1981

The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems

Subhash C. Agrawal

Jeffrey P. Buzen

Report Number:
81-384

The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems

Subhash C. Agrawal

Purdue University

Jeffrey P. Buzen

BGS Systems, Inc.

CSD-TR-384

ABSTRACT

The aggregate server method is an approximate, iterative method for analyzing computer systems containing serialization delays. Examples of serialization delays include delays encountered while waiting for entry into critical sections, non-reentrant subroutines, and locks. The method involves introduction of aggregate servers into a queueing network to represent the serialization delay. Service time requirements at all servers are modified appropriately to account for the contention at physical devices among serialized and non-serialized customers.

The algorithm is developed for single class closed queueing networks of single load independent servers employing processor sharing scheduling discipline. Results of validation based on comparison with exact numerical solution are presented. Some factors affecting the accuracy of the method are also discussed. Extensions to multi-class queueing networks, variable rate servers and other scheduling disciplines are also suggested.

Key Words and Phrases: aggregate server method, approximation, critical sections, meta-modeling, operational analysis, performance evaluation, product form, queueing networks, serialization delays.

CR Categories: 4.32, 4.33, 4.35, 4.6, 8.1

This work was supported, in part, by NSF grant MCS78-01729. Authors' Present Addresses: Subhash C. Agrawal, Purdue University, Department of Computer Sciences, West Lafayette, Indiana, 47907. Jeffrey P. Buzen, BGS Systems, Inc. 1 University Office Park, 29 Sawyer Road, Waltham, Massachusetts 02254;

The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems

Subhash C. Agrawal

Purdue University

Jeffrey P. Buzen

BGS Systems, Inc.

1. Background

When using queueing networks as models of computer systems, it is conventional to represent programs as customers circulating in the network, and to represent physical devices such as CPUs, disks and drums as servers. This approach is well suited for analyzing queueing delays caused by contention among programs for the active physical devices in a system. If these device contention delays are computed correctly, satisfactory models of overall system performance can be obtained in many cases.

There are, however, a large number of cases where satisfactory models must include delays for passive resources, delays that are materially different from active device contention delays. (An *active* resource contains processor(s) that render service. The delay per visit includes queueing and service from the processor(s). A *passive* resource is one that must be held as a precondition for holding active resources. The delay is queueing plus delays at active devices.) An important example is the delay that programs experience while waiting for main memory, or mounting of tapes and disk packs.

Note that programs usually do not begin execution until all the passive resources they require have been allocated. In addition, a program usually holds all its passive

resources until it terminates. When these conditions hold, and if the passive resources are indistinguishable, the technique of decomposition can be used to obtain a satisfactory approximation for the delays due to passive resource contention. It should be noted, however, that the extension of decomposition techniques from single class to multi-class models is quite difficult in practice. Also, the computation of auxiliary parameters for the decomposition can sometimes be complex, as in the case where the passive resource is permission to access granules in a database [14].

A second important category of delay not treated by conventional queueing network models is the stretchout of device service times (active servers) because of contention for I/O subsystem components such as channels, control units, and heads of string. A number of auxiliary models for approximating this stretchout factor have been developed for specific architectures [2]. The integration of such auxiliary models into a higher level queueing network presents a number of practical difficulties, but the use of auxiliary models to deal with service time stretchout in I/O subsystems appears, in principle, to be sound. An alternative approach based on the "method of surrogates" has also been proposed for this class of problems [12]. This approach has several advantages and can be generalized to other types of simultaneous resource possession problems. However, like decomposition technique, this method's generalization to multi-class models appears difficult.

A third category of delay that is not represented in conventional queueing network models is the *serialization delay* that arises because of contention for critical sections, non-reentrant subroutines, and other software control structures that cause processing to become serialized. The most common sources of serialization delays are routines that perform resource allocation, modify internal data structures, or update external files and databases.

Note that programs experience serialization delays after they have been allocated their required passive resources and have begun active processing. Also, programs

pass in and out of serialized phases during their execution. These two factors distinguish serialization delays from delays for passive resources such as main memory, tape drives, or the database granules discussed by Potier and Leblanc [14].

Serialization delays can often be neglected without affecting the accuracy of a model significantly. This is because the removal of serialization delays will generally cause an increase in the queueing delays at the servers that are accessed within the serialization phases; in effect, part of the queue for entry into a serialization phase is shifted to the original servers, and thus the net impact of ignoring serialization delays may be small. In fact, if the serialized processing consists only of a single processing burst at a single server with FCFS scheduling, neglecting the serialization delays will not introduce any error at all. However, if significant amount of processing is serialized, serialization delays must be included explicitly in the model to yield satisfactory results.

The literature on serialization delays is limited. Smith and Browne [15] have proposed an approach for treating this problem, but this approach was not subject to systematic validation. There was, however, a limited validation based on measurements of a real system. Unfortunately, the measurements were taken during interval when serialization delay contributed only a small fraction of the overall response time. Thus, even though excellent response time validation were obtained, it is not possible to make conclusive statements about the validity of Smith and Browne's [15] serialization delay model from the data that was presented.

Kumar and Gonsalves [13] present another method for modeling software structures in distributed systems and discuss an example of modeling of critical sections. They consider software modules to be servers in the queueing network, and physical resources circulate in the network as customers. When a customer (a device) visits a server (a software module), it means that the device wants to do processing on behalf of the software module. While their method is suitable for problems like modeling of

delays due to software locking, it does not seem to model different kinds of resources (e.g., CPU and I/O devices) adequately. For example, at any given time, a customer usually can be queued either at a CPU or at an I/O device, but not at both. Their methodology does not appear to allow this constraint to be represented.

Another approach to analyzing serialization delays is the aggregate server method originally developed by Buzen, Liu and Shum [8] for use with BEST/1 modeling package [1,4]. In this paper, we present a detailed discussion of the rationale and the conceptual basis for the aggregate server technique. We then present a new algorithm for evaluating aggregate server models and a systematic validation of the aggregate server method based on comparison with exact numerical solutions of detailed models that incorporate serialization delays explicitly.

