},\bar{q}}(\theta)$ be its Laplace transform. Then,

Proposition 5

$$T_{\bar{\jmath},\bar{q}}(\theta)=\sum_{k=0}^{\bar{\jmath}}a_{\bar{\jmath},k}z_k(\theta)^{\bar{q}}+\frac{\lambda_L}{\lambda_L+\theta},$$

where

$$z_k(\theta)=\frac{(\lambda_H+\lambda_L+(m-k)\mu_H+k\mu_L+\theta)-\sqrt{(\lambda_H+\lambda_L+(m-k)\mu_H+k\mu_L+\theta)^2-4(m-k)\mu_H\lambda_H}}{2\lambda_H}$$

and $a_{\bar{\jmath},k}$, $\forall\ 0\le k\le\bar{\jmath}\le m$, *are constants uniquely determined by* $\{T_{\bar{\jmath},0}(\theta),T_{\bar{\jmath},1}(\theta)\}$.

Proof

The proof will proceed in three steps:

1. Show that
$$T_{j,\bar{q}}(\theta)=\sum_{k=0}^{j}a_{j,k}(z_k(\theta))^{\bar{q}}+a'_j(z'_j(\theta))^{\bar{q}}+\frac{\lambda_L}{\lambda_L+\theta},\forall j,\bar{q}, \tag{92}$$
for the z_ks given in the proposition statement and some $z'_j>1$.

2. Show that $\lim_{\bar{q}\to\infty}T_{j,\bar{q}}(\theta)=\frac{\lambda_L}{\lambda_L+\theta},\forall\ j$.

3. From the first two steps, we must have $a'_j=0,\forall j$, because $z'_j>1$.

Let $f_j(z,\theta)\stackrel{\text{def}}{=}\sum_{\bar{q}=0}^{\infty}T_{j,\bar{q}}(\theta)z^{\bar{q}}$, then it suffices to show

$$f_j(z,\theta)=\sum_{k=0}^{j}\frac{a_{j,k}}{1-z_k(\theta)z}+\frac{a'_j}{1-z'_j(\theta)z}+\frac{\lambda_L/(\lambda_L+\theta)}{1-z}\qquad\forall\ j.$$

In what follows, when it is clear from the context we will again suppress the θ notation to simplify the exposition.

Again, we let $\tilde{T}_s$ denote the time to next overflow if the in-house call center is in state s. Based on (7), these random variables are sufficient to specify the general inter-overflow time distribution. Moreover, it is sufficient for us to focus on the states where all CSRs are busy. For brevity, let $\tilde{T}_{j,\bar{q}}$ denote the time to next overflow if the in house call center is in state $(m-j,j,\bar{q})$. Moreover, let $\omega_j=\lambda_H+\lambda_L+(m-j)\mu_H+j\mu_L$, and let $\tilde{e}_{\omega_j}$ denote an exponential random variable with rate ω_j. Then we have the following equations, which are analogues to (8) for the case in which $\mu_H=\mu_L$:

$$\tilde{T}_{j,\bar{q}}\ =\ \tilde{e}_{\omega_j}+\begin{cases}\tilde{T}_{j,\bar{q}-1} & \text{w.p. } (m-j)\mu_H/\omega_j\\ 0 & \text{w.p. } \lambda_L/\omega_j\\ \tilde{T}_{j,\bar{q}+1} & \text{w.p. } \lambda_H/\omega_j\\ \tilde{T}_{j-1,\bar{q}-1} & \text{w.p. } j\mu_L/\omega_j\end{cases}\qquad\forall\ \bar{q}\ge 1 \tag{93}$$

and

$$\tilde{T}_{0,\bar{q}} = \tilde{e}_{\omega_0} + \begin{cases} \tilde{T}_{0,\bar{q}-1} & \text{w.p. } m\mu_H/\omega_0 \\ 0 & \text{w.p. } \lambda_L/\omega_0 \\ \tilde{T}_{0,\bar{q}+1} & \text{w.p. } \lambda_H/\omega_0 \end{cases} \qquad \forall\ \bar{q} \geq 1. \tag{94}$$

If we let $\gamma_1(j) = \frac{(m-j)\mu_H}{\omega_j+\theta}$, $\gamma_2(j) = \frac{\lambda_H}{\omega_j+\theta}$, $\gamma_3(j) = \frac{\lambda_L}{\omega_j+\theta}$, and $\gamma_4(j) = \frac{j\mu_L}{\omega_j+\theta}$, then the above transition equations result in the following equations for the corresponding Laplace transforms:

$$\begin{aligned} T_{j,\bar{q}} &= \frac{(m-j)\mu_H}{\omega_j+\theta}T_{j,\bar{q}-1} + \frac{\lambda_L}{\omega_j+\theta} + \frac{\lambda_H}{\omega_j+\theta}T_{j,\bar{q}+1} + \frac{j\mu_L}{\omega_j+\theta}T_{j-1,\bar{q}-1} \\ &= \gamma_1(j)T_{j,\bar{q}-1} + \gamma_3(j) + \gamma_2(j)T_{j,\bar{q}+1} + \gamma_4(j)T_{j-1,\bar{q}-1}\ \forall \bar{q} \geq 1, j \geq 1 \end{aligned} \tag{95}$$

and

$$T_{0,\bar{q}} = \frac{m\mu_H}{\omega_0+\theta}T_{0,\bar{q}-1} + \frac{\lambda_L}{\omega_0+\theta} + \frac{\lambda_H}{\omega_0+\theta}T_{0,\bar{q}+1}. \tag{96}$$

Note that we have $\gamma_1(j) + \gamma_2(j) < 1,\ \forall j$. So if we let z_j and z'_j be the two roots of $\gamma_2(j)z^2 - z + \gamma_1(j) = 0$ such that $z_j \leq z'_j$, then we must have $0 < z_j < 1 < z'_j, \forall j$.

We first solve (96). Using the same generating function approach we used to prove Proposition 4, we obtain the following solution:

$$T_{0,\bar{q}} = a_{0,0}z_0^{\bar{q}} + a'_{0,0}z_0^{'\bar{q}} + \frac{\lambda_L}{\lambda_L+\theta}.$$

To complete the first step, we then use induction on j (starting from $j = 0$) to show that the solution to (95) is (92). The induction proof is similar to that in [21] for Lemma 10. Moreover, the proof of steps 2 and 3 are similar to that for Proposition 4; we will not repeat them here. □

As in the case of $\mu_H = \mu_L$, Proposition 5 allows us to reduce the infinite number of linear equations (of the Laplace transforms) to a finite number, and we can solve these linear equations completely. By (7), we obtain the Laplace transform of the inter-overflow time distribution. Even though the Laplace transform may be difficult to invert, we can repeatedly differentiate it to obtain its moments, just as we did in (25) and (27) for the case of $\mu_H = \mu_L$.