Collecting Unused Processing Capacity: An Analysis of Transient Distributed Systems

Leonard Kleinrock and Willard Korfhage*
Computer Science Department
University of California, Los Angeles

Abstract

Distributed systems frequently have large numbers of idle computers and workstations. If these could be harnessed, then considerable computing power would be available at low cost. We analyze such systems using a very simple model of a distributed program (a fixed amount of work) to see how the use of transient processors affects the program's service time.

1 Networks of Transient Processors.

Networks of computers are fairly common in business and research environments throughout the world. Originally motivated by a desire to ease data and device sharing, many networks have grown in speed and sophistication to the point that distributed processing can be performed on them. These networks vary in size from a handful of personal computers on a low-speed network, to thousands of workstations and a variety of larger machines on a high-speed, fiber-optic network. A typical example is that of workstations on a high-speed, local area network in a research laboratory. Not only are there many machines, well connected by the network, but the users are likely to demand more and more computing power.

On these networks, we often have the situation that many of the personal computers and workstations are sitting idle, waiting for their users, and thus being wasted. If we could recover this wasted time for useful processing, then we would have considerable computing power available to us at low cost [6]. We refer to these processors, which are sometimes busy and sometimes not, as *transient* processors.

This situation has analogies to time-sharing. In the past, institutions had one large resource, a mainframe computer, that was shared by many users. As networks of workstations develop, the largest computing resource is no longer a single machine, but is instead the aggregate computing power of the workstations. In the same manner that it has been possible for many people to share a single machine, by using the idle time of one person to run the program of another, so it is now possible to share a network of workstations for large, distributed computations.

Whether this is technically feasible or not depends on a variety of factors, such as the characteristics of the communications medium, the characteristics of the computers, and the statistical characteristics of the user population. Mutka and Livny [13], and Nichols [14] have shown that under at least some circumstances, this is very practical and useful.

From an analytical point of view, we would like to have a queueing model of a network of transient processors executing distributed programs. In this paper we take a first step toward this end by analyzing the service time for a very simple model of a distributed program (a fixed amount of work) to see how the use of transient processors affects what would otherwise be a deterministic service process.

2 The Model.

Assume that we have a network of M identical processors, and we wish to run a program that will require a total of W seconds of work. In general, a program consists of multiple stages of work, each of which must be completed before the start of the next. For this paper we assume that the program has only one stage of work, and furthermore, we assume that the work in any stage is infinitely divisible, and therefore is always spread evenly among the available processors.

Each processor has a capacity of one second of work per second. A processor alternates between a *non-available period*, when someone is using it, and an *available period*, when it is sitting idle. We assume that the length of non-available periods is randomly distributed with mean t_n and variance σ_n^2, and that the length of available periods is also randomly distributed from a (possibly different) distribution with mean t_a and variance σ_a^2. We wish to run our program on the network of processors, using processors while they are in their available periods. The finishing time of the program is given by a random variable $\tilde{f}$, with probability distribution $f(t)$, average $\overline{f}$, and Laplace transform $F^*(s)$. The purpose of this paper is to find

*This work was supported by the Defense Advanced Research Projects Agency under contract MDA 903-82-C0064, Advanced Teleprocessing Systems, and contract MDA 903-87-C0663, Parallel Systems Laboratory.

$f(t)$, or, in broader terms, to examine the potential use of currently wasted cycles. To do so, we examine a related function, $w(u \mid t)$, the probability density function of the amount of work, u, completed by time t, which has mean $\overline{W_t}$ and variance $\mathrm{Var}[W_t]$. Then, for a given W, $f(t)$ is the distribution of the time for the completed work to *first* reach W, a point also known as the *first passage time*.

If the amount of work that we wish to do is small, relative to t_a and t_n, then the finishing time is highly dependent upon the states of the processors when the program arrives. If, for example, we have a small amount of work to do on a single processor, either the processor is available with no delay, or the processor is non-available, and we must wait, on average, t_n seconds (assuming exponentially distributed non-available periods) before we can even start the work. If t_n is not small relative to W, then state of the processor when the program arrives strongly affects the finishing time distribution. In this paper, we use two techniques to mitigate the effect of this on the analysis. The single processor models make assumptions about the time that the program arrives (either in an available period or at the beginning of a non-available period). The multiprocessor model assumes that W is large relative to t_a and t_n, so the effect of initial conditions is negligible. Work is underway for the situations where W is not large.

