ANALYSIS OF THE NUMBER OF OCCUPIED PROCESSORS IN A MULTI-PROCESSING SYSTEM

Abdelfettah Belghith
Leonard Kleinrock

August 1985
CSD-850027

ANALYSIS OF THE NUMBER OF OCCUPIED PROCESSORS IN A MULTI-PROCESSING SYSTEMS *

Abdelfettah Belghith
Leonard Kleinrock
University of California, Los Angeles

ABSTRACT

We consider a multi-processor system consisting of a set, say P, of identical processors. A computer job is represented as a set of tasks partially ordered by some precedence relationships and represented as a directed acyclic graph called Process Graph. In such a graph, nodes represent tasks and edges represent precedence relationships between these tasks. Many parameters are in play to characterize the terrain of our multi-processing system. These are: the job arrival process,the process graph description (number of nodes, number of levels, number of tasks and their distribution among the levels, and the precedence relationships among the tasks), the task processing requirement, and the number of processors P in the system.

In this report, we investigate the distribution of the number of occupied processors, the average and variance of such number of occupied processors, and the probability distribution of the interarrival times between tasks to the system. We find that the average number of occupied processors in the system is independent of the number of levels in the process graph, the placement of tasks among levels, the precedence relationships among the tasks, the distribution of the task service time requirement, the distribution of the job arrival process and the number of available processors in the multi-processing system. Such average number of occupied processors in the system is found to be solely a function of the average number of tasks per job, the average rate of the job arrival process and the average task service time.

* This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense under contract MDA 903-82-C-0064

Table of Contents

List of Figures

page

CHAPTER 1
MODEL DESCRIPTION

A computer job is a set of tasks partially ordered by some precedence relationships. We represent a job by a Directed Acyclic Graph (DAG), hereafter called a Process Graph (PG). A node in the process graph represents a given task, and an edge (i,j) between node i and node j represents the precedence relationship between task i and task j. Edge (i,j) is used to prevent the start of task j execution unless task i execution has been completed. The tasks (i.e., nodes) in the process graph are, therefore, distributed into levels. Tasks at level one are said to be"starting tasks," and tasks at the last level in the process graph are said to be "terminating tasks." Any two tasks can be executed concurrently (i.e., in parallel) if, and only if, every predecessor of one task does not include the other task and vice versa. Figure (1) gives an example of such a process graph. The edges are implicitly directed downward.

The multi-processor system under consideration consists of a set, say P, of identical processors. Each processor is capable of executing any task. The number of processors, P, can be finite or infinite. Let us define the following quantities:

N = random variable counting the number of task in a job, $N \geq 1$

r = random variable counting the number of levels in a job, $1 \leq r \leq N$

X = random variable representing the task service time

Y = random variable representing the number of occupied processors

Since each level in the process graph must contain at least one task, it follows that the total number of process graphs having N tasks and r levels is equal to the number of ways to distribute (N-r) tasks among the r levels. This number of ways is [*]:

$$\binom{(n-r)+r-1}{r-1} = \binom{N-1}{r-1} \tag{I.1}$$

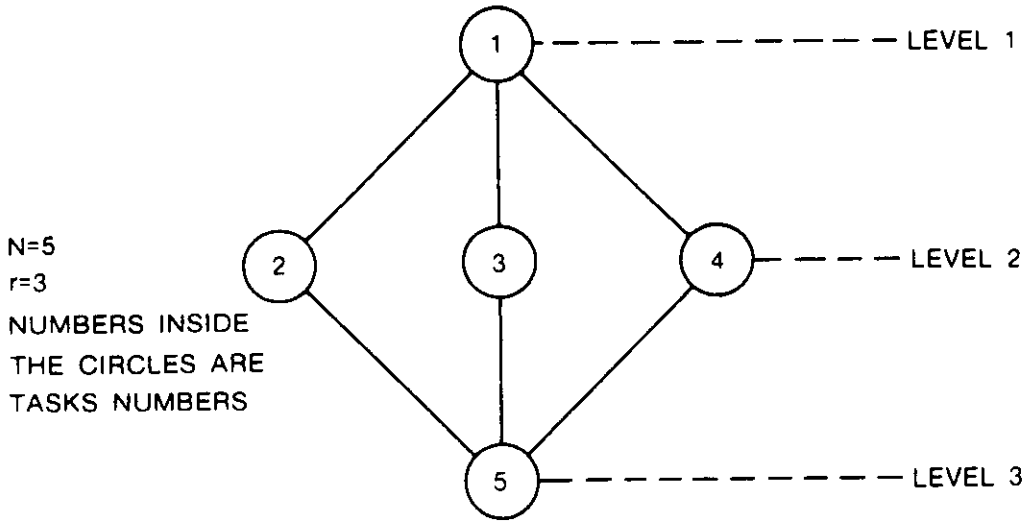


FIGURE I.1
Example of a Process Graph

PROPOSITION I.1

For a fixed number of tasks per job, say N, and fixed number of levels per job, say r, and for $1 \le n \le N-r+1$ and $1 \le k \le r$, the probability of having n tasks at level k is given by:

1. If r=1, then:

$$P[\ n \text{ tasks at level } 1\ ] = \begin{cases} 0 & \text{if } n \neq N \\ 1 & \text{if } n = N \end{cases}$$

---

* In fact, the ordinary generating function of such number of ways is:

$$\left[x + x^2 + + \cdots + x^k + \cdots\right]^r = x^r \left[1-x\right]^{-r}$$

since each level can have from 1 to N-r+1 tasks in it, the number of ways to distribute N tasks among the r levels such that no level is left empty, is the coefficient of x^N in the generating function. By the binomial theorem, Liu Introduction Combinatorial we have:

$$\left[1-x\right]^{-r} = \sum_{i=0}^{\infty} \binom{r+i-1}{i} x^i \quad \textit{thence,}$$

$$x^r \left[1-x\right]^{-r} = \sum_{i=0}^{\infty} \binom{r+i-1}{i} x^{r+i}$$

$$= \sum_{N=r}^{\infty} \binom{N-1}{r-1} x^N$$

2. If $r \geq 2$, then:

$$P[\ n \text{ tasks at level } k\ ] = \frac{\binom{N-n-1}{r-2}}{\binom{N-1}{r-1}}$$

PROOF

1. The proof of case 1 is trivial since for r=1 all tasks must be at this level.

2. For N and r fixed, from (I.1) we know that the total number of process graphs we can have is $\binom{N-1}{r-1}$.

   Consider now level k. We want n tasks at this level where $1 \leq n \leq N-r+1$. Thence, there are (N-n) tasks for the (r-1) remaining levels. The number of ways to distribute these (N-n) tasks among the (r-1) levels so that no level is left empty (i.e., the number of process graphs with (r-1) levels and (N-n) tasks) is given by:

$$\binom{\left[(N-n)-(r-1)\right]+(r-1)-1}{(r-1)-1} = \binom{N-n-1}{r-2}$$

   This number also represents the number of possibilities of level k $1 \leq k \leq r$ having n tasks out of a total of $\binom{N-1}{r-1}$ possibilities. The probability P[n tasks at level k] is therefore the ratio between them.

∎

Notice that level k can be any level, that is, $1 \leq k \leq r$. The minimum number of tasks any level can have is one and, therefore, the maximum number of tasks any level can have is (N-r+1).

PROPOSITION I.2

For a fixed number of tasks per job, say N, the probability that a randomly chosen process graph has r levels, $1 \leq r \leq N$, is given by:

$$P[\ \text{process graph has } r \text{ levels}\ ] = \frac{\binom{N-1}{r-1}}{2^{N-1}}$$

This conditional probability for a process graph with r levels, given that the number of tasks is fixed to N, is a binomial distribution b(r-1,N-1, $\frac{1}{2}$).

PROOF

$$P[\text{ process graph has r levels / N tasks in it }] = \frac{\text{number of process graphs with r levels and N tasks}}{\text{total number of process graphs with N tasks}}$$

But we have

$$\text{Total number of process graphs with N tasks} = \sum_{r=1}^{N} \binom{N-1}{r-1} = \sum_{r=0}^{N-1} \binom{N-1}{r} = 2^{N-1}$$

and we know that the number of process graphs with r levels and N tasks is $\binom{N-1}{r-1}$.

Thence,

$$P[\text{ process graph has r levels / N tasks in it }] = \frac{\binom{N-1}{r-1}}{2^{N-1}}$$

$$= \binom{N-1}{r-1} \left[\frac{1}{2}\right]^{r-1} \left[\frac{1}{2}\right]^{N-r}$$

$$= b(r-1, N-1, \frac{1}{2})$$

Let $\bar{r}$ denote the mean number of levels in a randomly chosen process graph, given that such graph has N tasks. Then, we have:

$$\bar{r} = \sum_{k=1}^{N} k\, P[\text{ process graph has k levels / N tasks in it }]$$

$$= \sum_{k=1}^{N} b(r-1, N-1, \frac{1}{2})$$

which gives

$$\bar{r} = \frac{N-1}{2} \tag{I.2}$$

The idea of a Markovian Process graph, denoted hereafter by MPG(X,Y), where X is the number of nodes (i.e., states) and Y is the number of levels, was first introduced by Towsley Towsley 78 and extensively used by Kung. Kung Kleinrock Random Graphs Kung Dissertation

A Markovian Process Graph is a Markovian state transition diagram generated for process graph PG, assuming infinite number of processors in the multi-processor system, where each state in the Markov chain represents a specific set of tasks in PG that can be executed in parallel. Following the notations used in Kung Dissertation , let C_α represent a state in the MPG where α is the set of tasks that are executed concurrently, and let $|\alpha|$ represent the number of tasks in the state α. The chain starts with state C_f, where f is the starting task (s) in PG. For each state C_α in the MPG, it will go to $|\alpha|$ other states, each branch corresponding to the termination of one of the tasks in α. The state $C_{\alpha'}$ at the end of one of these branches has the set of active tasks { α' }, where α' includes the tasks in α minus the completed task plus the activation of several other tasks, if any, due to the termination of this task. An exact algorithm and some example are given in . Kung Dissertation Figure (I.2) below gives the Markovian Process Graph for the Process Graph given in figure (I.1).

- circles represent states
- numbers inside circles are concurrent tasks numbers
- numbers near circles are state numbers
- numbers on edges are finishing task numbers

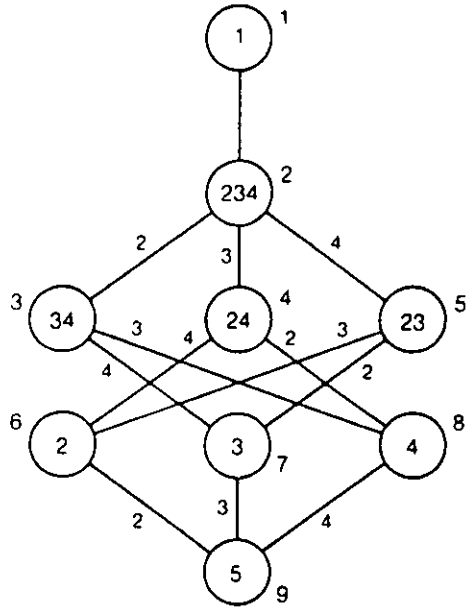


FIGURE I.2
Markovian Process Graph for the Process Graph Given in Figure I.1

In this technical report, we shall investigate the following:

1. the distribution of the number of occupied processors;

2. the average number of occupied processors; and

3. the distribution of the interarrival times between tasks to the system.

Many parameters are in play to characterize the terrain of the multi-processor system under investigation. These are:

1. *job arrival process:* Throughout this report, we assume that the job arrival process is Poisson with parameter λ. In chapter 5, we shall relax this assumption and consider a

general arrival process with independent job interarrival times.

2. *process graph:* characterized by:
   * the number of nodes;
   * the number of levels;
   * the distribution of tasks among the levels; and
   * the precedence relationships among the tasks.

3. *task processing requirement:* We shall consider three types. These are
   * constant service time per task;
   * exponential task service time; and
   * random task service time.

4. *number of processors:* Throughout this report, we assume an infinite number of processors. In chapter 5, though, we relax this assumption and consider the case of a finite number of processors.

This report is organized into six chapters. In chapter 2 we shall investigate the case of a Fixed Process Graph. All jobs have the same Process Graph with a fixed number of tasks and fixed number of levels. The Z-transform, the average and the variance of the number of occupied processors will be derived. The service time per task is considered to be constant, the same for all tasks. Chapter 3 will deal with semi-random Process Graphs with two or more levels. The case of just one level is treated in chapter 2. Each job has a process graph with a fixed number of tasks and fixed number of levels, but the distribution of tasks among levels varies from one job to another. The service time per task is kept constant, the same for any task. In this chapter, we shall also derive the Z-transform, the average and the variance of the number of occupied processors. Chapter 4 will deal with the case of a random Process Graph. Each job has a random Process Graph with fixed number of tasks but random number of levels (not exceeding the number of tasks). The Z-transform and the average number of occupied processors will be derived. The service time per task is considered to be fixed, the same for all tasks. Chapter 5 provides a detailed investigation of the average number of occupied processors. For the first time in this report we shall also consider the case of finite number of processors, exponential and random service time per task and completely random process graphs (e.g., the number of tasks per process graph is random, and the number of levels is also random but not exceeding the total number of tasks). Also, in this chapter, we shall relax the assumption that the job arrival process is Poisson and consider a general job arrival process with independent interarrival times between successive jobs. We shall find out in this chapter that the average number of occupied processors is independent of the number of levels in the process graph, the distribution of tasks among the levels, the precedence relationships among the tasks, the distribution of the task service time, the distribution of the job arrival process and the number of available processors in the system. Such average number of occupied processors will be shown to be only a function of the average number of tasks per process graph, the the average rate of the job arrival process and the average task service time. In chapter 6, we shall provide the distribution of the interarrival times between tasks to the system.

Two cases are investigated, exponential task service time and constant task service time.

CHAPTER 2
FIXED PROCESS GRAPH

In this chapter, we shall derive a closed form expression of the Z-transform of the distribution of the number of occupied processors. Closed expressions for the average and variance of the number of occupied processors will also be derived. Throughout this chapter, we shall make the following assumptions about the multi-processor system we have under investigation:

a. the number of processors in the system, P, is infinite;

b. the task service time, X, is constant, say 1 unit service time per task;

c. the process graph is fixed for all jobs . All jobs have the same process graph with a fixed number of tasks, say N, and fixed number of levels, say r. Moreover, if we let $J(n_1, n_2, \ldots, n_r)$ be the description of the process graph, where n_i is the number of tasks at level i in the process graph, then we require that all jobs have the same process graph description.

Let us define the following quantities:

Interval I = is the closed interval [t-r,t]; see figure (II.1).
Y_i = random variable counting the number of processors occupied at time t by the jobs that arrived in slot i in the interval I . A job arriving in slot i will participate with its (r-i+1)th level at time t. Such job will then occupy n_{r-i+1} processors at time t.
Y = random variable counting the total number of occupied processors at time t.

