A Stochastic Control Model for Deployment of Dynamic Grid Services

Darin England and Jon Weissman
Department of Computer Science and Engineering
University of Minnesota, Twin Cities
{england,jon}@cs.umn.edu

Abstract

We introduce a formal model for deployment and hosting of a dynamic grid service wherein the service provider must pay a resource provider for the use of computational resources. Our model produces policies that balance the number of required resources with the desire to keep the cost of hosting the service to a minimum. The two components of cost that we consider are the deployment cost and the cost to keep the service active, which we view as a lease. We cast the problem in a dynamic programming framework and we are able to show that the model makes good leasing decisions in the face of such uncertainties as random demand for the service and random execution times of service requests. The results show that the policies obtained from the model reduce the cost of hosting a service and significantly reduce the variance of that cost.

1. Introduction

The success of web services has influenced the way in which grid applications are being written [10]. Grid application designers are now beginning to make use of software services that provide a specific functionality to the application, such as solving a system of equations or performing a simulation remotely. Grid applications that make use of such services require consistent response times and high availability from those services. The service provider (SP), who develops the service and its interface, may charge users through subscriptions to the service or through metered usage [4]. In turn, we assume that there is a cost to the SP for maintaining the presence of a service in the grid. This cost is charged to the SP by the owner and maintainer of the computational resources, the resource provider [12]. This work focuses on controlling such a cost to the SP. The two components of cost are: 1) a deployment cost, and 2) a cost to keep the service active, which we model as the cost to hold a lease. If there were no costs to maintaining the presence of a grid service, then the SP could simply deploy the service in as many places as possible and leave it running. Therefore, the SP must balance the demand for service with the desire to keep the cost of providing it to a minimum.

The amount of resources needed may vary over time and is a function of the demand for the service and the compute-intensive nature of the service. We address the situation where the demand for the service and the execution times to process the service requests are unknown, but can be estimated. Even though the SP will know the processing requirements for a typical invocation of the service, the execution time of any particular instantiation of the service can vary due to input data dependencies as well as resource contention from other services if, as is likely in a grid, the service is deployed in a time-sharing environment. Our model allows for two types of service deployments: planned deployments, which take place in accordance with the normal leasing cycle, and unplanned dynamic deployments, which occur only in the presence of excess demand for the service.

In this article we propose a model for making service deployment and resource leasing decisions in the presence of random demand for the service and random execution times for processing service requests. The decisions are made periodically and are based on expected average demand and expected average execution times. We model the arrival and the processing of service requests as a stochastic process in which the inter-arrival times and the execution times come from known probability distributions. The problem is cast as a finite-horizon dynamic programming problem. The result is a leasing policy that indicates to the SP how many resources to lease in each decision period. An important result is that our approach greatly reduces the variance of the total cost. Low variance is important for maintaining consistency and predictability in the number of service deployments and hence in the number of leased resources. The contributions of this work are twofold: 1) a mathematical formulation of the leasing problem, the parameters of which may be adjusted to correspond to different economic scenarios, and 2) the reduction in costs which are a result of the model's leasing policies.

2. Related Work

A number of works have proposed service-oriented architectures and have tested high-performance applications in those environments [4, 13, 14, 12]. In [14], Weissman and Lee present an architecture and middleware for dynamic replica selection and creation in response to service demand. Their work answers the questions of when and where to deploy a grid service. In contrast, this work focuses on the question of how many resources are required to host a grid service in the presence of random demand and execution times.

Buyya et. al. [3] and Wolski et. al. [15, 16] examine the use of supply and demand-based economic models for the purpose of pricing and allocating resources to the consumers of grid services. In this work we assume a supply and demand-based economy in which both software services and computational resources are in demand. In particular, we assume a separation of interests between the service provider and the resource provider. The service provider obtains the necessary computational resources at a cost. The user then, is only concerned with the software services that are required for the application, rather than negotiating directly with a resource owner for computing time.