Our analysis ultimately allows for arbitrary distributions of the lengths of available and non-available periods, but in our examples, we assume exponential distributions. Using the average available and non-available times measured by Mutka and Livny, the examples in this paper are generally based on the following parameters: $t_a = 91$ minutes, $t_n = 31.305$ minutes, $W = 10^3$ minutes for single processor examples, and $W = 10^4$ minutes for multiprocessor ($M = 100$) examples. We choose W large relative to t_a and t_n for the reasons mentioned in the previous paragraph.

We ignore communications overhead and task precedence issues in this paper, and assume that our program can use all available processors at any given time. The results of this paper, then, provide upper (i.e. optimistic) bounds on the best performance achievable in this situation.

In the remainder of this paper, we first discuss previous work by others, then analyze the problem for 1 processor. We find expressions for $w(u \mid t)$ and $f(t)$ in the single processor case. We then use $w(u \mid t)$ to find $f(t)$ for M processors and compare it to the single processor case.

3 Previous Work.

A single transient processor can be modeled in a variety of ways: as a priority queue, as a queue with vacations, as an unreliable system, or as a cumulative, alternating renewal process. Of these methods, we choose that last because it provides the asymptotic distribution of $w(u \mid t)$ in a very simple form and for arbitrary available and non-available period distributions. We now briefly discuss the other alternatives.

3.1 Preemptive Priority Queues.

We can model a transient processor using a preemptive priority queueing system with two classes. The high priority class represents non-available periods that interrupts the service of distributed programs, represented by the jobs in the low priority class. Such systems do not completely model a transient processor because high priority jobs continue to arrive while a high priority job is in service, and this represents a queueing of non-available periods. If we do not object to this, and if we assume that both available and non-available periods arrivals are from (different) Poisson processes, then the Laplace transform of the "completion time" (our $F^*(s)$) is known [8], and from that we can get its mean and variance.

3.2 Queues with Vacation.

Another model of a transient processor is a queueing system in which the server goes on vacations. In particular, we require that the vacations occur randomly and preempt any customer in service. If the available periods are exponentially distributed, Gaver [2] provides an analysis. For non-exponential available periods, Federgruen and Green [5] have analyzed such queues.

3.3 Unreliable Systems.

Reliability analysis concerns itself with the availability of a system over time, and some work has been done to find the distribution of cumulative availability (the cumulative amount that a system is available over time). This corresponds exactly to accumulation of work in a network of transient processors. For a system with two states (available and non-available) Donatiello and Iyer [4] find the transform of the cumulative availability, and they derive a closed-form expression when the time in each state is exponentially distributed. Using their results, we can find the moments of the amount of work done in a given amount of time for any specific available and non-available period distributions. However, we were unable to derive general results in terms of the distributions' moments, and therefore we turned to the cumulative, alternating renewal analysis described later in this paper. Although, their results do not directly provide the distribution of the first passage time, the transform of this may be obtained using techniques from the cumulative, alternating renewal analysis.

De Souza e Silva and Gail [3] discuss the calculation of cumulative availability in systems which can be modeled as homogeneous Markov chains. They find a general method for calculating cumulative availability, and further find good techniques for numerical evaluation of their method. The SAVE (System Availability Estimator) program [7] implements their method. For numerical, not approximate analytic, results, this is an excellent approach.

4 Results for One Processor.

We use two methods for deriving for the finishing time of a program on one processor. The first analysis provides the distribution of the first passage time with some restrictions on the distributions of the available and non-available periods (see section 4.1); however, it does not yield the distribution of the amount of work done over time. The second analysis, using a cumulative, alternating renewal process, provides us with the distribution of $w(u\,|\,t)$ for arbitrary available and non-available period distributions, but only with the transform of the first passage time distribution. Given $w(u\,|\,t)$ from the second analysis, we can then develop a model for M processors.