2. Terminology

Consider a critical section entry to which is controlled by using a semaphore S [11]. A process wishing to enter this critical section performs a *wait* operation on the semaphore. If the count of the semaphore is 0 (or negative), the process will have to wait until the semaphore is signaled and this process is readied. After returning from the wait, the process will be the only process executing in the critical section. When the processing inside the critical section is completed, the process will exit the critical section by *signal*ing the semaphore S, and thereby, awakening a waiting process, if any.

The term *serialized phase* will be used to denote single threaded processing, for example, the critical section processing discussed above. That is, at most one customer may be actively executing in a serialized phase at any given time. When a customer (a process) is executing in a serialized phase, it will be called a *serialized customer*. Processing serialized using different semaphores occurs in different serialization phases.

The phase of processing in which different customers are not serialized will be called a *non-serialized phase*, and accordingly, a customer executing in the non-serialized phase will be called a *non-serialized customer*.

3. Overview of the Aggregate Server Method

The basic idea behind the aggregate server method is quite simple, and generalizes directly from consideration of the restricted case where there is a single customer class and a single critical section (a serialized phase). Consider such a network containing N customers and K servers. Let D_i be the total service time per job at server i. In the notation of Denning and Buzen [10], D_i is equal to $V_i S_i$, which is the product of the number of visits per job to device i and the service time per visit for device i.

Note that each D_i can be regarded as consisting of two components: D_{0i}, which is the total service time per job at server i that occurs outside critical section, and D_{1i}, which is the total service time per job at server i that occurs inside the critical section. The aggregate server technique is based on the idea of adding an additional (aggregate) server to the network to represent the serialized processing in the critical section, and then regarding the expanded $K+1$ server network as having a conventional product form solution.

Let Y_{K+1} be the service time per job at the aggregate server; in other words, Y_{K+1} represents the time spent per job in the critical section. As an initial approximation, Y_{K+1} can be set equal to the sum $D_{11}+D_{12}+\cdots+D_{1K}$ since the processing in the critical section is serialized. Also, let the service time per job (excluding critical section processing) at server i be $Y_i=D_{0i}$. Thus, in its simplest form, the aggregate server representation of the original problem is a product form network with $K+1$ servers where the service times per job are $Y_1, Y_2, \ldots, Y_{K+1}$. The queueing delay at the aggregate server represents the queueing delay for the critical section, and the overall throughput of the network represents the overall throughput of the original model.

In practice, the simple aggregate server model described above is too crude to yield satisfactory results. There are two main sources of error in this model:

(A) The service times per job at the original K servers (Y_i at server i) are represented simply as $D_{01}, D_{02}, \ldots, D_{0K}$. This ignores the fact that processing within the critical section can place an additional load on the original servers, and thus degrades their performance. To represent this degradation, assume instead the service times per job at the original servers are represented as $Y_1, Y_2, \ldots, Y_K$ where

$$Y_i = D_{0i} / H_{0i} \qquad i=1,2,\ldots,K \tag{1}$$

and $0<H_{0i}\leq 1$, $i=1,2,\ldots K$. One of the intermediate objectives of the aggregate server technique is to determine the values of *service time adjustment factors*, H_{0i}, that represent the service time elongation (or server degradation) properly.

(B) The service time per job at the aggregate server representing the critical section is represented as $Y_{K+1} = D_{11}+D_{12}+\ldots+D_{1K}$. Once again, this ignores the fact that processing outside the critical section can place an additional load on the servers that perform critical section processing, thus increasing the effective value of service time required at server i while in the critical section to $Y_{1i} = D_{1i} / H_{1i}$. To represent this effect, assume the service time per job at the aggregate server is

$$Y_{K+1} = \sum_{i=1}^{K} Y_{1i} = \sum_{i=1}^{K} D_{1i} / H_{1i} \tag{2}$$

and $0<H_{1i}\leq 1$, $i=1,2,\ldots,K$. Determining the values of the H_{1i} is another intermediate objective of the aggregate server technique.

The overall flow of the aggregate server technique can be now be specified. Note that steps 3 and 4 below account for points A and B respectively.

1. Consider a queueing network model that would satisfy product form conditions, except for the presence of 1 serialized phase (e.g., a critical section).

2. Add 1 aggregate server to the network for the serialized phase.

3. Using equation (1) and an initial approximation of H_{0i}, $i=1,\cdots,K$, compute the service time requirement at the original servers, accounting for contention from serialized processing.

4. Using equation (2) and an initial approximation of H_{1i}, $i=1,\cdots,K$, compute the service time at the aggregate server, accounting for contention at the original servers by non-serialized processing.

5. After computing the service times (as in steps 3 and 4), solve the network containing the original servers and the aggregate server using conventional product form techniques.

6. Compute new approximation to H_{zi}'s ($z=0,1$) from the solution obtained above. If there is no significant change in H_{zi}'s, we are done, otherwise, return to step 3 with new H_{zi}'s replacing old H_{zi}'s.

The six steps identified above were first proposed, in a different form, by Buzen, Liu and Shum in their original paper on aggregate servers [8]. This paper presents a new algorithm for implementing steps 3 and 4.

In our method, we first note that when there are n customers at a device ($n \neq 0$), on the average, each customer will be served at $1/n$ of the nominal rate. Using this observation and the state probabilities, we determine the effective rate at which a device serves nonserialized and serialized customers (i.e., we compute the service time adjustment factors, H_{zi}'s, where z is a serialized phase index). "Correct" values of the service time adjustment factors are computed iteratively.

4. Development of the Aggregate Server Method Delays

In this section, we will discuss development of the aggregate server method in an informal and intuitive way. A more formal and rigorous development of the aggregate server method in the framework of meta-modeling can be found in Buzen and Agrawal

[6].

The previous section discussed the aggregate server technique for a single serialization phase. Generalization to Z serialization phases is straightforward. Simply add Z aggregate servers to the network, one for each serialization phase. Figure 1 specifies the algorithm. Our notation has been summarized in Table 1.