Thence, we have:

$$Y=\sum_{i=1}^{r} Y_i \qquad \text{(II.1)}$$

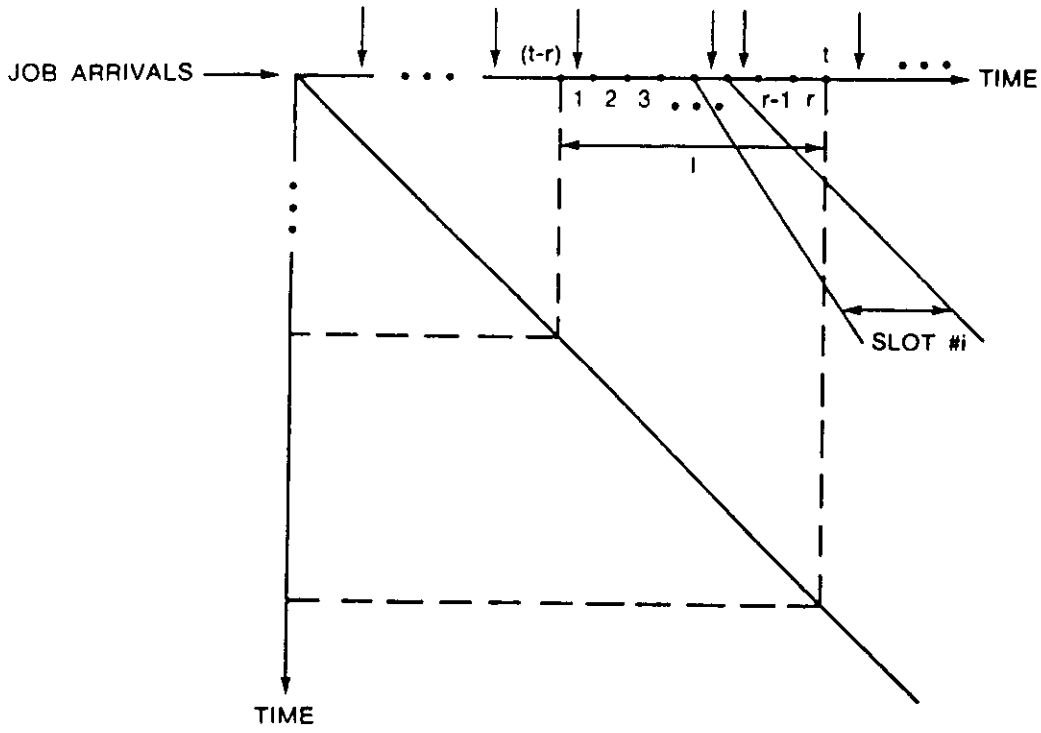


FIGURE II.1
Analysis of a Fixed Process Graph

$$P[Y_i=x] = \sum_{k_i=0}^{\infty} P[Y_i=x \ / \ k_i \text{ jobs arrived in slot } i].P[k_i \text{ jobs arrived in slot } i]$$

But,

$$P[Y_i=x \ / \ k_i \text{ jobs arrived in slot } i] = \begin{cases} 1 & \text{iff } x=k_i n_{r-i+1} \\ 0 & \text{otherwise} \end{cases}$$

Thence,

$$p[Y_i = k_i \ n_{r-i+1}] = P[k_i \text{jobs arrived in slot } i] \qquad \text{(II.2)}$$

Now, since the job arrival process is Poisson with rate λ, and the slot width is unity (same as the task service time), we obtain

$$P[Y_i = k_i \ n_{r-i+1}] = \frac{\lambda^k}{k_i!} e^{-\lambda} \qquad \text{(II.3)}$$

Now let us define the Z-transform of Y_i by $Y_i(Z)$; that is,

$$Y_i(Z) \triangleq \sum_{j=0}^{\infty} P[Y_i=j] \, Z^j$$

From (II.3) we get:

$$Y_i(Z) = \sum_{k_i=0}^{\infty} P\,[Y_i = k_i n_{r-i+1}]\, Z^{k_i n_{r-i+1}} = \sum_{j=0}^{\infty} \frac{\lambda^j}{j!} e^{-\lambda} Z^{n_{r-i+1}}$$

$$= e^{-\lambda} \sum_{j=0}^{\infty} \frac{\left[\lambda Z^{n_{r-i+1}}\right]^j}{j!}$$

Thus, we have:

$$Y_i(Z) = e^{-\lambda\left[1 - Z^{n_{r-i+1}}\right]} \tag{II.4}$$

Let $Y(Z)$ be the Z-transform of Y. Since the job arrival process is Poisson, then the arrivals and the number of such arrivals in any slot i, i=1, ... ,r are independent random variables. It follows that

$$Y(Z) = \prod_{i=1}^{r} Y_i(Z)$$

Using (II.4), we obtain

$$Y(Z) = \prod_{i=1}^{r} e^{-\lambda\left[1 - Z^{n_{r-i+1}}\right]}$$

$$= \prod_{j=1}^{r} e^{-\lambda\left[1 - Z^{n_j}\right]}$$

Therefore, we get:

$$Y(Z) = e^{-\lambda r}\, e^{\lambda \sum_{i=1}^{r} Z^{n_i}} \tag{II.5}$$

Notice from (II.5) that $Y(1) = 1$ and $Y(0) = e^{-\lambda r} = P_0$, where $P_0 = P$ [*no job arrivals in the interval I*].

II.1 THE AVERAGE NUMBER OF OCCUPIED PROCESSORS

We shall derive the average number of occupied processors. We have:

$$\bar{Y} = \frac{d}{dZ} Y(Z)_{/Z=1}$$

Thence, by using equation (II.5), we get:

$$\bar{Y} = e^{-\lambda r}\left\{\frac{d}{dZ}\left[\lambda\sum_{i=1}^{r} Z^{n_i}\right] . e^{\lambda\sum_{i=1}^{r} Z^{n_i}}\right\}_{/Z=1}$$

$$= e^{-\lambda r}\left\{\lambda \sum_{i=1}^{r} n_i\, z^{n_i-1} . e^{\lambda\sum_{i=1}^{r} Z^{n_i}}\right\}_{/Z=1}$$

where $n_i \geq 1 \ \forall i = 1,...,r$.

$$= e^{-\lambda r}\left\{\left[\lambda \sum_{i=1}^{r} n_i\right] . e^{\lambda r}\right\}$$

$$= \lambda \sum_{i=1}^{r} n_i$$

Thus, we arrive at:

$$\bar{Y} = \lambda N \qquad \text{(II.6)}$$

II.2 THE VARIANCE OF THE NUMBER OF OCCUPIED PROCESSORS

We have:

$$\overline{Y^2} - \bar{Y}^2 = \frac{d^2}{dZ^2} Y(Z)_{/Z=1}$$

and

$$\sigma^2_Y = \overline{Y^2} - \bar{Y}^2$$

where σ^2_Y denotes the variance of the number of occupied processors. Now,

$$\frac{d^2}{dZ^2} Y(Z) = \frac{d}{dZ}\left[\frac{d}{dZ} Y(Z)\right]$$

$$= e^{-\lambda r}\left\{\lambda \sum_{i=1}^{r} n_i \left[n_i - 1\right] Z^{n_i-2}\right\} e^{\lambda\sum_{i=1}^{r} Z^{n_i}} + e^{-\lambda r}\left\{\lambda \sum_{i=1}^{r} n_i\, Z^{n_i-1}\right\}^2 . e^{\lambda\sum_{i=1}^{r} Z^{n_i}}$$

Therefore,

$$\frac{d^2}{dZ^2}Y(Z)_{/Z=1} = e^{-\lambda r}\left[\lambda \sum_{i=1}^{r} n_i \left[n_i - 1\right]\right] e^{\lambda r} + e^{-\lambda r}\left[\lambda \sum_{i=1}^{r} n_i\right]^2 e^{\lambda r}$$

$$= \lambda \sum_{i=1}^{r} n_i \left[n_i - 1\right] + \lambda^2 N^2 = \lambda \sum_{i=1}^{r} n^2{}_i - \lambda N + \lambda^2 N^2$$

Thence, we get:

$$\sigma^2{}_Y = \lambda \sum_{i=1}^{r} n^2{}_i \qquad \text{(II.7)}$$

Let us try to find an upper and a lower bound for $\sigma^2{}_Y$. By using the loose inequality

$$\sum_i x_i{}^2 \leq \left[\sum_i x_i\right]^2$$

we get from (II.7)

$$\sigma^2{}_Y \leq \lambda N^2 \qquad \text{(II.8)}$$

or, equivalently,

$$\sigma^2{}_Y \leq N\ \bar{Y}$$

Indeed*, this is a tight upper bound for $\sigma^2{}_Y$; case of a process graph with N tasks and just one level (r=1). The case of a process graph with N tasks and N levels (r=N) gives the lower bound , that is $siga_Y{}^2=\lambda N$. Thus, we have:

$$\lambda N \leq \sigma^2{}_Y \leq \lambda N^2 \textit{ or equivalently } \bar{Y} \leq \sigma^2{}_Y \leq N\ \bar{Y}$$

Moreover, let $n_{\max} = \max n_i \quad i=1,...,r$. Thus:

$$\sum_{i=1}^{r} n^2{}_i \leq \sum_{i=1}^{r} n_i n_{\max}$$

$$\leq n_{\max} \sum_{i=1}^{r} n_i$$

$$\leq N n_{\max}$$

Also,

$$N \leq N - n_{\max} + n_{\max}{}^2 \leq \sigma^2{}_Y$$

Therefore, we obtain the following:

$$\lambda N \le \lambda \left\{(N-n_{max})+n_{max}^2\right\} \le \sigma^2_Y \le \lambda N n_{max} \le \lambda N^2 \qquad \text{(II.8)}$$

In the above analysis, no restriction was assumed for the choice of the shape of the PG(N,r). Equations (II.6), (II.7) and (II.8) are valid for any shape of the process graph PG(N,r). The only restriction is that all jobs have the same process graph description $J(n_1, n_2, \ldots, n_r)$.

Let us take *the discrete well-shaped diamond* process graph denoted by PG(r) (as a function of r only).

CASE 1: r odd

Figure (II.2) below gives some examples of such a process graph:

Thus, we have:

$$n_i = \begin{cases} i & 1 \le i \le \dfrac{r+1}{2} \\ r-i+1 & \dfrac{r+1}{2} \le i \le r \end{cases}$$

*Formerly, to obtain an upper bound on σ^2_Y, we have to solve the following maximization problem:

$$\max \sum_{i=1}^{r} n^2_i$$

$$subject\ to \sum_{i=1}^{r} n_i = N \quad \text{and} \quad 1 \le r \le N$$

For a given r, the solution to this problem is simply $n_k = N-r+1$, $n_j = 1 \ \forall j = 1,\ldots,N$ and $j \ne k$. It is also obvious that r=1 gives the upper bound
Likewise, to obtain a lower bound on σ^2_Y , we have to solve the following minimization problem:

$$\min \sum_{i=1}^{r} n^2_i$$

$$subject\ to \sum_{i=1}^{r} n_i = N \quad \text{and} \quad 1 \le r \le N$$

For a given r, the solution to this problem is simply $n_i = \dfrac{N}{r}$ for all i=1,...,N . It is easy to see that r=N gives the lower bound.

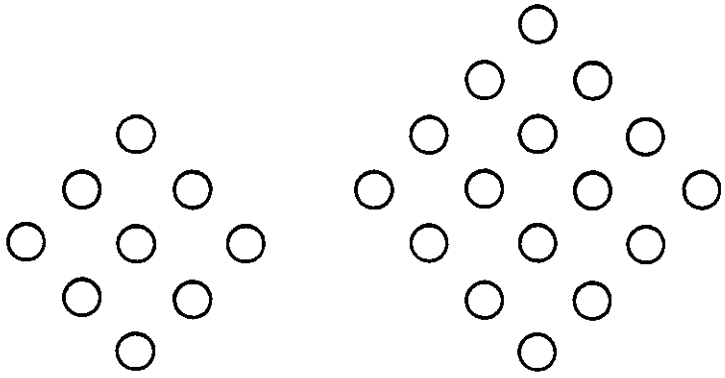


FIGURE II.2
Discrete Well-Shaped Process Graph with Odd Number of Levels

Thence:

$$N = \sum_{i=1}^{r} n_i$$

$$= \left[1+2+\ldots+\frac{r+1}{2}\right] + \left[\frac{r-1}{2}+\ldots+2+1\right]$$

$$= \frac{1}{2}\,\frac{r+1}{2}\,\frac{r+3}{2} + \frac{1}{2}\,\frac{r-1}{2}\,\frac{r+1}{2}$$

Thence:

$$N = \left[\frac{r+1}{2}\right]^2 \tag{II.9}$$

Substituting into (II.6), we get:

$$\bar{Y} = \frac{\lambda\left[\frac{r+1}{2}\right]^2}{4} \tag{II.10}$$

Using the known identity $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$, we get:

$$\sum_{i=1}^{r} n^2{}_i = 2 \sum_{i=1}^{\frac{r-1}{2}} i^2 + \left[\frac{r+1}{2}\right]^2$$

$$= \left[\frac{r+1}{2}\right]^2 \left[\frac{r}{6}+1\right] - \frac{r}{6}\left[\frac{r+1}{2}\right]$$

Now, by using (II.9), we obtain:

$$\sum_{i=1}^{r} n^2{}_i = N\frac{r+6}{6} - \frac{r}{6}\sqrt{N}$$

Finally, from (II.7) we get:

$$\sigma^2{}_Y = \lambda N\frac{r+6}{6} - \frac{\lambda r}{6}\sqrt{N} \qquad \text{(II.11)}$$

CASE 2: r even

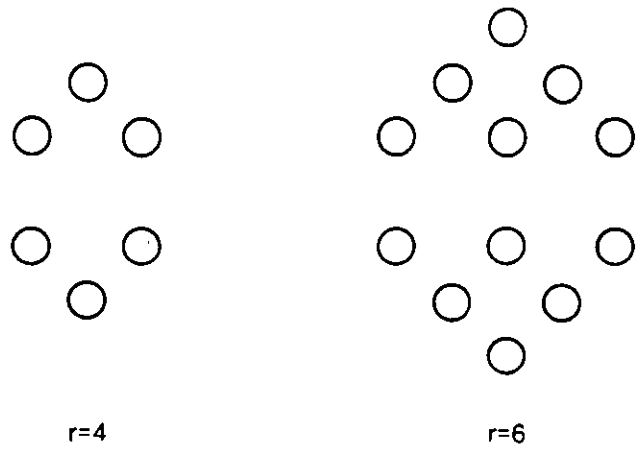


FIGURE II.3
Discrete Well-Shaped Process Graph with Even Number of Levels

Thence, we have:

$$n_i = \begin{cases} i & 1 \le i \le \frac{r}{2} \\ r-i-1 & \frac{r}{2} \le i \le r \end{cases}$$

Thus:

$$N = \sum_{i=1}^{r} n_i = \left[1+2+...+\frac{r}{2}\right] + \left[\frac{r}{2}+(\frac{r}{2}-1)+...+2+1\right]$$

$$= \frac{r(r+1)}{4}$$

Thus:

$$N = \frac{r(r+1)}{4} \tag{II.12}$$

and from (II.6) we obtain:

$$\bar{Y} = \frac{\lambda r(r+1)}{4} \tag{II.13}$$

and

$$\sum_{i=1}^{r} n^2{}_i = 2\sum_{i=1}^{\frac{r}{2}} i^2 = \frac{r+2}{3}\,\frac{r(r+1)}{4}$$

and, by using (II.12),

$$= \frac{r+2}{3}N$$

Finally, by using (II.7), we obtain

$$\sigma^2{}_Y = \lambda N\frac{r+2}{3} \tag{II.14}$$

or equivalently,

$$\sigma^2{}_Y = \bar{Y}\frac{r+2}{3} \tag{II.15}$$

Let us take *the uniform process graph of degree a* (see figure II.4). We have

$$N = ar \tag{II.16}$$

Therefore, from (II.6), we get

$$\bar{Y} = \lambda a r \qquad \text{(II.17)}$$

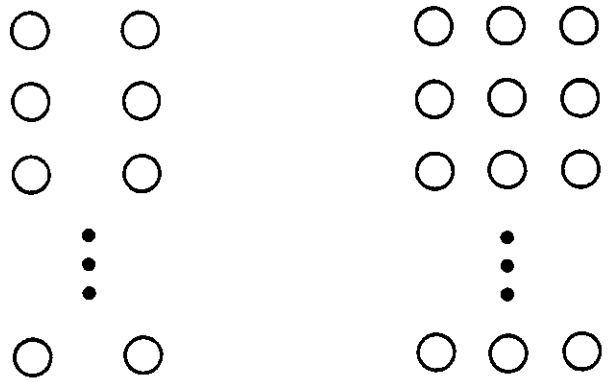


FIGURE II.4
Uniform Process Graph of Degree a

Now we have:

$$\sum_{i=1}^{r} n^2{}_i = \sum_{i=1}^{r} a^2 = a^2 r$$

and, by using (II.16) we get

$$\sum_{i=1}^{r} n^2{}_i = aN$$

Finally, by using (II.7), we obtain:

$$\sigma^2{}_Y = \lambda a N \qquad \text{(II.18)}$$

and, by using (II.6),

$$\sigma^2{}_Y = a\bar{Y}$$

II.3 FIXED PROCESS GRAPH WITH r=1

Since every job has a process graph with N tasks and 1 level, then all jobs arriving before time (t-1) should be finished before time t. Only those jobs arriving in the interval [t-1,t] will occupy any processors at time t. Each job arriving in the interval [t-1,t] will occupy N processors at time t. Thence, from equation (II.5), we get:

$$Y(Z) = e^{-\lambda[1-Z^N]} \qquad r=1 \tag{II.19}$$

The average number of occupied processors is then given by equation (II.6), and from equation (II.7) we get the variance of the number of occupied processors:

$$\sigma^2_Y = \lambda N^2 \qquad r=1 \tag{II.20}$$

II.4 CONCLUSION

In this chapter, we have investigated the case of a fixed process graph. We found closed expressions for the Z-transform, the average and the variance of the number of occupied processors in the system. Particular examples of fixed process graphs, namely, the discrete well-shaped process graph with both odd and even numbers of levels, the uniform process graph of degree a and the case of a process graph with just one level are considered.

