RC23094 (W0401-145) January 30, 2004
Mathematics

IBM Research Report

Optimal Control of Web Hosting Systems under Service Level Agreements

Alan J. King, Mark S. Squillante
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

Optimal Control of Web Hosting Systems under Service Level Agreements

Alan J. King and Mark S. Squillante

October 16, 2003

1 Introduction

The operation of a web-hosting facility involves control elements and decisions spanning many time scales. The physical computing facility contains many control points with parameters that can be tuned in real time to respond to performance statistics. At one extreme, high-performance routers operate at very fine time-scales and adjust their parameters accordingly. Operating system, middleware and application software may also monitor performance and set control parameters. At somewhat coarser time-scales, computer and disk farm partitions may be reassigned in response to changing workloads. Daily, weekly, monthly and longer-term forecasts may be used to schedule and plan allocation policies. At weekly and monthly time-scales, the capacity of the resource elements may be reviewed and cause the utility's supply chain model to process additional orders or to obtain short-term capacity. Finally, monthly and yearly performance reporting may require changes in the basic terms and conditions of the Service Level Agreement (SLA) and impact strategic models addressing the computing utility's profitability. The key operational point is that the overall solution must provide a unified framework that makes it possible to have various solution methods working together in a cohesive manner across a wide range of time scales in order to meet a common overall goal.

The physical computing facility includes control points at the router and server. Policies at these control points can be used to achieve QoS performance objectives through the allocation of resource elements, once information about the arrival and service processes are known or forecasted. The availability of such workload information is often significantly impacted by the time-scale of the control points. As a specific example, routers working at very fine time-scales often do not have any additional information about the arrival process beyond its current mean rate because the overhead of collecting more detailed information is too expensive at these fine time-scales. More detailed workload information is typically available as one moves to coarser time scales.

The varying assignment of resource elements over time is perhaps what most comes to mind when operational models of the computing utility are discussed and marketed: Additional resource elements can be brought to accommodate situations where customer user populations create bursts of demand. Different architectures can respond to such requirements with varying abilities, from hot repartitioning of a single large mainframe class machine to a cold restart of a rack mounted pizza-box. Reassigned resource elements are necessarily diverted from actual or potential use by other customers, so the operational models for these decisions typically must encompass a large number of potential actions and rewards.

1.1 Allocation of shared resources

The generic scenario is as follows. A dispatcher receives incoming client requests destined for different customer hosting sites and routes these requests to an appropriate downstream server element. Each server element in turn receives the client requests routed to it and schedules the execution of these requests among its resources. Both dispatchers and servers operate according to their current control policy parameters. The dispatcher observes incoming loads from the different customer hosting sites and performance statistics for the downstream server elements. Server elements are assigned to individual customer sites and often are not shared, either physically or virtually. At a relatively fine time-scale, the dispatcher's role can include deciding how to adjust its control policy parameters in an optimal, or near optimal, manner subject to the existing server element assignments. Similar control policy decisions can also be made at the server elements. At coarser time-scales, the role of resource allocation control policies includes deciding when to change the assignment of server elements among the different customer hosting sites. These decisions must encompass the load forecast, the system state, alternative control policies, the charging procedures, and the SLAs for the different customers.

One version of this generic scenario is the Bandwidth Broker in Section 2. Classes of customers differ in the price charged per "megabyte" carried, in the arrival rate of requests, and in the size of the request. The provisioning dispatcher, or bandwidth broker, can accept or reject the bandwidth request in order to optimize revenue subject to a capacity bound on available bandwidth. The feasibility bound on available bandwidth is a proxy for a QoS condition, in that as long as the bandwidth bound is respected then all users experience the desired QoS. This setting is similar to [35] and the dynamic programming solution to this problem is similar to the classical revenue management models [24, 31]. The main results from this analysis are: 1) different charging rates do indeed differentiate service levels achieved by customers; and 2) greedy allocation policies can be near-optimal in states characterized by excess capacity and states characterized by large numbers of requests in service.

Section 3 considers an operational model of the web hosting facility at fine time-scales that performs optimal downstream routing and server scheduling decisions for requests arriving from multiple classes of users for each of the many customer sites. The workload model encompasses state transitions for the user classes and allows general stochastic processes of the type found in practice. Revenue is calculated by charging users a price per request as in Section 2. The QoS conditions and profit estimates are modeled by the probability that the per-class response times are within a specified per-class bound, which is in turn bounded from above by an exact or approximate queueing-theoretic expression. In addition, requests that violate the QoS performance guarantee cause a penalty to be charged to the provider. Versions for various types of server scheduling policies among the user classes are also considered. The resulting class-constrained resource allocation problem is shown to be convex and an efficient solution algorithm is developed. The main results from this analysis demonstrate: 1) the viability and profitability of the general class of SLA and revenue/penalty charging models considered; 2) the effective use of stochastic bounds and approximations on per-class response time distributions given limited workload information; and 3) the efficiency of the solution methods making it possible to exploit them on-line.

Section 4 expands the set of resource allocation problems in the web hosting facility of Section 3 to span multiple time-scales. The SLA model is simplified by eliminating penalties for QoS noncompliance and by calculating the per-class revenues as functions of the per-class average server utilization rates per time interval. To meet QoS requirements, the resource allocation policies can assign more servers to the customer. (The charging scheme protects the customer from paying for over-allocations.) However, when server elements are reassigned, each such change takes the server out of service for an interval that reflects the transition time. The resulting resource assignment problem is a stochastic dynamic program. The solution approach takes advantage of the fact that this simplified revenue model causes the optimal routing and scheduling model of Section 3 to reduce to a linear program that has a direct solution. An approximation algorithm then provides resource assignment solutions demonstrating that the dynamic programming policies are far better than standard heuristics. Conclusions somewhat similar to Section 2 can be drawn, namely that greedy algorithms work well in states characterized by low utilization and that the dynamic programming solution significantly differs from greedy policies when the QoS constraints are at risk of violation — especially when the request processes are characterized by a considerable probability of bursting. In particular, as a result of employing the optimal routing and scheduling policies of Section 3, we find that the resource reallocation problem simplifies somewhat in the sense that small changes in the various workloads have a relatively small impact on overall profit, whereas adjusting server assignments to address major workload changes becomes a primary issue in satisfying QoS performance guarantees and maximizing overall profit.

The discussion so far describes resource allocation control policies that make operational decisions to meet QoS requirements. In Sections 3 and 4 the control policies have constraints on the per class response time QoS. Each of these settings assumes that a feasible solution exists and further assumes that the dispatcher pays penalties for out-of-compliance service — in other words, there must be a governing provision that user loads are within certain limits and that the service provider is protected from paying penalties for load levels above these limits. Moreover, when such limits are violated, the resource allocation control policies favor requests according to their importance relative to their QoS requirements and cost model.

1.2 Preface and Acknowledgements

This chapter is a survey of a few years of work and discussions among our colleagues in IBM Research. Many of these sections are based on material that has been published in the academic literature. Section 2 is adapted from the IBM Research report by Yuan-Chi Chang, Xin Guo, Tracy Kimbrel and Alan King (IBM Thomas J. Watson Research Center) that was never published [7]; the authors acknowledge Dr. Shabbir Ahmed (Georgia Institute of Technology) and Dr. David Gamarnik (IBM Thomas J. Watson Research Center) for valuable comments, and thank Menghui Cao (Columbia University) and Daniela de Farias (Stanford University) for their help in building the simulator and analyzing its runs during their summer internship at IBM. Section 3 is adapted from the paper by Zhen Liu, Mark Squillante and Joel Wolf (IBM Thomas J. Watson Research Center) that appeared in the proceedings of the IEEE Conference on Decision and Control [28]. Section 4 is adapted from the paper by Daniela de Farias (IBM Almaden Research Center), Alan King, Mark Squillante (IBM Thomas J. Watson Research Center) and Benjamin Van Roy (Stanford University) that appeared in the proceedings of the INFORMS Revenue Management Conference [8].

2 Dynamic Single-Resource Provisioning

We formulate a revenue-based single-resource allocation problem for multiple service classes. To fix ideas, the resource under management is the bandwidth on the link from the hosting facility network access router to the internet POP, although our methods and results can be applied more generally. Classes i are distinguished by the bandwidth request A_i and the rate R_i they are willing to pay for the request. The system can decide to allocate all of the bandwidth or to reject the service request. There is no best-effort service class.

The optimal policy is achieved by discrete-time dynamic programming. Theoretical and simulation results based on actual HTTP trace logs concur with conventional wisdom that as resource usage approaches 100%, headroom should be reserved for premium users. A greedy policy works well at low resource usage and in states when there are a large number of departures relative to arrivals, which suggests that one may simplify the implementation of dynamic programming policies by classifying states and applying class-based rules.

We also discuss the optimal off-line solution and report the revenue gap between online and off-line algorithms. Performance numbers from the off-line algorithm are not attainable in practice but they provide the ceiling on those of the online algorithm. We found that the gap narrows as the service capacity increases.

The rest of this section is organized as follows. §2.1 develops the discrete-time dynamic programming framework for bandwidth allocation. §2.2 describes approximation algorithms in the off-line setting. The simulation results are presented in §2.3 and implementation issues are discussed in §2.4. Finally, §2.5 contains directions for further research.

2.1 Formal framework

We develop a discrete-time dynamic programming formulation for allocating bandwidth to requests from multiple service classes. The problem is formulated as a discrete-time Markov decision process on a finite horizon. The state variable is the number of requests in progress in the system.

Consider the system with known capacity SLA, a set of m service classes, and a finite time T when the system returns to a renewal state. Let t denote the remaining time to T (time runs backwards in this section) and assume that allocation decisions are made at discrete time intervals of length 1. Thus, there are $T+1$ unit length time intervals. Decision stages are numbered in reverse order $t = T, T-1, \ldots, 0$ indicating the time remaining to renewal. Users arrive at random times $X_1, X_2, \ldots$. The i-th user stays connected for a random time D_i. The goal is to come up with the assignment policy which maximizes the total revenue over the time period T.

Notation. For any fixed time t (the remaining time to renewal), define:

- $\vec{N} = (n_1, n_2, \cdots, n_m)$ and $\vec{L} = (l_1, l_2, \cdots, l_m)$ are vectors denoting the numbers of requests in progress, each requiring bandwidth $\vec{A} = (A_1, A_2, \ldots, A_m)$ for the duration of the session. The SLA constraint mandates $\sum_i^m A_i n_i \le SLA$.
- $\vec{b} = (b_1, b_2, \cdots, b_m)$ is the vector denoting the number of different classes leaving the system; $\vec{b} > 0 \iff b_i > 0$ for some $i = 1, ...m$;
- $\vec{N} \ge \vec{b} (\vec{N} \ge \vec{L}) \iff n_i \ge b_i \ (n_i \ge L_i)$, for all $i = 1, \cdots, m$;
- p_{it} denotes the probability of a request from a service class i arriving during the time interval $[t, t-1)$;
- $q_{\vec{b}t}$ denotes the probability that $\vec{b}$ customers depart the system at time t;
- R_i denotes the revenue per unit time for the allocated bandwidth for class i.

General assumptions.

1. In each interval, up to one new request will be considered for service (one may easily relax this (apparently) restrictive assumption by aggregating thousands of requests to be a single "mega-request", and apply our formulation to this "mega-request").

2. Once a request arrives, a known bandwidth A_i $(i = 1, ...m)$ needs to be allocated.

3. The assignment of bandwidths occurs at the starting point of each time unit.

4. Instead of a best-effort allocation policy, an "all-or-nothing" policy is assumed that allocates to accepted requests the entire bandwidth A_i. R_i is the resulting revenue per request. To accommodate a best effort default service, one may view R_i as the value differential between an assigned bandwidth and a best effort class of service.

5. Unserved requests are lost without further impact on the system.

Request durations. It is not necessary to model individual request durations. One can derive the probability (q_{it}) of a service request from class i leaving the system at t via the arrival and service time assumptions.

Optimality equation. Consider the following action space $A = \{$ accept, reject $\}$. When the system is at t with $\vec{N}$ requests in progress, a reward $R(\vec{N}, a)$ is given for action $a \in A$. The corresponding optimality equation is:

$$V_t(\vec{N}) = \max_a \left\{ R(\vec{N}, a) + \sum_{\vec{L}} V_{t-1}(\vec{L}) P_{\vec{N},\vec{L}}(a) \right\} \tag{1}$$

subject to the SLA constraint and the boundary condition

$$V_0(0) = 0, \qquad V_0(\vec{N}) = -\infty \text{ when } \vec{N} \cdot \vec{A} > SLA, \tag{2}$$

where the summation in (1) is over any $\vec{L} \leq \vec{N}$ or $\vec{L} \leq \vec{N} + 1$ depending on whether the action a is to accept or to reject. $P_{\vec{N},\vec{L}}(a)$ denotes the transition probability from state $\vec{N}$ to state $\vec{L}$ under action a.

The above boundary conditions guarantee that SLAs are never violated. Note that the model also allows for violations of the SLAs via penalty functions $V_0(\vec{N})$.