We may easily derive the averages $\overline{W_t}$ and $\overline{f}$. In a long period of time, the processor spends, on average, a fraction $t_a/(t_a+t_n)$ of its time in available periods. Thus in t seconds, the average amount of work completed is t times this fraction, or

$$\overline{W_t} = \frac{t_a}{t_a+t_n}\,t. \tag{1}$$

Because the processor completes, on average, $t_a/(t_a+t_n)$ seconds of work per second, the reciprocal of this is the number of seconds it takes to complete one second of work. Multiplying this by W we find the average first passage time,

$$\overline{f} = \frac{t_a+t_n}{t_a}\,W. \tag{2}$$

If we wish to account for multiple processors, the average amount of work done per time period is increased by a factor of M, the number of processors, and the first passage time is likewise decreased by a factor of M, giving us:

$$\overline{W_t} = \frac{t_a}{t_a+t_n}\,Mt. \tag{3}$$

and

$$\overline{f} = \frac{t_a+t_n}{Mt_a}\,W \tag{4}$$

As we will find, this simple analysis accurately characterizes the system when our program takes a long time relative to the average length of the available and non-available periods.

4.1 Direct Analysis.

Assume, for the moment, that we have only one processor ($M=1$). If our program starts when the processor is available, as shown in Figure 1, it will finish at time $W+\hat{w}$, where $\hat{w}$ is the *additional* (wasted) time the

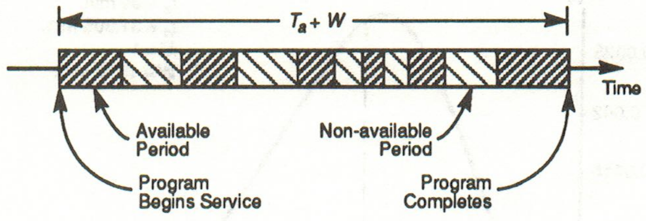

Figure 1: Time for one processor to finish W seconds of work.

program spends in the system because the processor is sometimes in non-available periods.

Because the non-available periods are independent and identically distributed random variables with exponential distributions, $P_{\hat{w}}(t)$, the probability distribution function of the additional time in non-available periods, given that we have i such periods, is an Erlang distribution; that is

$$\frac{dP_{\hat{w}}(t|i)}{dt} = \frac{(1/t_n)(t/t_n)^{i-1}}{(i-1)!}e^{-t/t_n} \tag{5}$$

The number of non-available periods that "arrive" during W seconds has a Poisson distribution with rate $1/t_a$. Thus, unconditioning on this number, we get that the density of $\hat{w}$ is

$$\frac{dP_{\hat{w}}(t)}{dt} = \tag{6}$$
$$\begin{cases} e^{-W/t_a}u_0(t) & \text{if } t=0 \\ \sum_{i=1}^{\infty} \frac{(1/t_n)(t/t_n)^{i-1}}{(i-1)!}e^{\frac{-t}{t_n}}\,\frac{(W/t_a)^i}{i!}e^{\frac{-W}{t_a}} & \text{if } t>0 \end{cases}$$

where $u_0(t)$ is a unit impulse at $t=0$. For $t>0$, this may be further reduced to

$$\frac{dP_{\hat{w}}(t)}{dt} = \frac{1}{t_n}\sqrt{\frac{tW}{t_at_n}}e^{-t/t_n}e^{-W/t_a}I_1(2\sqrt{\frac{tW}{t_at_n}}) \tag{7}$$

where $I_1(z)$ is the modified Bessel function of the first kind of order 1. Figure 2 shows the distribution of the first passage time using $t_a = 91$ and $t_n = 31.305$, and $W = 10^3$.

To find the mean and variance of this distribution, it simplifies the analysis to model the sequence of non-available periods using a series-parallel queueing server, shown in Figure 3, and then use known results for such servers. In this model, each of the infinite number of sub-servers (the numbered circles in the figure) has the same distribution as a single non-available period, and, in fact, represents a non-available period. We adjust the transition probabilities p_i so that the number of sub-servers a program passes through has the same distribution as the number of non-available periods arriving in W seconds. Upon entering the series-parallel server, the job immediately leaves the server with probability p_0, or the job enters sub-server