CHAPTER 3
SEMI-RANDOM PROCESS GRAPH $r \geq 2$

In this chapter, we shall derive a closed form expression of the Z-transform of the distribution of the number of occupied processors. Closed form expressions for the average and variance of the number of occupied processors will also be derived. Throughout this chapter, we shall make the following assumptions about the multi-processor system under consideration:

a. the number of processors in the system, P, is infinite;

b. the task service time, X , is constant, say one unit service time per task;

c. the process graph is semi-random. Each job has a process graph with a fixed number of tasks, say N, and fixed number of levels, say r. Jobs do not necessarily have the same process graph description $J(n_1,n_2,\ldots,n_r)$, where n_i i=1,...,r denotes the number of tasks at level i in the process graph. Proposition (I.1) gives the probability of having n tasks at a given level k, given that we have N tasks and r levels in such a process graph n=1,...,N-r+1 ,and k=1,...,r .

Let us define the following quantities:

I = the interval of time (t-r,t)

Y = random variable counting the total number of occupied processors at time t

Y_k = random variable counting the total number of occupied processors at time t, given that k jobs arrived in the interval I

X_i = random variable counting the number of occupied processors at time t and by the ith job arriving in the interval I

It is easy to see from Figure (III.1) that all jobs arriving before time (t-r) finished before time t. Such jobs will not occupy any processor at time t. Any jobs that occupy processors at time t are jobs that arrived in the interval I. Therefore,

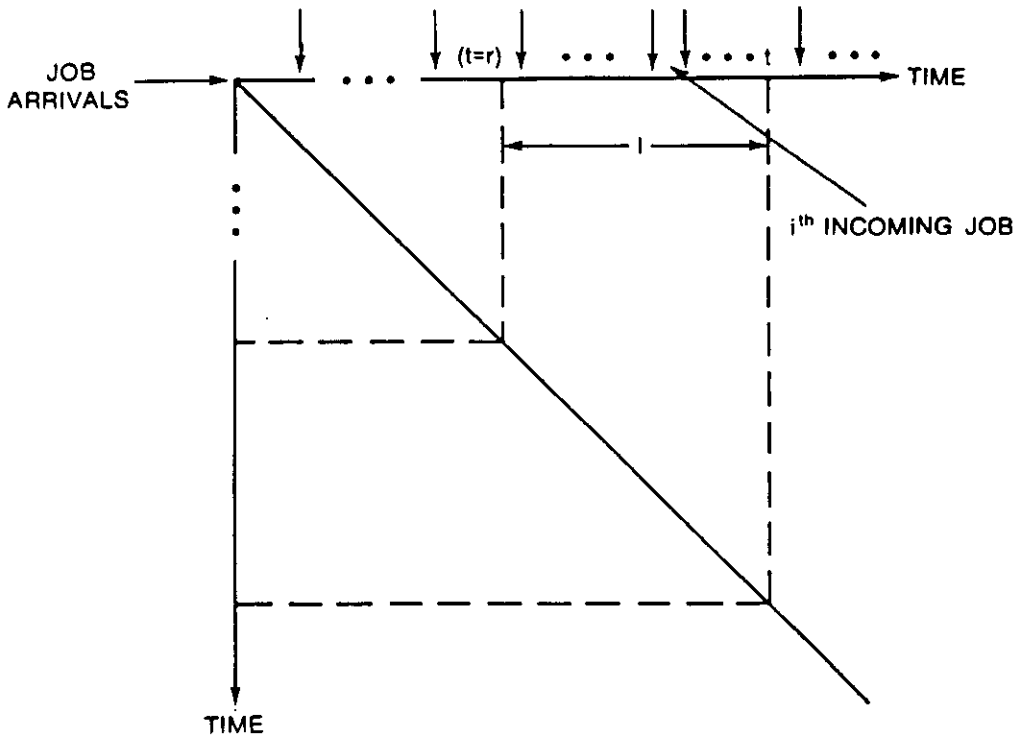


FIGURE III.1
Analysis of Semi-Random Process Graph

$$Y_k = \sum_{i=1}^{k} X_i \tag{III.1}$$

Also, we have

$$P[X_i=n_i] = \frac{\binom{N-n_i-1}{r-2}}{\binom{N-1}{r-1}} \quad \forall\, n_i=1,\ldots,N-r+1 \tag{III.2}$$

Let us define the following quantities:

$X_i(Z)$ = the Z-transform of X_i

$Y_k(Z)$ = the Z-transform of Y_k

$Y(Z)$ = the Z-transform of Y

Thence, we have:

$$X_i(Z) \triangleq \sum_{j=1}^{N-r+1} P[X_i=j]\, Z^j$$

$$= \sum_{j=1}^{N-r+1} \frac{\binom{N-j+1}{r-2}}{\binom{N-1}{r-1}} Z^j$$

That is,

$$X_i(Z) = \frac{1}{\binom{N-1}{r-1}} \sum_{j=0}^{N-r} \binom{N-j-2}{r-2} Z^{j+1} \tag{III.3}$$

Since the random variables X_is, i=1,...,k are independent and identically distributed (i.i.d.), we get

$$Y_k(Z) = \left[X(Z) \right]^k \tag{III.4}$$

where X(Z) is given by equation (III.3), and since the job arrival process is Poisson with parameter λ, we get

$$Y(Z) = \sum_{k=0}^{\infty} \left[X(Z) \right]^k \frac{(\lambda r)^k}{k!} e^{-\lambda r}$$

Thus,

$$Y(Z) = e^{-\lambda r \left[1 - X(Z)\right]} \tag{III.5}$$

Finally, by plugging equation (III.3) into equation (III.5), we get:

$$Y(Z) = e^{-\lambda r \left[1 - \frac{Z \sum_{j=0}^{N-r} \binom{N-j-2}{r-2} Z^j}{\binom{N-1}{r-1}}\right]} \tag{III.6}$$

III.1 THE AVERAGE NUMBER OF OCCUPIED PROCESSORS

Let us define the following quantities:

$\overline{Y}$ = *average number of occupied processors*

$$b(Z) = \sum_{i=0}^{M-R} \binom{M-i}{R} Z^i \tag{III.7}$$

$$a = \frac{\lambda r}{\binom{M+1}{R+1}} \tag{III.8}$$

where:

$$M = N-1 \qquad \text{and} \qquad R = r-1$$

Therefore, we can write (III.6) as follows:

$$Y(Z) = e^{-\lambda r}\, e^{aZb(Z)} \tag{III.9}$$

Since $\bar{Y} = \left.\frac{dY(Z)}{dZ}\right|_{Z=1}$, then, by using equation (III.9), we obtain:

$$\bar{Y} = ab(1) + a\left.\frac{db(Z)}{dZ}\right|_{z=1} \tag{III.10}$$

Let us then find $b(1)$ and $b'(1)$. Using equation (III.7), we get:

$$b'(Z) = \left.\frac{db(Z)}{dZ}\right|_{z=1} = \left.\sum_{i=1}^{M-R} i \begin{bmatrix} M-i \\ R \end{bmatrix} Z^{i-1}\right|_{Z=1} \quad thence,$$

$$b'(1) = \sum_{i=0}^{M-R} i \begin{bmatrix} M-i \\ R \end{bmatrix} \tag{III.11}$$

Now let us compute b(1). From equation (III.7) we get:

$$b(1) = \sum_{i=0}^{M-R} \begin{bmatrix} M-i \\ R \end{bmatrix}$$

$$= \begin{bmatrix} M \\ R \end{bmatrix} + \begin{bmatrix} M-1 \\ R \end{bmatrix} + \begin{bmatrix} M-2 \\ R \end{bmatrix} + \cdots + \begin{bmatrix} R \\ R \end{bmatrix}$$

Therefore, we can write b(1) as

$$b(1) = \sum_{i=0}^{M-R} \begin{bmatrix} R+i \\ R \end{bmatrix}$$

Let us consider the following function:

$$\beta(x) = (1+x)^R + (1+x)^{R+1} + \cdots + (1+x)^M$$

The coefficient of x^R in $\beta(x)$ is exactly b(1) since the coefficient of x^R in $(1+x)^{R+i}$ *is* $\begin{bmatrix} R+i \\ R \end{bmatrix}$.

Now let us rewrite the function $\beta(x)$ as:

$$\beta(x) = \frac{(1+x)^{M+1} - (1+x)^R}{x}$$

Thus, the coefficient of x^R in $\beta(x)$ is $\begin{bmatrix} M+1 \\ R+1 \end{bmatrix}$ and, therefore, we obtain

$$b(1) = \sum_{i=0}^{M-R} \binom{M-i}{R} = \binom{M+1}{R+1} \tag{III.12}$$

Now, by using equations (III.12), (III.8), (III.10) and (III.11), we get

$$\frac{dY(Z)}{dZ}_{/Z=1} = \lambda r \left[1+\frac{b'(1)}{\binom{M+1}{R+1}}\right] \tag{III.13}$$

Let us compute b'(1):

$$b'(1) = \sum_{i=0}^{M-R} i \binom{M-i}{R}$$

$$= 0 \binom{M}{R} + 1 \binom{M-1}{R} + \cdots + i \binom{M-i}{R} + \cdots + (M-R) \binom{R}{R}$$

$$= \sum_{i=0}^{M-R} (M-R-i) \binom{R+i}{R}$$

$$= (M-R) \sum_{i=0}^{M-R} \binom{R+i}{R} - \sum_{i=0}^{M-R} i \binom{R+i}{R}$$

and, using (III.7), we get

$$= (M-R)b(1) - \sum_{i=0}^{M-R} i \binom{R+i}{R}$$

Then, using (III.12), we get

$$b'(1) = (M-R) \binom{M+1}{R+1} - \sum_{i=0}^{M-R} i \binom{R+i}{R} \tag{III.14}$$

Let us define the following:

$$a_n = \sum_{i=0}^{n} i \binom{R+i}{R} \qquad n \geq 0 \tag{III.15}$$

Thence, we can define the following recurrence relation:

$$a_n = a_{n-1} + n \cdot \binom{R+n}{R} \qquad n \geq 1 \tag{III.16}$$

with the boundary condition

$$a_0 = 0 \tag{III.17}$$

Define the generating function of a_n by A(x); that is,

$$A(x) \triangleq \sum_{n=0}^{\infty} a_n x^n$$

Thus, equation (III.16) gives

$$\sum_{n=1}^{\infty} a_n x^n = \sum_{n=1}^{\infty} a_{n-1} x^n + \sum_{n=1}^{\infty} n \begin{bmatrix} R+n \\ R \end{bmatrix} x^n$$

That is,

$$A(x) - a_0 = xA(x) + x \sum_{n=0}^{\infty} n \begin{bmatrix} R+n \\ R \end{bmatrix} x^{n-1}$$

$$= xA(x) + x \frac{d}{dx} \left\{ \sum_{n=1}^{\infty} \begin{bmatrix} R+n \\ R \end{bmatrix} x^n \right\}$$

and, by using (A.3) of Appendix A, we get

$$= xA(x) + x \frac{d}{dx} \left\{ \frac{1}{(1-x)^{R+1}} - 1 \right\}$$

$$= xA(x) + x + \frac{R+1}{(1-x)^{R+2}}$$

Therefore, we obtain

$$A(x) = \frac{(R+1)x}{(1-x)^{R+3}}$$

We can rewrite A(x) as

$$A(x) = \frac{R+1}{(1-x)^{R+3}} - \frac{R+1}{(1-x)^{R+2}} \tag{III.18}$$

Using standard techniques Kleinrock Queueing Theory equation (III.18) can be inverted to give

$$a_n = (R+1) \begin{bmatrix} n+R+2 \\ R+2 \end{bmatrix} - (R+1) \begin{bmatrix} n+R+1 \\ R+1 \end{bmatrix}$$

and, by using the well known formula $\begin{bmatrix} n \\ k \end{bmatrix} = \begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$, we get

$$\binom{n+R+2}{R+2} = \binom{n+R+1}{R+2} + \binom{n+R+1}{R+1}$$

Thus, we obtain

$$a_n = (R+1)\binom{n+R+1}{R+2} \tag{III.19}$$

Now, using (III.14) and (III.19), we get:

$$\frac{db(Z)}{dZ}_{/Z=1} = (M-R)\binom{M+1}{R+1} - (R+1)\binom{M+1}{R+2} \tag{III.20}$$

and using equations (III.13) and (III.20), we obtain:

$$\frac{dY(Z)}{dZ}_{/Z=1} = \lambda r \frac{M+2}{R+2}$$

Finally, since M=N-2 and R=r-2, we obtain

$$\bar{Y} = \lambda N \tag{III.21}$$

III.2 THE VARIANCE OF THE NUMBER OF OCCUPIED PROCESSORS

Let σ^2_Y denote the variance of the random variable Y. Thus, we have

$$\sigma^2_Y = \frac{d^2Y(Z)}{dZ^2}_{/Z=1} + \frac{dY(Z)}{dZ}_{/Z=1} - \left[\frac{dY(Z)}{dZ}\right]^2$$

From equation (III.9), we get

$$\frac{d^2Y(Z)}{dZ^2}_{/Z=1} = 2ab'(1) + ab''(1) + \left[ab(1) + ab'(1)\right]^2 \tag{III.22}$$

Let us then compute $b''(1)$. From equation (III.7) we obtain

$$b''(Z) = \sum_{i=0}^{M-R} i(i-1)\binom{M-i}{R} Z^{i-2} \text{Thence},$$

$$b''(1) = \sum_{i=0}^{M-R} i(i-1)\binom{M-i}{R}$$

$$= 0\binom{M}{R} + 0\binom{M-1}{R} + 2\binom{M-2}{R} + 6\binom{M-3}{R} + \cdots + (M-R)(M-R-1)\binom{R}{R}$$

$$= \sum_{i=0}^{M-R} (M-R-i)(M-R-i-1) \begin{bmatrix} R+i \\ R \end{bmatrix}$$

which gives:

$$\frac{d^2 b(Z)}{dZ^2}_{/Z=1} = (M-R)(M-R-1) \sum_{i=0}^{M-R} \begin{bmatrix} R+i \\ R \end{bmatrix} - 2(M-R) \sum_{i=0}^{M-R} i \begin{bmatrix} R+i \\ R \end{bmatrix} + \sum_{i=0}^{M-R} i(i+1) \begin{bmatrix} R+i \\ R \end{bmatrix} \qquad \text{(III.23)}$$

We then need to calculate $\sum_{i=0}^{M-R} i(i+1) \begin{bmatrix} R+i \\ R \end{bmatrix}$. Let:

$$a_n = \sum_{i=0}^{n} i(i+1) \begin{bmatrix} R+i \\ R \end{bmatrix} \quad n \geq 0 \qquad \text{(III.24)}$$

Thence, we can obtain the following recurrence relationship:

$$a_n = a_{n-1} + n(n+1) \begin{bmatrix} R+n \\ n \end{bmatrix} \quad n \geq 1 \qquad \text{(III.25)}$$

with the boundary condition

$$a_0 = 0 \qquad \text{(III.26)}$$

Let us define the generating function of a_n by A(x); that is,

$$A(x) = \sum_{n=0}^{\infty} a_n x^n$$

Therefore, by using (III.25), we obtain

$$\sum_{n=1}^{\infty} a_n x^n = \sum_{n=1}^{\infty} a_{n-1} x^n + \sum_{n=1}^{\infty} n(n+1) \begin{bmatrix} R+n \\ R \end{bmatrix} x^n$$

which gives

$$A(x) = \frac{x}{1-x} \sum_{n=1}^{\infty} n(n+1) \begin{bmatrix} R+n \\ R \end{bmatrix} x^{n-1} \qquad \text{(III.27)}$$