Figure 1: **Aggregate Server Algorithm**

0: Initialize

initial service time requirements

$$D_{zi}, \quad z=0,1,\ldots,Z,\ i=1,2,\ldots,K$$

initial service time adjustment factors

$$H_{zi}=1.0, \quad z=0,1,\ldots,Z,\ i=1,2,\ldots,K$$

1: Compute stretched out service times

$$Y_{zi} = \frac{D_{zi}}{H_{zi}} \quad \begin{matrix} z=0,1,\ldots,Z \\ i=1,2,\ldots,K \end{matrix}$$

For non-serialized servers

$$Y_i = Y_{0i} \quad i=1,2,\ldots,K$$

For serialized servers

$$Y_{K+z} = \sum_{i=1}^{K} Y_{zi} \quad z=0,1,\ldots,Z$$

Fraction of time spent at a server while being processed in a phase

$$F_{zi}=\begin{cases} & z=0 \\ \dfrac{1}{Y_{K+z}} & \\ \dfrac{Y_{zi}}{Y_{K+z}} & z=1,\ldots,Z \end{cases}$$

2: Solve following queueing network under product form assumption

$$Y_1, Y_2, \cdots, Y_K, Y_{K+1}, \cdots, Y_{K+Z}$$

3: Compute new service adjustment factors, H_{zi}^{new} from old H_{zi} and F_{ji}.

4: If there is no significant change in H_{zi}'s, STOP,
otherwise, return to step 1 with new H_{zi}'s replacing old H_{zi}'s.

The next step is to develop a procedure for computing "correct" values of the service time adjustment factors, the H_{zi}'s. We will first show how one can obtain service time adjustment factors, given the exact solution of the network. Then, we will show

how to approximate these service time adjustment factors using a solution of the queueing network solved under product form assumption (steps 2 and 3 in figure 1).

4.1. Computation of H_{zi}'s from an Exact Solution

Consider server i which is visited during serialized and non-serialized phases of processing. This leads to contention among customers in different phases of processing for service at the server. Let n_{zi} ($i=1,2,...,K$ and $z=0,1,...,Z$) be the number of customers at device i that are in the serialized phase z. The phase 0 is the index of the non-serialized processing phase. K is the number of devices and Z is the number of serialized phases. Note that at any given instant only one customer may be actively receiving service in a serialized phase. Other customers wishing to enter the serialized phase are blocked, and await their turn. Thus, number of customers in serialized phase z is 0 or 1. Therefore, if n_{zi} represents the number of serialized phase z customers at server i, n_{zi} is either 0 or 1, $z=1,2,....,Z$, $i=1,\cdots,K$. Also note that at any given time, if there are k, $0 \le k \le Z$, serialized customers at the server i, there cannot be more than $N-k$ non-serialized customers at this server since N is the number of custmers in the network.

Assuming a processor sharing scheduling at the server, at any instant all customers present receive service at an equal rate. Thus, if there are c ($c=n_{0i}$) non-serialized customers and k ($k=\sum_{q=1}^{q=Z} n_{qi}$) serialized customers present at the server i, a customer will receive service at a rate equal to $1/(c+k)$ times the nominal rate of the server. This implies that $c/(c+k)$ is the fraction of the server capacity that is provided to the set of c non-serialized customers. It also means that $1/(c+k)$ is the fraction of the server capacity that will be provided to each serialized phase customer present at the server at that time.

Contention can thus be modeled as the allocation of a fraction of server capacity to customers in a given phase. The reciprocal of this fraction is the instantaneous

Table 1: Notation

N	total number of customers in the network (i.e., MPL)
K	number of servers in the network
Z	number of serialized phases
i	index of servers under consideration $i=1,\ldots,K$ original devices $i=K+1,\ldots,K+Z$ aggregate serialized servers
c	non-serialized phase customer index
z	phase index $z=0$ nonserialized phase $z=1,\ldots,Z$ serialized phase
Z_i	number of serialized phases in which a customer may visit device i, $i=1,\ldots,K$
n_{zi}	number of customers at device i while visiting phase z. (n_{0i} is the number of nonserialized customers receiving service at device i, and if $z>0$, $n_{zi}\in\{0,1\}$)
m_z	number of customers inside serialized phase z, $z\geq 1$, i.e., $m_z=\sum_{i=1}^{K} n_{zi}$. $m_z\in\{0,1\}$.
r_z	number of customers waiting for entry into the serialized phase z, $z\geq 1$.
n_z	number of customers at serialized phase z. $n_z=r_z+m_z$.
D_{zi}	total service time requirement (demand) at server i while in phase z
H_{zi}	service time adjustment factor for a phase z customer at device i
Y_{zi}	stretched out service time requirement at server i for phase z, i.e., service time modified to reflect the effect of contention due to customers in other phases $Y_{zi}=D_{zi}/H_{zi}$
Y_i	service time requirement at server i in the aggregate server network: $$Y_i=\begin{cases} Y_{0i} & i=1,\ldots,K \\ \sum_{t=1}^{K} Y_{i-K,t} & i=K+1,\ldots,K+Z \end{cases}$$
F_{zi}	fraction of residency time of phase z (i.e., time spent in phase z), $z\geq 1$ spent at device i, ($i=1,\ldots,K$) $$F_{zi}=\begin{cases} 0 & z=0 \\ \dfrac{Y_{zi}}{Y_{z+K}} & z=1,\ldots,Z \end{cases}$$
$S(Z,k\mid v_0,v_1,\cdots,v_l)$	set of partially ordered $Z+1$-tuples such that first $l+1$ elements of the tuples are $v_0,v_1,\cdots,v_l$, respectively. I.e., $$\{(t_0,t_1,\cdots,t_Z)\mid t_r=v_r\ r=0,\ldots,l;\ 0\leq t_r\leq Z\ r=l+1,\ldots,Z;$$ $$t_r\neq t_s \text{ if } r\neq s;\ t_r<t_s \text{ if } l<r<s\leq Z\ \}$$

value of the service time elongation factor. The fraction itself is the instantaneous value of the service time adjustment factor, H_{zi}. We can obtain the average value of the H_{zi}'s by averaging over those periods of times when a phase z customer is present at the server i.