More concretely, at each time period there are four possibilities: a request arrives and it is accepted; a request arrives and is rejected; no requests arrive; or no change (in arrival and departure) occurs. A more detailed description of the optimality conditions is as follows

$$\begin{aligned} V_t(\vec{N}) &= \sum_{i=1}^{m} \max \left\{ p_{it}(R_i + \sum_{\vec{L} \leq \vec{N} + \vec{e_i}} V_{t-1}(\vec{L}) P_{\vec{N}+\vec{e_i},\vec{L}}), p_{it} \sum_{\vec{L} \leq \vec{N}} V_{t-1}(\vec{L}) P_{\vec{N},\vec{L}} \right\} \\ &+ \prod_{i=1}^{m}(1 - p_{it}) \sum_{0 < \vec{b} \leq \vec{N}} q_{\vec{b}t} V_{t-1}(\vec{N} - \vec{b}) + \prod_{i=1}^{m}(1 - p_{it}) q_{0t} V_t(\vec{N}) \end{aligned}$$

where q_{0t} means no departure.

It is not hard to see that the optimal policy is completely state dependent. Because of the curse of dimensionality, it could be quite hard to derive the optimal policy in an explicit analytical form. But this computation can be done off-line. Solving the dynamic program yields a look-up table which specifies for each period t and $\vec{N}$ whether bandwidth A_i should be allocated for the arriving request.

Additional assumptions may reduce the computational complexity. For instance, if we assume that one and only one of the the following events occurs: (1) an arrival of a request from class i; (2) some departures of class i from the system; (3) no event. Then, the dynamic programming recursion can be written as:

$$\begin{aligned} V_t(\vec{N}) \quad &= \textstyle\sum_{i=1}^m p_{it} \max\{R_i + V_{t-1}(\vec{N}+\vec{e_i}), V_{t-1}(\vec{N})\} + \sum_{\vec{N}\geq\vec{b}>0} q_{\vec{b}} V_{t-1}(\vec{N}-\vec{b}) \\ &\quad +(1-\textstyle\sum_{i=1}^m p_i - \sum_{0<\vec{b}\leq\vec{N}} q_{\vec{b}})V_{t-1}(\vec{N}). \end{aligned} \tag{3}$$

This additional assumption is plausible and natural when one wants to approximate the corresponding continuous time problem with time discretization. Interested readers can find such applications in [24] for the airline seat control problem.

To further simplify the computations, one may wish to assume that

- interarrival time of customer requests is exponential with parameter λ_i for class i;
- service time for class i is exponential with parameter μ_i.

Nevertheless, one needs to be careful in choosing an appropriate time unit to make all assumptions consistent.

Control policy. The admission control policy is a simple threshold policy: Admit A_i at time t if the system has $\vec{N}$ requests in service when

$$R_i + \sum_{\vec{L}\leq\vec{N}+\vec{e_i}} V_{t-1}(\vec{L})P_{\vec{N}+\vec{e_i},\vec{L}} \geq \sum_{\vec{L}\leq\vec{N}} V_{t-1}(\vec{L})P_{\vec{N},\vec{L}} \tag{4}$$

This development is similar to dynamic programming approaches to the reservation of airline seat allocation problem with cancellations studied by Subramaniam *et al.* [39]. The principle differences are that in our model we permit variable resource requests A_i, and the exit probabilities q_{it} are class dependent. Moreover, their formulation relies heavily on a rather restrictive assumption that there is up to only one arrival or departure (not both) during one time period. It is worth pointing out that the approach in this section can be described in its most general form as an airline yield management problem with multiple fare classes, over-booking and cancellations. (Interested readers are referred to McGill and van Ryzin [31] for a survey of research in airline revenue management).

A similar problem was considered in [30]. However, their formulation is an admission control for a discounted Markov decision problem with an infinite time horizon, whereas the problem considered in this section has finite time horizon. The advantage of this approach is that the admission control policy can be adjusted to fit various traffic conditions by carefully choosing appropriate time windows. For example, look-up tables can be different for off-peak and peak hour traffic statistics. Also, Poisson arrival and service assumptions are crucial in the formulation of [30], while the formulation in this section can be applied to more general cases.

2.2 What if we know everything: off-line algorithms

In this subsection we consider the off-line problem of bandwidth allocation strategies. This provides a benchmark against which to compare the dynamic programming allocation strategy. Each request specifies an arbitrary bandwidth and revenue. Assume that the requested bandwidth w_i, value v_i (revenue), start time t_i and duration D_i of each request are known in advance. A subset of the requests is to be retained (called *intervals* in this subsection) that maximizes the sum of the retained values, subject to the constraint that at each time instant the sum of bandwidths of retained intervals that span that time instant is at most the total bandwidth capacity B.

This problem is NP -hard. There is a simple reduction from the knapsack problem [15]: for each item in the knapsack instance construct a bandwidth request with value equal to that of the item with bandwidth equal to the item's weight.

Thus one turns to approximation algorithms. An algorithm has an *approximation ratio* r if the value of the solution returned is at least $1/r$ times the optimal value for maximization problems, or at most r times the optimal value for minimization problems.

The allocation problem is isomorphic to the "general caching problem"; see [3], for instance. However, in the general caching problem, the optimization criterion is complementary: the goal is to minimize the sum of the values of the discarded intervals (requests) rather than to maximize the value of those retained. Irani [18] gives an algorithm for this problem with an approximation ratio logarithmic in the ratio of the cache size (which corresponds to total bandwidth) to the smallest page (smallest bandwidth of any request), for the special cases in which the values are the same for all intervals and in which they are proportional to the widths. Although this does not yield an approximation bound for the bandwidth allocation problem, the algorithm is appealing due to its simplicity, its natural adaptation to our problem, and its good performance in practice.

The algorithm classifies intervals by their widths, in geometrically increasing-width classes. Class j comprises requests of widths between 2^j and 2^{j+1}. The algorithm makes a left-to-right (i.e., in order of increasing time) scan of the input set of intervals, maintaining a set of "live" intervals that are candidates for the solution. When the right endpoint of a live interval is passed, the candidate is added to the solution. When the left endpoint of an interval is passed, the interval is added to the live set, and then some number of intervals is discarded from the live set if necessary because the width bound B is exceeded. Once a class is chosen from which to discard an interval, the interval chosen within that class is the one whose right endpoint is furthest away (i.e., greatest). Irani uses the brute force approach of discarding from every class whenever B is exceeded, but this is not necessary and is only used to show that the algorithm has a worst-case approximation guarantee. We can instead discard only enough intervals to satisfy the size bound B again, but we need to decide which class or classes to discard from.

Irani's algorithm can be adapted in a natural way to the bandwidth allocation problem. The idea is to balance the sums of the values of intervals discarded from each class. For each class, a counter is maintained; these counters are set to zero at the start of the algorithm. Each time an interval is discarded, the counter for the interval's class is incremented by the value of the interval. A victim class is chosen to minimize the maximum counter value, after incrementing the victim class' counter. As before, within a class that interval whose right endpoint is greatest is chosen. This is repeated until the bound B is met again. Recently, Bar-Noy *et. al.* [3] gave a simple and elegant algorithm for the bandwidth allocation problem with a constant approximation ratio.

We implemented Irani's algorithm and a simple off-line algorithm that discards the request with the smallest ratio of revenue to remaining time until the end of the request. Results of these simulations are reported below as benchmarks against which to compare online policies.

2.3 Simulation and comparison of control policies

In this subsection, two admission control policies: the dynamic programming (DP) policy and the Greedy policy (that assigns resources based solely on availability) are simulated by arrival rates and average service times derived from real HTTP log traces. The experiments show the DP policy, depending on the revenue assumptions, beats the Greedy policy by a wide margin in all occasions. In the rest of the subsection, the simulation setting and results are reported, followed by discussion and interpretation of the results.

2.3.1 Simulation setting

The simulation environment tracks the admission control decisions made by two policies, DP and Greedy. The arrival traffic processed by both policies is identical. However, due to different admission policies, the number of active requests is different and the number of completed tasks also differs. For easy comparison, only two classes of requests are assumed: Gold and Silver. This two-class assumption matches common distinctions of "browsing" and "shopping" customers in e-commerce. It is noted that multi-class arrivals can be easily handled by the DP formulation and solution developed in this section.

Arrival rates and average service time are derived from an actual one-day HTTP log trace from a major commercial Web site that recorded arrival times and requested file sizes. The time granulation of the log was on the order of several seconds. There were typically multiple arrivals in one second all sharing the same time stamp. The requested file sizes are generally different as the requests look for different files. Because of the insufficient temporal resolution and the single arrival per time slot assumption in the DP policy, the real traffic rates were estimated using an artificially generated arrival pattern.

The simulator estimates the arrival rates and service times in the following way:

- The logged requests are divided randomly into Gold group and Silver group according to a given fixed probability ratio. The Gold arrival rate is obtained by dividing the total number of Gold requests by the time duration of the log trace, e.g., 24 hours. The Silver arrival rate is the division of the number of Silver requests and the time duration.
- Average service times are computed from the requested file sizes. In the beginning of the simulation, the total available bandwidth (in SLA), the assigned bandwidth per Gold request, and the assigned bandwidth per Silver request, are specified by user. Average service times are computed by dividing the average requested file size with the assigned class bandwidth.
- The current simulation only applies the HTTP log traces to estimate rates. The interarrival times of every class is assumed to be i.i.d. Poisson and the service time of each class exponential.

The simulator also allows the assignment of revenues generated for each class. The revenue brought by each request is fixed regardless how long the request stays in the system. This assumption reflects how most HTTP 1.0 requests are processed, which rarely involve long connections. In HTTP 1.1 and later, connections may be much longer since all elements (text and images) in a document are aggregated and sent through a single HTTP connection. Nevertheless, the DP formulation can be easily modified to account for revenue rate if needed.

2.3.2 Greedy policy

The Greedy policy is used in conjunction with the DP policy as a reference benchmark. The Greedy policy operates by the First-Come, First-Serve (FCFS) rule and does not reject any request unless the SLA will be violated. It treats requests from different classes in the same way and ignores differential revenue brought by these requests. This simple policy reflects how most Web sites today handle HTTP requests.

2.3.3 Simulation results

Example. Many rounds of simulations with different bandwidth and revenue assignments were conducted. Since their results do not differ qualitatively, a representative case is reported here. We assume that ratio of service time between gold and silver is 2 and that

- Estimated arrival rate of Gold class (per time slot): 4.5
- Estimated arrival rate of Silver class: 8.6

Table 1: Comparison: DP vs. Greedy, online vs. off-line

Total Bandwidth (kbps)	1000	1000	1000	1000	1000
Gold Bandwidth (kbps)	56	56	56	56	56
Silver Bandwidth (kbps)	28	28	28	28	28
Gold Revenue ($)	3	3	3	3	4
Silver Revenue ($)	2	2	2	2	2
Gold Arrival Rate (/s)	10	15	10	10	10
Silver Arrival Rate (/s)	10	15	20	20	20
Average Gold Service Time (s)	1.7	1.7	1.7	1.7	1.7
Average Silver Service Time (s)	3.3	3.3	3.3	1.7	3.3
DP Total Revenue ($)	2597	2854	2590	3752	3388
Gold Acceptance Percent	72.94%	60.99%	74.67%	20.55%	82.34%
Silver Acceptance Percent	21.73%	4.14%	13.16%	77.72%	5.80%
Greedy Total Revenue ($)	2398	2436	2253	3605	2560
Gold Acceptance Percent	39.67%	23.99%	19.10%	38.76%	20.48%
Silver Acceptance Percent	59.84%	45.49%	42.27%	60.82%	43.64%
RatioOffline Total Revenue ($)	3639	4746	4438	5234	5182
CounterOffline Total Revenue ($)	3613	4644	4221	5155	4954

2.3.4 Discussion

From the reported simulation results in table 2.3.3, we observed:

1. The DP policy constantly beats the Greedy policy.
2. The DP policy does not fully utilize bandwidth all the time.
3. The DP policy favors requests with higher revenue.

Separately, we noted that both greedy and DP policies deliver similar performance at the beginning the simulation when the system is largely empty and is nowhere near overloading.

Intuitively, a good admission control policy should accept any request when the system is empty and should be very selective about what is accepted when the system is near its capacity. This is exactly what the DP policy is performing.

At the beginning, the DP policy is as aggressive as the Greedy policy in accepting requests, regardless of how much revenue they bring. Therefore, they deliver similar performance. As time progresses, the system gradually approaches its capacity and the DP policy selectively reserves bandwidth for the more valuable Gold requests and rejects most Silver requests. Bandwidth reservation makes the overall utilization lower in order that it be available when high-revenue customers arrive. As the simulation shows, this aggressive-on-revenue approach enables the DP policy to generate more revenue than the aggressive-on-usage Greedy policy.

It is well known that DP approach can be very costly to implement, since the computational complexity grows exponentially with the increase of customer classes. It is worth pointing out that in some cases when greedy is near optimal to DP, we may choose to replace greedy algorithm for DP. As noted, when the system is lightly loaded, greedy policy is optimal. Also, it is not hard to see that when a fairly large amount of requests are being served in the system — so that in the next time unit the ratio between a possible new arrival and possible departures is low — the greedy policy can be as near optimal as DP. This was confirmed by our experimental results.

2.4 Implementation issues

Most request dispatcher implementations emphasize low latency to prevent itself from being the bottleneck. A request admission policy must also make decisions quickly. Fortunately, to implement the DP policy one only needs to keep the look-up table in memory. Admitting or dropping a request only involves table look-up and a comparison.