We have

$$\sum_{n=1}^{\infty} n(n+1) \begin{bmatrix} R+n \\ R \end{bmatrix} x^{n-1} = \frac{d^2}{dx^2} \left\{ \sum_{n=1}^{\infty} \begin{bmatrix} R+n \\ R \end{bmatrix} x^{n+1} \right\}$$

$$= \frac{d^2}{dx^2} \left\{ x \sum_{n=1}^{\infty} \begin{bmatrix} R+n \\ R \end{bmatrix} x^n \right\}$$

and, by using (A.3) of Appendix A, we get

$$= \frac{d^2}{dx^2}\left\{ x\left[\frac{1}{(1-x)^{R+1}} - 1\right]\right\}$$

$$= \frac{R(1+R)x + 2(1+R)}{(1-x)^{R+3}}$$

Equation (III.23) thus becomes

$$A(x) = \frac{R(1+R)x^2 + 2(1+R)x}{(1-x)^{R+4}}$$

The above equation can be rewritten as:

$$A(x) = \frac{R(1+R)}{(1-x)^{R+2}} - \frac{2(1+R)^2}{(1-x)^{R+3}} + \frac{(1+R)(2+R)}{(1-x)^{R+4}} \qquad \text{(III.28)}$$

Equation (III.28) can be easily inverted Kleinrock Theory Queueing to obtain:

$$a_n = 2(1+R)\begin{bmatrix} R+n+1 \\ R+2 \end{bmatrix} + (1+R)(2+R)\begin{bmatrix} R+n+1 \\ R+3 \end{bmatrix} \quad n \geq 1 \qquad \text{(III.29)}$$

Let us now return to equation (III.23). By using equations (III.12), (III.11), (III.20) and (III.29), we get

$$b''(1) = (M-R)(M-R-1)\begin{bmatrix} M+1 \\ R+1 \end{bmatrix} - 2(M-R)\left\{(M-R)\begin{bmatrix} M+1 \\ R+1 \end{bmatrix} - (R+1)\begin{bmatrix} M+1 \\ R+2 \end{bmatrix}\right\}$$

$$+ 2(1+R)\begin{bmatrix} M+1 \\ R+2 \end{bmatrix} + (1+R)(2+R)\begin{bmatrix} M+1 \\ R+3 \end{bmatrix}$$

After some simplification we get

$$b''(1) = (M-R)(R-M-1)\begin{bmatrix} M+1 \\ R+1 \end{bmatrix} + 2(1+R)(1+M-R)\begin{bmatrix} M+1 \\ R+2 \end{bmatrix} + (1+R)(2+R)\begin{bmatrix} M+1 \\ R+3 \end{bmatrix} \qquad \text{(III.30)}$$

Now, let us return to the calculation of σ^2_Y. From equation (III.22) we have

$$\sigma^2_Y = ab(1) + 3ab'(1) + ab''(1) \qquad \text{(III.31)}$$

where a is given by equation (III.8), b(1) is given by equation (III.12), $b'(1)$ is given by equation (III.20) and $b''(1)$ is given by equation (III.30). Using the following,

$$\frac{\binom{M+1}{R+2}}{\binom{M+1}{R+1}} = \frac{M-R}{2+R} \quad \textit{and} \quad \frac{\binom{M+1}{R+3}}{\binom{M+1}{R+1}} = \frac{(M-R)(M-R-1)}{(2+R)(3+R)}$$

and after some simplifications, we obtain

$$\sigma^2_Y = \lambda r + \lambda r (M-R) \left\{ M-R+2 + \frac{1+R}{2+R}(2M-2R-1) + \frac{1+R}{3+R}(M-R-1) \right\} \qquad \text{(III.32)}$$

Now, replacing M and R by their respective values, we arrive at

$$\sigma^2_Y = lamdar + \lambda r (N-r) \left\{ N-r+2 + \frac{r}{r+1}(2N-2r-1) + \frac{r}{r+2}(N-r-1) \right\} \qquad \text{(III.32)}$$

III.3 CONCLUSION

In this chapter, we have investigated the case of a semi-random process graph. We found closed expressions for the Z-transform, the average and the variance of the number of occupied processors in the system.

CHAPTER 4
RANDOM PROCESS GRAPH

In this chapter, we shall derive a closed form expression of the Z-transform of the distribution of the number of occupied processors. Closed expressions for the average of the number of occupied processsors will also be derived. Throughout this chapter, we shall make the following assumptions:

a. the number of processors in the system, P, is infinite;

b. the task service time, X, is constant, say one unit of service time;

c. the process graph is random. Each job has a random process graph with a fixed number of tasks, say N, and a random number of levels r, $1 \le r \le N$. From proposition (I.2), we know that the probability of an incoming job having r levels is a Binomial given by

$$P[r=r] = \frac{\binom{N-1}{r-1}}{2^{N-1}} = b(r-1, N-1, \frac{1}{2})$$

Moreover, two jobs having the same number of levels, say r, do not necessarily have the same process graph description $J(n_1, \ldots, n_r)$, where n_i is the number of tasks at level i, $1 \le i \le r$. Proposition (I.1) gives the probability of having n tasks at a given level k, given that the job has r levels and N tasks, where n=1,...,N-r+1, k=1,...,r and r=1,...,N.

From Figure (IV.1), we see that any job arriving before time (t-N) will not participate (occupy any processor) at time t. A job arriving in the interval I=(t-N,t) will participate if, and only if, it has enough levels. Let us divide the interval I into N equal slots of duration one unit of time, each equal to the processing time of one task. We number such slots by 1,2,...,N (see Figure (IV.1)). It follows that a job arriving in slot number i will occupy some processors at time t if, and only if, it has at least i levels, where $1 \le i \le N$

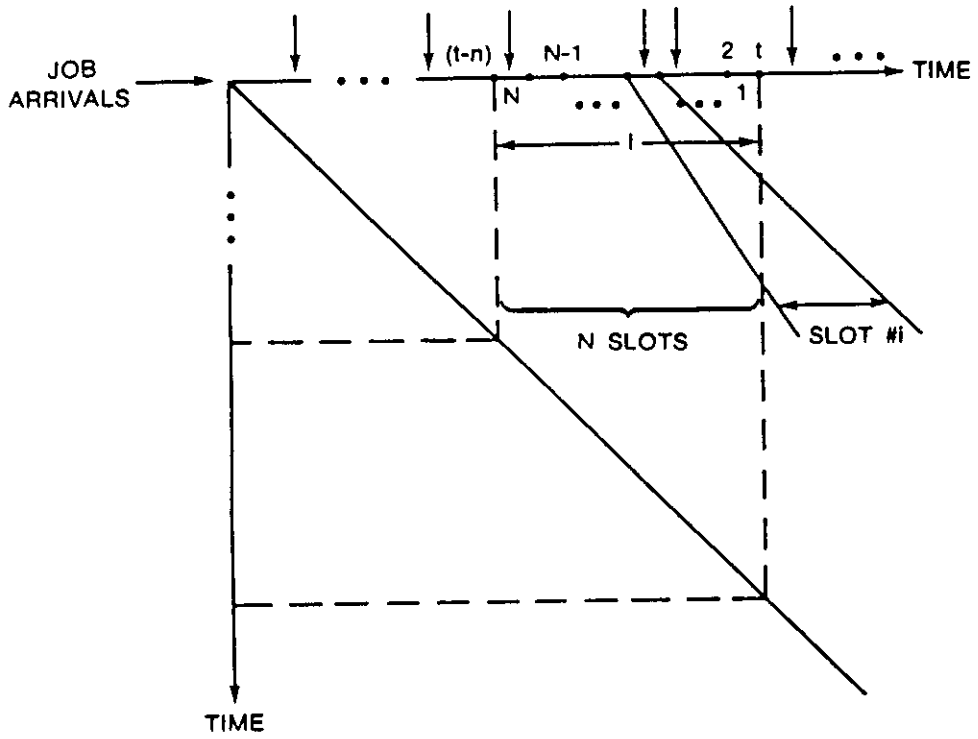


FIGURE IV.1
Analysis of Random Process Graph

PROPOSITION IV.1

The probability of a job arriving in slot i, $1 \le i \le N$, occupies a processor at time t is given by

$$P[\textit{ job arriving in slot i occupies some processors at time t }] = \frac{\sum_{j=0}^{N-i} \binom{N-1}{j}}{2^{N-1}}$$

PROOF

A job arriving in slot i, $1 \le i \le N$ occupies some processors at time t if, and only if, it has at least i levels. Thence, from proposition (I.2) we get

$$P[\textit{ job arriving in slot i occupies some processors at time t }] = \sum_{r=i}^{N} \frac{\binom{N-1}{r-1}}{2^{N-1}}$$

$$= \frac{1}{2^{N-1}} \sum_{j=1}^{N-i+1} \binom{N-1}{N-j}$$

and, since $\binom{N-1}{N-j} = \binom{N-1}{j-1}$, we get

$$\sum_{j=1}^{N-i+1} \binom{N-1}{N-j} = \sum_{j=1}^{N-i+1} \binom{N-1}{j-1} = \sum_{j=0}^{N-i} \binom{N-1}{j}$$

IIII

Let us now define the following quantities:

Y = random variable counting the total number of occupied processors at time t

$Y_{/k}$ = random variable counting the total number of occupied processors at time t if k jobs arrived in I

$Y_{/(k_1,\ldots,k_N)}$ = random variable counting the total number of occupied processors at time t if k_i jobs arrived in slot i, i=1,...,N

X_{ik_i} = random variable counting the number of occupied processors at time t due to jobs that arrived in slot i and given that k_i jobs arrived in slot i, i=1,...,N

X_{ij} = random variable counting the number of occupied processors by the jth job that arrived in slot i, i=1,..,N

It follows that

$$Y = \sum_{k=0}^{\infty} Y_{/k} \frac{(\lambda N)^k}{k!} e^{-\lambda N} \tag{IV.1}$$

$$Y_{/k} = \sum_{s.t.\ \sum_{i=1}^{N} k_i = k} Y_{/(k_1,\ldots,k_N)} \frac{k!}{k_1!\ldots k_N!} \frac{1}{N}^k \tag{IV.2}$$

$$Y_{/(k_1,\ldots,k_N)} = \sum_{i=1}^{N} X_{ik_i} \tag{IV.3}$$

$$X_{ik_i} = \sum_{j=1}^{k_i} X_{ij} \tag{IV.4}$$

Also, let us define the following Z-transforms:

$Y(Z) \triangleq \sum_{y=0}^{\infty} P[Y=y]Z^y$ the Z-transform of the random variable Y

$Y_{/k}(Z) \triangleq \sum_{y} P[Y_{/k}=y]Z^y$ the Z-transform of the random variable $Y_{/k}$

$Y_{/(k_1,\ldots,k_N)}(Z) \triangleq \sum_{y} P[Y_{/(k_1,\ldots,k_N)}=y]Z^y$ the Z-transform of the random variable $Y_{/(k_1,\ldots,k_N)}$

$X_{ik_i}(Z) \triangleq \sum_{x} P[X_{ik_i}=x]Z^x$ the Z-transform of the random variable X_{ik_i}

$X_{ij}(Z) \triangleq \sum_{x} P[X_{ij}=x]Z^x$ the Z-transform of the random variable X_{ij}

Since the X_{ij}'s are independent and identically distributed, $\forall$ j=1,..., k_i then it follows that, by using equation (IV.4), we get

$$X_{ik_i} = \prod_{j=1}^{k_i} X_{ij}(Z)$$

Let us denote by $X^i(Z) \triangleq X_{ij}(Z)$ since the X_{ij} 's are i.i.d. Thus, we get

$$X_{i_k}(Z) = \left[X^i(Z)\right]^k$$

and, from equation (IV.3), we get

$$Y_{/(k_1,\ldots,k_N}(Z) = \prod_{i=1}^{N} \left[X^i(Z)\right]^{k_i}$$

and, from equation (IV.2), we get

$$Y_{/k}(Z) = \sum_{s.t.\ \sum_{i=1}^{N} k_i = k} \frac{\left[X^1(Z)\right]^{k_1}}{k_1!} \cdots \frac{\left[X^N(Z)\right]^{k_N}}{k_N!}\, k! \left[\frac{1}{N}\right]^k$$

$$= \left[\frac{\sum_{i=1}^{N} X^i(Z)}{N}\right]^k$$

Finally, using equation (IV.1), we obtain

$$Y(Z) = \sum_{k=0}^{\infty} \left[\frac{\sum_{i=0}^{N} X^i(Z)}{N}\right]^k \frac{\lambda N^k}{k!} e^{-\lambda N}$$

which gives

$$Y(Z) = e^{-\lambda N} \cdot e^{\lambda \sum_{i=1}^{N} X^i(Z)} \qquad \text{(IV.5)}$$

IV.1 THE AVERAGE OF THE NUMBER OF OCCUPIED PROCESSORS

Let us denote by $\bar{Y}$ the average number of occupied processors. Thus, we have

$$\bar{Y} = \frac{dY(Z)}{dZ}_{/Z=1}$$

and we have

$$\frac{dY(Z)}{dZ} = e^{-\lambda N} \cdot e^{\lambda \sum_{i=1}^{N} X^i(Z)} \frac{d}{dZ}\left\{\lambda \sum_{i=1}^{N} X^i(Z)\right\}$$

Thus,

$$\bar{Y} = \lambda \sum_{i=1}^{N} \frac{dX_i(Z)}{dZ}_{/Z=1} \qquad \text{(IV.6)}$$

Concentrating on slot i, we have

$$P[X_{i,j}=x]=\sum_{r=1}^{N}P[X_{ij}=x/r=r]P[r=r] \quad \textit{where:}$$

For $i\geq 2$ we have

$$P[X_{i,j}=x/r=r]=\begin{cases}0 & \text{if } x\neq 0 \quad 1\leq r\leq i-1\\ 1 & \text{if } x=0 \quad 1\leq r\leq i-1\\ \dfrac{\binom{N-x-1}{r-2}}{\binom{N-1}{r-1}} & \text{if } x\geq 1 \quad i\leq r\leq N\end{cases} \tag{IV.7}$$

and for i=1 we have

$$P[X_{1j}=x/r=r]=\begin{cases}0 & \text{if } x\neq N \quad r=1\\ 1 & \text{if } x=N \quad r=1\\ N-x-1 & \\ r-2 & x\geq 1 \quad r\geq 2\\ 0 & \textit{otherwise}\end{cases} \tag{IV.8}$$

and we know that $P[r=r]=\dfrac{\binom{N-1}{r-1}}{2^{N-1}}$. Thence,

1. *for* $i\geq 2$

$$P[X_{i,j}=x]=\begin{cases}\dfrac{\sum_{r=1}^{i-1}\binom{N-1}{r-1}}{2^{N-1}} & x=0\\ \dfrac{\sum_{r=i}^{N}\binom{N-x-1}{r-2}}{2^{N-1}} & x\neq 0\end{cases} \tag{IV.9}$$

Therefore, we get

$$X_{i,j}(Z)=\frac{1}{2^{N-1}}\left\{\sum_{r=1}^{i-1}\binom{N-1}{r-1}+\sum_{x\geq 0}\left\{\sum_{r=i}^{N}\binom{N-x-1}{r-2}\right\}Z^{x}\right\} \qquad i\geq 2 \tag{IV.10}$$

Since we have $X_{ij}(Z)/_{Z=1}=1$, from equation (IV.10) we get

$$\sum_{x\geq 1}\sum_{r=i}^{N}\binom{N-x-1}{r-2}=2^{N-1}-\sum_{r=1}^{i-1}\binom{N-1}{r-1}\qquad i\geq 2 \tag{IV.11}$$

2. *for i=1*

From equation (IV.8) we obtain

$$P[X_{1j}=x]=\begin{cases}0 & x=0\\ \dfrac{1}{2^{N-1}} & x=N\\ \displaystyle\sum_{r=2}^{N}\frac{\binom{N-x-1}{r-2}}{2^{N-1}} & x\geq 1\end{cases} \tag{IV.12}$$

Therefore, we obtain:

$$X_{1j}(Z)=\frac{1}{2^{N-1}}\left\{Z^N+\sum_{x\geq 1}\left\{\sum_{r=2}^{N}\binom{N-x-1}{r-2}\right\}Z^x\right\}\qquad i=1 \tag{IV.13}$$

