*Proceedings of the 2011 Winter Simulation Conference*
*S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.*

# SIMULATION-BASED EVALUATION OF DISPATCHING POLICIES IN SERVICE SYSTEMS

Dipyaman Banerjee
Gargi Dasgupta
Nirmit Desai

IBM Research, India
New Delhi, India

## ABSTRACT

A service system is an organization of the resources and processes, which interacts with the customer and produces service outcomes. Since a majority of the service systems are labor-intensive, the main resources are the service workers. Designing such service systems is nontrivial due to a large number of parameters and variations, but crucial for business decisions such as labor staffing. The most important design point of a service system is how and when service requests are assigned to service workers a.k.a. *dispatching policy*. This paper presents a framework for evaluation of dispatching policies in service systems. A discrete event simulation model of a service system in the data-center management domain is presented. We evaluate four dispatching policies on five real-life service systems. We observe that the simulation-based approach incorporates intricacies of service systems and allows comparative analysis of dispatching policies leading to more accurate decisions on labor staffing.

## 1 INTRODUCTION

Service-based economies and business models have gained significant importance. The clients and service providers exchange value through service interactions and reach service outcomes. Given the focus on the individual customer's value and the uniqueness of the customer's needs, the service providers need to meet a large variety of expectations set by the customers. This is the primary reason for the service delivery to be labor-intensive where human intervention and interaction is unavoidable.

Service providers aim to maintain the quality, even in the face of unique customer expectations, by structuring their service delivery operations as *service systems (SS)*. A SS is an organization of the resources that support and the processes that drive service interactions so that the outcomes meet customer expectations (Alter 2008; Spohrer et al. 2007; Ramaswamy and Banavar 2008). This paper focuses on the SS in the data-center management domain. However, the contributions of this paper extend to all labor-intensive SS.

In the domain of data-center management, the customers own data centers and other IT infrastructures supporting their business. The size, complexity, and uniqueness of the technology installations drive outsourcing of the management responsibilities to specialized service providers. The service providers manage the data-centers from remote locations called *delivery centers* where groups of *service workers (SW)* skilled in specific technology areas support corresponding *service requests (SR)*. In each group, the processes, the people, and the customers that drive the operations of a delivery center constitute a SS. A delivery center is a system of multiple SS.

This paper focuses on the problem of designing the process of routing the SRs to the right technicians (a.k.a. dispatching policy) within a SS such that certain objectives are achieved and constraints are met. What is the basis for evaluating dispatching policies? We propose that the best dispatching policy is the one that: (1) satisfies the contractual constraints to the customer, (2) does not require SWs to work overtime, (3) minimizes the cost of running the SS and (4) minimizes the workload differential among the SWs. Whereas (1) and (3) are obvious, (2) and (4) are challenges unique to the domain of human resource allocation, unlike machine resource allocation as studied widely. While (1) and (2) are constraints

that must be satisfied, (3) and (4) are objectives that maybe at odds against each other. For example, minimizing the cost may lead to sub-minimum differential in the workload of the SWs. The importance of (3) versus (4) may vary depending on the business dynamics. Hence, we aim to measure (3) and (4) independently for each of the dispatching policies we evaluate.

Since (3) and (4) are minimization objectives, a search over the space of *SS configurations* (SSC) is performed. The decision variable that defines the space of configurations is the staff available at predefined shift schedules and the skill levels of the available staff. To evaluate each SSC in terms of (1)—(4), we employ a discrete event simulation model of the steady-state operations of SS (Heching et al. 2010). The experimental results presented in this paper are limited to SS in server support area in the data-center management domain. However, we believe the findings are applicable to other SS domains as well.

The key simulation model parameters include types of SRs and their arrival patterns, service time characteristics for each of the SR types, the contractual service level agreements (SLA) to the customers being supported, the shift schedules, a dispatching policy, staff break policy, and staff swing policies (defined next). SRs are organized in separate queues by their type and complexity and ordered by their priority within each queue. A SW may work primarily on SRs with complexity matching her skill level. The swing policy governs how exceptions to this guideline maybe made when a particular queue grows beyond a certain threshold and more SWs need to work on the queue even when they maybe overqualified given the complexity associated with the queue.

While the problems of optimal resource allocation and scheduling have been greatly studied (Brucker 2006), there are at least three new challenges when the resources are humans and the jobs are SRs.

- The due date $d_i$ of a job $j_i$ must be a scalar in the case of traditional formulations but it is an aggregate in the case of SS. For example, 95% of the SRs by customer X with "urgent" priority must be resolved within 4 hours of reporting. The 95% is computed over a fixed period – typically over a month – instead of being maintained at all times. Hence, within a month it is allowed that only 90% of the requests are resolved within 4 hours in the first week which is then offset by, say, 98% of them being resolved in 4 hours during the rest of the month, achieving the 95% overall. A scheduling formulation with $d_i$ = 4 hours would be over-constrained. There is not a natural way of mapping such aggregate constraints into scheduling problems formulation.
- The SR queues cannot be analyzed independently of each other because the swing policy may be invoked dynamically, moving SWs to growing queues and changing the rate of service for multiple queues in the system. This disables a large body of approaches on queue analysis.
- The processing time of a SR is not only stochastic but also its statistical distribution varies with the skill-level of the SW to whom it is assigned. This is an extension of the stochastic scheduling problems (Buzacott and Shanthikumar 1991) because the SRs may be assigned to SWs with a variety of a skill levels if the swing policy is invoked. Hence, the distribution parameters of the processing time are unknown at the time of job creation.

The above challenges in addition to the random breaks taken by SW and preemption of SRs make analytical modeling of SS a cumbersome exercise. Hence, we turn to simulation as a tool to model the operational characteristics of SS and estimate the values of the objectives (3) and (4) for a given SSC. There have been other comparisons of analytical models and simulation-based models that corroborate our choice of simulation as a tool (Franzese et al. 2009; Heching et al. 2010).