The only drawback of this approach is the potential large size of the look-up table. One option is to approximate the look-up table with some simple rules. For example, a rule may state when the overall bandwidth usage is below 60%, all requests should be accepted. Another rule may state when there are less than 20 Gold class customers and more than 80 Silver class customers, all new arrivals from Silver class customers should be rejected. These approximation rules are much less than precise than the look-up table, but they are easy to implement and require much less memory to store. Another simplification is to develop rules that characterize states where greedy policy is optimal or near optimal.

2.5 Concluding remarks

This section reports a successful implementation of revenue management control policies to the operational problem of service allocation to multiple user classes at a commercial Web hosting facility. Several conclusions and directions for further research are worth highlighting.

Revenue management appears to be an effective model for this purpose. Assigning revenue numbers to user requests does result in differentiated service. The service differentiation depends on the total free capacity available, so when there is capacity available then all classes receive the highest grade of service. This strikes us as fair: since the marginal cost of service is nearly zero so why not give good service when there is room to do so? On the other hand, the method preserves capacity to allocate to future arrivals so the site will never be "jammed". A static allocation model (like the "Paris subway" [34]) cannot do this.

The implementation includes several parameters that can be used for the tuning of differentiated services. These are: revenue by class, bandwidth allocation (as a proxy for all other Web-site resources) by class, and total capacity. Evidently, the setting of these parameters results in important service differentials. While one could outline an empirical procedure to assess the relative effects of tuning these parameters, one would like to have an analytical understanding of their impact on service level measurements. This is a topic for future research.

Solving the revenue management problem using dynamic programming techniques has obvious limitations. When the number of classes is more than two, there will be serious difficulties with the explosion of states. On the other hand, two classes may be enough for many applications. Certainly, the two class problem is well within the capabilities of the dynamic program even for very busy Web sites. It may be possible to handle larger numbers of classes by clever approximation and/or learning techniques [30]. Refer also to Section 4.

Perhaps a more serious consideration arises when the stochastic assumptions on the arrival and service processes are not met. Service times may exhibit a subexponential or heavy-tailed distribution, and arrivals may exhibit a short-range or long-range dependence structure. Generalizing the assumption of exponential service time distribution will considerably complicate the calculation of the departure probabilities in the dynamic programming recursion. A possible way through this may be to analyze a broader class of online algorithms and rank them by their efficiency relative to the optimal off-line solution under the subexponential/heavy-tailed and short-range/long-range dependent assumptions. Other approaches are considered in Sections 3 and 4.

To summarize: it appears that revenue management can be applied to the problem of resource allocation at Web sites and other IT service applications. A realistic implementation of such an idea appears in this section. Much interesting and important work remains to be done.

Appendix: Transition probability for the MDP. We will compute the probability of the transition $\langle N \rangle \to \langle N' \rangle$ when the decision $1 \leq i \leq m$ (allocate bandwidth A_i to the arrived request) is taken.

If we are in state $\langle N \rangle = (n_1, \ldots, n_m)$ and the customer arrives, the available decisions (bandwidths) are subject to the capacity constraint. Suppose, the decision A_1 is taken. At the next arrival we can be at any state $\langle N' \rangle = (n'_1, n'_2, \ldots, n'_m)$ with $n'_1 \leq n_1 + 1, n'_2 \leq n_2, \ldots, n'_m \leq n_m$.

For each individual bandwidth level i the transition $n_i \to n'_i$ means that n'_i requests are not over by the next arrival and $n_i - n'_i$ requests are over by the next arrival. For each individual request, the probability that it is over by the next arrival is $\mu_i/(\lambda_i + \mu_i)$ and correspondingly, the probability of not being over is $1 - \mu_i/(\lambda_i + \mu_i) = \lambda_i/(\lambda_i + \mu_i)$. This holds because the interarrival and service times are exponentially distributed with parameters λ_i and μ_i. Using a Bernoulli formula, the probability of transition $\langle N\rangle \rightarrow \langle N'\rangle$ under action A_1 can be derived.

3 Dynamic Multi-Resource Scheduling

We formulate a revenue/penalty-based multi-resource allocation problem for multiple QoS classes. In the present context, the control policies concern the load dispatching router and the per-server resources, although our methods and results can be applied more generally. Most previous optimal resource allocation studies of this type have focused on QoS performance guarantees based on throughput or mean response time measures. However, a critical issue for Web hosting applications and services concerns the per-request efficiency with which the differentiated services are handled, since delays experienced by clients of each service provider customer can result in lost revenue and clients for the customer. Further, such QoS performance guarantees may not be fully captured by the more standard performance metrics. To address these issues, we consider a general class of SLA models that include the tail distributions of the per-class response times in addition to standard metrics. The decision variables of the corresponding optimization problem are concerned with allocating resources to maximize the profit of hosting the customer Web sites under these SLA constraints and a general cost model.

Our problem falls within the general class of optimal resource allocation problems, but based on the foregoing non-conventional performance metrics. Recently, Menascé et al. [33] considered the related problem of resource scheduling in Web sites with the aim of maximizing revenue. A priority scheme is proposed for scheduling requests based on the states (navigation and purchase) of the client, who will lose patience if the response time is too long. Simulations were performed to evaluate the gains of such a policy in terms of revenue generated by the e-shopping carts. The study in [33] is the most closely related to our research in the literature, but it differs substantially from our study in several important respects; e.g., by considering a suboptimal allocation policy, and by taking a simulation-based approach.

The general optimal resource allocation problem of interest involves decisions at different time scales. In this section we present the general problem and consider the specific problem of optimal routing and scheduling of server resources at a relatively fine time scale, whereas Section 4 considers the related problem of optimal server assignments at a relatively coarse time scale. Specifically, we propose and investigate an analytic approach to obtain a provably optimal server capacity routing and scheduling policy with the use of methods from probability theory, queueing theory and combinatorial optimization. The optimization of SLA profits is formulated as a network flow model with a separable set of concave objective functions at the servers that are based on queueing-theoretic formulas we derive to capture the system dynamics. Our solution is computed using the most efficient algorithms for this class of optimization problems, and we show this solution to be globally optimal within assumptions of our formulation. A large number of experiments are performed to evaluate the effectiveness of our approach for optimizing SLA profits, quantifying the significant benefits of the optimal solution over existing methods across a wide range of system loads.

The remainder of the section is organized as follows. §3.1 describes the set of mathematical models used in our study, and then in §3.2 we present our analysis of the resource allocation optimization problem. Our experimental results are provided in §3.3, and concluding remarks are presented in §3.4.

3.1 Formal framework

We consider a Web hosting environment in which a common service provider hosts the Web sites for a set of N customers by providing Web applications and services to the clients of each customer. This includes the allocation of a set of M resources used to host the Web sites for these customers in order to satisfy the QoS performance requirements for every class of service. An SLA is created for every QoS class that a customer wants to provide to its clients, with an overall total of K SLA classes. The service provider gains revenue for all requests satisfying its per-class SLA, and incurs a penalty otherwise. The only exceptions to this are best-effort requests, for which a flat-rate pricing policy can be defined. Our objective then is to find the optimal resource allocation control parameters that maximize the profit of hosting the customer Web sites. We assume that the revenues and penalties for the service provider are coupled with the corresponding revenues and penalties of each of the customers being hosted, and thus optimizing SLA profits benefits every customer as well as the service provider.

3.1.1 Web hosting environment

A Web server farm is a distributed computer system consisting of M heterogeneous servers that independently execute K SLA classes of request streams, where each request is destined for one of N different Web sites. To accommodate any and all restrictions that may exist in the possible assignments of class-site pairs to servers, we assume that these possible assignments are specified via a general mechanism. In particular, let $I(i, j, k)$ be the indicator function matrix for these assignments: $I(i, j, k)$ takes on the value 1 when class k requests destined for site j can be served by server i, and 0 otherwise. This assignment function simply defines the set of class-site requests that can be served by a given server, with no implication that the requests will be served there. The problem of setting the values of the indicator matrix $I(i, j, k)$, which occurs at a coarser time scale due to the overheads involved in changing server assignments, are considered in Section 4.

We assume a system architecture in which there is a request dispatcher in front of the Web server farm that immediately routes all incoming class-site requests to one of the eligible servers. One of the optimization problems considered in this section is the control of the routing decisions between each pair of class-site requests and each server eligible to serve such requests. More precisely, we determine the optimal rate (or proportion) of traffic of different classes from different sites to be routed to each of the servers. Thus, our solution of the corresponding optimization problem will determine which requests are actually served by which servers.

3.1.2 Server scheduling policies

The primary policy for scheduling different classes of requests on each server is assumed to be Preemptive Priority Scheduling (PPS), which is a natural candidate when the SLA classes have a strict ordering among the per-class performance guarantees, revenues and penalties. It is quite reasonable to expect this to be the case for the class of Web hosting environments of interest, either in a strict sense or in a grouping sense (as discussed below). Moreover, class-based priority scheduling disciplines have been shown to be optimal for the scheduling of classes of differentiated service in single-server systems (e.g., refer to [37, 22, 23, 2]). We shall therefore assume throughout this section a strict ordering among the SLA classes across all servers and all sites.

Of course, there will be instances of our general problem where such a strict ordering does not exist among all of the SLA classes. A common scenario for such cases consists of the hosting of multiple Web sites each of which has levels of QoS performance guarantees that are very similar but not identical. Our proposed approach for this case consists of partitioning the individual SLA classes into groups each containing those which have very similar yet not identical QoS performance guarantees, revenues and penalties. The optimal solution among these equivalence groups of classes under the PPS discipline is obtained via the analysis of §3.2. The final step of our approach consists of taking this optimal resource assignment for each equivalence group and determining the optimal partition of this assignment among the SLA classes comprising the group under the Generalized Processor Sharing (GPS) discipline. GPS is considered as a natural candidate for the deployment of QoS response time guarantees within each equivalence group because of its properties of isolation and proportional sharing among classes. In the interest of space, we do not consider here the details of our combined PPS-GPS approach, but instead refer the interested reader to [27].

Finally, we note that it is certainly possible to consider the resource allocation problem using GPS instead of PPS for scheduling different classes of requests on each server. This approach is considered in [26] and it yields a sub-optimal solution to the resource allocation problem. Section 4 also considers a variant of the resource allocation problem of this section under the GPS server scheduling policy.

3.1.3 Queueing network model

The Web server farm is modeled by a queueing network composed of a set of M multiclass single-server queues and a set of $N \times K \times K$ single-class infinite-server queues. The former represents the collection of heterogeneous Web servers and the latter represents the client-based delays, or think times, between the service completion of one request and the arrival of the subsequent request within a client session. We shall henceforth index the Web servers by i, the delay servers by (j, k, k'), the Web sites by j, and the SLA classes by k, which are assumed to satisfy the natural conditions $i = 1, \dots, M$, $j = 1, \dots, N$, and $k = 1, \dots, K$, unless noted otherwise. Fig. 1 illustrates the queueing network model under consideration.

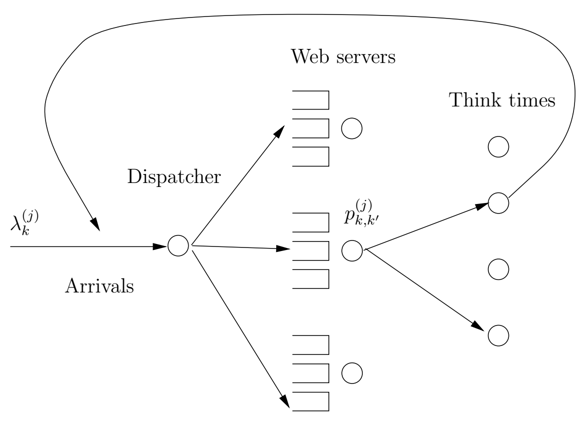

Figure 1: Queueing Network Model for Web Server Farm

Without loss of generality, we assume that each of the M single-server multiclass queues representing the Web servers can accommodate all classes of requests subject to $I(i, j, k)$. PPS is used to control the allocation of resources among the different service classes on each server. Thus, class 1 has absolute priority over class 2, which in turn has absolute priority over class 3, and so on. We assume that the service times of class k requests on server i follow a general distribution with mean $b_{i,k} = C_i^{-1} \mu_{i,k}^{-1}$, where C_i is the capacity of server i. Requests within each class on every server are assumed to be executed in an FCFS manner. This is a reasonable assumption within each class of a multiclass single-server queue under PPS, from both a practical and theoretical perspective. Furthermore, this use of FCFS minimizes the waiting time variance within each class [19].

