TR 04-007

A Stochastic Control Model for Leasing Computational Resources

Darin England and Jon Weissman

February 20, 2004

A Stochastic Control Model for Leasing Computational Resources

Darin England and Jon Weissman

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

March 2004

Abstract

We present a model for determining the optimal resource leasing policy for a dynamic grid service. The model assumes that the demand for the service as well as the actual execution times are unknown, but can be estimated. We cast the problem in a Dynamic Programming framework and we are able to show that the model can make good resource leasing decisions in the face of such uncertainties. In particular, we use the model to decide how many resources should be leased for the service and for how long. The results show that use of the model reduces the cost of leasing computational resources and significantly reduces the variance of the cost.

1 Introduction

The success of web services has influenced the way in which grid applications are being written [1]. Grid application designers are now beginning to make use of software services, which provide a specific functionality to the application, such as solving a system of equations or performing a simulation. Just as software is viewed as a service, a provider of computational resources may also be viewed as providing a service. Our view is that grid applications will make use of grid services and will be dynamic in that they can can adapt to the changing conditions in the grid [8]. In such an environment, a Service Provider (SP) may acquire computational resources through a leasing arrangement. One concern for the SP is the amount of computational resources that are required in order to provide an adequate quality of service. The amount of resources needed may vary and is a function of the demand for the service and the compute-intensive nature of the service. In this article, we address the situation where the demand for the service is unknown, but comes from a known probability distribution. In addition, even though the SP will know the processing requirements of the service, the actual execution times will not be known prior to executing a service request due to differences in input data as well as competition from other applications and services (i.e. we assume a time-sharing environment.) Another concern for the SP is how long to host the service. At some point, demand for the service will trail off and it may be more economical to process service requests in an "on-demand" fashion. In this article, we propose a model for making resource leasing decisions in the face of such uncertainties. The decisions are made periodically and are based on the expected average demand for the service and the expected average performance of the resources. We model the arrival and the processing of service requests as a stochastic process in which the interarrival times and the execution times come from known probability distributions. The resource leasing problem is then cast in a Dynamic Programming framework. The result is a leasing policy which tells the SP how many resources to lease in each period in order to provide an adequate level of service while keeping the cost of deployment and hosting to a minimum. An important result is that the DP approach to resource leasing can greatly reduce variance of the total leasing cost.

2 Dynamic Programming

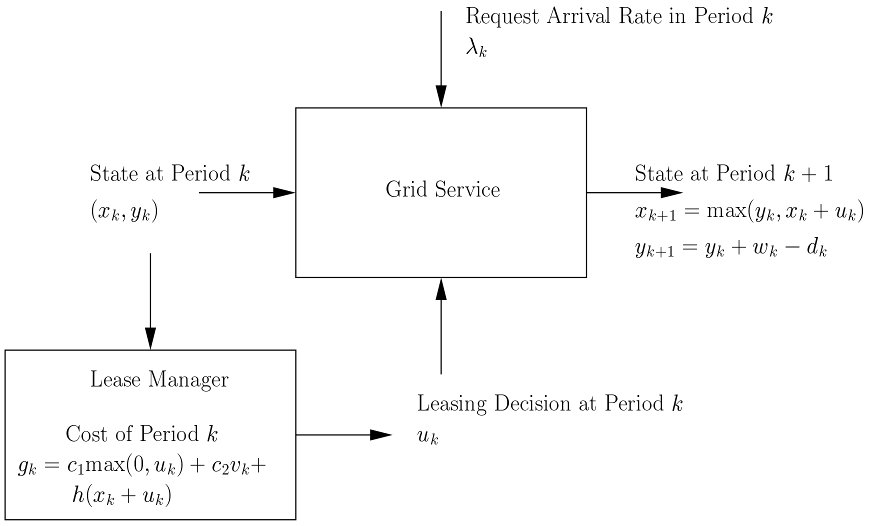
Dynamic Programming (DP) is an approach for modelling and for solving optimization problems in which periodic decisions must be made under some level of uncertainty. The DP approach is also known as stochastic optimal control [2]. The idea of DP is to solve a minimization problem in each period, beginning with the last period and ending with the initial period. The optimal decision will depend 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. At each stage, 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 allocate more resources than are currently available in the resource pool. The overall solution will give us an optimal policy for leasing additional resources in each period.

3 Model for Resource Leasing

We use the DP approach to model and solve a stochastic decision problem for leasing computational resources in order to deploy and and host a dynamic grid service. At each stage, 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.