The rest of this paper is organized as follows. Section 2 formally defines SS and describes its properties. Section 3 presents the details of our discrete event simulation model with its various parameters and the optimization approach that is wrapped around simulation. Section 4 defines two main classes of dispatching policies in the context of SS. Section 5 evaluates four dispatching policies from one of the two classes with real data from five SS within IBM and presents the results and observations. Section 6 compares this paper with literature from the call-center and SS domains and Section 7 concludes the paper.

## 2 SERVICE SYSTEMS

Informally, a SS is a configuration of technology and organizational networks designed to deliver services that satisfy the needs of customers. The following is a novel formal definition of SS in general.

**Definition 1** *A service system is a tuple ‹id, C, W, H, T, P, X, γ, τ, α, π, β› where,*
*id is the unique identifier of the SS,*
*C is a set of customers supported by the SS,*
*W is a set of SWs in SS, |W| is the staffing level of SS,*
*H is a set of shifts that run the operations of SS,*
*T is a set of time intervals, each having a unique Poisson arrival rate,*
*P is a set of priority levels,*
*X is a set of complexity levels,*
*γ: ‹$C_i$, $P_j$› → ‹$r_1$, $r_2$›, $r_1$, $r_2 \in \Re$ is a map from each customer – priority pair to a pair of real numbers representing the SLA percentage and resolution time target (in hours), respectively,*
*τ: ‹$P_i$, $X_j$› → ‹$r_1$, $r_2$›, $r_1$, $r_2 \in \Re$ is a map from each priority – complexity pair to a pair of real numbers representing the mean and standard deviation of the Lognormal service time distribution, respectively,*
*α: ‹$C_i$, $T_j$› → $\Re$ is a map from a customer – time interval pair to a real number representing the Poisson inter-arrival time in minutes of SRs from $C_i$ during time interval $T_j$,*
*π: W → H is a map from a SW to the shift in which she is available for work,*
*β: W → X is a map from a SW to the maximum complexity SR that she is skilled to support.*

All sets are non-empty and finite, and all mappings are total. Each element of H describes the start and end times of the shift as well as the days of week on which the shift is on. For example, a shift may be on every Mon – Fri 8am to 5pm and off on the weekends. T allows modeling of variation in arrival rate with time. Typically, T has one element per hour of week. The elements of P are ordered. Typically, we have four levels of priority of SRs. Similarly, the elements of X are also ordered and typically we have three levels of complexity. Contractual SLAs are given by γ for each customer and priority. An example would be $\gamma(\text{customer}_1, P_1) = \langle 95, 4 \rangle$, meaning that 95% of all SRs from $\text{customer}_1$ with priority $P_1$ in a month must be resolved within 4 hours. Service time distributions are characterized by τ. Service time for each priority and complexity combination is a Lognormal distribution characterized by the corresponding mean and standard deviation. Similarly, Poisson arrival distributions are characterized by α. The inter arrival rate varies for each customer and time interval combination. For example, α(customer1, Tuesday13)=140 means that for customer1, between 1pm and 2pm on Tuesday, SRs arrive with inter-arrival time of 140 minutes. Having hourly Poisson arrival rates allows accounting for the natural variations in arrivals based on the customer's business pattern and time zone. Each SW's schedule is given by π. At a point in time, a SR can only be assigned to SWs available in the shift covering the point in time. Finally, each SW's skill level is given by β.

**Definition 2** *A service request is a tuple ‹id, $ss_{id}$, $C_{id}$, $W_{id}$, $P_{id}$, $X_{id}$, $tic_{id}$, $toc_{id}$› where,*
*id is the unique identifier of the service request,*
*$ss_{id}$ is the identifier of the SS to which the service request is assigned,*
*$C_{id}$ is the customer who reported the service request with identifier id, $C_{id} \in C \cup \{\text{None}\}$,*
*$W_{id}$ is the SW who is assigned the service request with identifier id, $W_{id} \in W \cup \{\text{Unassigned}\}$,*
*$P_{id}$ is the priority of the service request with identifier id, $P_{id} \in P$,*
*$X_{id}$ is the complexity of the service request with identifier id, $X_{id} \in X \cup \{\text{All}\}$,*
*$tic_{id}$ is the arrival time of the service request with identifier id,*
*$toc_{id}$ is the completion time of the service request with identifier id.*
.

SRs model all work carried out in the SS. These include not only the requests originated by the customers, but also work internal to SS such as team meetings, emails, HR activity, reporting activity, and anything else that SWs spend their time on. In labor-intensive systems such as SS, this is important because significant time needs to be spent on such overhead activities (see Table 1 in Section 5.1 for evidence). As a result, there maybe SRs that do not originate from a customer and $C_{id}$ in such cases may assume special value None. Also, for such SRs, $X_{id}$ would assume special value All, which means that SWs with any skill level may be assigned such SRs. Note that all SRs are assigned dynamically according to the dispatching policies. Hence, $W_{id}$ may assume the special value Unassigned until the SR is assigned to a SW. In addition to these, there are auxiliary policies of SS worth describing here.

**Swing Policy** Although a highly skilled SW possesses the skills to work on low complexity requests, SWs are typically assigned SRs of the maximum complexity they have skills for, i.e., a SR with $X_j$ is assigned to $W_i$ if $\beta(W_i)=X_j$. A swing policy specifies maximum number of pending SRs with complexity $X_j$. When this maximum is observed, the swing is "invoked" and SWs $W_k$ where $\beta(W_k)=X_m$, maybe assigned SRs with $X_j$ where $X_m > X_j$. In other words, although $W_k$ are overqualified to handle $X_j$ , they may be assigned requests with $X_j$ when swing is invoked.