Thus, for phase 0, i.e., the non-serialized processing phase, the service time adjustment factor at server i is

$$H_{0i}=\frac{\sum_{c=1}^{N}\sum_{k=0}^{min(Z,N-c)}\frac{c}{c+k}p\left(n_{0i}=c\wedge\sum_{q=1}^{Z}n_{qi}=k\right)}{p\left(n_{0i}\geq 1\right)} \tag{3}$$

For serialized phase z, $(z\geq 1)$ at server i, the service time adjustment factor is:

$$H_{zi}=\frac{\sum_{k=1}^{min(Z,N)}\sum_{c=0}^{N-k}\frac{1}{c+k}p\left(n_{zi}=1\wedge n_{0i}=c\wedge\sum_{q=1}^{Z}n_{qi}=k\right)}{p\left(n_{zi}\geq 1\right)} \tag{4}$$

Note that the above expressions have been written in terms of the exact probabilities, without approximations. We expand these expressions further in Appendix A.

4.2. Approximating the H_{zi}'s

We will now outline our method of computing approximations to H_{zi}'s from the solution of the product form model, solved during the iteration of the Figure 1.

When expressions for service time adjustment factors, H_{zi} (equations 3 and 4), are reduced to a more amenable form, as shown in equation A-9 and A-10 (Appendix A), we need $p(n_{0i}\geq c\bigwedge_{q\in\tau_{1k}}n_{qi}=1)$, $\tau_{1k}=\{t_1,t_2,\ldots,t_k\}$, the probability that among the customers present at the server i, c are nonserialized, and the other k customers are in $t_1,t_2,\cdots,t_k$ serialized phases of processing. $(t_1,t_2,\cdots,t_k,\ (t_z\neq 0)$ are k unique serialized phase indices). We give an approximation for this probability term below.

First consider the probability that a customer in z^{th} serialization phase is at server i, $p\left(n_{zi}=1\right)$. Assuming that a customer spends F_{zi} $(F_{zi}=Y_{zi}/\sum_{i=1}^{K}Y_{zi},\ z\geq 1)$ fraction of the time spent in the serialization phase z at server i, we have, using

$p(a,b)=p(a)p(b\,|\,a)$,

$$\begin{aligned} p(n_{zi}=1) &= p(n_{zi}=1 \wedge n_z \geq 1) \\ &= p(n_{zi}=1 | n_z \geq 1)\, p(n_z \geq 1) \\ &= F_{zi}\, p(n_z \geq 1) \end{aligned}$$

We note here that if we know correct values of F_{zi}, no approximation is introduced above.

Returning to the expression $p(n_{0i} \geq c \bigwedge_{q \in \tau_{1k}} n_{qi}=1)$, $\tau_{lm} = \tau_{l,m} = \{t_l, t_{l+1}, \ldots, t_m\}$, we have,

$$\begin{aligned} p(n_{0i} \geq c \bigwedge_{q \in \tau_{1k}} n_{qi}=1) &= p(n_{t_1 i}=1)\; p(n_{0i} \geq c \bigwedge_{q \in \tau_{2k}} n_{qi}=1 | n_{t_1 i}=1) \\ &= F_{t_1 i}\; p(n_{t_1} \geq 1)\; p(n_{0i} \geq c \bigwedge_{q \in \tau_{2k}} n_{qi}=1 | n_{t_1 i}=1) \\ &\approx F_{t_1 i}\; p(n_{t_1} \geq 1)\; p(n_{0i} \geq c \bigwedge_{q \in \tau_{2k}} n_{qi}=1 | n_{t_1} \geq 1) \qquad (@) \\ &= F_{t_1 i}\; p(n_{0i} \geq c \wedge n_{t_1} \geq 1 \bigwedge_{q \in \tau_{2k}} n_{qi}=1) \\ &\ldots \\ &\approx \Big(\prod_{q \in \tau_{1k}} F_{qi}\Big)\; p(n_{0i} \geq c \bigwedge_{q \in \tau_{1k}} n_q \geq 1) \qquad (5) \end{aligned}$$

In the step marked with an @, the following homogeneity assumption

$$p(n_{0i} \geq c \bigwedge_{q \in \tau_{2k}} n_{qi}=1 | n_{t_1 i}=1) = p(n_{0i} \geq c \bigwedge_{q \in \tau_{2k}} n_{qi}=1 | n_{t_1} \geq 1). \qquad (6)$$

has been used. This is roughly equivalent of saying that when a serialized phase z is busy (i.e., there is a customer in the serialized phase), the long term behavior of customers in other phases is independent of this serialized customer's whereabouts. Equation 5 will be exact if this assumption holds and will be regarded as an approximation otherwise. Using equation 5, approximate service time adjustment factors can be computed as shown in equations B1 and B2 in Appendix B. Equations B3 and B4 (in Appendix B) give the expressions which can be evaluated using the convolution algorithm[5, 3].

5. Validation

The aggregate server approximation presented in this paper was tested by comparing it with exact numerical solutions obtained by a program that solved the "global

(1) Keeping the ratio of load on various devices in each phase same, vary the ratio of total serialized ($\sum_{z=1}^{Z}\sum_{i=1}^{K} D_{zi}$) to total activity ($\sum_{z=0}^{Z}\sum_{i=1}^{K} D_{zi} = \sum_{i=1}^{K} D_i$).

(2) Given the ratio of total serialized to total activity and otherwise balanced system (i.e., the ratio D_{zi}/D_{zl} is fixed for all z), vary the distribution of load among various serialized phases.

(3) In an otherwise balanced system, for given distribution of activity amongst phases, vary the ratio of activity amongst devices, in the same ratio in each phase.

(4) Vary the distribution of load among devices in a phase keeping the distribution among other phases unchanged.

We believe that our approach has enabled us to study the effects of some relevant parameters on the error in a systematic way.