Client sessions for site j that begin with a class k request arrive to the system from an exogenous source with rate $\lambda_k^{(j)}$. Upon completion of a class k request, the corresponding site j client session either returns as a class k' request with probability $p_{k,k'}^{(j)}$ following a delay at an infinite-server queue having mean $(\delta_{k,k'}^{(j)})^{-1}$, or completes with probability $1 - \sum_{\ell=1}^{K} p_{k,\ell}^{(j)}$. Let $L_k^{(j)}$ denote the aggregate arrival rate of site-j class-k requests to the set of servers, which is determined by the exogenous arrival rates and the transition probabilities:

$$L_k^{(j)} \;=\; \sum_{k'=1}^{K} L_{k'}^{(j)} p_{k',k}^{(j)} \;+\; \lambda_k^{(j)}. \tag{5}$$

Define $\widehat{\mathbf{P}}^{(j)} \equiv [p_{k,k'}^{(j)}]$ to be the corresponding request feedback matrix for site j, which is substochastic and has dimension $K \times K$. This matrix defines how each type of site j client session flows through the queueing network as a sequence of server requests and client think times, and thus it is used in our model to explicitly capture the correlations between the arrivals of class k and class k' requests of the same client session. The client think times can have arbitrary distributions, depending solely on the site j and the classes k and k'. It is important to mention that this model of client navigational behavior and its variants for capturing Web site traffic patterns are very general and are validated by various Web characterization studies; e.g., see [32, 27] and the references cited therein.

There are two important aspects of our resource allocation problem in practice that have direct restrictions and implications on our formulation and solution approach. On the one hand, the navigational behavior of client sessions explicitly captured in our queueing network model create per-class arrival processes that have strong dependence structures which in turn have a significant impact on the corresponding response time distributions. In particular, these types of complex arrival processes tend to result in response time distributions that have heavier, or longer, tails than the corresponding queue under simpler arrival processes (e.g., see [38]). On the other hand, the speed at which the dispatcher operates prevents it from collecting detailed statistics about the per-class arrival processes and typically the only available information is the corresponding mean arrival rates. Thus, the more detailed statistics required to properly characterize these arrival processes in order to obtain accurate tail distributions for the corresponding response times are simply not available to us. We therefore need an approach based solely on the mean arrival rate as the only characterization of the per-class arrival process that captures in a sufficient manner the tail of the corresponding response time distribution found in the Web hosting environments of interest.

Our approach is based on bounding from above the long-tail response time distributions of single-server queues under arrival processes with strong dependence structures by the response time tails of corresponding M/G/1 queues for relatively small response time values. More precisely, letting T_{LT} be the generic random variable for the response time process of a general single-server queue that has a long-tail response time distribution and letting $T_{\mathrm{M/G/1}}$ be the generic random variable for the response time process of a corresponding M/G/1 queue with $\mathsf{E}[T_{\mathrm{LT}}] = \mathsf{E}[T_{\mathrm{M/G/1}}]$, it then can be easily shown that $\mathsf{P}[T_{\mathrm{LT}} > x] \le \mathsf{P}[T_{\mathrm{M/G/1}} > x]$ for sufficiently small values of x. While most recently published work on the tail of the response time distribution has focused on the asymptotic behavior of the tail distribution, i.e., characterizing $\mathsf{P}[T > x]$ in the limit as $x \to \infty$, our interests here are quite different due to the very nature of the strict QoS performance guarantees motivating our study which makes us more interested in characterizing (or bounding) the measure $\mathsf{P}[T > x]$ for relatively small values of x. These and related arguments [27] suggest that we can use the corresponding response-time tail distributions from a multiclass preemptive-priority M/G/1 queue as an upper bound on the per-class response-time tail distributions for each server under PPS. Our results in §3.3 clearly demonstrate the effectiveness of this approach, especially those results based on access logs from production commercial Web sites.

3.1.4 Cost model

Our cost model is based on the notion that revenue is gained for each request satisfying its per-class SLA, and a penalty is paid otherwise. Let T_k be the generic random variable for the class k response time process, across all servers and all sites. Associated with each request of class k is an SLA constraint of the form

$$\mathsf{P}[T_k > z_k] \;\le\; \beta_k. \tag{6}$$

Our cost model is therefore based on receiving revenue $\mathbf{R}_k$ for each class k request having a response time of at most z_k and incurring a penalty $\mathbf{P}_k$ for each class k request which has a response time exceeding z_k. Our analysis can further handle multiple QoS performance guarantee intervals together with profit or loss parameters for each SLA class [27].

As is common in practice, there is at least one additional request class that is assumed not to have an SLA contract, and is instead served on a best-effort (BE) basis. Our cost model for each BE class k is based on the assumption that a fixed revenue $\mathbf{R}_k$ is gained for the entire class, independent of the number of class k requests executed, $k > K$. To help simplify the presentation, we will further assume that there is a single BE class $K + 1$, noting that it is straightforward to extend our analysis to handle multiple BE classes.

3.2 Optimization of SLA profits under PPS

We now consider the resource optimization problem of allocating server capacity among the set of class-site requests with the goal of maximizing profits under the above cost model. Let $\lambda_{i,k}^{(j)}$ denote the rate of class-k requests destined for site j that are assigned to server i. Our objective is to determine the optimal traffic assignments $\lambda_{i,k}^{(j)}$ that maximize SLA profits, given the matrix $I(i, j, k)$ and the exogenous arrival rates $\lambda_k^{(j)}$ which yield the aggregate arrival rates $L_k^{(j)}$ through the formula given in (5).

Our analysis is based on the assumption that the routing decisions at the request dispatcher are probabilistic: A class-k request for site j is routed to server i with probability $\lambda_{i,k}^{(j)} / \sum_{i'=1}^{M} \lambda_{i',k}^{(j)}$, independent of all else. When other routing mechanisms are used (e.g., weighted round robin), our optimal solutions in this probabilistic framework can be applied to set the parameters of these mechanisms (e.g., the per-class weights).

Our approach consists of first decomposing the queueing network into separate queueing systems and formulating each per-class optimization problem in terms of the profits of these queueing systems, and then solving the resulting optimization problems. In particular, we exploit the strict ordering of SLA classes and the properties of PPS to isolate the per-class queues at each server by decomposing the per-class performance characteristics of each server i in a hierarchical manner such that the analysis of the decomposed model for each class k in isolation is based on the solution for the decomposed models of classes $1, \ldots, k-1$. To elucidate the exposition, we first provide the theoretical framework for our decomposition-based approach and the corresponding formulation of the optimization problem; then we present the algorithms used to efficiently compute our optimal solutions to these resource allocation problems.

3.2.1 Formulation of optimization problem

To simplify the analysis, suppose the optimal traffic assignments satisfy $\rho_{i,k} \equiv \lambda_{i,k}/(C_i \mu_{i,k}) < 1$. We then divide the optimization problem into separate formulations for the SLA classes and the BE class. Our formulation for the SLA classes is given by:

$$\begin{aligned}
\text{(SLA-PPS)} \qquad \max \qquad & \sum_{i=1}^{M} \sum_{k=1}^{K} (\mathbf{R}_k \lambda_{i,k} - (\mathbf{R}_k + \mathbf{P}_k) \lambda_{i,k} \mathsf{P}[T_{i,k} > z_k]) \qquad (7) \\
\text{s.t.} \qquad & \mathsf{P}[T_{i,k} > z_k] \le \beta_k \omega_k; \\
& \sum_{j=1}^{N} \lambda_{i,k}^{(j)} = \lambda_{i,k}; \\
& \sum_{i=1}^{M} \lambda_{i,k}^{(j)} = L_k^{(j)}; \\
& \lambda_{i,k}^{(j)} = 0, \quad \text{if } I(i,j,k) = 0; \\
& \lambda_{i,k}^{(j)} \ge 0, \quad \text{if } I(i,j,k) = 1;
\end{aligned}$$

where $1 \le \omega_k \le \beta_k^{-1}$. The $\lambda_{i,k}^{(j)}$ are the decision variables we seek to obtain, and $\mathbf{R}_k$, $\mathbf{P}_k$, C_i, $\lambda_k^{(j)}$, z_k, β_k, ω_k, $\mu_{i,k}$ are input parameters. Note that the first constraint represents the SLA relationship among the variables, which depends on $\lambda_{i,k}$. Note also that, by allowing $z_k = \infty$ for any k, or equivalently by setting $\mathbf{P}_k = -\mathbf{R}_k$, our cost model makes it possible to include the objective of optimizing the throughput for class k. Furthermore, we note that there exists an equivalent formulation involving only the per-class tail distributions in the objective function of (SLA-PPS); this differs from the original objective function value by a constant.

In our formulation of the optimal allocation problem for the BE class we attempt to minimize the weighted sum of the mean response time for class

$K+1$ requests over all eligible servers subject to the decisions for the SLA classes, which yields the problem:

$$\begin{aligned}
\text{(BE-PPS)} \qquad \min \qquad & \sum_{i=1}^{M} \xi_{i,K+1} \left(\frac{\sum_{\ell=1}^{K+1} \lambda_{i,\ell} b_{i,\ell}^{(2)}}{2\sigma_{i,K}\sigma_{i,K+1}} + \frac{b_{i,K+1}}{\sigma_{i,K}} \right) \qquad (8) \\
\text{s.t.} \qquad & \lambda_{i,K+1} \le \overline{C}_i \mu_{i,K+1}; \\
& \sum_{i=1}^{M} \lambda_{i,K+1}^{(j)} = L_{K+1}^{(j)}; \\
& \lambda_{i,K+1}^{(j)} = 0, \ \text{ if } I(i,j,K+1) = 0; \\
& \lambda_{i,K+1}^{(j)} \ge 0, \ \text{ if } I(i,j,K+1) = 1;
\end{aligned}$$

where $\sigma_{i,k} = 1 - \rho_{i,k}^{+}$; $\overline{C}_i = C_i \, \sigma_{i,K}$; $\xi_{i,K+1}$ is the relative importance factor for the mean response time of the BE class $K+1$ on server i; $b_{i,\ell}$ and $b_{i,\ell}^{(2)}$ are the first two moments of the service times (recall that $b_{i,\ell} = C_i^{-1} \mu_{i,\ell}^{-1}$); and $\rho_{i,\ell}^{+}$ is the total load of classes $1, \ldots, \ell$: $\rho_{i,\ell}^{+} = \sum_{\ell'=1}^{\ell} \rho_{i,\ell'} = \sum_{\ell'=1}^{\ell} \lambda_{i,\ell'} b_{i,\ell'}$. The $\lambda_{i,K+1}^{(j)}$ are the decision variables that we seek to obtain, and the remaining variables are input parameters. Note that the relative importance weights $\xi_{i,K+1}$ are included in our formulation as they may be of greater use when there are multiple BE classes.

The response time expression in the objective (8) is derived from known results for the multiclass preemptive-priority M/G/1 queue; e.g., see [40]. Specifically, the set of SLA classes have absolute priority over the BE class of requests, and thus from the perspective of the BE class, the system behaves as a $K+1$-class preemptive-priority queue where BE requests represent the lowest priority class. Moreover, following the bounding arguments made above and based on the approximations developed below, we use the Poisson arrival process for the higher priority (SLA) classes of requests.

In the above formulation of (SLA-PPS), we also introduced the scaling factors ω_k to generalize the optimization problem. Several practical considerations motivate the use of such scaling factors. Most importantly, the scaling factors $\omega_k > 1$ make it possible for the hosting company to violate the SLA to a controlled degree in an attempt to increase profits under equation (7), whereas the hosting company will strictly follow the pre-defined SLA whenever $\omega_k = 1$. This can be particularly effective when the revenues and penalties for the hosting service provider are coupled with the corresponding revenues and penalties of each of the customer sites being hosted, as assumed in our study.

Another important aspect of the problem (SLA-PPS) concerns an explicit expression for the per-class response time tail distributions. Following the queueing-theoretic bounding arguments of §3.1.3, we consider this optimization problem within the context of a set of bounding multiclass preemptive priority M/G/1 queues. Our approach within this framework depends upon the per-class service time distribution, where the exact response time tail distribution for a single-class M/G/1 queue is used under exponential service times and a proposed approximation for the response time tail distribution in a single-class M/G/1 queue is used under general service times. To elucidate the exposition of this approach, we shall consider here the specific case where class 1 service times are exponentially distributed and the other classes have general service time distributions. More generally, one would exploit the class 1 analysis below for each of the SLA classes with exponential service times, and exploit the remaining analysis below (classes $2 \leq k \leq K$) for each of the SLA classes with a general service time distribution.

We first consider class 1 in isolation under the previously stated assumptions. Following the bounding arguments above, we suppose the arrival process to the class 1 queue at each server i to be a Poisson process. It then follows from standard results in the queueing theory (e.g., see [20]) that the left-hand-side of the SLA constraint is given by $\mathrm{e}^{-(C_i\mu_{i,1}-\lambda_{i,1})z_1}$. Hence, the SLA constraint (6) is satisfied when

$$\mathsf{P}[T_1 > z_1] \;=\; \mathrm{e}^{-(C_i\mu_{i,1}-\lambda_{i,1})z_1} \;\leq\; \beta_1. \tag{9}$$

As a result of (9), and since the lower priority classes do not interfere with the execution of class 1 requests under PPS, our formulation for the class 1 optimal resource allocation problem (SLA(1)-PPS) is identical to (SLA-PPS) but with the objective function in (7) replaced by:

$$\text{(SLA(1)-PPS)} \qquad \max \qquad \sum_{i=1}^{M}\Big(\mathbf{R}_1\lambda_{i,1} - (\mathbf{R}_1+\mathbf{P}_1)\lambda_{i,1}\mathrm{e}^{-(C_i\mu_{i,1}-\lambda_{i,1})z_1}\Big), \tag{10}$$

with the SLA constraint modified according to (9), and with $k = 1$ in the remaining constraints. Note that the third constraint is the resource allocation constraint, which ensures all offered traffic will be allocated in an attempt to optimize SLA profits. However, if we instead change this constraint to be $\sum_{i=1}^{M}\lambda_{i,1}^{(j)} \leq L_1^{(j)}$ and appropriately modify the objective function, then some fraction of the offered load might not be scheduled. In particular, whenever the optimal values of $\lambda_{i,1}^{(j)}$ sum to something less than the offered load, then admission control will be activated by denying service to the corresponding fraction of clients. Within the context of this formulation, all such clients can be assumed to be lowered to the BE class, and assumed to not satisfy their SLA constraint with probability 1 [27].