Using equations (IV.10) and (IV.13), we get

$$\sum_{i=1}^{N}\left\{\frac{dX_i(Z)}{dZ}_{/Z=1}\right\}=\sum_{i=2}^{N}\frac{1}{2^{N-1}}\sum_{x\geq 1}x\left\{\sum_{r=i}^{N}\binom{N-x-1}{r-2}\right\}+\frac{1}{2^{N-1}}\left\{N+\sum_{x\geq 1}x\left\{\sum_{r=2}^{N}\binom{N-x-1}{r-2}\right\}\right\}$$

Let us define the following quantities:

$$A=\sum_{i=2}^{N}\frac{1}{2^{N-1}}\sum_{x\geq 1}x\left\{\sum_{r=i}^{N}\binom{N-x-1}{r-2}\right\}$$

$$B=\sum_{x\geq 1}x\left\{\sum_{r=2}^{N}\binom{N-x-1}{r-2}\right\}$$

Thus, equation (IV.6) becomes

$$\bar{Y}=\lambda\left\{A+\frac{1}{2^{N-1}}\left[N+B\right]\right\} \tag{IV.14}$$

Now, let us compute A.

$$i=N \Rightarrow \begin{matrix} x=1 \\ r=N \end{matrix} \qquad \binom{N-2}{N-2}$$

$$i=N-1 \Rightarrow \begin{matrix} x=1,2 \\ r=N-1,N \end{matrix} \qquad \binom{N-2}{N-3}+\binom{N-2}{N-2}+2\binom{N-3}{N-3}$$

$$i=N-2 \Rightarrow \begin{matrix} x=1,2,3 \\ r=N-2,N-1,N \end{matrix} \qquad \binom{N-2}{N-4}+\binom{N-2}{N-3}+\binom{N-2}{N-2}+2\binom{N-3}{N-4}+2\binom{N-3}{N-3}+3\binom{N-4}{N-4}$$

$$\vdots$$

$$i=j \Rightarrow \begin{matrix} x=1,\ldots,N-j+1 \\ r=1,\ldots,N \end{matrix} \qquad \binom{N-2}{j-2}+\cdots+\binom{N-2}{N-2}+2\binom{N-3}{j-2}+\cdots+\binom{N-3}{N-3}+\ldots+\ldots+(N-j+1)\binom{j-2}{j-2}$$

$$\vdots$$

$$i=2 \Rightarrow \begin{matrix} x=1,\ldots,N-1 \\ r=2,\ldots,N \end{matrix} \qquad \binom{N-2}{0}+\ldots+\binom{N-2}{N-2}+2\binom{N-3}{0}+\ldots+2\binom{N-2}{N-2}+\ldots+\ldots+(N-1)\binom{0}{0}$$

Therefore, for any x, x=1,...,N-1, the participation is

$$x\left\{\binom{N-x-1}{0}+2\binom{N-x-1}{1}+3\binom{N-x-1}{2}+\cdots+(i+1)\binom{N-x-1}{i}+\cdots+(N-x)\binom{N-x-1}{N-x-1}\right\}$$

which gives

$$\textit{participation of the value } x,\ x=1,\ldots,N-1 = x\sum_{i=0}^{N-x-1}(i+1)\binom{N-x-1}{i} \qquad \text{(IV.15)}$$

which finally gives us

$$A=\frac{1}{2^{N-1}}\sum_{x=1}^{N-1}x\sum_{i=0}^{N-x-1}(i+1)\binom{N-x-1}{i} \qquad \text{(IV.16)}$$

On the other hand, we have

$$\sum_{i=0}^{N-x-1}(i+1)\binom{N-x-1}{i}+\sum_{i=0}^{N-x-1}\binom{N-x-1}{i}+\sum_{i=0}^{N-x-1}i\binom{N-x-1}{i}$$

but, by using the binomial theorem, Liu Introduction Combinatorial we have

$$\sum_{i=0}^{N-x-1} \binom{N-x-1}{i} = 2^{N-x-1} \qquad \textit{and}$$

$$\sum_{i=0}^{N-x-1} i \binom{N-x-1}{i} = (N-x-1)\, 2^{N-x-2}$$

Therefore, A becomes

$$A = \frac{1}{2^{N-1}} \sum_{x=1}^{N-1} x\, (N-x-1)\, 2^{N-x-1} \tag{IV.17}$$

Let us define the following quantities:

$$A_1 = \sum_{x=1}^{N-1} x\, (\frac{1}{2})^{x-1}$$

and

$$A_2 = \sum_{x=1}^{N-1} x\, (x-1)\, (\frac{1}{2})^{x-2}$$

Therefore, A becomes

$$A = \frac{1}{8} \left\{ 2NA_1 - A_2 \right\} \tag{IV.18}$$

Computation of A_1*:*

$$A_1 = \sum_{i=1}^{N} i \left(\frac{1}{2}\right)^{i-1} = \frac{d}{dx} \sum_{i=1}^{N-1} x^i \quad \textit{where } x = \frac{1}{2}$$

$$= \frac{d}{dx} \left[x\, \frac{1-x^{N-1}}{1-x} \right]$$

$$= \frac{\left[1-Nx^{N-1}\right](x-1) + x - x^N}{(1-x)^2}$$

and, replacing x by $\frac{1}{2}$, we get

$$A_1 = 4 - \left[\frac{1}{2}\right]^{N-2} (N+1) \tag{IV.19}$$

Computation of A_2*:*

$$A_2 = \sum_{i=1}^{N-1} i\, (i-1)\, x^{i-2}$$

$$= \frac{d^2}{dx^2} \sum_{i=1}^{N-1} x^i \quad where\ x=\frac{1}{2}$$

$$= \frac{N\,(N-1)\,x^{N-2}}{x-1} + \frac{2}{1-x} A_1(x)$$

and, replacing x by $\frac{1}{2}$, we get

$$A_2 = 16 - \left[N^2+N+2\right]\left[\frac{1}{2}\right]^{N-3} \qquad \text{(IV.20)}$$

Finally, by using equations (IV.19), (IV.20) and (IV.18), we obtain

$$A = N - 2 + \left[\frac{1}{2}\right]^{N-1} \qquad \text{(IV.21)}$$

Now, let us compute B. We have

$$B = \sum_{x\geq 1} x \left\{ \sum_{r=2}^{N} \binom{N-x-1}{r-2} \right\}$$

$x=N-1 \Rightarrow r=2 \qquad (N-1)\binom{0}{0} \qquad \Rightarrow (N-1)2^0$

$x=N-2 \Rightarrow r=2,3 \qquad (N-2)\binom{1}{0}+(N-2)\binom{1}{1} \qquad \Rightarrow (N-2)2^1$

$\vdots$

$x=i \Rightarrow r=2,\ldots,N-i+1 \qquad i\binom{N-i+1}{0}+i\binom{N-i+1}{1}+\ldots+i\binom{N-i+1}{N-i+1} \qquad \Rightarrow i\,2^{N-i+1}$

$\vdots$

$x=1 \Rightarrow r=2,\ldots,N \qquad \binom{N-2}{0}+\ldots+\binom{N-2}{i}+\ldots+\binom{N-2}{N-2} \qquad \Rightarrow 2^{N-2}$

Therefore, we obtain:

$$B = \sum_{i=1}^{N-1} i\,2^{N-i-1} = 2^{N-2} \sum_{i=1}^{N-1} i \left(\frac{1}{2}\right)^{i-1}$$

and, by using equation (IV.19), we get

$$B = 2^N - (N+1) \tag{IV.22}$$

Then, using equations (IV.21) and (IV.22), equation (IV.14) becomes

$$\sum_{i=1}^{N} \left\{ \frac{dX_i(Z)}{dZ}_{/Z=1} \right\} = N$$

Finally, equation (6) becomes

$$\bar{Y} = \lambda N \tag{IV.23}$$

IV.2 CONCLUSION

In this chapter, we have investigated the case of a random process graph. We were able to find closed expressions for the Z-transform and the average number of occupied processors in the system.

CHAPTER 5
AVERAGE NUMBER OF OCCUPIED PROCESSORS

In this chapter, we shall investigate the average number of occupied processors under very general conditions, given that the multi-processor system under investigation is in equilibruim. We shall prove that such average number of occupied processors depends solely on:

a. the job arrival rate;

b. the average number of tasks per job; and

c. the average service time per task.

In particular, it does not depend on:

a. the shape of the process graph (i.e., the precedence relationships among tasks in the process graph);

b. the distribution of the number of tasks per job;

c. the (conditional) distribution of the number of levels per job and the repartition of tasks among such levels;

d. the distribution of the task service time;

e. the number of processors in the system; and

f. the distribution of the job arrival process.

In section V.1, we shall deal with the case of an infinite number of processors. For such case, the system is always in equilibrium. Section V.2 deals with the case of a finite number of processors. Finally, in the conclusion, we provide a pictorial representation of the system utilization and the average number of occupied processors as a function of the number of available processors

in the multi-processor system, and a pictorial representation of the average time insystem and the average of the number of occupied processors as a function of the average rate of the job arrival process.

V.1 INFINITE NUMBER OF PROCESSORS

Throughout this section, we shall assume that the number of processors in the system is infinite. In section V.1.1, we shall deal with the case of constant service time per task, section V.1.2 is an investigation of the case of exponential service time per task, and in section V.1.3, we shall provide a more general result by letting the service time per task to be random.

V.1.1 CONSTANT SERVICE TIME PER TASK

In this section, we assume that the number of processors in the system is infinite and the service time per task is constant. We shall first state a theorem that does not depend on the particularities of the process graph.

THEOREM V.1

The average number of occupied processors, $\bar{Y}$, in the case of

1. infinite number of processors;

2. constant service time per task, say one unit; and

3. Poisson job arrival process with parameter λ,

is given by

$$\bar{Y} = \lambda \bar{N}$$

where $\bar{N}$ is the average number of tasks per job.

PROOF

Notice first that $\bar{Y}$ does not depend on the shape of the process graph and, in particular, does not depend on the distribution of tasks between the levels inside the process graph.

In the previous chapters we have already shown that, for N fixed, we have $\bar{Y} = \lambda N$, that is, for the cases

1. CASE 1: fixed process graph, N fixed and r fixed (see chapter 2)

2. CASE 2: semi-random process graph, N fixed, r fixed and random distribution of tasks among levels inside the process graph, (see chapter 3)

3. CASE 3: random process graph, N fixed and r random with $1 \le r \le N$, (see chapter 4)

Let us first give an alternate, but rather simple, proof for the above cases and then show that such result still holds for the more general case of a completely random process graph, N random and r random $1 \le r \le N$.

CASE 1: Fixed Process Graph, N fixed and r fixed

Since the job arrival process is Poisson with parameter λ and since only and every job that arrived during the interval [t-r,t] will occupy some processors at time t, then the average number of jobs present in the system is λr. On the other hand, the average number of tasks per level in the process graph is $\frac{N}{r}$. It follows that

$$\bar{Y} = \lambda r \frac{N}{r} = \lambda N$$

CASE 2: Semi-Random Process Graph, N fixed and r fixed

The proof exactly follows the proof of Case 1 since the average number of tasks per level in the process graph is again $\frac{N}{r}$.

CASE 3: Random Process Graph, N fixed and r random

Since the job arrival process is Poisson with parameter λ and, since $r = 1,\ldots,N$, then we can see such Poisson source of jobs as N Poisson sources, each with parameter λ_i, where

$$\lambda_i = \lambda P[r = i]$$

The ith source is, then, the source of jobs having N fixed and $r=i$ fixed. Let
$\bar{Y}_i$ = the average number of processors occupied by jobs generated by the ith source. Thence, we have

$$\bar{Y} = \sum_{i=1}^{N} \bar{Y}_i$$

On the other hand, since the jobs generated by the ith source have a fixed N and a fixed r=i, then by case 1 we obtain

$$\bar{Y}_i = \lambda_i N$$

Therefore, we get

$$\bar{Y} = \sum_{i=1}^{N} \bar{Y}_i$$

$$= \sum_{i=1}^{N} \lambda_i N$$

$$= N \sum_{i=1}^{N} \lambda P[r = i]$$

$$= \lambda N \sum_{i=1}^{N} P[r = i]$$

$$= \lambda N$$

ADDITIONAL CASE: Completely Random Process Graph N Random and r Random

Define

$$\bar{\bar{N}} = \sum_{n \geq 1} nP[N = n]$$

The average number of tasks per job over all jobs. Since the job arrival process is Poisson with parameter λ, then we can see it as an infinity of Poisson sources, each with parameter λ_i, where

$$\lambda_i = \lambda P[N = i]$$

That is, the ith source is the source of jobs having a fixed number, i, of tasks. Moreover, source i by itself can also be decomposed into i other Poisson sources, the jth (j=1,...,i) of which has parameter λ_{ij}, where

$$\lambda_{ij} = \lambda_i P[r_i = j] \quad j=1,...,i \text{ and } i=1,2,3,...$$

where r_i is a random variable counting the number of levels in a process graph given that the job is generated from the ith Poisson source. Thus,

$$\lambda_{ij} = \lambda P[N = i] P[r_i = j] \quad j=1,...,i \text{ and } i=1,2,3,...$$

Let us define the following quantities:

$\bar{Y}_i$ = average number of processors occupied by jobs generated by the ith source, i=1,2,3,...

$\bar{Y}_{ij}$ = average number of processors occupied by jobs

generated from the source ij (i.e., the jth source of the ith source), i=1,2,3,... and j=1,...,i

Thus, we have

$$\bar{Y}_i = \sum_{j=1}^{i} \bar{Y}_{ij} \quad and \quad \bar{Y} = \sum_{i=1}^{\infty} \bar{Y}_i$$

On the other hand, by using the result of case 1, we can write

$$\bar{Y}_{ij} = \lambda_{ij}\, i$$

Finally, we obtain:

$$\bar{Y} = \sum_{i=1}^{\infty} i \sum_{j=1}^{i} \lambda P[N=i]\, P[r_i=j]\, i$$

$$= \lambda \sum_{i=1}^{\infty} iP[N=i] \sum_{j=1}^{i} P[r_i=j]$$

$$= \lambda \sum_{i=1}^{\infty} iP[N=i]$$

$$= \lambda \bar{N}$$

V.1.2 EXPONENTIAL TASK SERVICE TIME

In this section, the number of processors in the system is assumed to be infinite, and the task service time is exponentially distributed. Ultimately, we shall provide, as in the previous section, a theorem that is independent of the particulars of the process graph. First, though, we shall take into account the particulars of the process graph.

LEMMA V.1

The average number of occupied processors $\bar{Y}$ for the case of

1. an infinite number of processors,
2. exponential task service time with parameter $\frac{1}{\mu}$,
3. Poisson job arrival process with parameter λ, and
4. N fixed and r fixed

is given by

$$\bar{Y} = \frac{\lambda N}{\mu} \quad and\,for\ \mu=1 \quad \bar{Y} = '\lambda N$$

PROOF

Let PG(N,r) denote a process graph with N tasks and r levels and random distribution of the tasks among the r levels. Let MPG(X,Y) denote a Markovian process graph (see chapter 1) with X tasks and Y levels.