3.1 State

We need to define a representation of the state of the dynamic grid service. The state should be a compact summary of the available information that affects the decision to lease



Request Arrival Rate in Period k
λ_k
State at Period k
(x_k, y_k)
Grid Service
State at Period $k+1$
$x_{k+1} = \max(y_k, x_k + u_k)$
$y_{k+1} = y_k + w_k - d_k$
Lease Manager
Cost of Period k
$g_k = c_1\max(0, u_k) + c_2v_k + h(x_k + u_k)$
Leasing Decision at Period k
u_k

Figure 1: A Stochastic Control Model

resources. 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 [2]. 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.

3.2 Decision Variable

Suppose that an SP wishes to host a grid service for some number of periods N. At the end of the Nth period, the service is no longer hosted. The objective of the leasing problem is to acquire enough resources to meet the expected demand for the grid service while keeping the cost of leasing to a minimum. If too many resources are leased, then the service provider incurs unnecessary cost. If too few resources are leased, then the SP must acquire additional resources in an *on-demand* fashion, but at a higher cost. An architecture that can accommodate such dynamic deployment of a grid service is described in [8]. 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. u_k may be negative, in which case the decision is to "take down" the grid service on u_k resources.

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 $y_k - x_k$

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 in the resource pool.

3.3 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 computing systems[1] and it is also very amenable to analytical analyses. We denote the demand for a grid service in period k by λ_k, which represents the average arrival rate of service requests for period k. The arrival rate may vary with time, indicating a non-stationary Poisson process.

3.4 Execution Times

We assume that the execution time of a service request is unknown until the request has finished executing. Although the SP will know the performance characteristics of the service, we assert that *exact* execution times cannot be predicted due to 1) unknown characteristics of the input data for any particular request, and 2) the request may need to compete with other applications/services for processing time. The execution time is then 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 and decision variables and using random demand and execution times in this manner results in a discrete-time dynamic system as shown in Figure 2. The state of the system evolves according to

$$x_{k+1} = \max(y_{k+1}, x_k + u_k). \tag{3.1}$$

Note that y_{k+1} is a random variable that is a function of the number of service requests and their execution times in period k. Specifically,

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

where a_k is the number of arrivals in period k and d_k is the number of departures in period k.

3.5 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 uses bandwidth and disk space. Executing service requests uses CPU cycles and memory. The objective of our model is to provide an

[1]However, the Poisson model is not appropriate for modelling the arrival of individual packets [5].

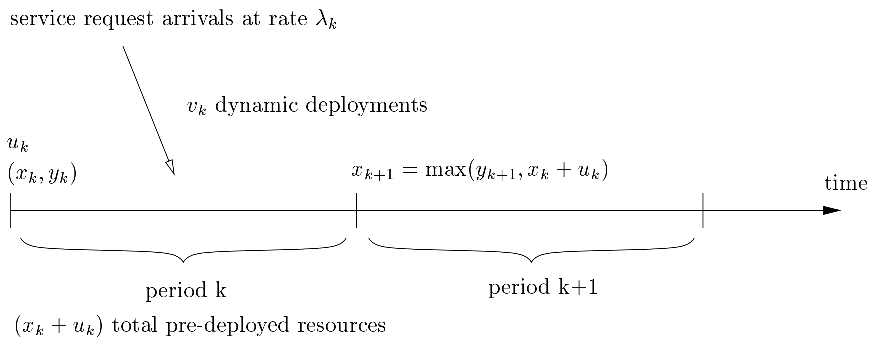

service request arrivals at rate λ_k
v_k dynamic deployments
u_k
(x_k, y_k)
$x_{k+1} = \max(y_{k+1}, x_k + u_k)$
time
period k
period k+1
$(x_k + u_k)$ total pre-deployed resources

Figure 2: A Discrete-time Dynamic System

adequate number of resources for the grid service while keeping the cost to a minimum. We assume that the resource provider charges for two different types of deployment as well as charging for leasing the computational resources. The first type of cost is associated with pre-planned deployment of the grid service. These deployments occur at the beginning of a period and last for at least one full period. A pre-planned deployment means that the service provider deploys the grid service in anticipation of service request arrivals. The second type of cost is incurred whenever there is an unplanned dynamic deployment of the grid service. This type of deployment may happen in the middle of a period whenever a service request arrival cannot be served because all resources are busy serving other requests[2]. We denote the number of such deployments during a period k as v_k. The third type of cost is the cost to hold a resource for one or more periods. One may think of the deployment cost as the setup cost, and the cost to hold a resource as the traditional cost of leasing. The three cost components are

$c_1 \max(0, u_k)$	The cost of pre-planned static deployment, where c_1 is the cost per leased resource
$c_2 v_k$	The cost of unplanned dynamic deployment, where c_2 is the cost per deployment
$h(x_k + u_k)$	The cost to hold a lease, where h is the cost per resource per period.