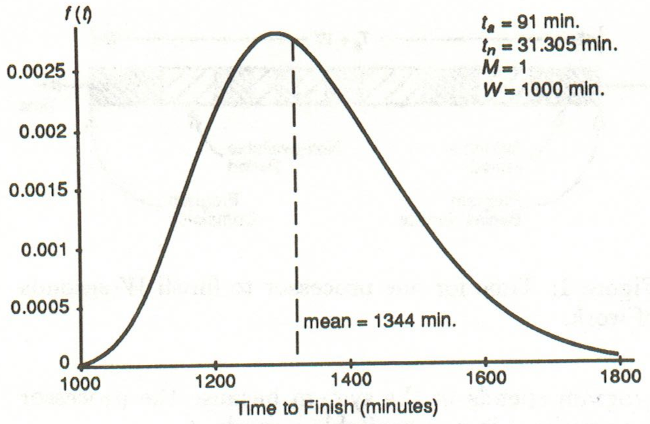

Figure 2: First Passage Time Density for Direct Analysis.

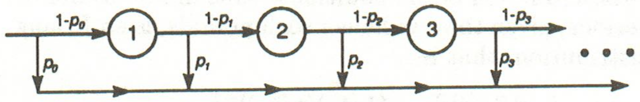

Figure 3: Series-parallel model of extra time needed to complete W seconds of work.

1 with probability $1-p_0$. Then each time a job leaves a sub-server i, a similar choice is made between leaving the series-parallel server (with probability p_i) or continuing on to the next sub-server (with probability $1-p_i$). For our purposes, we adjust the probabilities p_i so that the number of sub-servers passed through has a Poisson distribution. To compute these p_i, we note first that $p_0 = e^{-t_a W}$. For notational convenience, let $\mathcal{P}_\lambda(i) = \frac{\lambda^i}{i!}e^{-\lambda}$, the i^{th} term of a Poisson distribution with parameter λ. Next, we see that $(1-p_0)p_1 = \mathcal{P}_{W/t_a}(1)$, and we immediately have that

$$p_1 = \frac{\mathcal{P}_{W/t_a}(1)}{1-\mathcal{P}_{W/t_a}(0)}.$$

It may be proved by induction that

$$p_i = \frac{\mathcal{P}_{W/t_a}(i)}{1-\sum_{j=0}^{i-1}\mathcal{P}_{W/t_a}(j)}. \tag{8}$$

Using these p_i's, and the expression in Kleinrock [10] for $\hat{W}^*(s)$, the Laplace transform of the density of the time required to pass through a series-parallel server, we get

$$\hat{W}^*(s) = p_0 + \sum_{i=1}^{\infty} \frac{(W/t_a)^i}{i!} e^{-(W/t_a)} \left(\frac{1/t_n}{s+1/t_n}\right)^i \tag{9}$$

which, after some manipulation, becomes

$$\hat{W}^*(s) = \mathcal{P}^*_{W/t_a}\left(\frac{1/t_n}{s+1/t_n}\right) \tag{10}$$

where $\mathcal{P}^*_\lambda(z) = e^{\lambda(z-1)}$ is the z-transform of a Poisson distribution with parameter λ.

We multiply this by e^{-Ws} to account for the W seconds that we are required to work, and finally find the transform for the distribution of time to finish W seconds of work, i.e.,

$$F^*(s) = e^{-Ws}e^{(W/t_a)((1/t_n)/(s+1/t_n)-1)}. \tag{11}$$

Taking derivatives, we find that the mean time required to finish W seconds of work is

$$\overline{f} = W\frac{t_a+t_n}{t_a}, \tag{12}$$

and the variance of this time is

$$\sigma_f^2 = 2\frac{t_n^2 W}{t_a} \tag{13}$$

Note that Equation 12 agrees with Equation 2. This mean and variance may also be derived by viewing the distribution as a constant plus a random (Poisson-distributed) sum of Erlang random variables, and using the well-known formulas for the mean and variance of a random sum of random variables [10].

We can modify this analysis for non-exponential non-available time distributions, but it still requires exponential available times so that the number of non-available periods in W seconds has a Poisson distribution. If we wish to allow arbitrary available period distributions as well, then we need a more sophisticated method of counting non-available periods, and this leads to analyzing the situation as a renewal process.