Since the task service time is exponentially distributed with parameter μ, then we can convert our PG(N,r) into a Markovian MPG(X,N). Notice that X is finite since both N and r are finite. Each job using some processors at time t is occupying only one state of the MPG(X,N). Thence,

$$\bar{Y} = \begin{bmatrix} average\ number\ of\ jobs \\ occupying\ some\ processors \end{bmatrix} \cdot \begin{bmatrix} average\ number\ of\ tasks \\ per\ state\ over\ all\ MPG \end{bmatrix} \qquad (V.1)$$

Let S = average system time per job. Thus, from M/G/∞ results, we obtain

$$average\ number\ of\ jobs\ occupying\ some\ processors = \lambda S \qquad (V.2)$$

On the other hand, we can rewrite S as

$$S = \frac{N}{\mu\ (average\ number\ of\ tasks\ per\ MPG\ state)}$$

which gives

$$average\ number\ of\ tasks\ per\ MPG\ state = \frac{N}{\mu S} \qquad (V.3)$$

Finally, by using equations (V.1), (V.2) and (V.3), we get

$$\bar{Y} = \lambda S \frac{N}{\mu S} = \frac{\lambda N}{\mu}$$

iiii

LEMMA V.2

The average number of occupied processors $\bar{Y}$ for the case of

1. an infinite number of processors,
2. exponentially distributed task service time with parameter μ,
3. Poisson job arrival process with parameter λ, and
4. N fixed and r random, $r = 1,...,N$

is given by

$$\bar{Y} = \frac{\lambda N}{\mu} \quad and for \ \mu=1 \quad \bar{Y} = \lambda N$$

PROOF

It is enough to notice that the proof of Lemma (V.1) does not depend on the value of r.

||||

LEMMA V.3

The average number of occupied processors $\bar{Y}$ for the case of

1. an infinite number of processors,
2. exponentially distributed service time per task with parameter μ,
3. Poisson job arrival process with parameter λ, and
4. N random, $N = 1,2,3,...$ and r random, $r = 1,...,N$

is given by

$$\bar{Y} = \frac{\lambda N}{\mu} \quad and for \ \mu=1 \quad \bar{Y} = \lambda N$$

PROOF

Let the Poisson source with parameter λ be seen as an infinity of Poisson sources with the ith i=1,2,3,... having parameter λ_i where

$$\lambda_i = \lambda P[N = i].$$

with P[N =i] = P[job has i tasks] i=1,2,3,...

Let $\bar{Y}_i$ be the average number of processors occupied by jobs generated from the ith Poisson source. Thus, by Lemma (V.2), we obtain

$$\bar{Y}_i = \frac{\lambda_i i}{\mu}$$

and, finally,

$$\bar{Y} = \sum_{i=1}^{\infty} \bar{Y}_i$$

$$= \sum_{i=1}^{\infty} \frac{\lambda_i i}{\mu}$$

$$= \frac{1}{\mu} \sum_{i=1}^{\infty} \lambda P[N = i] i$$

$$= \frac{\lambda N}{\mu}$$

||||

The following theorem presents the same statement as theorem (V.1) set forth above but deals with the case of exponential service time per task.

THEOREM V.2

The average number of occupied processors $\bar{Y}$, in the case of

1. an infinite number of processors,

2. exponentially distributed service time per task with parameter μ, and

3. Poisson job arrival process with parameter λ

is given by

$$\bar{Y} = \frac{\lambda \bar{N}}{\mu} \quad and for \ \mu=1 \quad \bar{Y} = \lambda \bar{N}$$

where $\bar{N}$ is the average number of tasks per job.

PROOF

Notice first that $\bar{Y}$ is insensitive to the shape of the process graph and, in particular, does not depend on the distribution of tasks among the levels inside the process graph.
The proof is completely provided by Lemmas (V.1), (V.2) and (V.3).

||||

V.1.3 RANDOM SERVICE TIME PER TASK

In this section, we assume that the number of processors in the system is infinite but that the service time per task is considered to be random. As in the previous two sections, we shall formulate a theorem that is independent of the particulars of the process graph. Moreover, we shall relax the assumption that the job arrival process is Poisson. In fact, all we shall assume in this section is that the job arrivals are independent.

Let us define the following parameters:

λ	= the job arrival rate
$\bar{N}$	= the average number of tasks per job
$\bar{C}_j$	= the average concurrency per job over all jobs
$\bar{K}$	= the average number of jobs present in the system
S	= the average time a job spends in the system
$\bar{X}$	= the average service time per task

THEOREM V.3

The average number of occupied processors $\bar{Y}$ for the case of

1. an infinite number of processors,
2. random service time per task with average $\bar{X}$,
3. random job arrival process with average arrival rate λ (job arrivals independent), and
4. random process graph, that is,
   * N random
   * r random, r=1,...,N
   * random repartition of tasks among levels, and
   * random precedence relationships among levels

is given by

$$\bar{Y} = \lambda \bar{N} \bar{X} \quad \textit{and for} \quad \bar{X}=1 \quad \bar{Y} = \lambda \bar{N}$$

PROOF

Since the average number of occupied processors, $\bar{Y}$, represents the average concurrency in the system, it follows that

$$\bar{Y} = \bar{K}\ \bar{C}_j$$

Notice that $\overline{KC_j} = \bar{K}\ \bar{C}_j$, due to the fact that P is infinite. By using Little's formula Little , we have

$$\bar{K} = \lambda\, S$$

where S can be written as

$$S = \frac{\bar{N}}{\bar{C}_j}\,\bar{X}$$

It then follows that

$$\bar{Y} = \lambda S \bar{C}_j$$

$$= \lambda \frac{\bar{N}}{\bar{C}_j}\,\bar{X}\ \bar{C}_j$$

Thence,

$$\bar{Y} = \lambda \bar{N} \bar{X}$$

V.2 FINITE NUMBER OF PROCESSORS

In this section, the number of processors in the system is finite, say P. We shall prove that the average number of occupied processors, $\bar{Y}$, is still given by $\bar{Y} = \lambda \bar{N} \bar{X}$. Throughout this section, we assume that the multi-processor system is in equilibrium.

THEOREM V.4

If the multi-processor system is in equilibrium and work-conservative, then the average number of occupied processors $\bar{Y}$ for the case of

1. finite number of processors, say P,
2. random service time per task (possibly different distribution for each task) with overall average $\bar{X}$,
3. random job arrival process with average arrival rate λ (job arrivals independent), and
4. random process graph per job, that is for each job:
   * N random
   * r random, r=1,...,N
   * random repartition of tasks among levels, and
   * random precedence relationships among levels

is given by:

$$\bar{Y} = \lambda \bar{N} \bar{X} \quad \textit{and for} \quad \bar{X}=1 \quad \bar{Y} = \lambda \bar{N}$$

Moreover, if the system is overloaded then

$$\bar{Y} = P$$

PROOF

Let:

$\bar{n}_p$ = the average number of tasks per job processed by processor p, p=1,...,P

ρ_p = utilization of processor p, p=1,...,P

ρ = system utilization

The equilibruim condition is then $\forall$ p=1,...,P $\rho_p < 1$ and thus $\rho < 1$. We have

$$\rho_p = \lambda \bar{n}_p \bar{X}$$

$$\bar{N} = \sum_{p=1}^{P} \bar{n}_p$$

$$\bar{Y} = \sum_{p=1}^{P} \rho_p$$

$$= \sum_{p=1}^{P} \lambda \bar{n}_p \bar{X}$$

$$= \lambda \bar{N} \bar{X}$$

If the system is overloaded (i.e., system utilization greater than one) then it is easy to see that $\bar{Y} = P$ since all the processors are being used all the time.

Also, notice that, in equilibrium, we have

$$\rho = \frac{\bar{Y}}{P}$$

V.3 CONCLUSION

In this chapter, we have proved that the average number of occupied processors in a multi-processor system with P=1,2,3,... processors is given by $\bar{Y} = \lambda \bar{N} \bar{X}$, where $\bar{N}$ and $\bar{X}$ represent the average number of tasks per job and the average service time per task, respectively. It is interesting to note that such average number of occupied processors does not depend on the description of the jobs (e.g., the distribution of the number of tasks per job, the distribution of the number of levels in the process graph, the repartition of the tasks among the levels, the precedence relationships among the levels inside the process graph, the distribution of the task service time, the distribution of the job arrival process and the number of processors in the system given that such multi-processor system is in equilibium).

Figure (V.1) and figure (V.2) below give a pictorial representation of the system utilization and the average number of occupied processors in the system as a function of the total number of available processors in the system. In Figure V.1, we have $\lambda \bar{N} \bar{X} < 1$, that is. the utilization of the system when P=1 is less than unity. In Figure V.2, we have $\lambda \bar{N} \bar{X} \geq 1$, that is, the utilization of the system when P=1 is greater than unity.

versus P, $\lambda \bar{N} \bar{X} < 1$ "

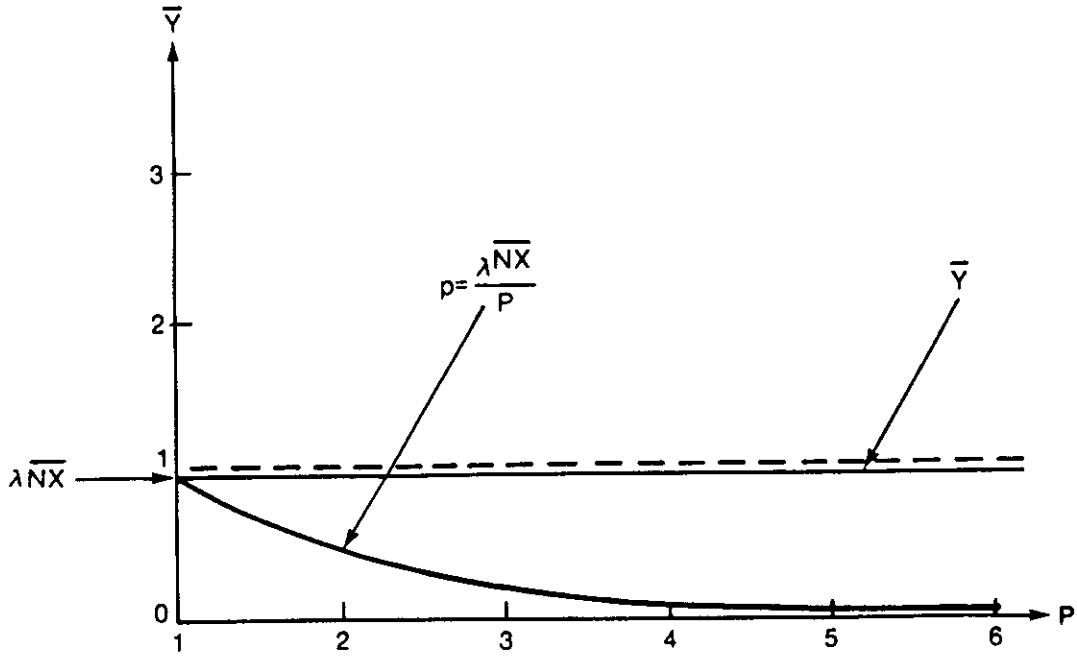


FIGURE V.1
System Utilization and Average Number of Occupied Processors versus P, $\lambda\overline{NX}<1$

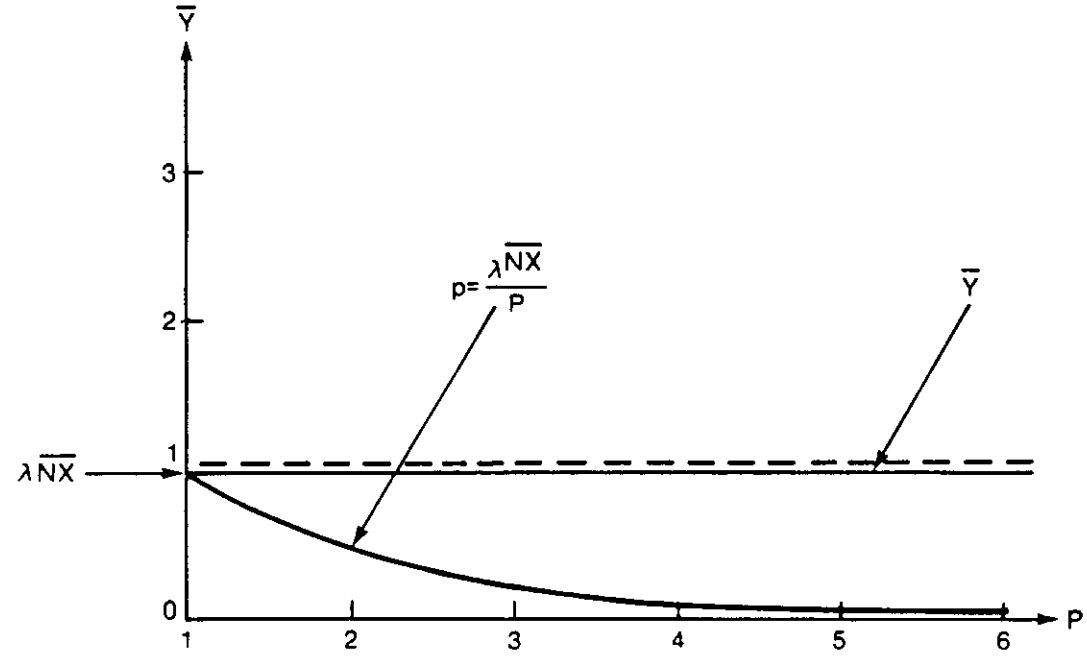


FIGURE V.2
System Ultilization and Average Number of Occupied Processors versus P, $\lambda\overline{NX}\geq 1$ versus P, $\lambda\overline{NX}\geq 1$ "

Figure (V.3) below gives a pictorial representation of the average number of occupied processors $\bar{Y}$ and the average system time S as a function of the job arrival rate λ for a given number of processors P.

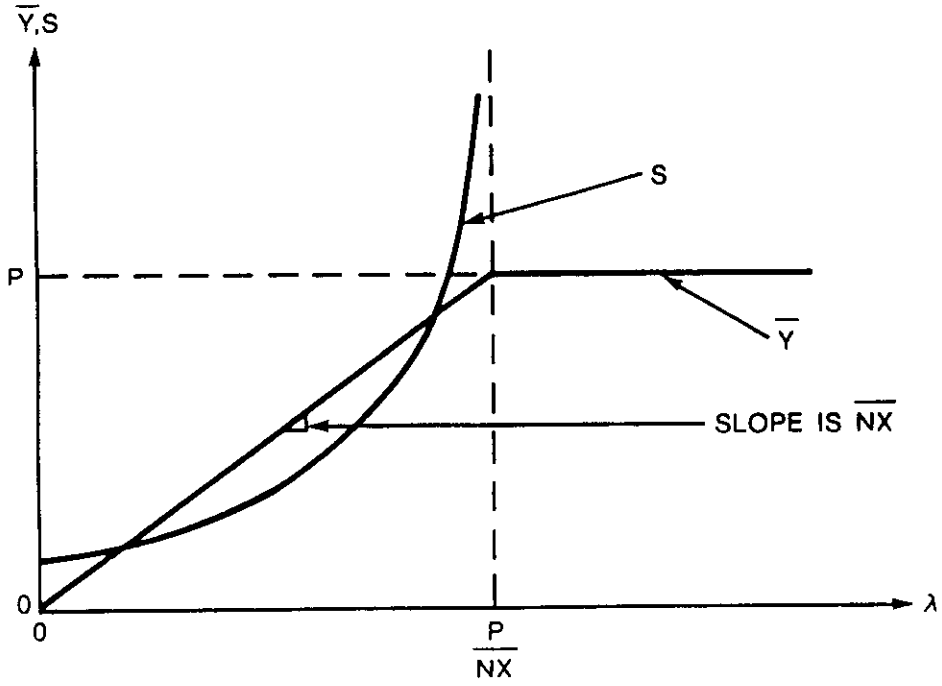


FIGURE V.3
System time and Average Number of Occupied Processors, versus λ

CHAPTER 6
TASK INTERARRIVAL TIME DISTRIBUTION

In this chapter, we shall derive a closed form expression of the distribution of the task interarrival times to the system. Throughout the chapter, we assume that the number of available processors is infinite and that the job arrival process is Poisson with parameter λ. In section VI.1, we investigate the case of exponential task service time, and in section VI.2, we shall consider the case of constant service time per task.

VI.1 EXPONENTIAL TASK SERVICE TIME

In this section, we shall find the distribution of the interarrival time between tasks to the system for the special case of N tasks and N levels process graph, PG(N,N). A job is then represented as a vertical string of tasks. We also consider that the job arrival process is Poisson with parameter λ and that the task service time is exponential with parameter μ.

Let us define the following:
t_j = random variable measuring the interarrival time between jobs
t_a = random variable measuring the interarrival time between tasks to the system

Let us place ourselves just at a task arrival (hereafter called a tagged arrival), and let t_0 be the arrival instant of such tagged arrival. Notice that the next incoming task can be

1. from the same job as the tagged task,
2. from another job that is already in the system at time t_0,
3. from a newly arriving job (such a job must arrive to the system after time t_0 but no later than $(t_0 + t)$).