We assume that $c_1 < c_2$, otherwise it would always be cheaper to just dynamically deploy the grid service for every request arrival. The total cost, g_k, charged by the resource provider in a given period is the sum of the three cost components. Because v_k and x_k are random

[2]Thus, we assume that there is no queueing of service requests.

variables, we must compute the expected cost 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 (3.1), 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.

4 DP Algorithm for esource Leasing

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 $J_k(x_k, y_k)$.

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

In the equation above, 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 from $N-1$ to 0, the cost-to-go is computed for every possible state (x_k, y_k) and for every admissible leasing decision u_k. The optimal allocation decision for a given state is the one that minimizes $J_k(x_k, y_k)$. We denote the optimal decision as u_k^*, and the optimal cost-to-go as $J_k^*(x_k, y_k)$. The DP algorithm for the resource leasing problem is presented as Algorithm 1. The pseudo-code is presented in matlab-*like* notation. Algorithm 1 results in a lookup table for each period. The table for period k gives the optimal allocation decision for every possible state (x_k, y_k) that could occur.

5 esults

The purpose of using a Dynamic Programming approach to model and solve the resource leasing problem is to make the best possible leasing decisions in the presence of uncertain demand and execution times. Execution of the DP algorithm provides us with a minimum cost leasing decision for each possible state that the system may enter. Our hypothesis is that, by using the results from the DP algorithm, we can reduce not only the cost of leasing computational resources, but also the variability of the cost. Thus, we can reduce the amount of uncertainty when leasing resources for the deployment of a dynamic grid service. Figure 3 graphically shows how the leasing decision varies with the number of already leased resources and the number of requests in execution at the beginning of a period. This graph represents the optimal leasing decision u_k^* for a single period. In general, as the number of

Algorithm 1: DP Algorithm for Resource Leasing

```
input  : Number of time periods $N$
input  : Total number of resources available $R$
input  : Static deployment cost $c_1$, dynamic deployment cost $c_2$, leasing cost $h$
input  : Arrival Rate $\lambda$
input  : Service Rate $\mu$
output: Lookup table $J$, gives an optimal leasing policy
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 $J_{k+1}(x_{k+1})$;
                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
```

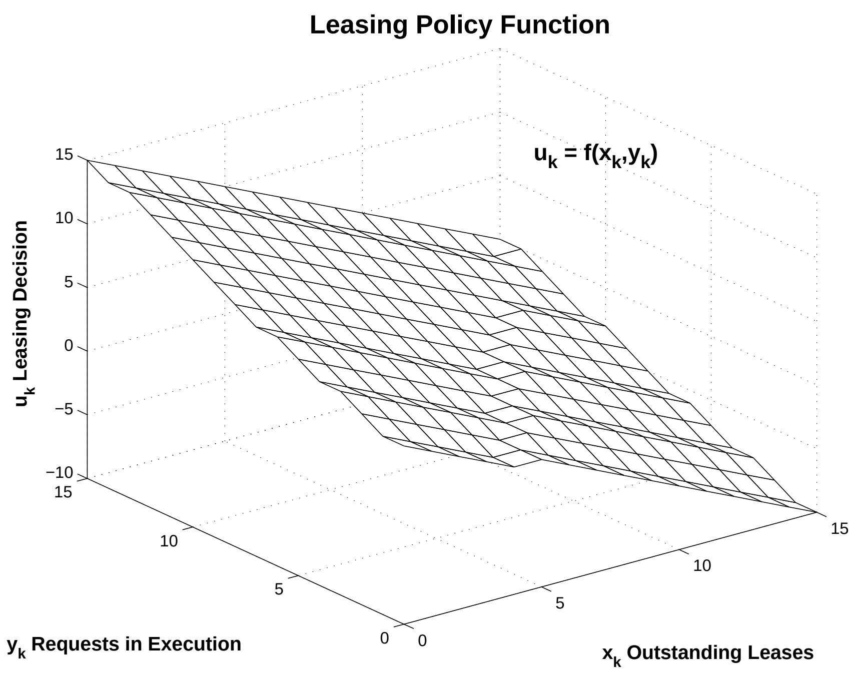


Figure 3: Optimal Leasing Policy

already leased resources, x_k, increases, the number of newly leased resources, u_k, decreases. Also, as the number of requests in execution at the beginning of a period, y_k, increases, the number of newly leased resources, u_k, increases.