Upon solving (SLA(1)-PPS) to obtain the optimal decision variables ${\lambda_{i,1}^{(j)}}^*$, we seek to approximate the tail distribution for the class 2 queue within the same queueing-theoretic framework, which then will be used recursively to formulate and solve the optimization problem for the subsequent classes under the PPS ordering. Thus, for any class $k \geq 2$, we shall exploit results on preemptive priority M/G/1 queues to approximate the tail distributions of the response times at server i. More precisely, we assume that there exist constants $\gamma_{i,k}$ and $\theta_{i,k}$ such that

$$\mathsf{P}[T_{i,k} > z_k] \;\simeq\; \gamma_{i,k}\mathrm{e}^{-\theta_{i,k}z_k}. \tag{11}$$

These types of approximations are justified by various known results in the queueing literature, including the exponential distribution of response times in an M/M/1 queue (e.g., refer to equation (9) and [20]), the heavy-traffic approximation of an exponential waiting time distribution in a GI/G/1 queue due to Kingman (e.g., see [21]), the exponential upper and lower bounds on the response time in G/GI/1 queues (e.g., refer to [25]), the large deviations upper and lower bounds on queues (e.g., see [6, 14]), and the asymptotic results for single-server queues (e.g., refer to [1]).

Note that the exact tail distributions of the response times in preemptive priority M/G/1 queues can be obtained by numerically inverting the corresponding Laplace transforms (e.g., refer to [40]). However, such an approach will not be very useful for our optimal allocation problem. Indeed, for our solution approach, the cost functions should be concave in the rate variables $\lambda_{i,k}$, which does not hold true in general for the exact tail distributions.

Assuming that we have solved the optimization problem for the higher priority classes $1, \ldots, k-1$, then our formulation of the allocation problem for class k (SLA(k)-PPS) is identical to (SLA-PPS) but with the objective in (7) replaced by:

$$\text{(SLA}(k)\text{-PPS)} \qquad \max \qquad \sum_{i=1}^{M} \left(\mathbf{R}_k \lambda_{i,k} - (\mathbf{R}_k + \mathbf{P}_k) \lambda_{i,k} \gamma_{i,k} \mathrm{e}^{-\theta_{i,k} z_k} \right) \tag{12}$$

In order to apply the efficient optimization algorithms of §3.2.2, we need to appropriately choose the parameters $\theta_{i,k}$ and $\gamma_{i,k}$. While different schemes can be envisioned, in this section we propose to fit the two parameters with the first two moments of the response time distribution. Since $\mathsf{E}[X^m] = m \int_0^\infty x^{m-1} \mathsf{P}[X > x] dx$ for any nonnegative random variable X, it then follows from (11) that

$$\mathsf{E}[T_{i,k}] \;=\; \gamma_{i,k}/\theta_{i,k} \qquad \text{and} \qquad \mathsf{E}[T_{i,k}^2] \;=\; 2\gamma_{i,k}/\theta_{i,k}^2, \tag{13}$$

and thus

$$\theta_{i,k} \;=\; \frac{2\mathsf{E}[T_{i,k}]}{\mathsf{E}[T_{i,k}^2]} \qquad \text{and} \qquad \gamma_{i,k} \;=\; \frac{2\mathsf{E}[T_{i,k}]^2}{\mathsf{E}[T_{i,k}^2]}. \tag{14}$$

Using the known formulas for $\mathsf{E}[T_{i,k}]$ and $\mathsf{E}[T_{i,k}^2]$, we have [40]

$$\mathsf{E}[T_{i,k}] \;=\; \frac{\sum_{k'=1}^{k} \lambda_{i,k'} b_{i,k'}^{(2)}}{2\sigma_{i,k-1}\sigma_{i,k}} + \frac{b_{i,k}}{\sigma_{i,k-1}}; \tag{15}$$

$$\mathsf{E}[T_{i,k}^2] \;=\; \frac{\sum_{k'=1}^{k} \lambda_{i,k'} b_{i,k'}^{(3)}}{3\sigma_{i,k-1}^2\sigma_{i,k}} + \frac{b_{i,k}^{(2)}}{\sigma_{i,k-1}^2} + \left(\frac{\sum_{k'=1}^{k} \lambda_{i,k'} b_{i,k'}^{(2)}}{\sigma_{i,k-1}\sigma_{i,k}} + \frac{\sum_{k'=1}^{k-1} \lambda_{i,k'} b_{i,k'}^{(2)}}{\sigma_{i,k-1}^2} \right) \mathsf{E}[T_{i,k}] \tag{16}$$

where $b_{i,k}$, $b_{i,k}^{(2)}$ and $b_{i,k}^{(3)}$ are the first three moments of the class k service times on server i, and $\sigma_{i,k} = 1 - \rho_{i,k}^{+}$.

We now establish an important result based on this formulation that shows our solution to be globally optimal, within the assumptions of this section.

Theorem 1 *Assume that the service time distributions of classes* $1, \ldots, k$ *at each server* i *belong to the family of mixtures of exponentials. Then, the objective functions in (10) and (12) are concave in* $\lambda_{i,k}$*, for all* $i = 1, \ldots, M$*.*

Proof The objective function in (10) is obviously concave for each i as the first term is linear in $\lambda_{i,k}$ and the second term is negative and convex in $\lambda_{i,k}$. As for the objective function in (12), it suffices to show that $\gamma_{i,k}$ is increasing and convex in $\lambda_{i,k}$ and that $\theta_{i,k}$ is decreasing and concave in $\lambda_{i,k}$. These properties are readily verified.

Indeed, we are confident that this result holds under more general assumptions. Furthermore, it follows from this theorem that we can recursively apply the network flow model algorithms of §3.2.2 to the K subproblems (SLA(k)-PPS) corresponding to classes $k = 1, 2, \ldots, K$.

3.2.2 Optimization algorithms

We now present a brief description of the basic optimization algorithm employed in the solution for each class k. Additional details, generalizations and references can be found in [27, 17].

Consider a directed network with a single source node and multiple sink nodes. There is a function associated with each sink node. This function is required to be increasing, differentiable and concave in the net flow into that sink, and the overall objective function is the (separable) sum of these concave functions. We wish to optimize this objective function. There can be both upper (capacity) and lower bound constraints on the flows on each directed arc. We call this the network flow resource allocation problem (NFRAP). To be precise, consider a directed network consisting of nodes $\mathbf{V}$ and directed arcs $\mathbf{A}$. The arcs $a_{v_1 v_2} \in \mathbf{A}$ carry flow $f_{v_1 v_2}$ from nodes $v_1 \in \mathbf{V}$ to nodes $v_2 \in \mathbf{V}$. The flow is a real variable that is constrained to be bounded below by a constant $l_{v_1 v_2}$ and above by a constant $u_{v_1 v_2}$. That is, $l_{v_1 v_2} \leq f_{v_1 v_2} \leq u_{v_1 v_2}$ for each arc $a_{v_1 v_2}$. It is possible, of course, that $l_{v_1 v_2} = 0$ or $u_{v_1 v_2} = \infty$. There will be a single source node $s \in \mathbf{V}$ satisfying $\sum_{a_{s v_2}} f_{s v_2} = \mathbf{R} > 0$. This value $\mathbf{R}$, the net outflow from the source, is a constant that represents the amount of resource available to be allocated. There are N sinks $v_2 \in \mathbf{N} \subseteq \mathbf{A}$ which have the property that their net inflow $\sum_{a_{v_1 v_2}} f_{v_1 v_2} > 0$. All other nodes $v_2 \in \mathbf{A} - \{s\} - \mathbf{N}$ are transshipment nodes that satisfy $\sum_{a_{v_1 v_2}} f_{v_1 v_2} - \sum_{a_{v_2 v_3}} f_{v_2 v_3} = 0$. There is a single increasing, concave and differentiable function F_{v_2} of the net flow into each sink node j. Thus, the overall objective function is given by $\sum_{v_2 \in \mathbf{N}} F_{v_2}(\sum_{a_{v_1 v_2}} f_{v_1 v_2})$, which we wish to optimize subject to the lower and upper bound constraints described above.

In addition to the source node s, there are N nodes corresponding to the sites, followed by a pair of M nodes corresponding to the servers, the latter set being the sinks. In the first group of arcs, the jth node has capacity equal to $L_k^{(j)}$. The second group of arcs corresponds to pairs (j, i) for which $I(i, j, k) = 1$, and these arcs have infinite capacity. The capacities of the third group of arcs on (i, k) correspond to the SLA constraints. All lower bounds are 0.

A special case of NFRAP is:

$$\begin{aligned}
\max \quad & \sum_{v_2=1}^{N} (F_{v_2}(x_{v_2})) && (17)\\
\text{s.t.} \quad & l_{v_2} \le x_{v_2} \le u_{v_2}; && (18)\\
& \sum_{v_2=1}^{N} x_{v_2} = \mathbf{R}; && (19)
\end{aligned}$$

where each $F_{v_2}(\cdot)$ is an increasing, concave and differentiable function of the real decision variables x_{v_2}. The optimal solution for this so-called separable concave resource allocation problem (SCRAP) occurs at the place where the derivatives $F'_{v_2}(x_{v_2})$ are equal and the resource allocation constraint in equation (19) holds, modulo the bound constraints in (18). More precisely, the algorithm proceeds as follows. If $\sum_{v_2=1}^{N} l_{v_2} > \mathbf{R}$ or $\sum_{v_2=1}^{N} u_{v_2} < \mathbf{R}$, there is no feasible solution and the algorithm terminates. Otherwise, the algorithm consists of an outer bisection loop that determines the value of the derivative D and a set of N inner bisection loops that find the value of $l_{v_2} \le x_{v_2} \le u_{v_2}$ satisfying $F'_{v_2}(x_{v_2}) = D$ if $F'_{v_2}(l_{v_2}) \le D$ and $F'_{v_2}(u_{v_2}) \ge D$. Otherwise, we set $x_{v_2} = l_{v_2}$ (in the first case), or $x_{v_2} = u_{v_2}$ (in the second). The initial values for the outer loop can be taken as the minimum of all values $F'_{v_2}(l_{v_2})$ and the maximum of all values $F'_{v_2}(u_{v_2})$. The initial values for the v_2-th inner loop can be taken to be l_{v_2} and u_{v_2}.

Now the general network flow problem is solved by recursive calls to a subroutine that solves the problem with a slightly revised network and with generalized bound constraints $l'_{v_1v_2} \le f_{v_1v_2} \le u'_{v_1v_2}$ instead of those described above. As the algorithm proceeds it makes calls to the SCRAP solver. More precisely, we start by solving the problem obtained by ignoring all but the source and sink nodes. Let x_{v_2} denote the solution to that optimization problem. In the next step we add a supersink t to the original network, with directed arcs jt from each original sink, forming a revised network $(\mathbf{V}', \mathbf{A}')$. We set $l'_{jt} = 0$ and $u'_{jt} = x_{v_2}$ for all arcs connecting the original sinks to the supersink. For all other arcs the lower and upper bounds remain the same. Thus $l'_{v_1v_2} = l_{v_1v_2}$ and $u'_{v_1v_2} = u_{v_1v_2}$ for all arcs $a_{v_1v_2}$. We then solve a so-called maximum flow problem to find the largest possible flow $f_{v_1v_2}$ through the network $(\mathbf{V}', \mathbf{A}')$ subject to the lower and upper bound constraints described above. A simple routine for the maximum flow problem is the so-called labeling algorithm combined with a path augmentation routine. Using the residual network one can simultaneously obtain the so-called minimum cut partition. Those original sink nodes j which appear in the same partition as the supersink are now regarded as saturated. The flow f_{v_2t} becomes the lower and upper bounds on that arc. Thus we set $l'_{v_2t} = u'v_2t = f_{v_2t}$. For all remaining unsaturated arcs j we set $l'_{v_2t} = x_{v_2}$ and $u'_{v_2t} = f_{v_2t}$. Now we repeat the entire process, solving the SCRAP for the unsaturated nodes only, with suitably revised total resource, and then solving the revised network flow problem. This process continues until all nodes are saturated, or we reach an infeasible solution.

3.3 Experimental results

In this subsection we discuss some experimental results to illustrate the effectiveness of our approach for optimizing SLA profits based on QoS performance guarantees. A large number of experiments have been conducted under a wide range of parameter settings. In each case, we numerically determine the optimal solution using the models and methods of §3.1 and §3.2, and then we investigate through numerical experiments and simulation the benefits of our approach. A representative set of these experiments are discussed here; additional results and details can be found in [27].