4.2 Analysis as a Cumulative, Alternating Renewal Process

To allow arbitrary distributions for the available and non-available periods, we follow the presentation of Cox [1] in his treatment of cumulative, alternating renewal processes. As before, the single processor has a capacity of one second of work per second, and we wish to find the distribution of the time required for this processor to finish W seconds of work. Let X_i' be the duration of the i^{th} non-available period, and X_i'' be the duration of the i^{th} available period, as shown in Figure 4. Let the renewal points for our alternating renewal process be the beginning of the non-available periods, shown as heavy dots in the figure; the time between renewal points is then $X_i = X_i' + X_i''$. This has mean $E[X] = t_a + t_n$ and variance $Var[X] = \sigma_a^2 + \sigma_n^2$. We assume that $t = 0$ occurs at the beginning of a renewal period.

To form a cumulative process from this, define W_i to be the amount of work completed in each renewal period: $W_i = X_i''$, with mean t_a and variance σ_a^2. Let Z_t be the sum of all the available time up to time t, excepting that in the current available period, if the

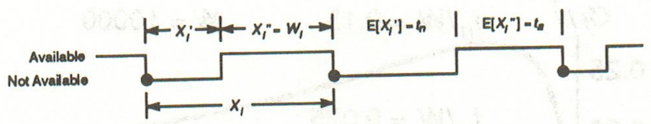

Figure 4: Cumulative Renewal Process

process is in such a period at time t.

$$Z_t = \sum_{i=1}^{N_t} W_i \quad (N_t = 1, 2, \ldots) \qquad (14)$$
$$= 0 \quad (N_t = 0)$$

where N_t is the number of renewals in $(0, t]$. Asymptotically, Z_t has the same properties as $w(u \mid t)$, the process that accumulates the true total available time up to time t; however, Z_t's analysis is more tractable than that of $w(u|t)$.

Cox's analysis allows the available and non-available periods to have arbitrary distributions. For t large, Z_t is the sum of many independent random variables, and it is asymptotically normal with mean

$$\mathrm{E}[Z_t] = \frac{t_a}{t_a + t_n}\, t \qquad (15)$$

and variance

$$\mathrm{Var}[Z_t] = \frac{2(\sigma_a^2 t_n^2 + \sigma_n^2 t_a^2)}{(t_a + t_n)^3}\, t. \qquad (16)$$

Using exponentially distributed available and non-available periods, these become:

$$\mathrm{E}[Z_t] = \frac{t_a}{t_a + t_n}\, t \qquad (17)$$

$$\mathrm{Var}[Z_t] = \frac{2(t_a t_n)^2}{(t_a + t_n)^3}\, t. \qquad (18)$$

As noted before, the asymptotic properties of Z_t are identical to the asymptotic properties of $w(u|t)$.

Comparing this to the result of the direct analysis, we note that for t equal to the mean first passage time (Eqn. 12), we have done, on average, W seconds of work, as we expected.

In his derivation, Cox, assuming that W_i is independent of X_i'' for all i, derives a double transform for $f(t)$. Unfortunately, this transform is very difficult to invert, but the asymptotic distribution of Z_t is really what we need so we can use it in the next analysis.

5 Results for M Processors

Because the amount of work done by one transient processor is the sum of a (possibly large) number of available periods, the total work done in time t is asymptotically normal with mean and variance given in Equations 17 and 18, respectively. If $t_a \ll W$ and $t_n \ll W$, then it is reasonable to use Brownian motion as a model of an M processor system. All M processors are assumed to be independent, so the amount of work done by time t is the sum of M independent, (approximately) normally distributed random variables, with mean

$$\bar{b}t = \frac{t_a}{t_a + t_n} Mt = p_a Mt \qquad (19)$$

and variance

$$\sigma_b^2 t = \frac{2(\sigma_a^2 t_n^2 + \sigma_n^2 t_a^2)}{(t_a + t_n)^3}\, Mt. \qquad (20)$$

For exponential available and non-available distributions, Eqn. 20 becomes:

$$\sigma_b^2 t = \frac{2(t_a t_n)^2}{(t_a + t_n)^3} Mt = \frac{2p_a^2(1 - p_a)Mt}{t_n} \qquad (21)$$

where $p_a = t_a/(t_a + t_n)$.