3. Dynamic Programming

Dynamic programming (DP), or stochastic optimal control, is an approach for modelling and for solving optimization problems in which periodic decisions must be made under some level of uncertainty. The idea of DP is to solve a minimization problem in each period, beginning with the last period and ending with the initial period [1]. The optimal decision depends on the *state* of the system, which we define to be the number of previously leased resources and the number of currently executing service requests. In each decision period the expected costs of all admissible decisions are computed. An admissible decision is one that is valid given the current state of the system, e.g. we may not lease more resources than are currently available in the resource pool. The overall solution provides an optimal policy for leasing additional resources in each period[1].

4. Service Deployment and Resource Leasing

We use the DP approach to model and solve a stochastic decision problem for leasing computational resources. The grid service is then deployed and hosted on those resources for a certain length of time. The total length of time for which the service is to be deployed is divided into N periods, with the index of the current period labelled k, $k = 0 \ldots N$. The zeroth period represents the initial starting state of the system just before the first period. At the beginning of each period, the SP must decide how many resources are needed. Figure 1 shows how the leasing decision is applied in an arbitrary period. The variables in Figure 1 are defined in the following subsections and the model is fully discussed.

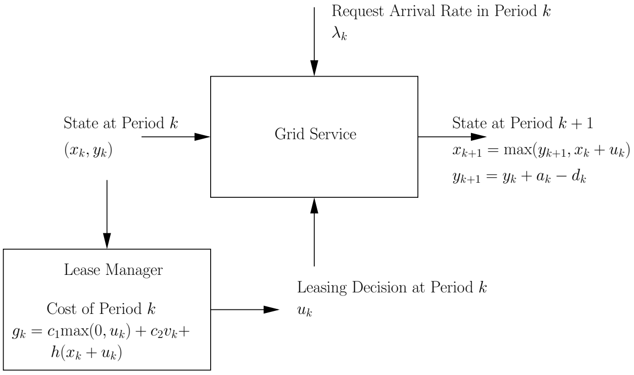
Request Arrival Rate in Period k
λ_k
State at Period k
(x_k, y_k)
Grid Service
State at Period $k+1$
$x_{k+1} = \max(y_{k+1}, x_k + u_k)$
$y_{k+1} = y_k + a_k - d_k$
Lease Manager
Cost of Period k
$g_k = c_1\max(0, u_k) + c_2 v_k +$
$h(x_k + u_k)$
Leasing Decision at Period k
u_k

Figure 1. Leasing Decision

4.1. State

We now define a representation of the state of the system. The state should be a compact summary of the available information that affects the decision to deploy the service. A tenet of DP is that the information that a decision-maker uses should only depend on the current state, and not on past history [1]. We describe the state of the grid service by a pair of variables (x_k, y_k) defined as follows.

x_k number of currently leased resources at the beginning of period k.

y_k number of currently executing service requests at the beginning of period k.

4.2. Decision Variable

In our model there is a single decision variable, u_k, that represents the number of (additional) resources to lease at the beginning of a period k. We use the qualifier "additional" because at the beginning of period k, there are already x_k resources held in lease and some or all of those leases may be renewed. u_k may be negative, in which case the decision is to "take down" the grid service on u_k resources. The state of the system evolves according to

$$x_{k+1} = \max(y_{k+1}, x_k + u_k). \tag{1}$$

Note that y_{k+1} is a random variable and is a function of the number of service requests and their execution times in pe-

[1] The policy is *optimal* in a probabilistic sense due to the random demand and execution times.

riod k. Specifically, for the number of requests in execution at the beginning of a period, we write

$$y_{k+1} = y_k + a_k - d_k,$$

where a_k is the number of requests for the grid service in period k and d_k is the number of requests that finished execution in period k.

We mention here that the decision variable u_k may only take on admissible values. An admissible value is one that is valid for the current state of the system. If, at the beginning of period k, there are y_k service requests in execution, then we must lease at least $y_k - x_k$ additional resources just to cover the current load. Also, we may not lease more resources than are available. Thus,

$$y_k - x_k \leq u_k \leq R - x_k,$$

where R is the maximum number of resources that could be leased, that is, the total number of resources available to the SP.