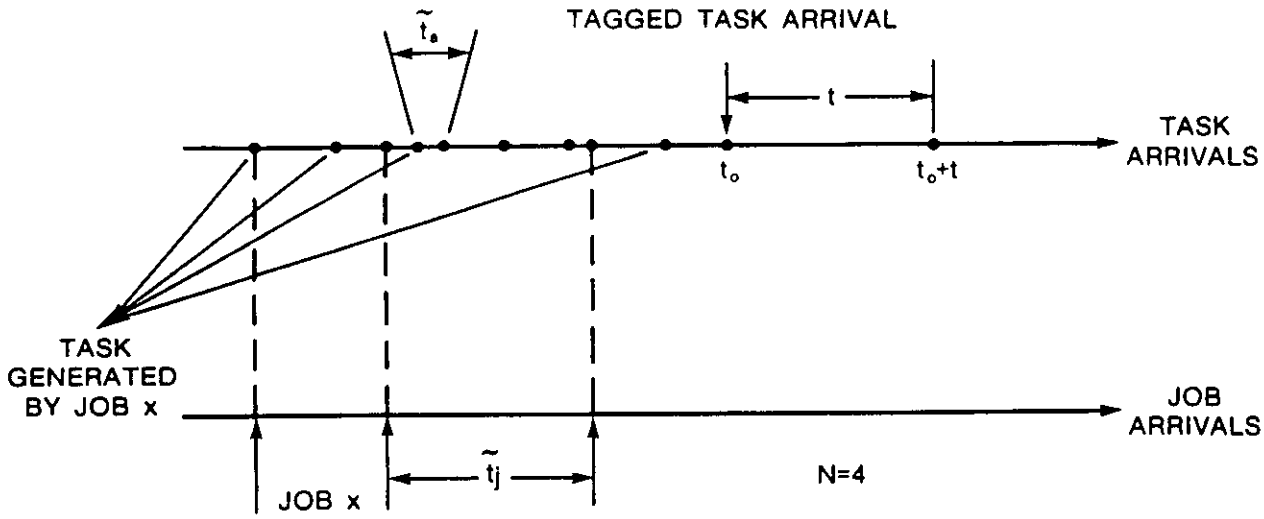


FIGURE VI.1.1
Task Arrivals Diagram; Exponential Task Service Time

CASE 3

The next incoming task is from a newly arriving job. We then have

$$P[\text{ a task arrival in the interval } [t_0, t_0+t\,]\,] = P[\text{ job arrival in } [t_0, t_0+t\,]\,] = 1 - e^{-\lambda t}$$

CASES 1 AND 2

The next incoming task can be either from a newly arriving job or from a job that is already in the system. Let us assume that we have k jobs in the system at time t_0 (just after time t_0). Notice that only those jobs that are not executing their last task will generate further tasks; those jobs executing their last task will not generate any more tasks. Let us then find the probability of a job executing its last task, given that such a job is already in the system.

$$P[\text{job is executing its last task / it is in the system}] = P[\text{ remaining tasks in such job} = 0]$$

$$= \frac{1 - P[\text{ number of tasks in a job} \le 0\,]}{\text{average number of tasks per job}}$$

$$= \frac{1-0}{N}$$

Thus, we have

$$P[\text{ job is executing its last task / it is in the system }] = \frac{1}{N} \qquad \text{(VI.1)}$$

and, subsequently,

$$P[\text{ job is not executing its last task / it is in the system }] = \frac{N-1}{N} \qquad \text{(VI.2)}$$

We have the following:

$P[t_a > t] = P[\text{ no task arrival in the interval } [t_0, t_0+t] \text{ / task arrival at time } t_0]$

$$= P\left\{\begin{array}{l}\text{no job arrival in the interval } [t_0, t_0+t] \text{ and no task arrivals from existing} \\ \text{jobs at time } t_0 \text{ the interval } [t_0, t_0+t] \text{ , given that at least one job is in the system at time } t_0\end{array}\right\}$$

Since the event { no job arrival in [t_0, t_0+t] } and the event { no task arrivals in [t_0, t_0+t] from existing jobs in the system at time t_0 } are disjoint, then,

$$P[\ t_a > t\ ] = P[\text{ no job arrival in } [t_0, t_0+t]\ ]\ .\ P\left[\begin{array}{l}\text{no task arrivals in } [t_0, t_0+t] \text{ from existing} \\ \text{jobs in the system at time } t_0 \text{ / at} \\ \text{least one job is in the system at time } t_0\end{array}\right].$$

$$= P[\text{ at least one job is in the system at time } t_0\ ]$$

On the other hand, since the job arrival process is Poisson with parameter λ, we have:

$$P[\text{ k jobs are in the system at time } t_0\ ] = \frac{\left(\lambda \bar{X}\right)^k}{k!} e^{-\lambda \bar{X}}$$

where

$\bar{X}$ is the average job service time, that is $\bar{X} = \frac{1}{\mu}$ and,

$$P[\text{ at least one job is in the system at time } t_0\ ] = 1 - e^{-\lambda \bar{X}}$$

Also, we have

$$P[\text{ no job arrivals in the interval } [\ t_0, t_0+t\ ] = e^{-\lambda t}$$

Now,

$$P\left\{\begin{array}{l}\text{no task arrivals in } [\ t_0, t_0+t\ ] \text{ from existing} \\ \text{jobs in the system at time } t_0 \text{ / at least} \\ \text{one job is in the system at time } t_0\end{array}\right\} = \sum_{k=1}^{\infty} P\left\{\begin{array}{l}\text{no task arrivals in } [\ t_0, t_0+t\ ] \\ \text{from existing jobs in the system} \\ \text{at time } t_0 \text{ / k jobs are in the} \\ \text{system at time } t_0\end{array}\right\} P\left\{\begin{array}{l}\text{k jobs are in the} \\ \text{system at time } t_0\end{array}\right\}$$

But, we have:

$$P\left\{\begin{array}{l}P[\text{ no task arrivals in } [\,t_0,t_0+t\,]\\ \text{from existing jobs in the system}\\ \text{at time } t_0 \,/\, k \geq 1 \text{ jobs exist}\\ \text{in the system at time } t_0\end{array}\right\} = \sum_{i=0}^{k} P\left\{\begin{array}{l}\text{no task arrivals in } [\,t_0,t_0+t\,]\\ \text{from the } k\geq 1 \text{ existing jobs in the}\\ \text{system at time } t_0 \,/\, i \text{ jobs}\\ \text{among the } k \text{ are not executing}\\ \text{their last tasks}\end{array}\right\} . P\left\{\begin{array}{l}i \text{ jobs among the } k \text{ are not}\\ \text{executing their last tasks}\end{array}\right\}$$

Therefore, it follows that:

$$P\left\{\begin{array}{l}\text{no task arrivals in } [\,t_0,t_0+t\,]\\ \text{from existing jobs in the}\\ \text{system at time } t_0 \,/\, k \geq 1 \text{ jobs}\\ \text{in the system at time } t_0\end{array}\right\} = \sum_{i=0}^{k} \left[e^{-\mu t}\right]^i \binom{k}{i} \left[\frac{N-1}{N}\right]^i \left[\frac{1}{N}\right]^{k-i}$$

$$= \left[\frac{N-1}{N}\, e^{-\mu t} + \frac{1}{N}\right]^k$$

since

$$P[\text{ no task arrivals in } [\,t_0,t_0+t\,] \text{ from the } i \text{ jobs not executing their last tasks }] = \left[e^{-\mu t}\right]^i$$

Thus, it follows that

$$P\left\{\begin{array}{l}\text{no task arrivals in } [\,t_0,t_0+t\,]\\ \text{from existing jobs in the}\\ \text{system at time } t_0 \,/\, \text{at least one}\\ \text{job is in the system at time } t_0\end{array}\right\} = \sum_{k=1}^{\infty} \left[\frac{N-1}{N}\, e^{-\mu t} + \frac{1}{N}\right] . \frac{\left[\lambda \bar{X}\right]^k}{k!}\, e^{-\lambda \bar{X}}$$

$$= e^{-\lambda \bar{X}} \left\{ e^{\frac{\lambda \bar{X}}{N}\left[1+(N-1)e^{-\mu t}\right]} - 1 \right\}$$

and $P\,[t_a > t\,]$ becomes

$$P\,[t_a > t\,] = e^{-\lambda t} \,.\, e^{-\lambda \bar{X}} \left\{ e^{\frac{\lambda \bar{X}}{N}\left[1+(N-1)e^{-\mu t}\right]} - 1 \right\} . \frac{1}{1-e^{-\lambda \bar{X}}}$$

which finally gives

$$P\,[t_a \leq t\,] = 1 + \frac{e^{-\lambda \bar{X}}}{1-e^{-\lambda \bar{X}}}\, e^{-\lambda t} - \frac{e^{-\lambda \bar{X}\left[\frac{N-1}{N} - \frac{N-1}{N} e^{-\mu t}\right]}}{1-e^{-\lambda \bar{X}}} . e^{-\lambda t} \quad t \geq 0 \qquad \text{(VI.2)}$$

VI.1.1 AVERAGE INTERARRIVAL TIME OF TASKS TO THE SYSTEM

Let us denote by $\bar{t}_a$ the average interarrival time between tasks to the system. We then have

$$\bar{t}_a = \int_0^\infty \left\{1 - P\,[t_a \le t\,]\right\} dt$$

$$= \int_0^\infty \left\{ -\frac{e^{-\lambda \bar{X}}}{1 - e^{-\lambda \bar{X}}}\, e^{-\lambda t} + \frac{e^{-\lambda \bar{X}\frac{N-1}{N}}}{1 - e^{-\lambda \bar{X}}}\, e^{\lambda \bar{X}\frac{N-1}{N} e^{-\infty}}\, e^{-\lambda t} \right\} dt$$

which gives

$$\bar{t}_a = \frac{e^{-\lambda \bar{X}\frac{N-1}{N}}}{1 - e^{-\lambda \bar{X}}} \sum_{i=0}^{\infty} \frac{\left[\lambda \bar{X}\frac{N-1}{N}\right]^i}{\left[\lambda + i\mu\right] i\,!} - \frac{e^{\left[-\lambda \bar{X}\right.}}{\lambda \left[1 e^{-\lambda \bar{X}}\right.} \qquad \text{(VI.3)}$$

Notice that for N=1, equation (VI.3) gives:

$$\bar{t}_a = \frac{1}{\lambda}$$

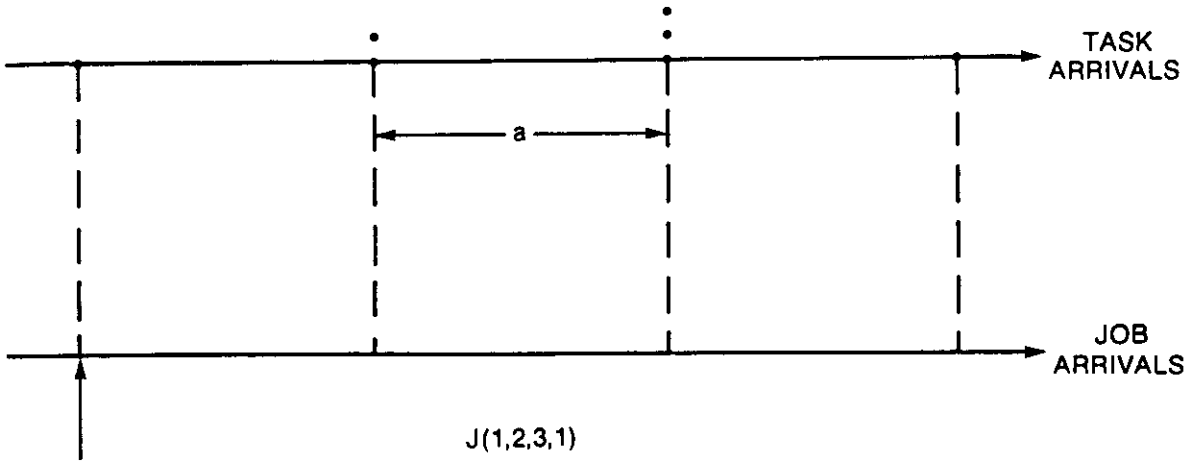
In fact, this is exactly the average interarrival time of a usual Poisson arrival process.

VI.2 CONSTANT TASK SERVICE TIME

Our multi-processor system can be viewed as an FCFS queueing system with infinite number of processors. In this section, we are interested in finding the interarrival time of tasks to the system. A job is represented as a process graph with N tasks and r levels, where $1 \le r \le N$, and N and r are fixed for all jobs. The task service time is constant, say a units of time.

Let $J = \left(n_1, n_2, \ldots, n_r\right)$ be a given process graph (a given job) where n_i is the number of tasks at level i, i=1,..,r. Thence, n_1=1, n_r=1 and $\sum_{i=1}^{r} n_i = N$. Since the number of processors is infinite, then a job, upon its arrival, immediately creates its first task. Each a seconds thereafter, the job creates all the tasks of its next level as shown in the figure below.

Each job is then represented as a vertical string of super-tasks. Super-task i comprises n_i tasks. Super-task i is the set of tasks at level i in the process graph. Let us first consider the interarrival time of super-tasks to the system.



FGURE VI.2.1
Internal Creation of Tasks

LEMMA VI.1

Provided that $a \geq 0$, no simultaneous super-task arrivals can occur.

PROOF

Super-task arrivals from the same job are separated by exactly a seconds. Since $a \geq 0$, then no simultaneous super-task arrivals from the same job can occur. On the other hand, to have simultaneous super-task arrivals from 2 different jobs, the arrival of these two jobs must be separated by exactly ia seconds where $0 \leq ia \leq r-1$. However, since the job arrival process is Poisson with parameter λ, then we have

$$P[\text{ job arrival in } [\ t,t+dt\ ] \text{ and job arrival in } [\ t+ia,t+ia+dt\ ]\ ]$$

$$= P[\text{ job arrival in } [\ t,t+dt\ ]\ ].\ P[\text{ job arrival in } [\ t+ia,t+ia+dt\ ]\ ]$$

$$= [\ \lambda\, dt+O(t)\ ].[\ \lambda\, dt+O(t)\ ]$$

$$= \lambda^2 dt^2 + O(t) = O(t)$$

since dt is very small.

The above can also be seen by noticing that the probability of having two arrivals separated by exactly ia seconds is the same as the probability of having simultaneous arrivals.

IIII

Lemma (VI.1) says that, to characterize the distribution of the tasks interarrival time process, we need to find:

1. the distribution of the super-task interarrival times; and

2. the distribution of the super-task size.

First, we shall find the distribution of the super-task interarrival times. For such purpose, a job is represented by a process graph with r levels and r super-tasks. Let us define the following quantities:
t = random variable measuring the interarrival time between jobs
t_a = random variable measuring the interarrival time between super-tasks

Our objective is to find $P[t_a \le t]$.

CASE r=1

Since each job creates just one super-task and this is exactly upon its arrival to the system, then the distribution of the interarrival time between tasks is the same as the job interarrival time distribution. We have

$$P[t_a \le t] = 1 - e^{-\lambda t} \qquad t \ge 0 \tag{VI.4}$$

CASE r ≥2
We first distinguish two cases.