**Preemption Policy** SRs with higher priority may preempt those with lower priority. The policy is a relation $P_i \Rightarrow P_j$ meaning SRs with priority $P_i$ can preempt those with priority $P_j$. The relation $\Rightarrow$ is anti-reflexive, transitive, and anti-symmetric.

**Breaks Policy** SWs take multiple breaks during shift hours and hence become unavailable. The breaks policy specifies the number of breaks and the total duration of breaks in a shift per SW. It also specifies the maximum priority SRs that can be preempted by the SW taking a break. For example, if the maximum preemptible priority is P3, then the SWs can go for a break even while working on an P3 SR but they may not go for a break while working on a SR with P2 priority (where P2 > P3).

**On-Call Policy** A SR may arrive at a time when there are no shifts scheduled in the SS. At least one SW is designated as "on-call" for all non-shift hours. The on-call policy defines the minimum priority level above which the SRs arriving in non-shift hours would be routed to the "on-call" SW.

**Infrastructure Down Policy** The infrastructure can go down at regular intervals (due to maintenance, outage) and can disrupt the operation of a pool. When the infrastructure is down, SRs continue to arrive but cannot be attended to. This policy specifies the mean-time-between-failures for infrastructure and mean down-time.

## 3 SIMULATION MODEL OF SERVICE SYSTEMS

In this section, we describe the simulation model for SS according to its definition in Section 2 and adapted from prior work by Heching et al. (2010). As our experiments have been conducted with data from SS in server support area in the data-center management domain, we describe the simulation framework for the same. The framework may need minor changes when applied to other areas.

### 3.1 Simulation Model

All SS in server support domain extend the SS of Definition 1 with following specializations.

- T contains one element for each hour of week. Hence, |T| = 168. Each time interval is one hour long.
- $P$={P1, P2, P3, P4}, where, P1 > P2 > P3 > P4.
- $X$={High, Medium, Low}, where, High > Medium > Low.

- **Swing:** Swing is invoked when Low queue length>10. When this happens, one $W_i$ is randomly chosen such that $\beta(W_i)$=Medium and $W_i$ is assigned SRs from Low queue until Low queue length<10.
- **Preemption:** Preemption relation $\Rightarrow$ is the transitive closure of the tuples $P_1 \Rightarrow P_2$, $P_2 \Rightarrow P_3$, $P_2 \Rightarrow P_4$.
- **Breaks:** Each SW takes 3 breaks in a shift independently of other SWs with a total duration of 75 minutes. The maximum priority SRs preemptible by breaks is $P_3$. In other words, breaks are modeled as SRs with priority $P_2$.
- **On-call:** Minimum priority above which SRs arriving out of shift hours routed to on-call SW is $P_2$. Hence, SRs with $P_3$ or $P_4$ would wait till the next shift if they arrive in non-shift hours.
- Infrastructure down: The mean-time-between-failures is 178 hours and the mean down time is 104 minutes (based on historical data) and they are distributed exponentially.

Figure 1 shows the main components of the simulation model. The SRs arrive from multiple customers as well as internal work. Within each hour of week $T_j$, and for customer $C_i$, the arrivals follow a Poisson distribution with inter-arrival averages given by $\alpha(C_i, T_j)$. The function $\alpha$ is learned from historical data of at least 6 months of arrivals for each of the customers as well as internal work.

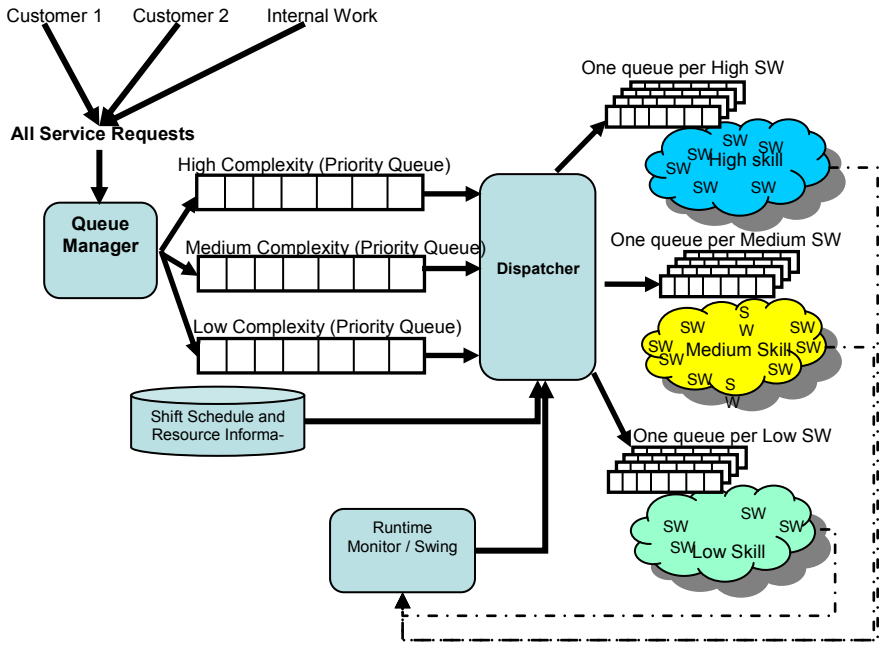


Figure 1: Discrete Event Simulation Model of SS in Server Support Area

As soon as a SR arrives, the queue manager assigns it to the matching complexity queue. The complexity queues are prioritized. The priority associated with SRs in complexity queues may be based on the $P_{id}$ of the SR but may not be identical. Priority of SRs in the complexity queues depends on the dispatching policy adopted. Hence, we call these as *dispatch priority* of SRs to differentiate it from its $P_{id}$. The dispatch priorities are not recorded but computed on-the-fly and hence can change continuously with time.