To test the effectiveness of using the DP approach, we ran the DP algorithm for increasing values of c_2, the cost of unplanned dynamic deployment. 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 for making resource leasing decisions. 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 determined by minimization (over u_k) of the cost function. For Static leasing, the same number of pre-planned resources were leased in each period, with unplanned dynamic deployment occurring whenever an arriving request found that all resources were busy. 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 in the resource pool
$c_1 = 1$	Cost of pre-planned deployment
$c_2 = \{1, 2, 3, 4, 5, 6\}$	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 replications of each simulation. The averaged results for the mean and the variance of the total leasing cost, $g = c_1 \max(0, u) + c_2 v + h(x + u)$, are presented in Figure 4. The plot on the left is the mean total cost. We see that a lower cost is achieved with DP leasing, although the difference is only about 3 units. However, the mean does not tell the complete story. The real benefit to using the DP approach is revealed in the plot on the right, which shows the variances 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 much faster rate than for DP leasing. One may think that the percentage of unplanned dynamic deployment cost is much greater for Static leasing as opposed to DP leasing, thereby contributing to the greater variance in total cost. Figure 5 shows that this is not the case. The percentage of dynamic deployment cost is only slightly greater for Static leasing, and not enough to account for the increased variance. Figure 5 was made for the case where $c_2 = 3$. The percentages are similar for other values of c_2. We conclude that using the DP approach can significantly reduce the variability of leasing costs for the deployment and hosting of a dynamic grid service.

6 Conclusion

We have presented a stochastic control model for leasing computational resources for the purpose of deploying and hosting a dynamic grid service. The model is useful for making resource leasing decisions in the face of such uncertainties as unknown demand for the service and unknown execution times. By employing a Dynamic Programming approach, we were able to obtain both a model and a solution. Our cost function captures the cost of deployment as well as the cost to hold a lease. The results from the simulation experiments show that this cost is lower when using the DP approach to resource leasing. More importantly, as the cost of unplanned dynamic deployments increases, use of the DP approach considerably reduces the variability of the total cost.