4.3. Demand

Requests for the grid service arrive at random points in time. We model the arrivals as a Poisson process due to its practicality. The Poisson process closely matches many arrival processes in real computing systems and is also amenable to analytical analyses. In the appendix we present a probabilistic argument for the computation of the expected values of two random variables that are needed in our model. The tractability of the result depends, in part, on the assumption that the demand for the grid service follows a Poisson process. We denote the demand in period k by λ_k, which represents the average arrival rate of service requests. The arrival rate may vary with time, indicating a non-stationary Poisson process.

4.4. Execution Times

We assume that the execution time of a service request is unknown until the request finishes execution. Although the SP will know the performance characteristics of the service, we assert that *exact* execution times cannot be predicted because 1) the input data will be different for any particular request, and 2) requests may need to compete with other applications and services for processing time. The execution time is therefore modelled as a random variable. For the model itself, we make no assumptions on the distribution of execution times. However, for our computational experiments with the DP algorithm, we model the executions times as Exponential random variables with parameter μ. Modelling the state, decision variables, demand, and execution times in this manner results in the discrete-time dynamic system shown in Figure 2.

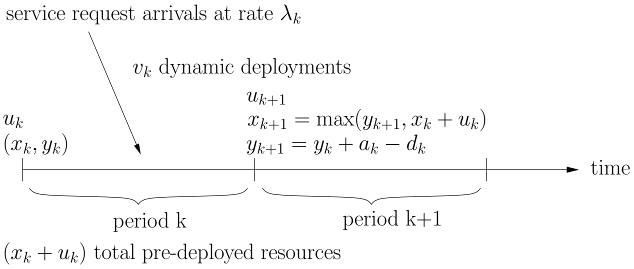
service request arrivals at rate λ_k
v_k dynamic deployments
u_{k+1}
$x_{k+1} = \max(y_{k+1}, x_k + u_k)$
$y_{k+1} = y_k + a_k - d_k$
u_k
(x_k, y_k)
time
period k
period k+1
$(x_k + u_k)$ total pre-deployed resources

Figure 2. Discrete-time Dynamic System

4.5. Cost

Hosting a grid service is not free. There are both direct and indirect costs for the use of computational resources: deploying a grid service requires bandwidth and disk space, processing service requests requires CPU cycles and memory. Our model employs a two-tier cost structure for deployment of a grid service, plus a separate cost to the keep the service active for more than one period after the initial deployment. The two types of deployment are

1. planned deployment with cost $c_1 \max(0, u_k)$, and
2. unplanned dynamic deployment with cost $c_2 v_k$.

A planned deployment means that the service provider deploys the grid service at the beginning of a leasing cycle in anticipation of service request arrivals. An unplanned dynamic deployment occurs whenever a service request cannot be processed because all resources are busy serving other requests[2]. We denote the number of such deployments during a period k as v_k. Unplanned deployments require on-demand leasing of resources. Our experiments assume that the resource provider will charge more for on-demand leases (i.e. $c_1 \leq c_2$) because the service provider is willing to pay the cost in order to maintain quality of service. However, no such assumption is required by the model itself and other economic scenarios can be accommodated. Another reason for the higher cost of on-demand leases is the single-service nature of the deployment. With planned deployment, one instantiation of the grid service may serve multiple consecutive requests, but not multiple simultaneous requests.

The cost to keep the grid active once it is deployed, that is, the cost to hold the lease on a resource, is h per period. The total cost, g_k, charged by the resource provider in a given period is the sum of the three cost components[3]. Since v_k and x_k are random variables, we must compute the ex-

[2] We assume that there is no queueing of service requests.

[3] The units of c_1, c_2, and h are monetary.

pected cost, $E\big[g_k\big]$, during execution of the DP algorithm.

$$\begin{aligned} E\big[g_k\big] &= E\big[c_1\max(0,u_k) + c_2 v_k + h(x_k + u_k)\big] \\ &= c_1\max(0,u_k) + c_2 E\big[v_k\big] + h(E\big[x_k\big] + u_k). \end{aligned}$$

From Equation (1) we see that the computation of $E\big[x_k\big]$ amounts to the computation of $E\big[y_k\big]$. The computations of both $E\big[v_k\big]$ and $E\big[y_k\big]$ are described in the appendix.