The SRs move from the complexity queues to individual SWs as soon as they are dispatched to the SW according to the dispatching policy implemented by the dispatcher. The dispatcher may or may not be automated. In general, the dispatcher decides (D1) when to dispatch a SR from the complexity queues to an individual SW, (D2) how to compute the dispatch priority of SRs, and (D3) which SW to dispatch the SR. Each SW has a priority queue associated with her. The SRs in these individual SW queues are prioritized according to their dispatch priority. A SW typically completes working on a SR and takes the next

SR to work on from the head of her queue. The time taken by a SW on a SR follows a Lognormal distribution with mean $r_1$ and standard deviation $r_2$. Distributions are computed for each complexity and each severity of work. The distribution function $\tau(P_{id}, X_{id})$ is learned by conducting time and motion exercises where each SW records the actual "touch time" devoted to each of the SRs. The "touch time" is distinct from the duration for which the SR remains assigned to a SW. This is because the SWs often multi-task, defer a SR and move on to the next one, and go for a break while working on a SR.

If a SR with preemptible priority is being worked on by a SW and it is preempted by a newly dispatched SR, the preemptible SR is inserted back to the SW's queue and its service time is updated to discount the time already spent on it by the SW. The SW starts working on the preempting SR. Whenever a SR is completed by a SW, the completion time $toc_{id}$ is recorded and the SR exits the system.

The runtime monitor collects statistics on the performance of the SS against the SLAs as well as triggers invocation of the swing policy. When the swing is invoked, the dispatcher designates one of the SWs from Medium skill group to work on Low complexity requests until the queue goes down to below 10.

Among other statistics, the output of a simulation run is the actual SLA attainment percentages for each customer and priority pair given by function $\gamma'$: $\langle C_i, P_j \rangle \rightarrow \Re$. For example, if $\gamma'(\text{customer}_1, P_1)=92$ and $\gamma(\text{customer}_1, P_1)= \langle 95, 4 \rangle$ then the actual attainment of 92% is below the SLA target attainment of 95% for $\text{customer}_1$ and $P_1$ combination. $\gamma'$ is computed based on the arrival and completion times of all SRs during simulation.

$$\underset{C_i \in C}{\forall} \underset{P_j \in P}{\forall} \gamma'(c_i, p_j) = \underset{id \in ID}{\forall} 100 \times \sum \frac{(toc_{id} - tic_{id} \leq \gamma(C_{id}, P_{id}).r_2)}{|ID|}$$

For each combination $\langle C_i, P_j \rangle$, ID is the set of identifiers of all SR id observed in a simulation run of one month where $C_{id}=C_i$ and $P_{id}=P_j$. The expression $\gamma(C_{id}, P_{id}).r_2$ denotes the $r_2$ component (target time in hours) of $\gamma(C_{id}, P_{id})$. The numerator counts all instances where the completion happened before the target time and the denominator counts all SRs of the customer-priority pair.

### 3.2 Optimization

A SSC can be optimized with respect to staffing level by varying the set W, shift staffing levels $\pi$ and skills of SWs $\beta$. The rest of the SS components are fixed inputs. The objective of the optimization is to find a SSC such that it minimizes the staffing level |W| while satisfying the SLA constraints $\gamma$. We say that a SSC is *feasible* if (a) it satisfies the SLA constraints $\gamma$, i.e., $\underset{C_i \in C}{\forall} \underset{P_j \in P}{\forall} \gamma'(c_i, p_j).r_1 \geq \gamma(c_i, p_j).r_1$ and (b) the complexity queues are not growing unbounded. The (b) part is significant because $\gamma'$ is based only on SRs that exit the SS during the simulation period. Because analytical formulation of this optimization problem is not possible (as discussed in Section 1), optimization is implemented as an iterative heuristic search. We use OptQuest optimization toolkit (Laguna (1998)) packaged within AnyLogic (Anylogic Tutorial (2008)). In each iteration, the following steps are performed.

1. Set $W_{min} = \infty$, $SSC_{best}$=NIL
2. Choose a SSC to evaluate.
3. If $|W| > W_{min}$, reject the SCC and go back to Step 2. For every SR complexity $X_i$ present in the workload, ensure there exists at least one SW $W_j$ such that $\beta(W_j) = X_i$ . If not, reject the SSC and go back to Step 2.
4. The chosen SSC is evaluated via simulation. Simulation consists of 100 replications of 7 days of warm up followed by 30 days simulation runs. At the end of each replication $\gamma''$ is computed and stored. At the end of all replications, $\gamma'$ is computed as an average of all $\gamma''$. The replications help counter anomalies caused by stochastic variations, especially when the arrivals are low.
5. If the chosen SSC is feasible based on $\gamma'$, continue to Step 6. Else, go back to Step 2.
6. Set $W_{min} = |W|$, $SSC_{best}$=SSC. Go back to Step 2.

$SSC_{best}$ is the optimal configuration of the given SS with corresponding staffing of $W_{min}$. In our experiments with multiple SS, we observed that after 5000 iterations, no better $SSC_{best}$ are found. Hence, we perform 5000 iterations for experiments described in Section 5.

## 4 DISPATCHING POLICIES

Deciding on D1, D2, and D3 introduced in Section 3.1 defines a dispatching policy. In this paper, we consider two main classes of dispatching policies defined in the following.

### 4.1 Pull-based dispatch

These are distributed dispatching policies in that the individual SW choose the highest priority SR to work on. Hence, (D1) the SR with the highest dispatch priority is dispatched from the complexity queues to individual queues as soon as one of the SW becomes available and (D3) the SR is assigned to the first available SW with a matching skill level. Note that as a result of D1 and D3, the individual SW queues can have at most one SR in them which is the SR that the SW is currently working on. The rest of the SRs remain in the complexity queue. On D2, there are several ways in which the dispatch priority may be computed. Each of the ways completes the definition of a dispatching policy.