3.3.1 Configuration of experiments

Throughout this subsection we shall focus on the following parameter settings: $M = 12$; $N = 3$; $K = 3$; $C_i = 1.0$ for $i = 1, \ldots, 6$; $C_i = 2.0$ for $i = 7, \ldots, 12$; and $\lambda_k^{(1)} = 0.08, 0.16, 1.2$; $\lambda_k^{(2)} = 0.06, 0.12, 0.8$; $\lambda_k^{(3)} = 0.04, 0.08, 0.4$; $\mu_{ik}^{-1} = 0.15, 0.3, 0.6$; $\beta_k = 0.05, 0.1, 0.1$; $z_k = 0.6, 1.2, 1.8$; $\mathbf{R}_k = 0.3, 0.2, 0.1$; $\omega_k = 15, 8, 8$; for $k = 1, 2, 3$. There is a strict ordering of QoS performance guarantees, revenues and penalties from class 1 to class 3. We shall consider the effect of the penalty-revenue ratio $r := \mathbf{P}_k/\mathbf{R}_k$ by using $\mathbf{P}_k = r \cdot \mathbf{R}_k$ in our experiments. (One would typically have $r \geq 1$, and usually $r \gg 1$.) We also have investigated the case where the SLA for class 3 is based solely on throughput by setting $\mathbf{P}_3 = -\mathbf{R}_3$, where the corresponding results are only slightly different from those provided below; refer to [27].

Although we are discussing experiments with a relatively modest number of servers, these results can be easily used to infer the results corresponding to Web server farms which are larger (in a uniform sense). For example, consider a system where each of the 12 servers actually represents 10 real servers that are identical in every way. In this case, the optimal solution for the 120 server farm can be easily inferred from the solution provided in our experiments.

A multiplicative load factor $\eta > 0$ is used to scale the base arrival rates $\lambda_k^{(j)}$ to consider different traffic intensities. Thus, the load factor η provides a relative scaling such that $\lambda_k^{(j)}\eta$ is used in the experiments for the site-j class-k arrival rate. Note that, at least for the experiments discussed below, the service requirements depend only on the classes k, and not on the servers i.

We consider 3 configurations $I1$, $I2$ and $I3$ for the indicator matrix $I(i, j, k)$. Matrix $I1$ corresponds to the fully clustered server farm: any server can process requests for any class and any site. Matrix $I2$ corresponds to a partially clustered server farm: servers 1, 2 and 3 are dedicated to site-1 requests, servers 4 and 5 are dedicated to site-2 requests, and server 6 is dedicated to site-3 requests; whereas servers 7, 9, 10 and 12 are shared by site-1 and site-2 requests, and servers 8 and 11 are shared by site-1 and site-3 requests. Matrix $I3$ corresponds to a fully partitioned server farm: servers 1, 2, 3, 7, 8, and 9 are dedicated to site-1 requests, servers 4, 5, 10, and 11 are dedicated to site-2 requests, and servers 6 and 12 are dedicated to site-3 requests.

For comparison with our optimal allocation algorithm, we consider the proportional assignment scheme that employs:

$$\lambda_{i,k}^{(j)} = \frac{\lambda_k^{(j)} C_i \mu_{i,k} I(i,j,k)}{\sum_{\ell=1}^{M} C_\ell \mu_{\ell,k} I(\ell,j,k)}; \qquad (20)$$

$$\lambda_{i,k} = \sum_{j=1}^{N} \lambda_{i,k}^{(j)}; \qquad (21)$$

to allocate the per-class per-site traffic among the eligible servers. This is a natural way to assign the traffic and server capacity, and it is provably the best load balancing scheme in terms of stability regions. Moreover, for a more competitive comparison with our optimal solution, we consider a PPS discipline at every server. If on the other hand existing Web server scheduling disciplines were used together with proportional allocation instead of PPS, then the results under our optimal solutions would provide significantly larger profits than those discussed in this subsection.

3.3.2 Comparison of profits

We now quantitatively compare the profits obtained under our optimal resource allocation algorithm with those obtained under the proportional assignment scheme. For each of the experimental configurations considered we only discuss profit results for the SLA classes, because the BE classes do not impact the profit value under our cost model. Various comparisons are discussed under different stochastic assumptions.

Poisson Arrivals and Exponential Service Times. Consider first the case where the arrivals form a Poisson process, and the service times are exponentially distributed. We consider 4 values of r, starting with $r = 10$ and doubling up through $r = 80$. Since we are considering the profit per unit time, ideal performance under this metric corresponds to a linear function.

We observe that both algorithms yield a profit for small values of r, with considerably larger profits under the optimal assignment as well as strictly positive profits for somewhat higher values of r. For very large values of r, it is not possible to be profitable, with greater losses under proportional assignment than the optimal solution. At light to moderate loads the profits under the optimal assignment essentially grow linearly, and the curves are nearly on top of each other. This is the ideal situation: Penalties are rare. The profits do start to tail off at heavy loads. Naturally, the higher r values degrade more seriously, and the ordering of the curves is strictly determined by the value of r.

We also note that the optimal curves for $I2$ are, for all practical purposes, almost identical to those for the fully clustered matrix $I1$. Indeed, this is a further indication of the robustness of the overall optimal resource allocation algorithm. Even though the partially clustered and fully partitioned matrices correspond to distinctly different scenarios, there is enough flexibility to allow the optimizer to find solutions equivalent to the globally optimal solution given in the fully clustered case. Naive schemes such as proportional will not be as robust, which is further illustrated next.

Renewal Arrival Process and General Service Times. Now consider a more general case in terms of the distributional model assumptions. We focus on $I1$ with the penalty to profit ratio fixed at $r = 20$. The interarrival and service time sequences are each assumed to be independent and identically distributed, but otherwise arbitrary. We consider a comparison between the optimal solution and the proportional scheme under different coefficients of variation for these distributions, where coefficients of variation of 5 and 10 for the interarrival times and the service times are studied.

In comparison to the previous case where the coefficient of variation equals 1 for both interarrival and service times, the profits under both optimal and proportional solutions are decreased. These profits are decreasing in the coefficients of variation of both service times and interarrival times. This is not very surprising, as it is well known that the response times increase as the coefficients of variation increase. It is, however, important to note that the gaps in profits between the two solutions is increased with the increase of the coefficient of variation.

Access Logs from a Production Commercial Web Server. We next consider the case where the request arrival times and service times are taken from the access logs of a production commercial Web site, which exhibit long-range dependent arrival patterns with a Hurst parameter of around 0.78 and subexponential (Weibullian) service requirements. The corresponding results for I matrix configurations and various penalty to profit ratios r are studied.

It is interesting to observe from these results that the gap between the profits under both optimal and proportional solutions is even bigger than those discussed above for the stochastic interarrival and service processes. Moreover, we see that all of the trends observed under the above workloads are also discuss in the results based on a production commercial Web site.

3.4 Concluding remarks

The growth in Web usage creates a vital need to provide QoS guarantees for each differentiated service class across a wide range of Web hosting environments. In this section we explored the problem of optimizing profits under SLA contracts based on strict QoS performance guarantees and a general cost model. Our optimal resource allocation solution is obtained in a hierarchical manner using methods from probability theory, queueing theory and combinatorial optimization.

Our results provide important insights into the fundamental problem of optimizing SLA profits in Web hosting environments. In particular, we observe that the optimal resource allocation algorithm provides significantly larger profits per unit time than those obtained under the natural scheme of proportional assignment combined with PPS. This also illustrates the validity of our overall approach, including the viability and profitability of the class of SLA contracts used in our study. The optimal allocation consistently provides large profits over a wide range of system loads, whereas the naive algorithm typically yields losses which can be quite considerable. While it is possible to increase the capacity of the system to make a profit under proportional allocation, much greater profits can be obtained under the optimal allocation algorithm with fewer system resources, thus making a profit in a more efficient manner. Our results further demonstrate the effective use of bounds and approximations on the per-class response time distribution when closed-form expressions are not known, at least within the context of our SLA contracts and related models. Finally, numerical experiments also demonstrate our methods to be extremely efficient in practice, making it possible to exploit them on-line in Web hosting environments.

4 Dynamic Multi-Resource Assignment

We continue our investigation of a general revenue/penalty-based multi-resource allocation problem by building upon the optimal routing and scheduling of server resources at a relatively fine time scale of the previous section and by focusing on the related control problem of optimal server assignments at a relatively coarse time scale. In particular, one of the key tasks of the hosting service provider is to allocate servers to each of the Web sites to satisfy the agreed upon QoS performance guarantees for the different classes of incoming requests at each point in time, while maximizing its profits. Doing so requires consideration of what might happen over multiple periods of time. However, the number of scenarios to which the system can transition in just a short amount of time grows quickly with the system dimensions, making it computationally infeasible to find the optimal control policy for dynamically assigning servers, as well as adding new servers, within the context of the set of SLAs. We propose a solution to the Web server allocation problem based on approximate dynamic programming, and compare our algorithm against a deterministic policy that optimizes the allocation based on the average Web site traffic.

The remainder of the section is organized as follows. Aspects of our formal framework for the resource allocation problem are presented in §4.1. We then formulate the server allocation problem in the MDP framework in §4.2 and discuss applications of approximate linear programming in §4.3. We demonstrate how problems stemming from state space and action space complexity can be addressed in §4.4 and §4.5. We present experimental results in §4.6, and offer closing remarks in §4.7.

4.1 Formal framework

4.1.1 Solving the scheduling and routing problems

We consider a variant of the formal Web hosting framework in Section 3.1 consisting of a collection of M Web servers that are shared by N customer Web sites. The scheduling policy at each Web server is GPS. We will solve the server capacity scheduling and routing problem over 2.5-minute intervals. The Web server assignments and arrival rates are assumed to remain fixed within each of these intervals. We use the symbol I to denote a particular encoding of the Web server assignment and let L denote a vector of aggregated endogenous/exogenous arrival rates L_k. The solution of the scheduling and routing problem is a function of the pair (I, L).

Using an approach based on stochastic bounds and approximations, our routing and scheduling decision variables reduce to $\lambda_{i,k}$, the arrival rate of class k requests routed to server i, and $\phi_{i,k}$, the fraction of server i capacity assigned to class k. The arrival rates across servers for any given class have to equal the total arrival rate for that class L_k. Furthermore, the total assigned capacity for any given server cannot exceed 1.

The SLA establishes that the response time T_k for each class k request must satisfy

$$P(T_k > z_k) \leq \beta_k,$$

for given parameters z_k and β_k. As discussed in [26], we bound this constraint from above by

$$e^{(\lambda_{i,k} - \phi_{i,k}\mu_k)z_k} \leq \beta_k,$$

via arguments based on stochastic bounds and approximations.

We also consider a simplified variant of the cost model in Section 3. In particular, a usage-based cost model is considered in which server usage is charged per time with rate $\mathbf{P}_k$ for class k. The expected time server i devotes to class k in each 2.5-minute interval is given by $\lambda_{i,k}/\mu_k$, provided that arrival rates and service times are expressed in the correct time scale, and therefore the expected profit generated by class k requests processed on server i is given by

$$\mathbf{P}_k \frac{\lambda_{i,k}}{\mu_k}.$$

We thus have the following optimization problem for determining server capacity scheduling and routing policies:

$$\begin{aligned}
\text{(Usage-GPS)} \qquad \max \quad & \sum_{i=1}^{M} \sum_{k=1}^{K} \mathbf{P}_k \frac{\lambda_{i,k}}{\mu_k} \qquad (22) \\
\text{s.t.} \quad & \lambda_{i,k} \leq \frac{\ln(\beta_k)}{z_k} + \phi_{i,k}\mu_k, \text{ if } I(i,k) = 1, i = 1, \ldots, M, k = 1, \ldots, K; \\
& \sum_{i=1}^{M} \lambda_{i,k} \leq L_k, k = 1, \ldots, K; \\
& \lambda_{i,k} = 0, \text{ if } I(i,k) = 0, k = 1, \ldots, K, i = 1, \ldots, M; \\
& \lambda_{i,k} \geq 0, \text{ if } I(i,k) = 1, k = 1, \ldots, K, i = 1, \ldots, M; \\
& \sum_{k=1}^{K} \phi_{i,k} \leq 1, i = 1, \ldots, M; \\
& \phi_{i,k} = 0, \text{ if } I(i,k) = 0, k = 1, \ldots, K, i = 1, \ldots, M; \\
& \phi_{i,k} \geq 0, \text{ if } I(i,k) = 1, k = 1, \ldots, K, i = 1, \ldots, M.
\end{aligned}$$

The $\lambda_{i,k}$ and $\phi_{i,k}$ are the decision variables we seek to obtain. Here our use of $I(i,k)$ is a slight abuse of notation and it indicates whether server i is assigned to the Web site associated with class k.

The optimal value of problem (Usage-GPS) is denoted by $R(I,L)$, corresponding to the expected profit over a 2.5-minute interval when arrival rates are given by L and the Web server assignment is given by I.

The LP (Usage-GPS) can be solved analytically, which speeds up computation. Note that it decomposes into N smaller problems of scheduling and routing for each Web site. Moreover, we can show that the following greedy strategy is optimal:

1. assign the minimum capacity $-\ln(\beta_k)/z_k\mu_k$ to each class k;

2. assign the remaining capacity as needed to classes based on a priority scheme, serving classes according to profit $\mathbf{P}_k$.