5. DP Algorithm

The DP algorithm proceeds in stages from period $N-1$ backward in time to period 0. At each stage, the algorithm computes the expected cost to get to the last stage, which is the expected cost for the current stage plus the expected costs for all future stages. This is known as the *cost-to-go* and is denoted by $J_k(x_k, y_k)$.

$$\begin{aligned} J_k(x_k, y_k) = \min_{y_k - x_k \le u_k \le R - x_k} &\Big[c_1\max(0,u_k) + c_2 E\big[v_k\big] \\ &+ h(E\big[x_k\big] + u_k)\Big] + J_{k+1}(x_{k+1}, y_{k+1}) \end{aligned}$$

In the equation above, the index k goes from $N-1$ to 0. The cost at the end of the last period, $J_N(x_N, y_N)$, is called the terminal cost. Traditionally, the terminal cost represents the cost of having unused resources at the end of the planning horizon. For our model, the terminal cost is $\max(0, x_N - y_N)$. At every other stage, the cost-to-go is computed for every possible state (x_k, y_k) and for every admissible leasing decision u_k. The optimal deployment and leasing decision for a given state is the one that minimizes the cost-to-go. We denote the optimal decision as u_k^*, and the optimal cost-to-go as $J_k^*(x_k, y_k)$. Algorithm 1 is the DP algorithm for the resource leasing problem. The pseudo-code is presented in matlab-*like* notation. The functions compute_V and compute_Y correspond to the computations of $E\big[v_k\big]$ and $E\big[y_k\big]$ given in the appendix. Algorithm 1 results in a lookup table for each period. The table for any period k gives the optimal leasing decision for every possible state (x_k, y_k) that could occur.

One drawback of using the standard DP algorithm in this way is that the size of the state space necessarily limits the size of the problem that we may consider. However, there exist suboptimal techniques that can be used to overcome the combinatorial explosion in the size of the state space. One such approach is based on neuro-dynamic programming (NDP), which is also known as reinforcement learning [2]. In the NDP approach a scoring function approximates the true cost function, and a compact representation of the state space is used which captures only the salient features of the state. Another approach to reducing the size of the state space in the underlying Markov Decision Process is found in [9]. In their work, Song et. al. model the state using separate Markov processes for each variable. When a change of state occurs in one process, a probabilistic mechanism based on the correlation between the variables effects a change in the other process. We are currently experimenting with these methods in order to solve larger instances of the leasing problem.

Algorithm 1: DP Algorithm for Resource Leasing

```
input  : Number of time periods $N$
input  : Total number of resources available $R$
input  : Deployment costs $c_1$ and $c_2$, leasing cost $h$
input  : Arrival Rate $\lambda$
input  : Service Rate $\mu$
output : Optimal leasing policy $J$
for k = N-1:0
    Compute $J_k(x_k, y_k)$ for each possible state;
    for x = 0:R
        for y = 0:R
            Calculate the expected cost for each
            admissible leasing decision;
            for u = y-x:R-x
                Compute the expected number of dynamic
                deployments;
                v = compute_V(t, y, x+u, mu,
                lambda);

                Compute the expected number of requests
                in execution at the beginning of period
                $k+1$;
                y_new = compute_Y(t, y, mu,
                lambda);

                Compute $x_{k+1}$ so that we can lookup
                $J_{k+1}(x_{k+1}, y_{k+1})$;
                x_new = max(y_new, x+u);
                cost_to_go = lookup(x_new,
                y_new, k);

                Compute the expected cost;
                g = c1*max(0, u) + c2*v +
                h*(x+u) + cost_to_go;

                Store g if it is the minimum cost found so
                far and keep track of the associated $u_k$;
                if g < min_cost then
                    min_cost = g;
                    u_star = u;
                end
            end
            Store the leasing decision $u_k$ that achieved
            the minimum cost;
            J(row, col, k) = u_star;
        end
    end
end
```