1. Earliest deadline first (EDF-PULL): Dispatch priority is inversely proportional to time left to SLA target deadline, i.e., $[(tic_{id} + \gamma(C_{id}, P_{id}).r_2) - time]$, where, *time* is the current time.
2. Least margin first A (LMF-A-PULL): Dispatch priority is inversely proportional to the margin left to SLA target deadline considering the mean service time, i.e., $[(tic_{id} + \gamma(C_{id}, P_{id}).r_2) - \tau(P_{id}, X_{id}).r_1 - time]$, where, *time* is the current time and expression $\tau(P_{id}, X_{id}).r_1$ denotes the $r_1$ (mean) of $\tau$.
3. Least margin first B (LMF-B-PULL): Dispatch priority is inversely proportional to the margin to SLA target deadline considering the mean and standard deviation of service time, i.e., $[(tic_{id} + \gamma(C_{id}, P_{id}).r_2) - (\tau(P_{id}, X_{id}).r_1 + 3 * \tau(P_{id}, X_{id}).r_2) - time]$, where, *time* is the current time, expression $\tau(P_{id}, X_{id}).r_1$ denotes the $r_1$ (mean) of $\tau$, and expression $\tau(P_{id}, X_{id}).r_2$ denotes the $r_2$ (standard deviation) of $\tau$.
4. Priority-Based (PRIO-PULL): Dispatch priority is the same as priority.

We consider both LMF-A-PULL and LMF-B-PULL because without experiments, it is not clear which of these would result in a better estimation of the expected service time. Note that we do not treat a sample from the Lognormal service time distribution for $(P_{id}, X_{id})$ as a good estimate because the difference between two consecutive random samples (the first used as an estimate and the second as the actual service time) represents estimation error and it could be quite large.

### 4.2 Push-based dispatch

These are centralized dispatching policies in that the dispatcher is a central entity that assigns SR to SW queues actively. Hence, (D1) SRs are dispatched from the complexity queue to individual SW queues as soon as they arrive. Due to D1, the complexity queues can have at most one SR that is being dispatched. On D2, the dispatch priority can be determined in ways similar to the pull-based policies. D3 depends on various scheduling objectives as described below.

1. Balance queue length (BAL-Q): Dispatch to the SW with the least number of tickets in her queue.
2. Balance workload (BAL-LOAD): Dispatch to the SW with the least amount of work in her queue computed as the sum of mean service times of queued SRs, i.e., for all *id* in SW queue, $\Sigma\ \tau(P_{id}, X_{id}).r_1$.
3. Minimize max lateness (MIN-LATE): Dispatch to the SW such that it minimizes the maximum *lateness* in SS. The maximum lateness is the max of the difference between expected completion time and target time for all tickets in SS, i.e., for all *id*, $max[(\tau(P_{id}, X_{id}).r_1 + time) - (tic_{id} + \gamma(C_{id}, P_{id}).r_2)]$.
4. Minimize max tardiness (MIN-TARD): Dispatch to the SW such that it minimizes the maximum tardiness in the system. The maximum tardiness is the max of the difference between expected comple-

tion time and target time for tickets that are expected to complete after their targets after this dispatch, i.e., for all *id* completing after their target time, $max[(\tau(P_{id}, X_{id}).r_1 + time) - (tic_{id} + \gamma(C_{id}, P_{id}).r_2)]$.

5. Minimize number of misses (MIN-MISS): Dispatch to the SW such that it minimizes number of tickets that would miss their targets in the SS, i.e., $min\ \Sigma\ [(\tau(P_{id}, X_{id}).r_1 + time) > (tic_{id} + \gamma(C_{id}, P_{id}).r_2)]$.
6. Balance utilization (BAL-UTIL): Dispatch to the SW having least utilization so far.

So far we have assumed that once a SR is assigned to a SW it cannot be moved. However, in certain situations, the above mentioned push based schemes may make use of movement of tickets among SW queues to further increase efficiency.

## 5 EVALUATION OF DISPATCHING POLICIES

In this section, we describe experimental evaluation of the various pull-based dispatching policies based on data from five real SS from the server support area and present our observations. We employ the AnyLogic simulation toolkit with OptQuest optimization engine for the experiments. As each optimization run of 5000 iterations took several days to complete on a standard PC, we could not complete our experiments on the push-based policies. We hope to have them completed at the time of the conference.

### 5.1 Experiments and Data

We choose five real-life SS from two different countries providing server support to IBM's customers. Collectively, the five SS cover a variety of characteristics such as high vs. low workload, small vs. large number of customers being supported, small vs. big in size in terms of the staffing level, and stringent vs. lenient SLA constraints. We collect the following data from each of the SS to allow us to construct the various sets and mappings of SS.

- Demographic data describes the static entities and constraints of the SS and is the basis for constructing the sets C, W, and H, and mappings $\gamma$, $\pi$, and $\beta$.
- Workload data is the historical data on SR arrivals and is the basis for constructing the mapping $\alpha$.
- Effort data is the output of time and motion study that captures the "touch time" data and is the basis for constructing the mapping $\tau$.

Tables 1 and 2 show summary of the workload and effort data of the five SS, respectively. Columns 2-4 in Table1 shows the Customer work (C SR) and the Internal work (I SR) both in terms of weekly volume and hours worked per SW, per day.