With these as the parameters for our Brownian motion, the density of the time, t, that it takes for M

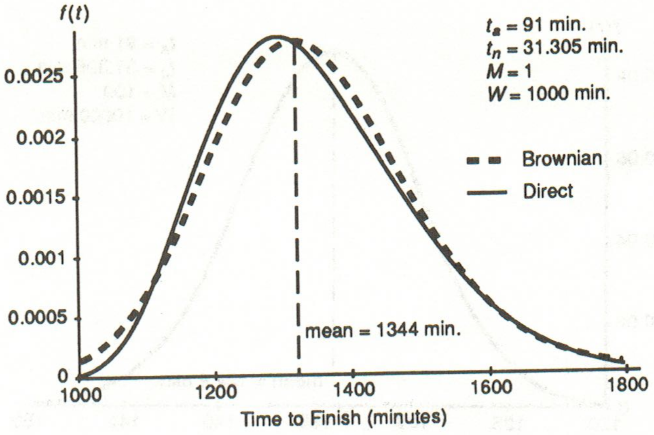

Figure 5: First Passage Time Distributions for Direct and Brownian Motion Analyses.

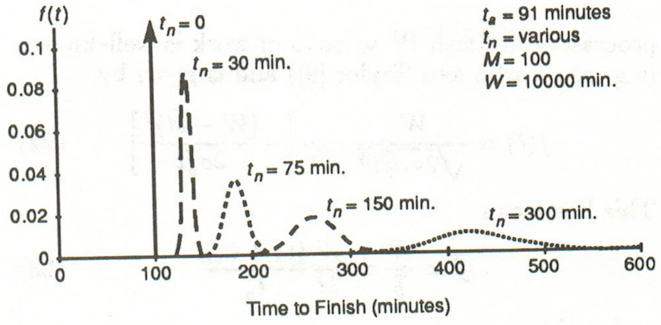

Figure 6: First Passage Time Densities for Brownian Motion Model for Various t_n.

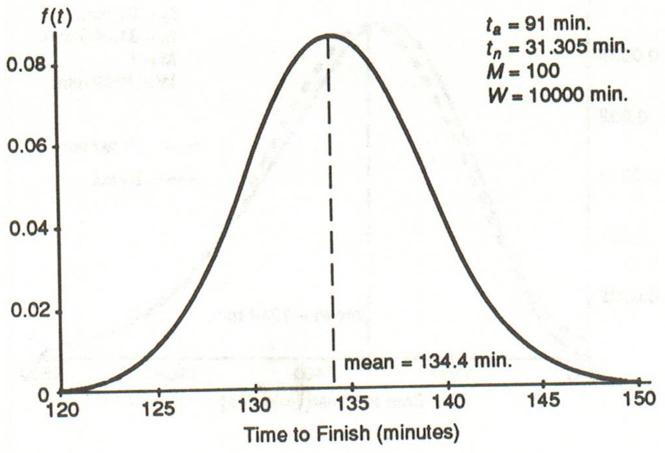

Figure 7: First Passage Time Density for Brownian Motion Analysis.

processors to finish W seconds of work is well-known (e.g. see Karlin and Taylor [9]) and is given by:

$$f(t) = \frac{W}{\sqrt{2\pi\sigma_b^2 t^3}} \exp\left[-\frac{(W-\bar{b}t)^2}{2\sigma_b^2 t}\right] \tag{22}$$

This has mean

$$\bar{f} = \frac{W}{\bar{b}} = \frac{W}{M}\frac{(t_a+t_n)}{t_a} \tag{23}$$

and variance

$$\sigma_f^2 = \frac{W}{\bar{b}}\frac{\sigma_b^2}{\bar{b}^2}. \tag{24}$$

and for exponential distributions, the latter becomes

$$\sigma_f^2 = \frac{2W}{M^2}\frac{t_n^2}{t_a}. \tag{25}$$

Equation 22 is the main result of this paper. Note that it makes no assumptions about the distributions of the available and non-available periods, except that their variances are finite, and only the distributions' means and variances appear in the first passage time.

Note that for the case $M = 1$, this mean and variance agree with our single processor analysis of section 4.1. The first passage time densities for both the single processor analysis and the multiprocessor analysis with $M = 1$ are shown in Figure 5.