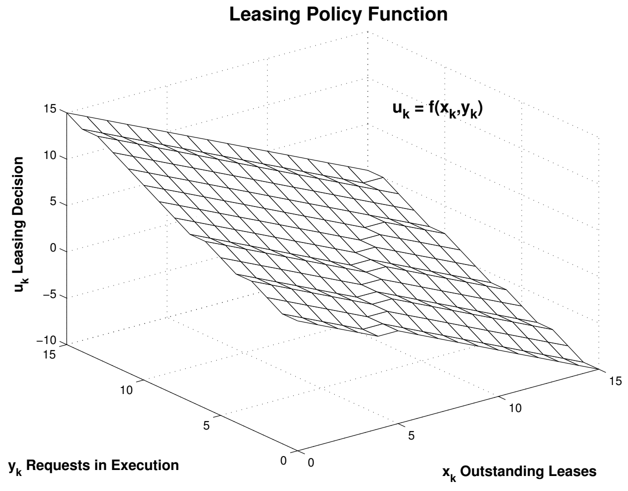

Figure 3. Optimal Leasing Policy

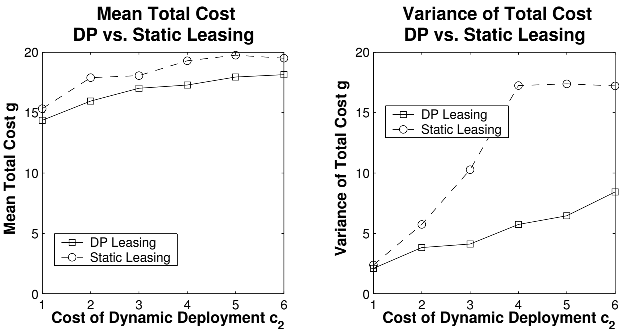

Figure 4. Mean and Variance of Total Cost

6. Results

In this section, we present results from a simulation study. The results show that by using the policy obtained from the DP algorithm, we can reduce not only the deployment and leasing costs, but also the variability of those costs. As a consequence, the SP can reduce the amount of uncertainty in the cost of hosting a grid service. Figure 3 shows how the leasing decision varies with the state. This graph represents the optimal leasing decision, u_k^*, for a single period. As shown in the figure, when the number of outstanding leases, x_k, increases, the number of new leases, u_k, decreases. Also, as the number of requests in execution at the beginning of a period, y_k, increases, the number of new leases, u_k, increases.

In our experiments, the largest component of the cost function was c_2, the cost of unplanned dynamic deployments. Therefore, we ran the DP algorithm for increasing values of c_2. We then ran two separate simulations for resource leasing for each value of c_2. One of the simulations made use of the results from the DP algorithm. We refer to this scenario as *DP leasing*. The other simulation did not use the DP results at all. In this scenario, which we refer to as *Static leasing*, the number of resources acquired at the beginning of each period was pre-determined by minimization (over u_k) of the cost function. In the Static leasing scenario the same number of planned resources were leased in each period, with unplanned dynamic deployment occurring whenever necessary. The specific parameters used during execution of the DP algorithm and during the subsequent simulations were

$N = 10$	Number of time periods
$R = 20$	Total number of resources available
$c_1 = 1$	Cost of planned deployment
$c_2 = 1 \ldots 6$	Cost of unplanned dynamic deployment
$h = 1$	Cost of holding a lease for one period
$\lambda = 10$	Average arrival rate of service requests (requests per period)
$\mu = 1$	Average service rate of computational resources (requests per period).

We performed 100 repetitions of each simulation. The averaged results for the mean and the variance of the total cost are presented in Figure 4. The plot on the left is the mean total cost. DP leasing results in an approximate 10% decrease in average total cost over Static leasing. Moreover, an even greater benefit can be seen in the plot on the right, which shows the variance of the total cost. We see that as the cost of unplanned dynamic deployments increases, the variance of the total cost for Static leasing increases at a much faster rate than for DP leasing. The variance for Static leasing ceases to increase when it gets very close to the mean value. (The cost is always positive.) We initially thought that the percentage of total cost attributed to unplanned dynamic deployments would be much greater for Static leasing as opposed to DP leasing, thereby contributing to the greater variance in total cost. Table 1 shows that this is not the case. The percentage of cost due to unplanned dynamic deployments is only slightly greater for Static leasing, and is not enough to account for the increased variance. Thus, the reduction in variance for DP leasing is due to the use of the policy derived from the DP algorithm. Table 1 was made for the case where $c_2 = 3$. The percentages are similar for other values of c_2. We conclude that the DP approach significantly reduces the variability of the cost of hosting of a dynamic grid service.