Table 1: Customer SR (C SR), Internal SR (I SR) and Total work volume statistics

| | C SR Per Week | I SR Per Week | C SR Work / SW / Day (Hrs) | I SR Work / SW / Day (Hrs) | Total Work / SW / Day (Hrs) |
|---|---|---|---|---|---|
| **SS1** | 2127.0 | 983.5 | 3.8 | 2.7 | 6.5 |
| **SS2** | 1752.0 | 961.0 | 5.3 | 7.0 | 3.0 |
| **SS3** | 544.6 | 773.7 | 2.7 | 5.8 | 8.5 |
| **SS4** | 111.5 | 750.3 | 0.7 | 4.9 | 5.6 |
| **SS5** | 1932.5 | 973.7 | 7.6 | 5.8 | 13.4 |

Table 2: Estimated service times mapping $\tau$ (all numbers in minutes), for a SS

| Service Time ($\tau$) | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| **Low Count** | 6 | 38 | 249 | 37 |
| **Low Avg** | 93.8 | 52.1 | 64.8 | 78.6 |
| **Low Stddev** | 126.6 | 59.8 | 62.1 | 43.6 |
| **Medium Count** | 5 | 79 | 198 | 17 |
| **Medium Avg** | 179.5 | 53.4 | 70.5 | 44.9 |
| **Medium Stddev** | 81.6 | 42.8 | 63.4 | 41.8 |
| **High Count** | 2 | 11 | 33 | 11 |
| **High Avg** | 76.9 | 43.6 | 80.3 | 73.3 |
| **High Stdev** | 17.4 | 13.7 | 63.9 | 95.9 |

As highlighted in Section 1, we have a dual basis for evaluating dispatching policies. Hence, we compute the following two metrics during each optimization run of an SS with each of the four pull-based dispatching policies.

- Minimum $W_{min}$ with which the SS can be run. Lower $W_{min}$ is better.
- Fairness to the SWs computed as the difference between the maximum utilization and the minimum utilization of SWs. Utilization for a SW is the percentage of the simulation time worked by the SW. We propose *fairness index (FI)* as a measure of fairness, computed as the weighted sum of the difference in the maximum and minimum utilization of SW at each skill level normalized by the staffing level of SSC. Lower fairness index is better. For a $SSC_{best}$ achieved by a dispatching policy, if $U_j^{max}$ denotes the maximum utilization of a SW at skill level $X_j$, $U_j^{min}$ denotes the minimum utilization of a SW at skill level $X_j$, and $|W^j|$ denotes the number of SW at skill level $X_j$, then

$$FI = \frac{\sum_j \left( (U_j^{max} - U_j^{min}) \times |W^j| \right)}{|W|}$$

### 5.2 Results and Observations

Figure 2 shows the $W_{min}$ as achieved by the pull-based dispatching policies on each of the five SS. It is clear that the PRIO-PULL fares much worse than the other three approaches. This is due to the fact that in PRIO-PULL there are only four priority levels and hence the highest priority SRs of multiple customers collide in the priority queue. SRs with the same priority maybe ordered randomly. However, the SLA target times vary by account even for the same priority.

**Observation 1** *In PRIO-PULL, priority inversion may occur when an SR from an account having lenient SLA is put in front of another SR with the same priority but a different account having stringent SLA.*

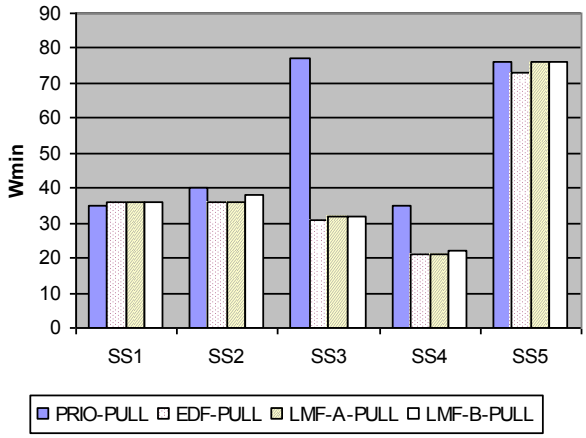


Figure 2. $W_{min}$ achieved by the pull-based dispatching policies on five real SS

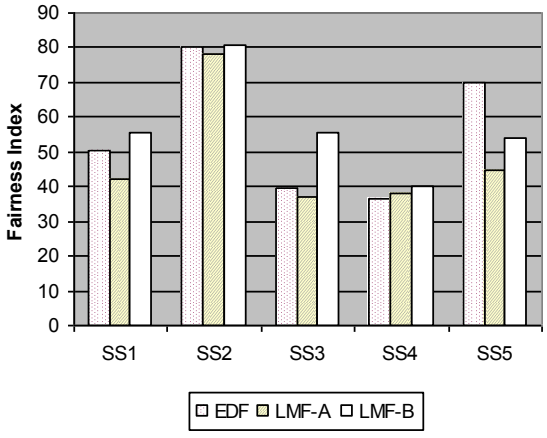


Figure 3. Fairness Index of the three pull-based dispatching policies on three SS

We then investigate EDF, LMF-A and LMF-B. The three recommend staffing that are very close to each other with EDF often being the best. It is slightly surprising that EDF, which considers only SLA deadline performs better than LMF-A and LMF-B, which consider the margin left to the SLA deadline. The margin-based approaches ought to work better because an SR having deadline in 2 hours with pending work of 1 hour should have a lower dispatch priority than a SR having deadline in 3 hours with pend-

ing work of 2.5 hours. Whereas in call-centre domain, policies based on service times have proved to be very effective, in the SS domain, the service time distributions usually show large variance as evident from Table 2. The margin-based approaches depend on a good estimation of the expected service time (pending work). Large variance in service times causes the margin-based policies to degenerate to EDF.