Below we discuss the results of our experiments with a network consisting of two servers, and two serialized phases (two critical sections). Total service time requirement in the serialized phase 1, CS1, equals $\sum_{i=1}^{2} D_{1i}$. The total serialized service requirement CS equals $\sum_{z=1}^{2}\sum_{i=1}^{2} D_{zi}$. The total service time requirement per customer is the sum of its serialized and nonserialized service requirements at each device, and it equals $(\sum_{i=1}^{2} D_i)$. Thus, the ratio CS/TOTAL is the ratio of total serialized service time requirement per customer to the total service requirement, and the ratio CS1/CS is the ratio of total service requirement for serialized phase CS1 to total amount of serialized processing per customer.

In the discussion below, by error we mean relative error in throughput, which is

$$\%\ ERROR = \frac{(\text{approximate throughput - exact throughput})}{\text{exact throughput}} * 100$$

Figure 2 illustrate the effect of customer population on the error. Figure 3 and 4 will examine the effect of varying relative loading between phases (steps 1 and 2 above).

The network examined in figures 2, 3 and 4 is balanced in the sense that within a phase service time requirements at all servers are equal. In figure 5 and 6, we will examine the effect of the relative loading of various devices (steps 3 and 4 above).

Figure 2 shows the relative error in throughput as a function of change in population. Note that the error levels off as customer population increases. The error for the unbalanced specific representative system (to be examined in later in Section 4.2) levels of much earlier than for balanced systems. It, in fact, decreases slightly for the specific case example. We attribute it to the unbalance in the network. We think it is because an unbalanced system reaches saturation earlier than an equivalent balanced system due to the presence of specific bottlenecks which limit the throughput.

Figure 3 displays the relative error for the balanced system examples as a function of the ratio CS/TOTAL. Note that the error peaks about CS/TOTAL $\approx$ 0.5, i.e., when the total serialized service time requirement approximately equal the total nonserialized service requirement. This demonstrates that the aggregate server method is more accurate for unbalanced systems. Fortunately, real systems are rarely balanced.

Figure 4 shows what happens to the relative error as the distribution of load among various serialized phases is changed for the balanced systems under consideration. We note that the error again peaks when load is distributed equally between the two serialized phases - a balance condition. However, the error is much less sensitive to the distribution of activity among serialized phases as compared to the distribution between tte serialized and the nonserialized activity.

In Figure 5, we study the effect of changing the servers' relative speed (server 1's speed/server 2's speed) only. This keeps everything else, except the relative utilizations of the two servers, unchanged. In particular, the network with CS/TOTAL=0.5, CS1/CS=0.5, and with servers' relative speed of 1 (both servers are capable of serving same number of customer-visits per unit time) is the "completely balanced case" considered earlier. We see that the error in throughput decreases dramatically as the