Cost Component	DP Leasing	Static Leasing
Planned deployment	10%	8%
Dynamic deployment	10%	15%
Cost to hold lease	80%	77%

Table 1. Percentages of Total Cost

7. Conclusion and Future Work

This work introduces a stochastic control model for deployment and hosting of a dynamic grid service. The objective of the model is to produce policies for leasing computational resources so that quality of service is maintained while the costs of deployment and leasing are kept to a minimum. The model is useful for making resource leasing decisions in the face of such uncertainties as random demand for the service and random execution times of service requests. By employing a dynamic programming approach, we were able to obtain both a model and a solution. Our cost function captures two types of deployment costs and the cost to hold a lease. Execution of the DP algorithm resulted in leasing policies that were subsequently used in a simulation study. The results from the simulation experiments show that the cost of deployment and leasing (hosting) is lower when using the DP policies. Just as important, we show that as the cost of unplanned dynamic deployment increases, use of the DP policies considerably reduce the variability of the total cost to the service provider.

The formulation and solution of the problem in this work are for a finite-horizon problem in which the SP intends to deploy the grid service for a particular length of time. Temporary need for high-performance simulations during and after natural or man-made disasters typify such services. We also want to consider the case where a grid service persists indefinitely. Our future work will include formulations for infinite-horizon problems of this type and experimentation with different solution methods. Associated with this work, we will provide a stopping criterion which will indicate, as demand for the service trails off, when it is more economical to take down the service and just provide on-demand dynamic deployment.

8. Acknowledgements

The authors would like to acknowledge the support of the National Science Foundation under grants NGS-0305641 and ITR-0325949, the Department of Energy's Office of Science under grant DE-FG02-03ER25554, and the Minnesota Supercomputing Institute and the Digital Technology Center at the University of Minnesota.

References

[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, second edition, 2000.

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

[3] R. Buyya et al. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[4] I. Foster et al. Grid services for distributed system integration. *Computer*, 35(6), 2002.

[5] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Margan Kaufmann, 1998.

[6] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[7] P. G. Hoel, S. C. Port, and C. J. Stone. *Introduction to Stochastic Processes*. Waveland Press, Inc., 1987.

[8] S. M. Ross. *Introduction to Probability Models*. Academic Press, Inc., fourth edition, 1989.

[9] B. Song, C. Ernemann, and R. Yahyapour. Parallel computer workload modelling with markov chains. In D. G. Feitelson and L. Rudolph, editors, *10th Workshop on Job Scheduling Strategies for Parallel Processing*, 2004. New York, NY.

[10] The Globus Alliance. The WS-Resource Framework. `http://www.globus.org/wsrf`.

[11] K. S. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications*. John Wiley and Sons, Inc., second edition, 2002.

[12] J. B. Weissman, S. H. Kim, and D. A. England. A Dynamic Grid Service Architecture. in submission, 2004.

[13] J. B. Weissman and B.-D. Lee. The service grid: Supporting scalable heterogenous services in wide-area networks. In *IEEE Symposium on Applications and the Internet*, 2001. San Diego, CA.

[14] J. B. Weissman and B.-D. Lee. The virtual service grid: An architecture for delivering high-end network services. *Concurrency: Practice and Experience*, 14(4):287–319, Apr. 2002.

[15] R. Wolski et al. G-commerce: Market formulations controlling resource allocation on the computational grid. In *International Parallel and Distributed Processing Symposium*, 2001.