**Observation 2** *When the service times show large variance, simpler approaches like EDF work well. When service times can be accurately predicted, the margin-based approaches works well.*

Next, we study the fairness index achieved by each dispatching policy for the same five SS. Since PRIO fares worse in the $W_{min}$ comparison, we do not include it in the fairness analysis. Assuming a shift of 9 hours, an SS having SWs working close to 7 hours (considering 2 hours of breaks) can be called heavily loaded SS. From Table 2, last column, SS1, SS3, and SS5 are heavily loaded whereas SS2 and SS4 are lightly loaded. Figure 3 shows the FI of all five SS by the three dispatching policies. Overall, LMF-A achieves the lowest FI. This is because in LMF-A, a SW always chooses a SR with the least margin. Because the margin takes the mean service time into account, the chosen SR may end up with varying actual service times, balancing out the utilizations across SW. In EDF, a SW may end up choosing more SR having larger actual service times than other SWs. In LMF-B, the margin depends also on the standard deviation that are large. As a result, same situation as EDF arises. Also, these differences are emphasized only in the heavily loaded SS. In such SS, commonly, there are SRs in the complexity queues waiting to be picked up not allowing SWs to be idle. Being idle would have helped balance out the utilizations.

**Observation 3** *When a SS is heavily loaded, mean service time-based policies can provide better fairness. When a SS is lightly loaded, all policies dispatch fairly.*

## 6 LITERATURE REVIEW

There exists a large body of work in the scheduling and resource allocation area. However, as discussed in Section 1, there are at least three new challenges in the domain of SS. These challenges render unsuitable most of the traditional scheduling and resource allocation literature. We present the remaining literature in the following logically grouped areas.

### 6.1 Call-Center Domain

Several approaches consider optimal allocation of agents to calls in the call-center domain. Robbins and Harrison (2008) (R&H) consider the problem of determining optimal staff schedules in a call-center scenario. They evaluate the system given a staffing level with an analytical model, which is possible in their simplified domain. They apply simulation to search for the optimal staffing level based on a heuristic. Iravani, Kolfal, and van Oyen (2007) (IK&O) apply simulation to evaluate cross-training patterns among call-center agents. Unlike R&H, they associate calls to underlying call types. An agent needs to be trained for a call type to service a call of the type. While this makes IK&O's domain similar to ours, their aim is to evaluate cross-training patterns against the metric of average waiting times of customers. Cross-training patterns help guide the training decisions for agents which is independent of work-assignments and thus are orthogonal to the dispatching policies.

In general, the call-center domain too presents the challenges of aggregate contractual SLAs and random arrival rates like the SS domain. While these are important similarities, SS present additional significant challenges. First, the staff in R&H's work is not organized by skill levels and the calls are not associated with complexity. Second, there is no preemption in the call-center domain. If a call is being answered by an agent, it must be completed before the agent takes another call. Also, there are no swing policies and an analytical model of the system is possible as demonstrated in R&H's paper. Third, unlike the "touch time" data, accurate call data is easily available through the underlying telecom infrastructure. AlCall durations can be readily used for learning the parameters of service time distributions. This is in

contrast to the SS domain where the SWs multi-task through multiple activities. Activities are put on hold in the interest of other activities. Hence, estimating the service time distributions is difficult. These factors contribute to the complexity of the SS domain. Next, we discuss literature applicable to the SS domain.

### 6.2 Service Systems Domain

Verma et. al. (2011) address the problem of dispatching of SRs within service systems. They propose a two-step mixed-integer program formulation with objectives of minimizing invocations of swing in the first step and minimizing the workload differential in the second step. While their objective is similar, their formulation does not model the stochastic variations of arrivals or processing times. Further, the SLA constraints in their formulation cannot be aggregates. Lastly, they consider only the BAL-LOAD dispatching policy. Wasserkrug et. al. (2008) propose a methodology for shift scheduling in the domain of third-level IT support which is very similar to the domain of SS. Of the three steps of their method, we focus only in the second step of "calculating staffing requirements" but with the aim of evaluating dispatching policies. The only dispatching policy considered in their work is PRIO-PULL. Unlike this paper, they do not validate their method against data from real-life third-level IT support. Chenthamarakshan et. al. (2010) propose an approach for assigning SWs to SRs. However, their domain is application development support rather than data-center management. Hence, the problem being addressed is closer to the one of hiring skilled SWs for known project needs than the dynamic dispatching of SRs as considered in this paper. Brickner et. al. (2010) apply ARENA simulation tool to analyze service systems. Unlike in our model, the system is not subjected to aggregate SLA constraints. Also, there is no preemption or swing policies in their model. Their model is slightly different than ours because they assume a pool of dedicated SWs for each customer as well as a separate pool of shared SWs. Nonetheless, they do not aim to evaluate dispatching policies either. Chan (2008) applies an agent-based simulation technique to study the emergent behavior of a service system consisting of a large number of cells. Each cell contains an analytical M/M/1 queue model. The simulation helps observe how cells die and neighborhood patterns emerge among cells. While Chan's work exemplifies the human aspects of service systems – which would be an important future work for this paper – it does not aim to evaluate dispatching policies.

In general, the current literature does not provide a decision support framework for choosing the right dispatching policy for a given SS. Also, these approaches do not seem to recognize the importance of simultaneously minimizing the fairness index while minimizing the cost. Lastly, few of the above approaches have been validated with as rigorous experiments as in this paper.

## 7 CONCLUSIONS

SS are of tremendous socio-economic importance. Studies of the various aspects of SS are beginning to emerge. From a holistic perspective, we formally defined SS and presented a discrete event simulation model of SS in the domain of server support. The simulation model wrapped within an optimization approach searches the optimal staffing for a given SS. We defined the three main questions that dispatching policies should answer and introduced two classes of dispatching policies applicable to the SS domain. Various dispatching policies can be plugged into the simulation model to enable a comparative analysis. We showed the results based on the data from five real-life SS and four pull-based dispatching policies. An analysis of the results lead us to three important observations. Our simulation framework can be readily used across a wide array of domains of SS to achieve labor efficiencies while preserving the workload balance among service workers.