Optimality of the procedure above is easily verified as follows. Suppose we have two classes k and k' with $\mathbf{P}_k < \mathbf{P}_{k'}$, with $\sum_i \lambda_{i,k'} < L_k$ and $\phi_{i,k} > 0$ for some i. Then we can reallocate server capacity according to $\bar{\phi}_{i,k} = \phi_{i,k} > 0 - \epsilon$, $\bar{\phi}_{i,k'} = \phi_{i,k'} + \epsilon$ so that $\sum_i \lambda_{i,k'} + \epsilon\mu_k \leq L_k$ and $\bar{\phi}_{i,k} \geq 0$, which is a new feasible solution to (Usage-GPS). This incurs a change in profit of

$$\frac{\mathbf{P}_{k'}}{\mu_{k'}}\epsilon\mu_{k'} - \frac{\mathbf{P}_k}{\mu_k}\epsilon\mu_k = \mathbf{P}_{k'} - \mathbf{P}_k > 0.$$

We conclude that an optimal policy must serve all requests of the most expensive classes first, hence the greedy policy is optimal.

Note that the constraint

$$\lambda_{i,k} \leq \frac{\ln(\beta_k)}{z_k} + \phi_{i,k}\mu_k,$$

corresponding to the SLA for class k, requires that a minimum server capacity be assigned to requests of class k even if that server is not processing any requests of that type. This is due to the fact that the SLA constraint is based on a stochastic bound, which is not tight for small (or zero) arrival rates $\lambda_{i,k}$. Ideally, problem (Usage-GPS) would be reformulated to correct for that, but this would lead to a nonconvex optimization problem. In a different relaxation of the original scheduling and routing problem, we may approximate the number of servers needed for serving all requests of class k by

$$\frac{L_k}{\mu_k + \frac{\ln(\beta_k)}{z_k}}, \tag{23}$$

and assign the available servers to the classes associated with a Web site according to that number. In this situation, server capacity is not assigned to classes with no requests being processed in a given server. Expression (23) is motivated by the fact that, if a server i is totally dedicated to class k requests ($\phi_{i,k} = 1$), it can process at most $\mu_k + \frac{\ln(\beta_k)}{z_k}$ requests of that type. With expression (23) as an approximation for the number of servers needed for each class, we have a nearly optimal policy by assigning available servers greedily according to profit $\mathbf{P}_k/\mu_k$.

The Web server allocation problem will next be solved by dynamic programming. We will use the optimal value of problem (Usage-GPS) — R(I,L) — as one-step rewards. Dynamic programming is called for due to the time-variant nature of arrival rates: changes in the incoming traffic will typically require adjustments in the number of servers allocated to each Web site. However, before tackling the Web server allocation problem, we will discuss the arrival rate process.

4.1.2 Arrival Rate Process

As explained in Section 3.1.3, there are two types of Web page requests in a Web server farm: exogenous, corresponding to users initiating a browsing session, and endogenous, corresponding to subsequent requests in an already initiated session. As mentioned before, we will consider the aggregated arrival rate, making no distinction between these two types of requests.

We consider the following model for the arrival rate process for requests of class k associated with Web site i:

$$L_k(t+1) = \max(\bar{L}_k + a_k(L_k(t) - \bar{L}_k) + \sigma_k N_k(t) + M_k B_i(t), 0) \tag{24}$$

where $\{N_k(t), k = 1, \ldots, K, t = 0, 1, \ldots\}$ is a collection of independent standard normal random variables and $\{B_i(t), i = 1, \ldots, N, t = 0, 1, \ldots\}$ is a collection of independent Bernoulli random variables.

We interpret the arrival rate process in (24) as follows. $\bar{L}_k$ represents a prediction for the average arrival rate associated with class k. We assume that arrival rates fluctuate around their average value $\bar{L}_k$, where a_k is a scalar between 0 and 1 representing how persistent deviations from the average arrival rate behave. The normal random variables $N_k(t)$ represent regular fluctuations around the average arrival rate, and the Bernoulli variables $B_i(t)$ capture arrival bursts. Note that the occurrence of a burst is associated with a Web site as a whole, not with any particular classes of users.

4.2 MDP Model for Web Server Assignment Problem

We model the Web server assignment problem in discrete time, with each time step corresponding to 2.5 minutes which represents the time necessary to allocate or deallocate a server. The state of this model should contain all information that is important for the assignment decision. In the Web server assignment problem, with the simplified model in (24) for arrival rates presented in §4.1.2, a natural choice for the state variable is the pair (I, L), where I indicates the servers assigned to each Web site and L is a K-dimensional vector of arrival rates.

Actions A take values on the same space as Web server configurations I and indicate new Web server configurations. Valid actions must satisfy the constraint that only currently deallocated servers can be assigned to a Web site. A state (I, L) under action A transitions to state $(A, \tilde{L})$ with probability $P(L, \tilde{L})$ determined by the arrival rate processes in (24).

We have to specify rewards associated with each state-action pair. For simplicity, we will assume that the arrival rate L remains constant over each time step in the Web server assignment problem, and consider the expected reward $R(I, L)$ for the static scheduling/routing decision given by the optimization problem (Usage-GPS).

We will seek to optimize the discounted infinite-horizon reward. We expect to use reasonably large discount factors (in the experiments, $\alpha = 0.99$) so that our criterion can be viewed as an approximation to average reward.

4.3 Approximate Linear Programming

In the previous subsection, we specified the parameters necessary to formulate the Web server assignment problem as a dynamic programming problem. Ideally, an optimal policy would be determined based on the *optimal value function* $J^*(I, L)$, which is the unique solution to Bellman's equation:

$$J^*(I, L) = \max_A \left\{ R(I, Z(L, A)) + \alpha \mathrm{E}\left[J^*(A, L(1)) | L_0 = L \right] \right\}.$$

Function $Z(L, A)$ determines the servers available to each Web site when the configuration is changing from L to A (for instance, if a server is being added to a Web site, it will only be available in the next time step; however, if it is being removed from that Web site, it becomes unavailable right away).

An optimal policy A^* can be derived from J^* as follows:

$$A^*(I, L) = \operatorname*{argmax}_A \left\{ R(I, Z(L, A)) + \alpha \mathrm{E}\left[J^*(A, L(1)) | L_0 = L \right] \right\}.$$

There are several algorithms for solving Bellman's equation. However, we observe that even with a reasonably simple model for the arrival rate processes such as that given by (24) — and certainly with more sophisticated models that one might eventually want to consider — our problem suffers from the *curse of dimensionality.* The number of state variables capturing arrival rate processes grows linearly in the number of Web sites being hosted, and the number of states for the mapping of servers to Web sites is on the order of $O(M^N)$. Clearly, for all but very small Web server farms, we will not be able to apply dynamic programming exactly, as it would require computing and storing the optimal value function over a huge state space. Alternatively, we use approximate linear programming [36, 9].

Approximate linear programming is based on the linear programming approach to dynamic programming [11, 12, 13, 29]. It involves an approximation of the optimal value function by a linear combination of prespecified *basis functions* $\phi_i, i = 1, \ldots, p$:

$$J^*(I, L) \approx \sum_{i=1}^{p} r_i \phi_i(I, L).$$

A reasonable set of weights $r_i, i = 1, \ldots, p$ to be assigned to each of the basis functions can be found by the solution of the following linear program:

$$\begin{aligned} \min_{r_i} \quad & \sum_i r_i \sum_{I,L} c(I,L)\phi_i(I,L) \\ \text{s.t.} \quad & R(I, Z(L,A)) + \sum_i r_i \mathrm{E}\left[\phi_i(A, L(1))|L_0 = L\right] \leq \sum_i r_i \phi_i(A,L), \quad \forall (I,L,A). \end{aligned}$$

We refer to this problem as the *approximate LP*. The objective function coefficients $c(I,L)$ are *state-relevance weights* and they determine how errors in the approximation of the optimal value function over different portions of the state space are weighted.

We face the following design decisions in the implementation of approximate linear programming:

- choice of basis functions ϕ_i;
- choice of "state-relevance weights" c;
- development of a mechanism for dealing with the intractable number of constraints involved in the approximate LP.

We address these design questions in the next subsection.

4.4 Dealing with State Space Complexity

A suitable choice of basis functions is dictated by conflicting objectives. On one hand, we would like to have basis functions that accurately reflect the advantages of being in each state. To satisfy this objective we might want to have reasonably sophisticated basis functions; for instance, values of each state under a reasonably good heuristic could be a good choice. On the other hand, choices are limited by the fact that the implementation of the approximate LP in acceptable time involves the ability to compute a variety of expectations of the basis functions relatively fast. In particular, we need to compute or estimate the objective function coefficients $c^T\Phi$, which correspond to the vector of expected values of each of the basis functions conditioned on the states being distributed according to the state-relevance weights c. Expected values of the basis functions also appear in the approximate LP constraints; specifically, a constraint corresponding to state (I_k, L_k) and action A_k involves computing the expected value of the basis functions evaluated at (A_k, L_{k+1}), conditioned on the current arrival rates L_k. To keep the running time acceptable, we would like to have basis functions that are reasonably simple to compute and estimate. We would also like to keep the number of basis functions reasonably small.

Our approach was to extract a number of features from the state that we thought were relevant to the decision-making process. The focus was on having smooth features, so that one could expect to find a reasonable scoring function by using basis functions that are polynomial on the features. After some amount of trial and error, we have identified the following features that have led to promising results:

- number of servers being used by each class, assuming that server capacity is split equally among all classes associated with each Web site. Denote by $I(i)$ the number of servers allocated to Web site i, and by $N(i)$ the number of classes associated with that site. The number of servers U_k being used by each class k associated with Web site i is the minimum of $I(k)/N(k)$ and expression (23) for the approximate number of servers needed by that class.
- current arrival rates per class, given by L_k.
- average server utilization for each Web site. This is computed as the ratio between the total number of servers being used by all classes associated with a given Web site, assuming that server capacity is split equally among all classes, and the total number of servers assigned to that Web site. More specifically, $\sum_k U_k/I(i)$.

We let server capacity be split equally among all classes associated with each Web site in the choice of features for the sake of speed, as that leads to simpler expressions for the features and allows for analytical computation of certain expected values of the features and functions thereof. We have a total of $N + 2K$ features, where N is the number of Web sites and K is the total number of classes. We consider basis functions that are linear in the features, for a total of $N + 2K + 1$ basis functions.

We need to specify a distribution over pairs (I, L) to serve as state-relevance weights. Recall that states are given by pairs (I, L) corresponding to the current server assignment and arrival rates. Following the ideas presented in [10], we sample pairs (I, L) with a distribution that approximates the stationary distribution of the states under an optimal policy. Since the arrival rate processes are independent of policies, it is not difficult to estimate their stationary distribution. We use the following approximation to the stationary distribution of arrival rates:

$$L_k(\infty) \approx \left(\bar{L}_k + \frac{\sigma_k}{\sqrt{1-a_k^2}} N(0,1) + J_i \sum_{t=0}^{\infty} a_k^t B_t, 0 \right)^+ .$$

In choosing state-relevance weights, we have simplified the expression further and considered

$$L_k(\infty) \approx \max \left(\bar{L}_k + \frac{\sigma_k}{\sqrt{1-a_k^2}} N(0,1) + \frac{J_i}{1-a_k} B_0 \right)^+ .$$

We sample arrival rate vectors L according to the distribution above, and then select a server assignment I based on L. While the evolution of arrival rates is independent of decisions being made in the system in our model, the same is not true for server assignments. Hence sampling Web server assignments I based on the arrival rates is a more involved task. Our approach is to choose a server configuration based on the approximate number of servers needed per class, according to expression (23). Servers are greedily assigned to Web sites corresponding to classes with the highest profits $\mathbf{P}_k/\mu_k$. As discussed in §22 in the context of the scheduling and routing problems, such a procedure is nearly optimal for fixed arrival rates. Hence our underlying assumption is that an optimal dynamic server allocation policy should somewhat track the behavior of the short-run optimal allocation.

Naturally we cannot expect that states visited in the course of running the system under an optimal policy would always correspond to pairs of arrival rates L and server assignments I that are optimal with respect to L. To account for that, we randomize the assignment being sampled for each vector of arrival rates by moving some servers between Web sites and between allocated and deallocated status with some probability, relative to the optimal fixed assignment. More specifically, the randomization is performed in two steps:

1. for each originally unallocated server, with probability p_a allocate it to a Web site chosen uniformly from all Web sites;

2. for each originally allocated server, with probability p_d deallocate it.

The choice of probabilities for the randomization step involved some trial and error. We found that relatively high values of p_a lead to the best performance overall, whereas p_d can be made reasonably small. Typical values in our experiments were $p_a = 0.9$ and $p_d = 0.1$.

The objective function coefficient $c^T\Phi$ is estimated by sampling according to the rules described above. Note that with our choice of basis functions, conditional expected values of $\phi_i(\cdot)$ involved in the constraints can be computed analytically.

To deal with the intractable number of constraints involved in the approximate LP, one approach is to sample state-action pairs according to the frequency with which they would be observed if the system were running under an optimal policy. It can then be shown that, for a sufficiently large and tractable number of samples, solving the approximate LP with only the sampled constraints yields a good approximation to the full problem; see [10]. Determining the frequency of state-action pairs under an optimal policy is not feasible; alternatively, we choose a distribution that is only approximately representative of how often state-action pairs are visited.