[16] R. Wolski et al. Grid resource allocation and control using computational economies. In F. Berman, G. Fox, and A. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, chapter 32, pages 747–769. John Wiley and Sons, 2003.

A. Computation of $E\left[v_k\right]$ and $E\left[y_k\right]$

We use a probabilistic argument to determine the expected number of unplanned dynamic deployments, $E\left[v_k\right]$, and the expected number of service requests in execution at the beginning of a period, $E\left[y_k\right]$. The analysis is based on the $M/G/\infty$ queue, an infinite server queueing system with a Poisson arrival process. In reality, there will not be an infinite number of resources available; however, for analysis purposes, we assume that additional resources are available (at a cost of c_2 per resource) whenever an unplanned dynamic deployment occurs. The arrival rate of the Poisson process is λ and the service rate is μ. Although we have used Exponential service times in this work, the model allows for the service times to come from a general probability distribution.

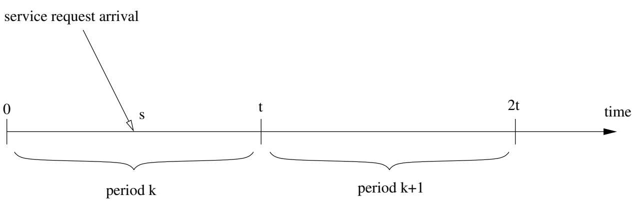

Figure 5. Arrival of a Service Request

Let us begin by computing $E\big[y_{k+1}\big]$. Consider Figure 5 and let the interval $(0,t]$ represent period k. We distinguish between two types of service requests: those that arrive in the interval $(0,t]$, and those that were already present and executing at time zero. For both types of service requests we are concerned with computing the probability that a request is still in execution at the beginning of period $k+1$, hence contributing to y_{k+1}. Consider a request that arrives at time s, $0 < s \leq t$. The probability that the request finishes execution by time t is, by definition, $F(t-s)$, where F is the cumulative probability distribution of the service times (i.e. the Exponential distribution in our case.) Consequently, the probability that the request is still in execution at time t is $1 - F(t-s)$. Given that the arrival occurred in the interval $(0,t]$, one result about the Poisson process states that the time of arrival s is uniformly distributed on $(0,t]$ (see, for example, Theorem 6.2 in [11]). Thus, for an arbitrary arrival in period k the unconditional probability that the request is still in execution at the beginning of period $k+1$ is

$$\begin{aligned} p_t = \frac{1}{t}\int_0^t 1 - F(t-s)ds = \frac{1}{t}\int_0^t 1 - F(x)dx \\ = \frac{1-e^{-\mu t}}{\mu t}. \end{aligned} \tag{2}$$

Let $N(t)$ be the total number requests that arrive during the interval $(0,t]$. Each request has an independent service time. Thus, each request has an independent probability p_t of still being in execution at time t. Let $W(t)$ be the number of requests still in execution at time t. Given $N(t) = n$, the conditional distribution of $W(t)$ is binomial with parameters n and p_t. Thus,

$$P\big[W(t) = j \mid N(t) = n\big] = \binom{n}{j} p_t^j (1-p_t)^{n-j},$$

and the unconditional distribution of $W(t)$ is, by the theorem of total probability,

$$\begin{aligned} P\big[W(t){=}j\big] &= \sum_{n=j}^{\infty} P\big[W(t){=}j \mid N(t){=}n\big] P\big[N(t){=}n\big] \\ &= \sum_{n=j}^{\infty} \binom{n}{j} p_t^j (1-p_t)^{n-j} e^{-\lambda t}\frac{(\lambda t)^n}{n!} \\ &= e^{-\lambda t p_t}\frac{(\lambda t p_t)^j}{j!}. \end{aligned}$$