Figure 6 shows the distribution of first passage time for various t_n with $t_a = 91, M = 100$, and $W = 10^4$. Using the $t_a = 91$ and $t_n = 31.305$, our job of 10^4 minutes would take about a week to run on a single, dedicated processor. When run on a network of 100 transient processors, it would take 134.06 minutes, or about 2.25 hours. The distribution of this first passage time is shown in Figure 7. Note that although $W \gg t_a$ and $W \gg t_n$, we have that W/M, the finishing time if the processors were fully dedicated to the program, is close to the average length of an available period, and it is remarkable that the curve is still so symmetrical.

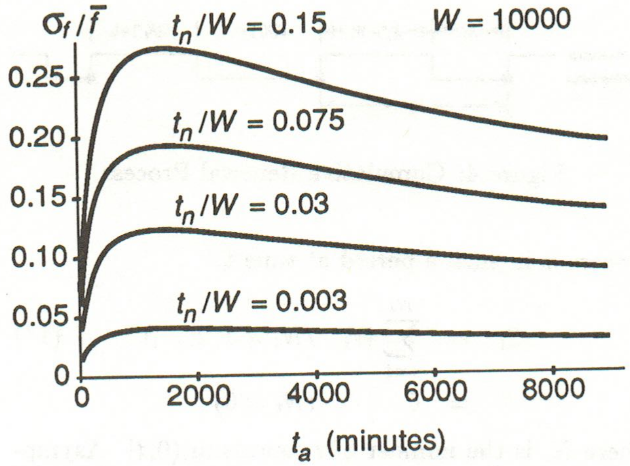

Figure 8: $\sigma_f/\bar{f}$ with various t_a and t_n.

It is useful to examine the ratio of σ_f to $\bar{f}$, namely:

$$\frac{\sigma_f}{\bar{f}} = \frac{\sqrt{\sigma_b^2}}{\sqrt{\bar{b}W}} \tag{26}$$

to see what happens to the distribution as the parameters change. For exponential distributions this becomes

$$\frac{\sigma_f}{\bar{f}} = \sqrt{\frac{2t_n}{W}\frac{\sqrt{t_n/t_a}}{1+t_n/t_a}}, \tag{27}$$

and it is this equation that we examine in more detail. Figure 8 plots Eqn. 27 for fixed t_n and W, and varying t_a. Note that in this figure, $t_n = 30$ minutes is the lowest line and $t_n = 1500$ minutes is the uppermost line.

Because we assumed $t_n \ll W$, this ratio tends to be less than 1, reaching a peak when $t_a = t_n$. If we fix t_n/W and let t_n/t_a go to infinity (which implies $t_a \to 0$), the ratio goes to 0. We explain this by noting that for small t_a, it takes very many available–non-available cycles before the work is finished. The law of large numbers insures that the first passage time distribution, which is the sum of these many periods, will then be tight about its mean.

If, on the other hand, we let $t_a \to \infty$, the ratio of the standard deviation to the mean goes to zero once again. When t_a is large relative to t_n, the non-available periods become negligible, as if the processors are always available. Again, the first passage time distribution becomes very tight about its mean because non-available time periods add little variability to the finishing time.

Given that the first passage time distribution is tight about its mean, (i.e. $t_a \gg t_n$, or $t_a \ll t_n$, or $W \gg t_n$), it may be accurate enough to consider the distribution as an impulse at the mean finishing time (in the spirit of the law of large numbers). Using the previous example again, we find $\sigma_f^2 = 21.53$, and approximating $f(t)$ as a normal distribution (discussed below), we find that 90% of the time, programs requiring 10^4 minutes of work will finish within 7.6 minutes of the 134.4 minute mean finishing time.

The central-limit theorem says that $f(t)$ will tend toward a normal distribution when many available—non-available periods occur before the program completes (i.e. $W \gg t_a$ and $W \gg t_n$). To approximate the first passage time, we use a normal distribution with the same mean and variance as the first passage time distribution:

$$\hat{f}(t) = \frac{1}{\sqrt{2\pi\sigma_f^2}} e^{-(t-\overline{f})^2/(2\sigma_f^2)} \tag{28}$$

When the mode of the first passage time distribution is close to its mean, a normal distribution well approximates the first passage time. The mode is:

$$t_{mode} = \frac{1}{2}\sqrt{\frac{9(\sigma_b^2)^2}{\overline{b}^4} + \frac{4W^2}{\overline{b}^2}} - \frac{3\sigma_b^2}{2\overline{b}^2}. \tag{29}$$

Comparing this to the mean, we find that the percentage difference between the two is approximately $3\sigma_b/2\overline{b}W$, which, as we would expect, shrinks as W grows.