7 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

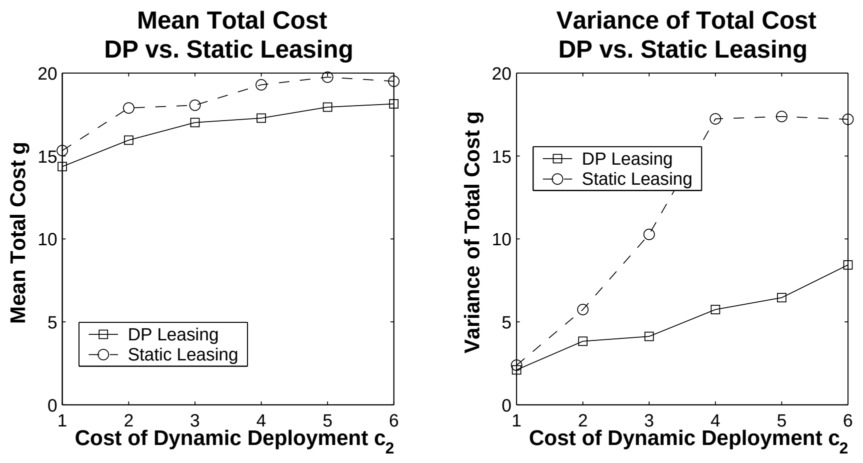


Figure 4: Mean and Variance of Total Cost

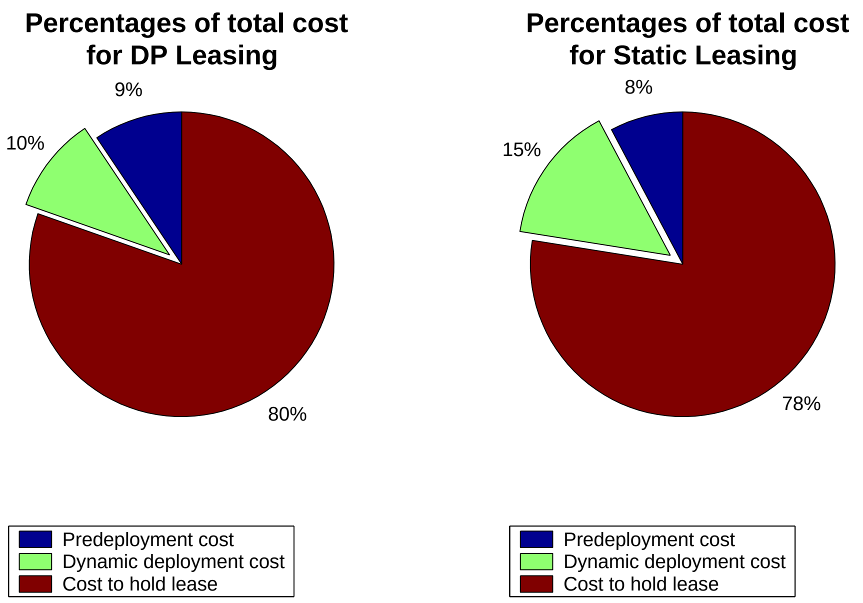


Figure 5: Percentages of Total Cost

under grant DE-FG02-03ER25554, and the Minnesota Supercomputing Institute and the Digital Technology Center at the University of Minnesota.

eferences

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

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

[3] Darin A. England and Jon B. Weissman. Costs and benefits of load sharing in the computational grid. Technical Report 2003-033, Army High-Performance Computing Research Center, 2003.

[4] Paul G. Hoel, Sidney C. Port, and Charles J. Stone. *Introduction to Stochastic Processes.* Waveland Press, Inc., 1987.

[5] V. Paxson and S. Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3), 1995.

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

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

[8] Jon B. Weissman, Seonho H. Kim, and Darin A. England. A Dynamic Grid Service Architecture. submitted to SC2004, 2004.

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, that is, 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.

Let us begin by computing $E\left[y_{k+1}\right]$. Consider Figure 6 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

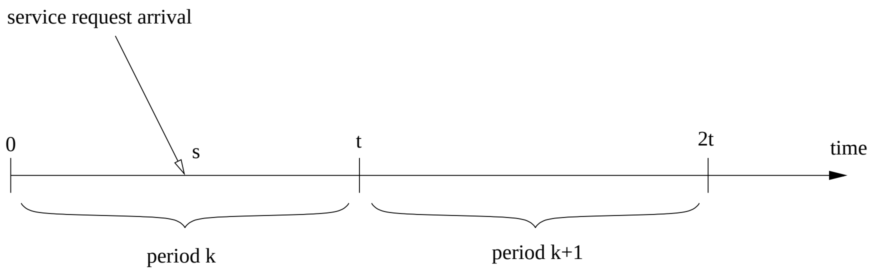


Figure 6: Service Request Arrival

still in execution at the beginning of period $k+1$, hence contributing to y_{k+1}. Consider a request that arrives at a 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 $(0,t]$[3]. 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

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

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 $X(t)$ be the number of requests still in execution at time t. Given $N(t) = n$, the conditional distribution of $X(t)$ is binomial with parameters n and p_t. Thus,

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

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

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

[3]See, for example, Theorem 6.2 in [7]

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 $Y(t)$. We note that $Y(t)$ is independent of $X(t)$ 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 $Y(t)$ is binomial with parameters y and $e^{-\mu t}$. Thus,

$$P\big[Y(t) = j \mid Y(0) = y\big] = \begin{matrix} y \\ j \end{matrix} \; e^{-\mu t j}(1 - e^{-\mu t})^{y-j}.$$

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

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

We now turn our attention to the computation of $E\big[v_k\big]$, that is, 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 pre-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 6.) 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[4] of the Poisson process states that

[4]See Proposition 3.3 in[6], for example.

$N_2(t)$ is a Poisson random variable with mean

$$E\big[N_2(t)\big] = E\big[v_k\big] = \lambda \int_0^t P_2(s)ds. \tag{A.2}$$

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 pre-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 $X(t)$ and $Y(t)$ for the $M/G/\infty$ queue. Suppose that there are j resources busy at time s. Then $X(s) + Y(s) = j$ because an executing request must have either began 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

$$\begin{aligned} P_2(s) &= P\big[X(s) + Y(s) \geq x_k + u_k \mid Y(0) = y\big] \\ &= \sum_{j=x_k+u_k}^{\infty} \Big[\sum_{k=0}^{\min(y,j)} P\big[X(s) = k\big] P\big[Y(s) = j - k\big]\Big] \\ &= \sum_{j=x_k+u_k}^{\infty} \Big[\sum_{k=0}^{\min(y,j)} \binom{y}{k} e^{-\mu s k}(1 - e^{-\mu s})^{y-k} \frac{(\lambda s p_s)^{j-k} e^{-\lambda s p_s}}{(j-k)!}\Big], \end{aligned}$$

where p_s is computed as in Equation (A.1). For numerical execution of the DP algorithm, the evaluation of the integral in Equation (A.2) 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.