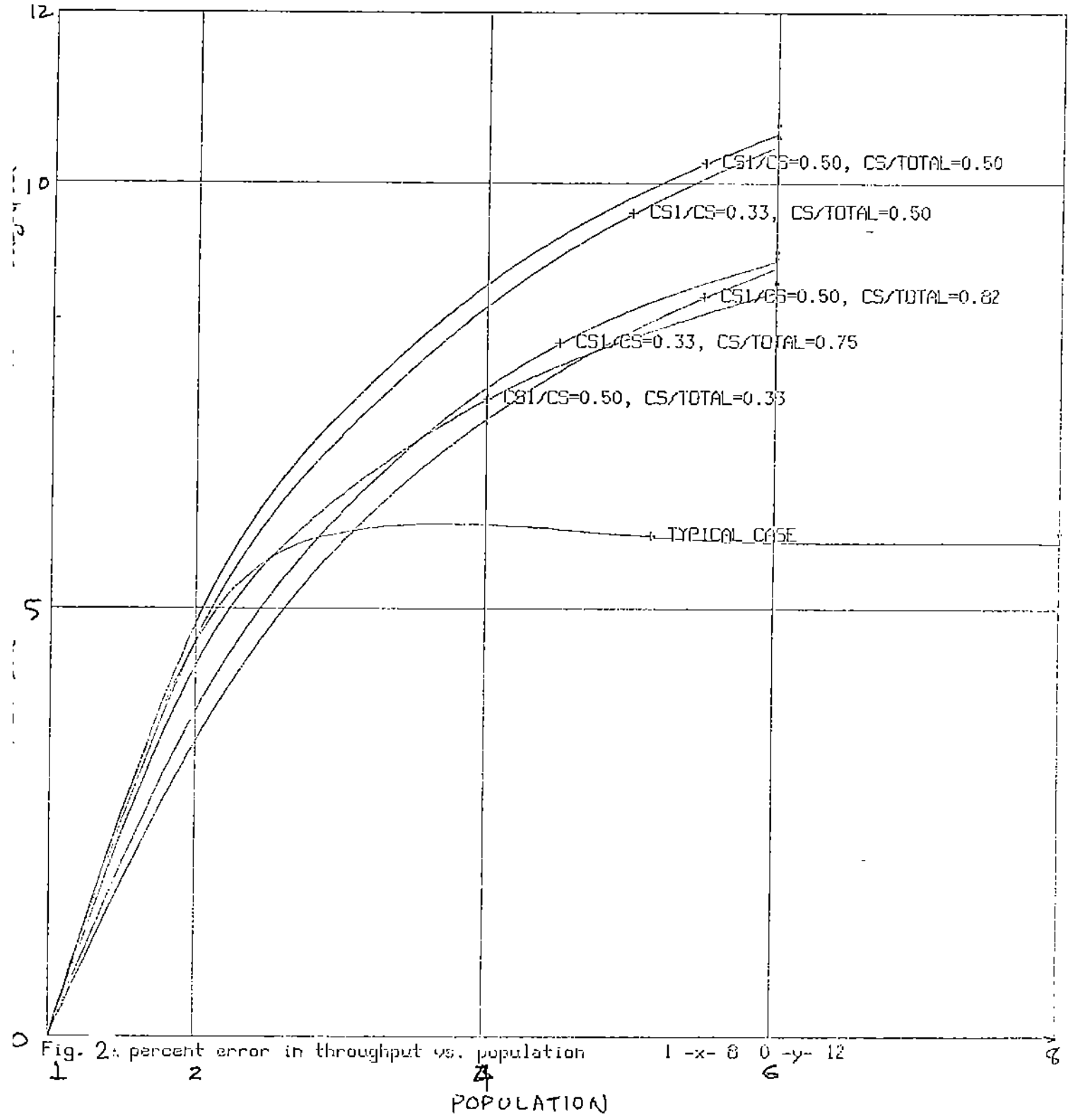


Fig. 2: percent error in throughput vs. population 1 -x- 8 0 -y- 12

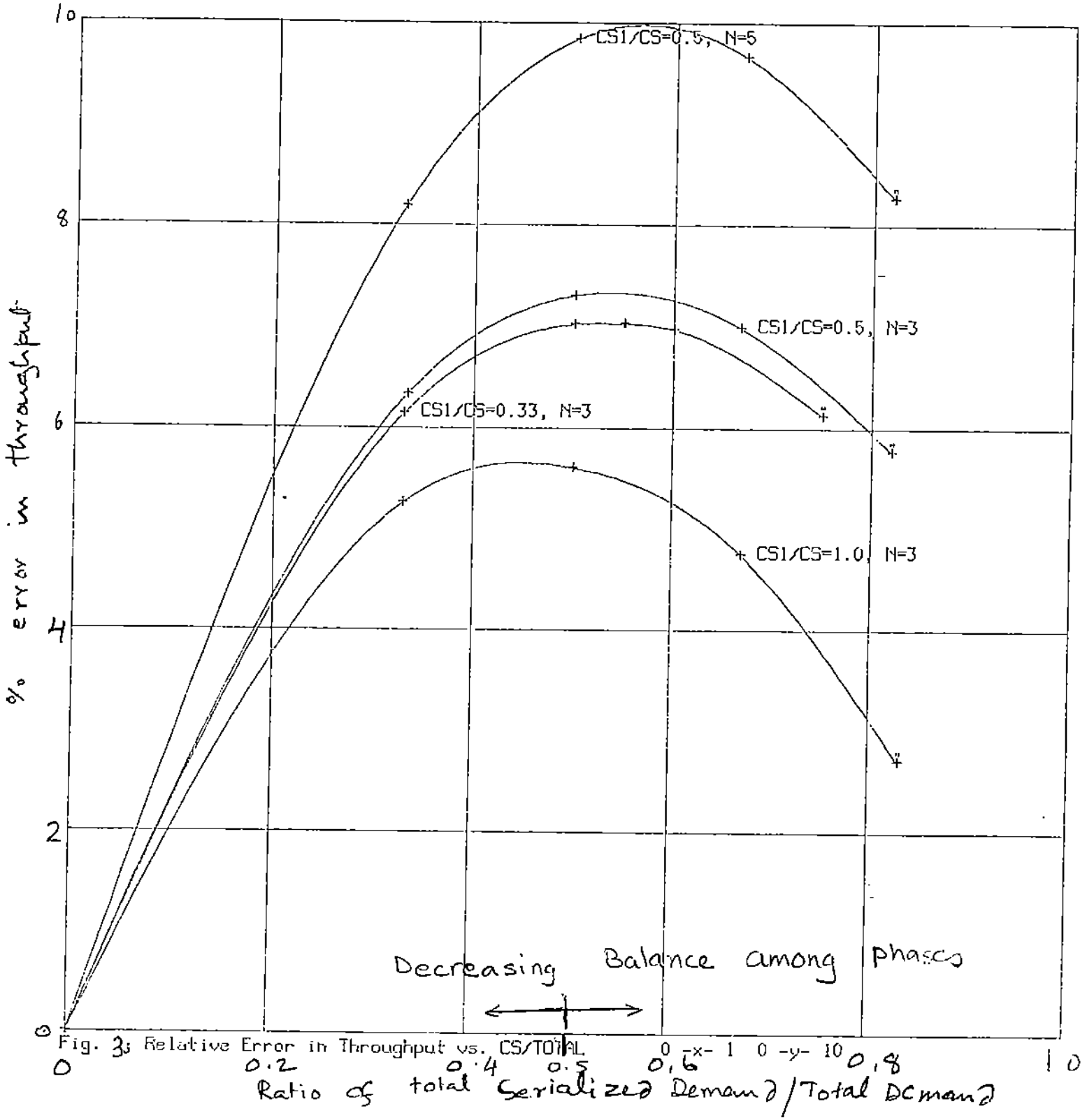


Fig. 3: Relative Error in Throughput vs. CS/TOTAL 0 -x- 1 0 -y- 10

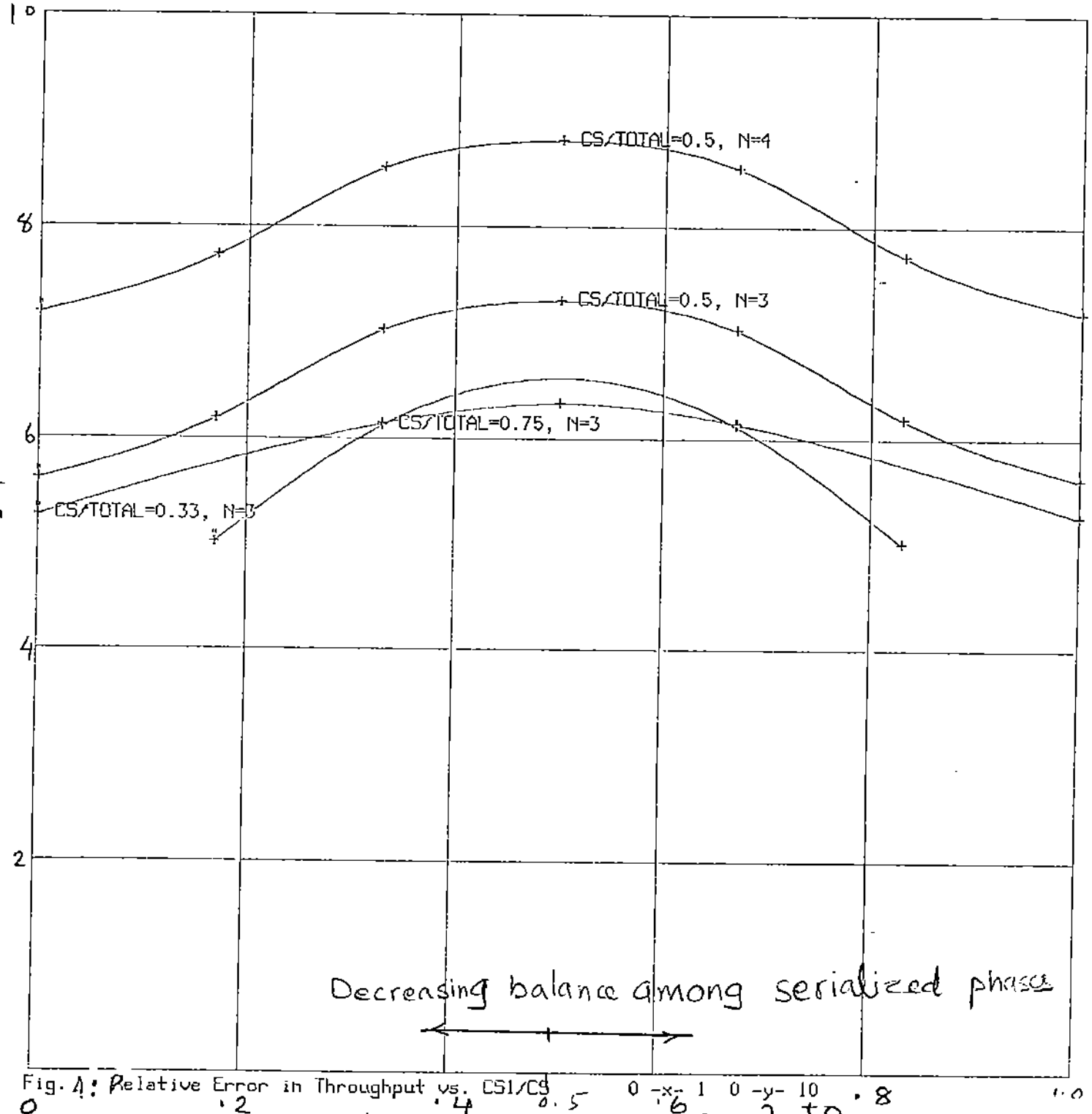


Fig. 4: Relative Error in Throughput vs. CS1/CS 0 -x- 1 0 -y- 10

relative speed (and the utilization) of servers deviates from 1. Equivalently, as the system becomes more unbalanced, the accuracy of the aggregate server technique improves.

Figure 6 illustrates the effect of changing the service requirements at two servers in a serialized phase (phase CS2), keeping the total service requirement in each phase constant. Once again, we note that the error decreases significantly as we depart from the completely balanced configuration.

5.2. A Specific Representative Case

This section presents a specific example that is intended to be representative of the load distribution found in real systems. It consists of 1 CPU, 3 disks and 2 critical sections (serialized phases). The network is depicted in figure 7, which also gives the server speeds, routing probabilities and visit counts. The service time requirements at each device in each phase and total service time requirements at each device are shown in Table 2. These numbers were picked so that the total service time requirement for noncritical section processing, critical sections CS1 and CS2 processing will be approximately 50, 20 and 30 percent of the total service requirement. Other details, such as the percent load for each device in each phase is given in Table 3. These figures should help one place the network in the frame work described in the previous subsection. Note that for this example, CS/TOTAL≈0.5.

Table 4 gives the device and critical section utilizations and relative error in throughput as population is varied from 1 through 8. Notice that the error is zero for population 1. This is expected of this method because it is exact for product form networks and the networks with critical sections do not violate product form when there is only one customer in the network.

We see that the error in throughput first rises with increase in population, then decreases slightly after peaking. We also noticed that the error in individual server's