6 Conclusion

We have analyzed a network of transient processors, and determined the probability density of the length of time it takes to finish a fixed amount of work. The main result for an M processor network is given in Equation 22, and it is valid for general available and non-available period distributions. Simulations confirm that Brownian-motion-with-drift is an accurate model of system performance under the assumptions given above. With large programs that run for a long time relative to the length of available and non-available periods, the central limit-theorem applies, and the Brownian-motion-with-drift model remains good regardless of the distributions of the available and the non-available periods. Under these assumptions, the distribution of finishing time will be very tight about its mean, and is well approximated by a normal distribution.

It does remain to account for communication overhead and precedence relationships, but it is likely that these can be accommodated, or at least approximated, within the model.

The analysis in this paper has not examined the effect of multiple programs in the network. We may now use the first passage time distribution (Eqn. 22), as the service time in a queueing system that represents the network. If each job gets the whole network and they must queue, then a G/G/1 queue is a good model. If all the programs in the network share the processors equally, then we could model the network as an M/G/1 processor-sharing system. The analysis of such systems remains for a future paper.

It is well known that, for a given total processing capacity, the average response time is shortest if we use one large processor rather than many small processors [11]. From this perspective, the trend toward individual workstations is a curious one. However, this result assumes that each program executes on only one processor. If we distribute the program over *all* the small processors, then we may recover, at least partially, the response time advantages of a large, central system, while retaining the advantages of individual workstations.

References

[1] D. R. Cox. *Renewal Theory.* Methuen and Co., Ltd., London, science paperbacks edition, 1962.

[2] Jr. D. P. Gaver. A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society*, B24:73, 1962.

[3] Edmundo de Souza e Silva and H. Richard Gail. Calculating Availability and Performability Measures of Repairable Computer Systems Using Randomization. *Journal of the ACM*, 36(1):171–193, January 1989.

[4] Lorenzo Donatiello and Balakrishna R. Iyer. Closed-Form Solution for System Availability Distribution. *IEEE Transactions on Reliability*, R-36(1):45–47, April 1987.

[5] A. Federgruen and L. Green. Queueing Systems with Service Interruptions. Research Working Paper 84-5, Columbia University, 1984.

[6] Robert Felderman, Eve Schooler, and Leonard Kleinrock. The Benevolent Bandit Laboratory: A Testbed for Distributed Algorithms. *IEEE Journal on Selected Areas in Communications*, 7(2):303–311, February 1989.

[7] Ambuj Goyal. System Availability Estimator (SAVE) User's Manual Version 2.0 (External). Technical Report RC 12517 (No. 56267), IBM Watson Research Center, February 1987.

[8] Daniel P. Heyman and Matthew J. Sobel. *Stochastic Models in Operations Research, Volume 1: Stochastic Processes and Operating Characteristics.* Series in Quantitative Methods for Management. McGraw Hill, 1982.

[9] Samuel Karlin and Howard M. Taylor. *A First Course in Stochastic Processes.* Academic Press, second edition, 1975.

[10] Leonard Kleinrock. *Queueing Systems, Volume 1: Theory.* John Wiley and Sons, 1975.

[11] Leonard Kleinrock. Distributed Systems. *Communications of the ACM*, 28(11):1200–1213, November 1985.

[12] Willard Korfhage. *Distributed Systems and Transient Processors.* PhD dissertation, University of California, Los Angeles, 1989.

[13] Matt W. Mutka and Miron Livny. Profiling Workstation's Available Capacity for Remote Execution. Computer Sciences Technical Report 697, CS Dept., Univ. of Wisconsin, May 1987.

[14] David A. Nichols. Using Idle Workstations in a Shared Computing Environment. In *Proceedings of the Eleventh ACM Symposium on Operating System Principles*, pages 5–12. ACM, November 1987.