While the SSC in this paper were obtained by varying staffing parameters, we can vary the SLA parameters to understand the impact of a stringent or a lenient SLA on required staffing. Such an analysis can inform the contractual agreements between customers and service providers. Similarly, we can vary the shift timings to find the best shift schedules for an SS. More importantly, what customers should be supported by what SS in a delivery center? Optimal allocation of customers to SS can be achieved by si-

mulating all SS of a delivery center together and varying customer to SS allocations. Significant future work opportunities such as the above lie ahead.

## ACKNOWLEDGMENTS

We express our sincere thanks and gratitude to Yixin Diao, Brian Eck, Aliza Heching, and David Northcutt of the IBM simulation team for their extended support and long discussions.

## REFERENCES

Alter, S. 2008. "Service System Fundamentals: Work System, Value Chain, and Life Cycle", *IBM Systems Journal* 47 (1): 71 – 85.

Anylogic Tutorial 2008. How to build a combined agent based/system dynamics model in Anylogic. http://www.xjtek.com/anylogic/articles/13/. *System Dynamics Conference*.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2000. *Discrete-Event System Simulation*. 3rd ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Brickner, C., D. Indrawan, D. Williams, S. R. Chakravarthy. 2010. "Simulation of a Stochastic Model for a Service System." In *Proceedings of the 2010 Winter Simulation Conference,* Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 1636 – 1647. Piscataway, NJ: Institute of Electrical and Electronic Engineers Inc.

Brucker, P. 2006. *Scheduling Algorithms*. 5th ed. Heidelberg: Springer.

Buzacott, J.A., and J.G. Shanthikumar. 1991. *Stochastic Models of Manufacturing Systems*. New Jersey: Prentice Hall.

Chan, W. K. 2008. "An Analysis of Emerging Behaviors in Large-Scale Queueing-Based Service Systems Using Agent-based Simulation." In *Proceedings of the 2008 Winter Simulation Conference*, Edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 872 – 878. Piscataway, NJ: Institute of Electrical and Electronic Engineers Inc.

Chenthamarakshan, V., K. Dixit, M. Gattani, M. Goyal, P. Gupta, N. Kambhatla, R. M. Lotlikar, D. Majumdar, G. R. Parija, S. Roy, S. Soni, and K. Visweswariah. 2010. "Effective decision support systems for workforce deployment." *IBM Journal of Research and Development* 54(6):1 – 15.

Franzese, L. A., M. M. Fioroni, P. J. de Freitas Filho, and R. C. Botter. 2009. "Comparison of Call Center Models." In *Proceedings of the 2009 Winter Simulation Conference,* Edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 2963 – 2970. Piscataway, NJ: Institute of Electrical and Electronic Engineers Inc.

Heching, A. R., D. Banerjee, G. Dasgupta, N. Desai, and Y. Diao. 2010. "Simulation Design to Improve Design of IT Service Delivery" In *Proceedings of INFORMS*. Linthicum, MD

Iravani, S. M. R., B. Kolfal, and M. P. Van Oyen. 2007. "Call-Center Labor Cross-Training: It's a Small World After All." *Management Science* 53 (7): 1102 – 1112.

Laguna, M. 1998. Optimization of complex systems with optquest. OptQuest for Crystal Ball User Manual, *Decisioneering.*

Ramaswamy, L., and G. Banavar 2008. "A Formal Model of Service Delivery." In *Proceedings of the 2008 IEEE International Conference on Service Computing*, 517 – 520.

Robbins, T. R., and T. P. Harrison. 2008. "A simulation based scheduling model for call centers with uncertain arrival rates." In *Proceedings of the 2008 Winter Simulation Conference,* Edited by S. Mason, R. Hill, L. Monch, and O. Rose, 2884 – 2890. Piscataway, NJ: Institute of Electrical and Electronic Engineers Inc.

Spohrer, J., P. P. Maglio, J. Bailey, and D. Gruhl. 2007. "Steps Toward a Science of Service Systems." *IEEE Computer*, 40 (1): 71 – 77.

Verma, A., N. Desai, A. Bhamidipaty, A. N. Jain, S. Barnes, J. Nallacherry, S. Roy. 2011. "Automated Optimal Dispatching of Service Requests." In *Proceedings of the SRII Global Conference*, 426 – 429.

Wasserkrug, S., S. Taub, S. Zeltyn, D. Gilat, V. Lipets, Z. Feldman, and A. Mandelbaum. 2008. "Creating operational shift schedules for third-level IT support: challenges, models and case study." *International Journal of Services Operations and Informatics* 3(3/4): 242 – 257.

## AUTHOR BIOGRAPHIES

**DIPYAMAN BANERJEE** is a Researcher at IBM Research – India. He works in the area of service analysis and optimization. He holds an M.S. degree in Computer Science from University of Tulsa, Oklahoma. His research interest include distributed systems, machine learning, and game theory. His email address is dipyaban@in.ibm.com.

**GARGI B. DASGUPTA** is a Researcher at IBM Research – India. She holds a Ph.D. degree in Computer Science from University of Maryland, Baltimore County. Her current research focus include IT service systems, analysis and optimization for large-scale distributed systems and power and energy management solutions for server and storage infra-structures. She is also interested in areas of cloud computing and networking. Her e-mail is gaargidasgupta@in.ibm.com.

**NIRMIT DESAI** is a Researcher at IBM Research – India. He received his Ph.D. in Computer Science from North Carolina State University, Raleigh. His research interests are next generation service systems, service processes, and formal aspects of service description and analysis. His e-mail address is nirmit.desai@in.ibm.com.