Thus, the number of requests that arrive during period k and that are still in execution at the beginning of period $k+1$ is Poisson distributed with parameter

$$\lambda t p_t = \frac{\lambda}{\mu}(1 - e^{-\mu t}).$$

Now we must consider the other type of service request that was mentioned earlier, that is, a request that was already present and executing at the beginning of period k. Denote the number of such requests that are still in execution at time t by $Z(t)$. We note that $Z(t)$ is independent of $W(t)$ and that each of these requests has an independent probability of finishing execution by time t. Given that y of these requests are present at the beginning of period k, the conditional distribution of $Z(t)$ is binomial with parameters y and $e^{-\mu t}$. Thus,

$$P\big[Z(t) = j \mid Z(0) = y\big] = \binom{y}{j} e^{-\mu t j}(1 - e^{-\mu t})^{y-j}.$$

The conditional probability of $Z(t)$ suffices since we may observe the value of $Z(0)$ (i.e. the value of y_k) at the beginning of each period. Because $W(t)$ and $Z(t)$ are independent random variables we may compute the expectation of their sum by taking the sum of their expectations. Thus,

$$\begin{aligned} E\big[y_{k+1}\big] &= E\big[W(t) + Z(t)\big] \\ &= E\big[W(t)\big] + E\big[Z(t)\big] \\ &= \frac{\lambda}{\mu}(1 - e^{-\mu t}) + y_k e^{-\mu t}. \end{aligned}$$

We now turn our attention to the computation of $E\big[v_k\big]$, the expected number of unplanned dynamic deployments that will occur during a period k. For this analysis we again make use of results from queueing theory for the $M/G/\infty$ queue as well as results from the Poisson process. To begin, let us classify a service request as one of two possible types. A Type 1 request is a request that executes on a resource that was acquired through a planned deployment. So, a request is a Type 1 request if there is a free resource available (and already leased) when the request arrives. A Type 2 request is one that executes on a resource that was acquired through an unplanned dynamic deployment. Thus, a request that arrives and finds that all leased resources are busy is a Type 2 request. It is the Type 2 requests that contribute to v_k.

We have defined two possible types of requests and we note that a request must be one of these two types. Suppose that a service request arrives at time s, $0 < s \leq t$ (referring again to Figure 5.) Let $P_1(s)$ be the probability that the request is a Type 1 request. Similarly, Let $P_2(s)$ be the probability that the request is a Type 2 request, where $P_1(s) + P_2(s) = 1$. Now let $N_2(t)$ be the number of Type 2 requests occurring by time t, that is, the number of unplanned dynamic deployments v_k. A useful result of the Poisson process states that $N_2(t)$ is a Poisson random variable with mean

$$E\big[N_2(t)\big] = E\big[v_k\big] = \lambda \int_0^t P_2(s)ds \tag{3}$$

(see Proposition 3.3 in[8], for example). The only remaining task is to derive an expression for $P_2(s)$, the probability that a service request arrival finds all leased resources busy and therefore an unplanned dynamic deployment will occur. In any period k the total number of planned leased resources is $x_k + u_k$. So an unplanned dynamic deployment will occur whenever a request arrives and $x_k + u_k$ or more resources are busy. Here we make use of the previous definitions of $W(t)$ and $Z(t)$. Suppose that there are j resources busy at time s. Then $W(s) + Z(s) = j$ because an executing request must have either begun execution in the interval $(0, s]$, or it must have already been in execution at time zero. Given that we begin period k with y requests already in execution, the expression for $P_2(s)$ is

$$\begin{aligned} P_2(s) &= P\big[W(s) + Z(s) \geq x_k + u_k \mid Z(0) = y\big] \\ &= \sum_{j=x_k+u_k}^{\infty} \Big[\sum_{k=0}^{\min(y,j)} P\big[Z(s) = k\big] P\big[W(s) = j - k\big] \Big], \end{aligned}$$

where

$$\begin{aligned} P\big[Z(s) = k\big] &= \binom{y}{k} e^{-\mu sk}(1 - e^{-\mu s})^{y-k}, \\ P\big[W(s) = j - k\big] &= \frac{e^{-\lambda s p_s}(\lambda s p_s)^{j-k}}{(j-k)!}, \end{aligned}$$

and p_s is computed as in Equation (2). For numerical execution of the DP algorithm the evaluation of the integral in Equation (3) was done using the `trapz` function in matlab. This function computes an approximation of the integral by constructing trapezoids which approximate the area under the function.