In our constraint sampling scheme, we sample states according to the same procedure used for the state-relevance weights. Once a state (I, L) is sampled, we still need to sample an action A corresponding to a new server assignment. We choose a feasible action as follows:

1. compute the approximate expected arrival rates at the next time step as follows:

$$L_k(+) = \bar{L}_k + a_k(L_k - \bar{L}_k) + s_k + 20 * P[B_i = 1]M_k.$$

   Note that we overestimate the arrival rates; the true expected arrival rate at the next time step is actually given by $\bar{L}_k + a_k(L_k - \bar{L}_k) + P[B_i = 1]M_k$.

However, experimental results suggested that overestimating future arrival rates led to improvements in the overall quality of the policy generated by approximate linear programming; it is helpful to plan for some slack in the capacity allocated to each Web site since true arrival rates are uncertain.

2. compute the number of needed servers per class k for arrival rates $L_k(+)$, as given by expression (23). For each class, with probability p_r randomize the number of needed servers by multiplying it by a uniform random variable between 0 and 1.

3. assign the unused servers according to need, with priority given to classes k with higher values of $\mathbf{P}_k/\mu_k$.

4. determine the number of servers with usage less than a threshold value (in our experiments, 40%). With probability p_r, randomize the number of underused servers by multiplying it by a uniform random variable between 0 and 1.

5. deallocate the underused servers.

In our experiments, p_r was set to 0.1. Note that the action sampling procedure corresponds to a randomized version of a greedy policy.

4.5 Dealing with Action Space Complexity

The Web server assignment problem suffers from complexity in the action space, in addition to having a high-dimensional state space. In particular, a naive definition of the problem leads to a large number of actions per state: there are two choices for each allocated server (to keep it allocated or to deallocate it) and $N + 1$ choices for each deallocated server (allocate it to each of the Web sites or keep it deallocated).

An approach to dealing with the large number of actions is to split them into sequences of actions, so that in each point in time it is necessary to choose from smaller sets of actions. A natural choice is to consider actions for each server in turn; we would have a sequence of M decisions, with each decision being a choice from at most $N+1$ values. This approach does not fully solve the problem of having a large action space; instead, we transfer the complexity to the state space. As suggested in the discussion of the previous subsection, we choose an alternative approach: we "prune" the action space, using the structure present in the Web server allocation problem to discard actions that are most likely suboptimal in practice.

Our pruning of the action space is based on different rules for allocating and deallocating servers. In deciding how many servers to assign to each Web site, we first generate a rough estimate of how many extra servers each class will need, given by expression (23), and we order the classes according to the associated profits $\mathbf{P}_k/\mu_k$. We then use the scoring function generated by approximate linear programming to decide on the *total number* of Web servers to be allocated, ranging from 0 to the total number of free servers. Servers are allocated based

	browsers	buyers
service rate (μ_k)	10	5
profit ($\mathbf{P}_k$)	1	3
response time threshold (z_k)	1	2
maximum fraction of requests with service time $\geq z_k$) (β_k)	0.1	0.2

Table 2: Characteristics of browsers and buyers.

class \ Web site	1	2	3	4	5	6	7	8	9	10
browsers	20	26	26	19	19	16	14	11	9.6	10
buyers	5	3	4	4	2.5	2	4	4	2	5

Table 3: Average arrival rates $\bar{L}_k$.

upon need, following the profit-based class ordering. In deciding how many servers to deallocate, we first estimate their usage over the current and next time steps, and order servers based on usage. We then consider deallocating a number of servers ranging from 0 to the number of servers with usage under some threshold (in our experiments, 40%). We decide on the *total number* of servers to be deallocated based on the scoring function generated by approximate linear programming. Actual servers being deallocated are decided based on the usage ordering (least used servers are deallocated first).

4.6 Experimental Results

To assess the quality of the policy being generated by approximate linear programming, we compared it to a fixed policy that is optimal for the average arrival rates $\bar{L}_k + M_k P[B_i = 1]/(1 - a_k)$. We considered problems with up to 65 Web servers and 10 Web sites, with 2 classes of requests per Web site ("browsers" and "buyers"), for a total of 20 classes.

The actual advantage of using approximate linear programming as opposed to a fixed allocation depends on the characteristics of the arrival rate processes. In particular, in our experiments the best gains were obtained when arrivals were bursty.

Tables 2–6 present data for a representative example with 65 servers and 10 Web sites. The two classes of requests per Web site — "browsers and buyers" —

class \ Web site	1	2	3	4	5	6	7	8	9	10
browsers	0.95	0.95	0.97	0.95	0.95	0.95	0.97	0.95	0.95	0.95
buyers	0.95	0.95	0.97	0.95	0.95	0.95	0.97	0.95	0.95	0.95

Table 4: Persistence of perturbations in arrival rates over time a_k.

class \ Web site	1	2	3	4	5	6	7	8	9	10
browsers	1.92	1.28	2.56	0.64	1.60	1.28	1.60	0.96	0.64	0.64
buyers	0.32	0.32	0.48	0.32	0.22	0.32	0.80	0.16	0.22	0.48

Table 5: Arrival rate fluctuation factors σ_k.

class \ Web site	1	2	3	4	5	6	7	8	9	10
browsers	60	100	170	80	100	58	130	53	50	70
buyers	40	20	90	24	14	10	70	32	20	70

Table 6: Magnitude of bursts in arrival rates M_k.

were assumed to have the same characteristics across Web sites. Service rates, price per unit of time and SLA values for browsers and buyers are presented in Table 2. Characteristics of Web site traffic are presented in Tables 3–6. We let the probability $P[B_i = 1]$ of observing a burst of arrivals in any time step for each Web site i be the same and equal to 0.005.

Simulation results comparing the policy generated by approximate linear programming with the fixed allocation policy optimizing for average arrival rates $\bar{L}_k + M_k/(1 - a_k)$ show that the policy generated by approximate linear programming led to average profits that are 15% higher than the profits obtained by the fixed policy, as well as to a dramatic increase in the QoS, with about half as many dropped requests. Furthermore, achieving approximately the same profit with a fixed allocation was empirically determined to require as many as 120 servers.

When the bursty component $M_k B_i(t)$ in the arrival rate processes is relatively small, the gains resulting from the use of approximate linear programming relative to that of the naive fixed allocation are smaller. Note that in this situation the fluctuations are driven by the term $\sigma_k N(t)$. We have two different possibilities in this case: σ_k is small, in which case there is little fluctuation and fixed policies should do well; and σ_k is large, in which case there is much fluctuation and one might think that dynamic allocations would do better. The problem with the latter is that there is a considerable amount of noise in the system, which makes it difficult for any dynamic policy to track the variations in arrival rates fast enough. In fact, in our experiments the policies generated by approximate linear programming in this case tended to perform few changes in the server allocation, settling in a fixed allocation that had average reward comparable that of the fixed allocation that is optimized for average arrival rates.

The approximate linear programming algorithm was implemented in C++ and the approximate LP was solved by CPLEX 7.5 on a Sun Ultra Enterprise 6500 machine with Solaris 8 operating system and a 400 MHz processor. Reported results were based on policies obtained with approximate LP's involving 200,000 sampled constraints. The constraint sampling step took ~46 seconds on average and solution of the approximate LP took 17 minutes.

4.7 Closing Remarks

We proposed a model for the optimal resource allocation problem in Web hosting systems and developed a solution via approximate linear programming. The experimental results suggest that our approach can lead to major improvement in performance compared to the relatively naive approach of optimizing allocation for average traffic, in particular when Web site traffic is bursty.

Several extensions of the current model and solution may be considered in the future:

- More extensive comparison with other heuristics;
- Refinement of the arrival rate processes, with the possibility of including forecasts and accounting for cyclical patterns often observed in Internet traffic;
- Refinement of the choice of basis functions;
- Further development of the action space pruning strategies;
- Extending our model for the scheduling and routing problems to support priority scheduling at the servers, either instead of or in addition to the proportional scheduling policy considered herein;
- Integration with the problems of scheduling and routing, by considering all of them within the dynamic programming framework, rather than solving the first two by stochastic bounds and approximations;
- Consideration of capacity planning, where the number of servers does not remain fixed but may increase as new servers are acquired over time at some cost.

References

[1] J. Abate, G. L. Choudhury, D. M. Lucantoni, and W. Whitt. Asymptotic analysis of tail probabilities based on the computation of moments. *Annals of Applied Probability*, 5:983–1007, 1995.

[2] F. Avram, D. Bertsimas, and M. Ricard. Fluid models of sequencing problems in open queueing networks; an optimal control approach. In F. Kelly and R. Williams, editors, *Stochastic Networks*, volume IMA 71, pages 199–234, 1995.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, pages 735–744. ACM, May 2000.

[4] D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[5] D. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[6] C. S. Chang. *Performance Guarantees in Communication Networks*. Springer-Verlag, London, 2000.

[7] Y.-C. Chang, X. Guo, T. Kimbrel, and A. King. Bandwidth broker: Revenue maximization policies for Web hosting. Technical report, IBM Research Division, 2001.

[8] D. P. de Farias, A. King, M. S. Squillante, and B. van Roy. Dynamic control of Web server farms. In *Proceedings of the INFORMS Revenue Management Section Conference*, June 2002.

[9] D. P. de Farias and B. V. Roy. The linear programming approach to approximate dynamic programming. Conditionally accepted to *Operations Research*, 2001.

[10] D. P. de Farias and B. V. Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. Conditionally accepted to *Mathematics of Operations Research*, 2001.

[11] G. de Ghellinck. Les problèmes de décisions séquentielles. *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, 2:161–179, 1960.

[12] E. V. Denardo. On linear programming in a Markov decision problem. *Management Science*, 16(5):282–288, 1970.

[13] F. D'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, 1963.

[14] N. G. Duffield and N. O'Connell. Large deviations and overflow probabilities for the general single-server queue, with applications. *Mathematical Proceedings of the Cambridge Philosophical Society*, 118:363–374, 1995.

[15] M. Garey and D. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[16] G. Gordon. *Approximate Solutions to Markov Decision Processess*. PhD thesis, Carneggie Mellon University, 1999.

[17] T. Ibaraki and N. Katoh. *Resource Allocation Problems: Algorithmic Approaches*. The MIT Press, Cambridge, Massachusetts, 1988.

[18] S. Irani. Page replacement with multi-size pages and applications to web caching. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 701–710, El Paso, Texas, May 1997.

[19] J. F. C. Kingman. The effect of queue discipline on waiting time variance. In *Proceedings of the Cambridge Philosophical Society*, volume 58, pages 163–164, 1962.

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

[21] L. Kleinrock. *Queueing Systems Volume II: Computer Applications.* John Wiley and Sons, 1976.

[22] G. P. Klimov. Time sharing service systems I. *Theory of Probability and Its Applications*, 19(3):532–551, 1974.

[23] G. P. Klimov. Time sharing service systems II. *Theory of Probability and Its Applications*, 23(2):314–321, 1978.

[24] S. A. Lippman and S. Stidham, Jr. Individual versus social optimization in exponential congestion systems. *Operations Research*, 25(2), March 1977.

[25] Z. Liu, P. Nain, and D. Towsley. Exponential bounds with an application to call admission. *Journal of the ACM*, 44:366–394, 1997.

[26] Z. Liu, M. S. Squillante, and J. L. Wolf. On maximizing service-level-agreement profits. In *Proceedings of the ACM Conference on Electronic Commerce (EC'01)*, October 2001.

[27] Z. Liu, M. S. Squillante, and J. L. Wolf. Optimal control of resource allocation in e-business environments with strict quality-of-service performance guarantees. Technical report, IBM Research Division, 2001.

[28] Z. Liu, M. S. Squillante, and J. L. Wolf. Optimal control of resource allocation in e-business environments with strict quality-of-service performance guarantees. In *Proceedings of the IEEE Conference on Decision and Control*, December 2002.

[29] A. S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[30] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis. Call admission control and routing in integrated service networks using neuro-dynamic programming. *IEEE Journal on Selected Areas in Communications*, 18(2):197–208, February 2000.

[31] J. McGill and G. J. van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233–256, May 1999.

[32] D. A. Menascé, V. A. F. Almeida, R. Fonseca, and M. A. Mendes. A methodology for workload characterization of e-commerce sites. In *Proceedings of the 1999 ACM Conference on Electronic Commerce*, 1999.

[33] D. A. Menascé, V. A. F. Almeida, R. Fonseca, and M. A. Mendes. Business-oriented resource management policies for e-commerce servers. *Performance Evaluation*, 42:223–239, 2000.

[34] A. Odlyzko. A modest proposal for preventing internet congestion. Technical report, AT & T Labs-Research, September 1997. http://www.research.att.com/~amo.

[35] I. C. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*, 8(2):171–184, February 2000.

[36] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

[37] W. E. Smith. Various optimizers for single-stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1954.

[38] M. S. Squillante, D. D. Yao, and L. Zhang. Web traffic modeling and web server performance analysis. In *Proceedings of the IEEE Conference on Decision and Control*, December 1999.

[39] J. Subramaniam, S. Stidham, Jr., and C. J. Lautenbacher. Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science*, 33(2):147–167, May 1999.

[40] H. Takagi. *Queueing Analysis: A Foundation of Performance Evaluation. Volume 1: Vacation and Priority Systems, Part 1.* North Holland, Amsterdam, 1991.