2. CASE WHERE $0 \le t < a$

We have

$$P[t_a \le t] = 1 - P[t_a > t]$$

and

$$P[t_a > t] = P[\textit{ super-task arrivals during the interval t }]$$

$$= P[\textit{ no job arrivals in the intervals } I_0, I_1, I_2, \ldots, I_r]$$

$$= \prod_{i=0}^{r-1} e^{-\lambda t}$$

which gives

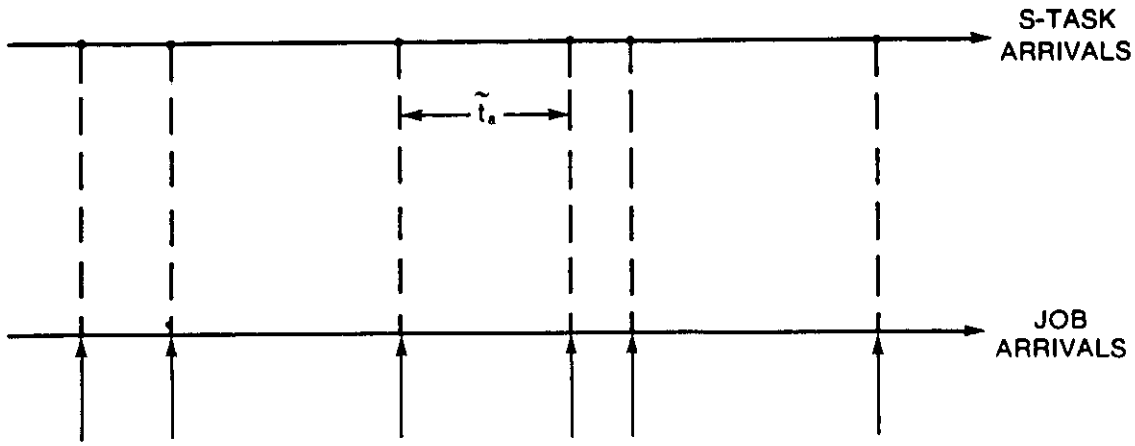


FIGURE VI.2.2
Task Interarrival Diagram, Case of r=1

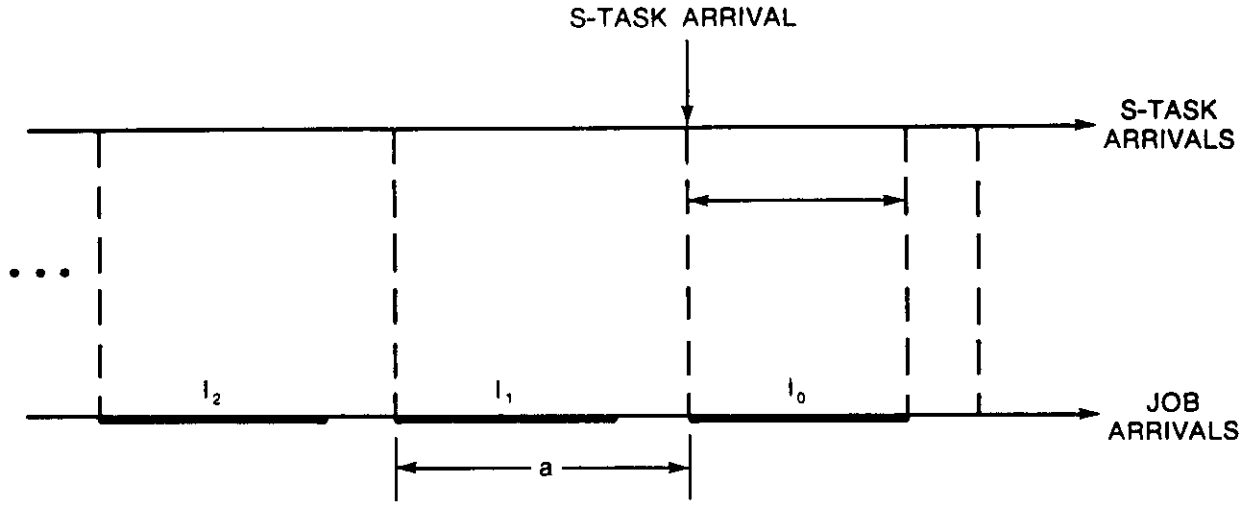


FIGURE VI.2.3
Task Interarrival Diagram, Case of $0 \leq t < a$

$$P[t_a \leq t] = 1 - e^{-\lambda t} \qquad \text{(VI.5)}$$

1. CASE $t \geq a$

We have

$$P[t_a \leq t] = 1 - P[t_a > t],$$

and, since $t \geq a$, we must distinguish between the following two cases:

1. t=a, by Lemma (VI.1) the super-tasks belonging to the same job;

2. $t > a$, the super-tasks belonging to different jobs.

1.1 CASE t=a

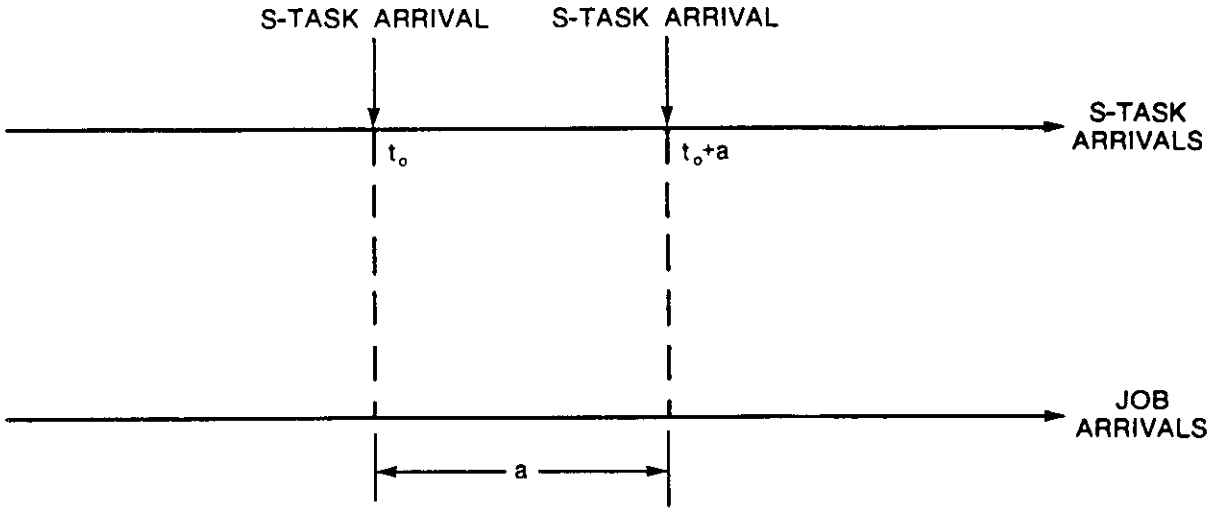


FIGURE VI.2.4
Task Interarrivals Diagram, Case of t=a

Since t=a, then P[$t_a \leq t$] = P[$t_a \leq a$] = 1 - P[$t_a > a$], but
P[$t_a > a$] =
P[no super-task arrivals in the interval (t_0, t_0+a)] and [no super-task arrival at time (t_0+a)].
Also, since both events are disjoint, we obtain
P[$t_a > a$] =
P[no super-task arrivals in the interval (t_0, t_0+a)] · P[no super-task arrivals at time (t_0+a)].
Finally, from case 1 where $0 \leq t < a$, we get

$$P[\text{ no super-task arrivals in the interval } (t_0, t_0+a)\ ] = e^{-\lambda r a}$$

and, by application of Lemma (VI.1), we get
P[no super-task arrivals at time (t_0+a)] = P[the tagged super-task is the last super-task of its job]
This probability can be calculated by two different methods:

METHOD 1

We know that a job has r super-tasks and that each super-task takes a seconds of processing time. Thence,

$$P[\textit{ job is executing its ith super-task / job is in the system }] = \frac{1}{r}$$

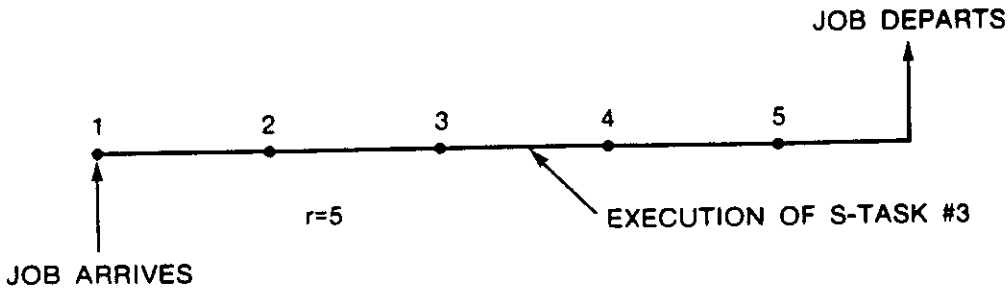


FIGURE VI.2.5
Execution of a Super-Task

METHOD 2

P[no super-task arrivals at time (t_0 +a)] = P[remaining number of super-tasks of the tagged job = 0]
Let us define the following PDF:

$$F_{ST}(n) = P[\textit{ number of super-tasks in a job} \le n]$$

Thus, we have

$$F_{ST}(n) = \begin{cases} 1 & n=r \\ 0 & otherwise \end{cases}$$

and the corresponding pdf $f_{ST}(n)$ = P[number of super-tasks in a job = n]. Thus, we have

$$f_{ST}(n) = \begin{cases} 1 & n=r \\ 0 & otherwise \end{cases}$$

and the average number of super-tasks in a job is equal to r. From the above definitions, we get:

$$P[\textit{ remaining number of super-tasks of the tagged job }] = \frac{1 - F_{ST}(0)}{r} = \frac{1}{r}$$

Notice that the above is just the residual life formula. It then follows that

$$P[t_a > a] = \frac{e^{-\lambda r a}}{r} \qquad t=a \tag{VI.6}$$

1.2 CASE $t > a$

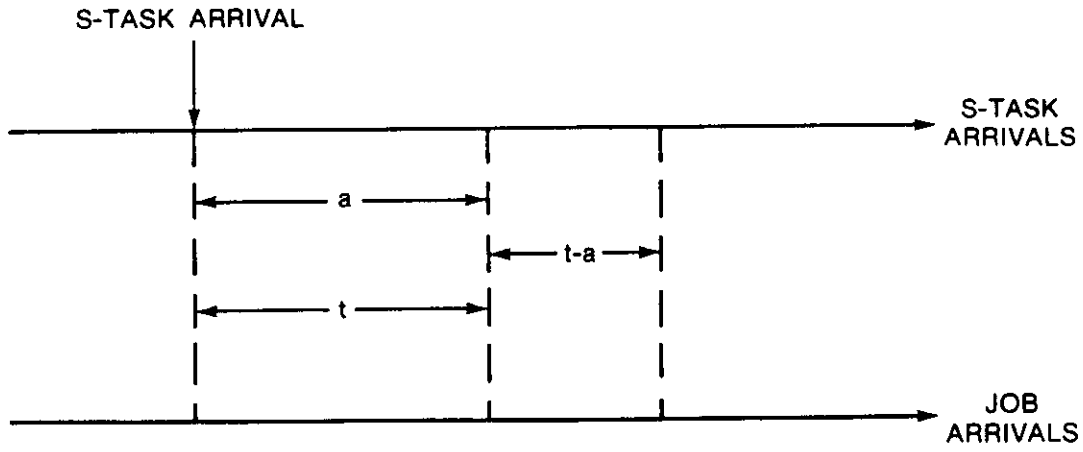


FIGURE VI.2.6
Task Interarrivals Diagram, case of $t > a$

We have:

$$P[t_a \le t] = 1 - P[t_a > t] \quad \textit{and;}$$

$$P[t_a > t] = P[\textit{ no super-task arrivals during } t\ ]$$

and, since by hypothesis t > a,

$$= P[\textit{ no super-task arrivals in } [\ t_0+a\ ,t_0+t\ ]\ ]$$

$$= P[\textit{ no job arrivals during } (t-a)\ ]$$

We then obtain

$$P[t_a > t] = e^{-\lambda(t-a)} \quad t > a \tag{VI.7}$$

Finally, putting the two cases together (i.e., for $t \ge a$), we get

$$P[t_a \le t] = 1 - P[t_a > t]$$

$$= 1 - P[\textit{ no super-task arrivals during } t\ ]$$

$$= 1 - P\begin{bmatrix}\textit{no super-task arrivals during } t\ / \\ \textit{no super-task arrivals during } a\end{bmatrix} \cdot P[\textit{ no super-task arrivals during } a\ ]$$

$$= -P \left[\begin{matrix} \textit{no super-task arrivals during } t \textit{ /} \\ \textit{super-task arrivals during } a \end{matrix} \right] \cdot P[\textit{ super-task arrivals during } a\ ]$$

Notice that:

$$P[\textit{ no super-task arrivals during } t \textit{ / super-task arrivals during } a\ ] = 0$$

Using equations (VI.6) and (VI.7), we obtain:

$$P[t_a \le t] = 1 - \frac{e^{\lambda a(1-r)}}{r} e^{-\lambda t} \quad t \ge a \tag{VI.8}$$

VI.1.1 AVERAGE INTERARRIVAL TIME BETWEEN SUPER-TASKS

Let: $\overline{t}_a$ = average interarrival time between super-tasks.

So,

$$\overline{t}_a = \int_0^{\infty} \left[1 - P[t_a \le t]\right] dt$$

and, using equations (VI.5) and (VI.8), we get

$$\overline{t}_a = \int_0^a e^{-\lambda r t}\, dt + \int_a^{\infty} \frac{e^{\lambda a(1-r)}}{r} e^{-\lambda t}\, dt$$

which gives

$$\overline{t}_a = \frac{1}{\lambda r} \quad r \ge 2$$

and, for the case of r=1, from equation (VI.4) we get

$$\overline{t}_a = \frac{1}{\lambda} \quad r = 1$$

Thence,

$$\overline{t}_a = \frac{1}{\lambda r} \quad r \ge 1 \tag{VI.9}$$

VI.2.2 VARIANCE OF THE INTERARRIVAL TIME BETWEEN SUPER-TASKS

Let $\sigma^2_{t_a}$ be the variance of the interarrival times between super-tasks. We have

$$\sigma^2_{t_a} = E[t_a^2] - E[t_a]^2$$

Using equations (VI.7) and (VI.8) (and after some calculations), we obtain

$$\sigma^2_{t_a} = \frac{1 - 2(1-r)\,e^{-\lambda a r}}{(\lambda r)^2} \qquad r \geq 1 \tag{VI.11}$$

THEOREM VI.1

For any random process graph with N tasks and r levels, r=1,...,N and for the case of an infinite number of processors and constant service time per task, the average number of jobs occupying some processors at any time t is given by λ r.

PROOF

The proof follows directly from equation (VI.9). Also, notice that since the processing time per task is constant and the number of processors is infinite, then every job arriving in the interval (t-r,t) will occupy some processors at time t. On the other hand, since the job arrival process is Poisson with parameter λ, then the average number of jobs arriving in an interval of length r is λ r.

IIII

THEOREM VI.2

For any random process graph with N tasks and r levels, r=1,..,N and for the case of an infinite number of processors and constant service time per task, a job that occupies some processors at time t participates on the average in $\frac{N}{r}$ tasks.

PROOF

Since the average number of occupied processors is λN and the average number of jobs occupying some processors is λr , it follows that the average number of participating tasks per job is $\frac{N}{r}$.

IIII

In the sequel, we shall find the distribution of the super-task size. Recall that a job is represented by $J = (n_1, n_2, \cdots, n_r)$, where n_i is the number of tasks at level i, i=1,...,r

LEMMA VI.2

The distribution of the size of an arrival (i.e., a super-task), given that all jobs have the same process graph, is given by

$$P[S_s = k] = \frac{1}{r} \sum_{\substack{n_i \in J \\ i=1,\ldots,r \\ s.t.\ n_i = k}} 1 \qquad k \geq 1$$

where S_s denotes the random variable measuring the size of a super task.

PROOF

Consider the arrival process of super-tasks to the system. Take any arrival and call it a tagged arrival. This tagged arrival belongs to a given job. Call such a job a tagged job. Thence, we have

$$P\left[\begin{array}{l}\textit{tagged arrival is the jth}\\ \textit{super-task of the tagged job}\end{array}\right] = P\left[\begin{array}{l}\textit{tagged job is executing its jth}\\ \textit{level / tagged job is in the system}\end{array}\right]$$

$$= \frac{1}{r}$$

We then obtain

$$P[S_z=k] = \frac{1}{r}(\textit{ number of levels having k tasks })$$

IIII

Now let us consider a more general situation where jobs can have a random process graph but still with N tasks and r levels with r=1,...,N.

LEMMA VI.3

For a random process graph with N tasks and r levels, r=1,...,N, the distribution of the size of an arrival (i.e., super-task) is given by

$$P[S_z=k] = \frac{\binom{N-k-1}{r-2}}{\binom{N-1}{r-1}} \qquad r\geq 2\,,\quad 1\leq k\leq r$$

and

$$p[S_z=k] = \begin{cases} 0 & \text{if } k\neq N \\ 1 & \text{if } k=N \end{cases} \qquad r=1$$

PROOF

The proof is a direct consequence of proposition (I.1).

IIII

VI.3 CONCLUSION

In this chapter, we have investigated the distribution of the task interarrival times to the system. Two cases are studied, namely, the case of exponential task service time and the case of constant service time per task.