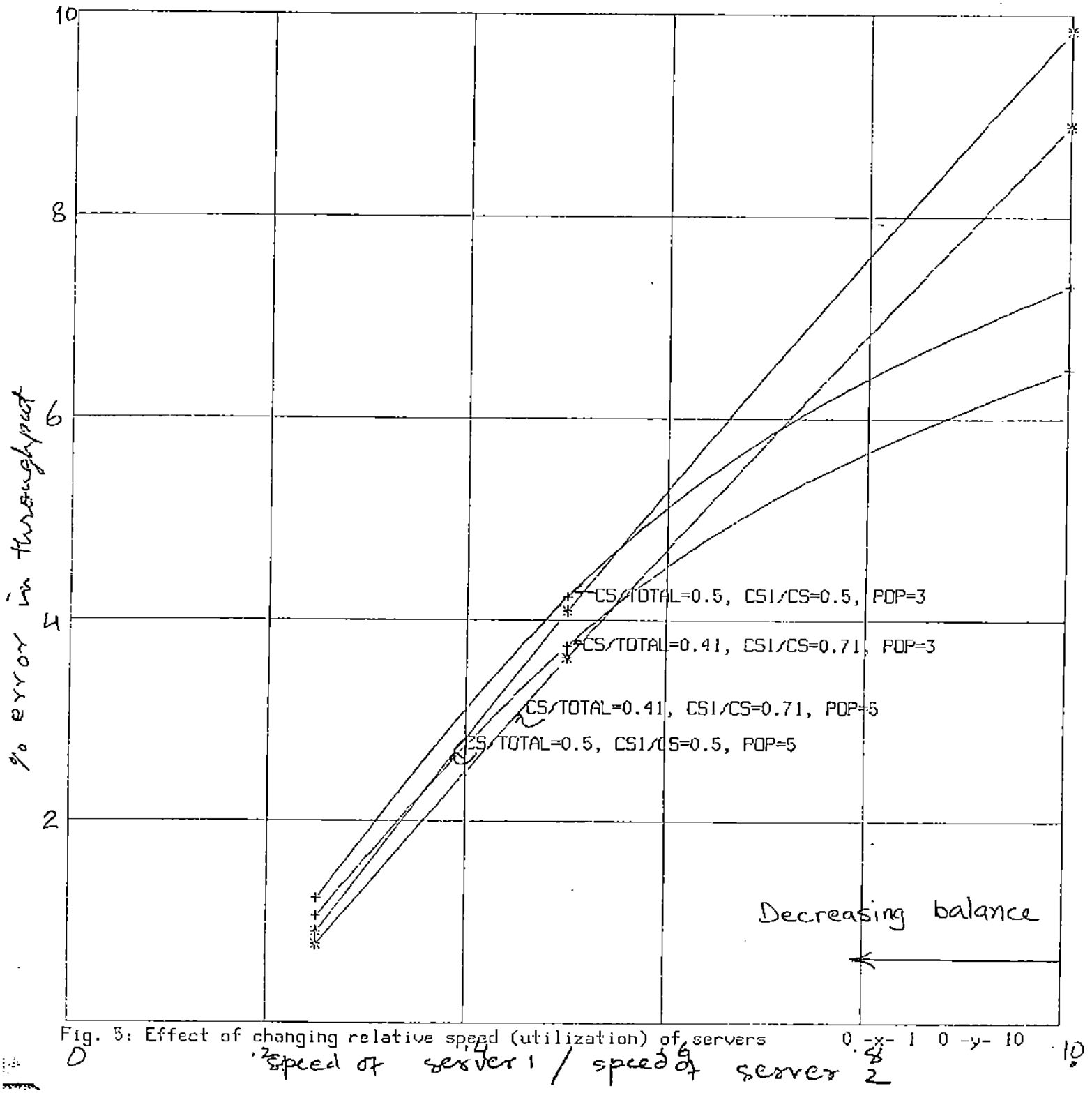


Fig. 5: Effect of changing relative speed (utilization) of servers

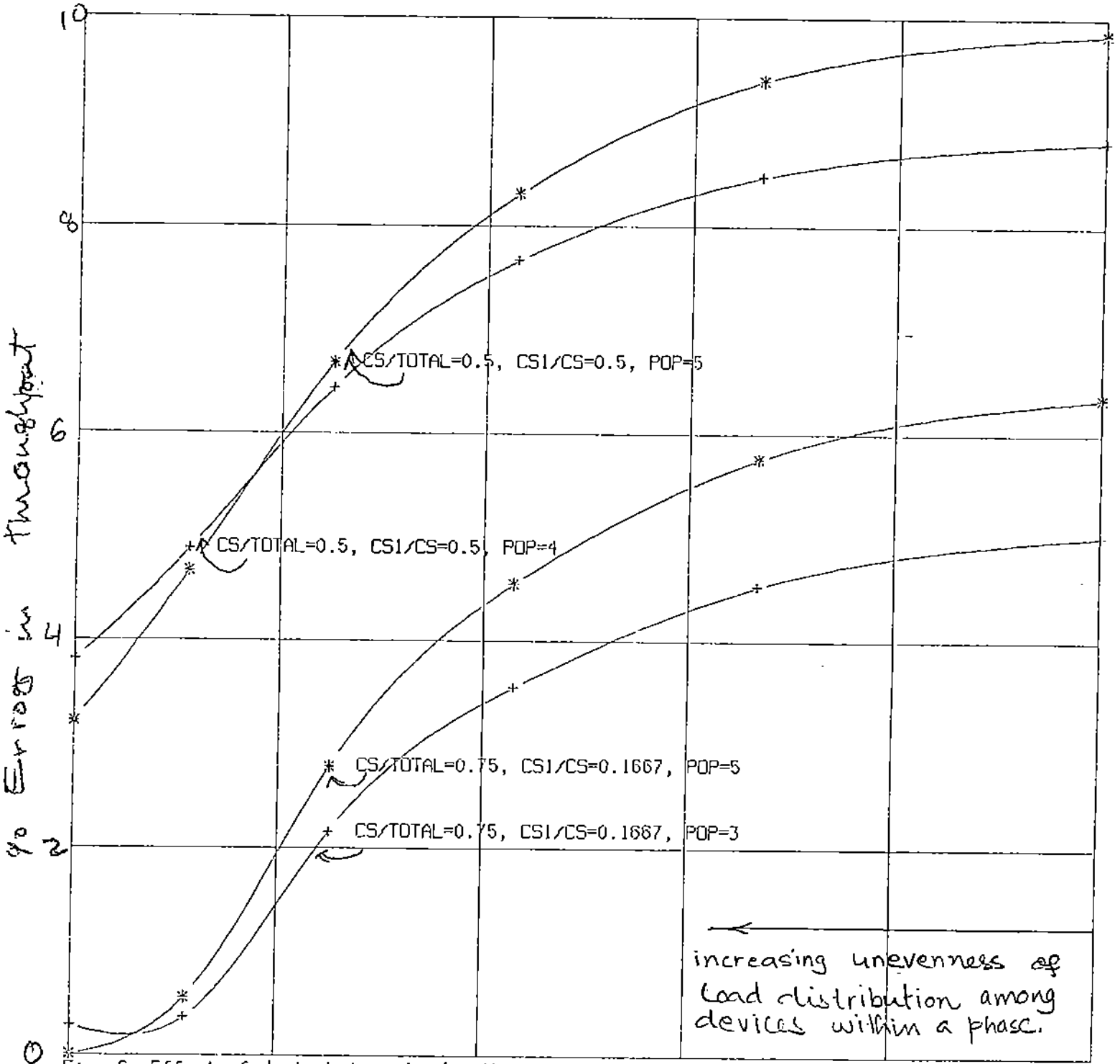


Fig. 6: Effect of distributing load within a given phase (CS2)

Unserialized phase 2 service time requirement at server 1

service time requirement at server 2

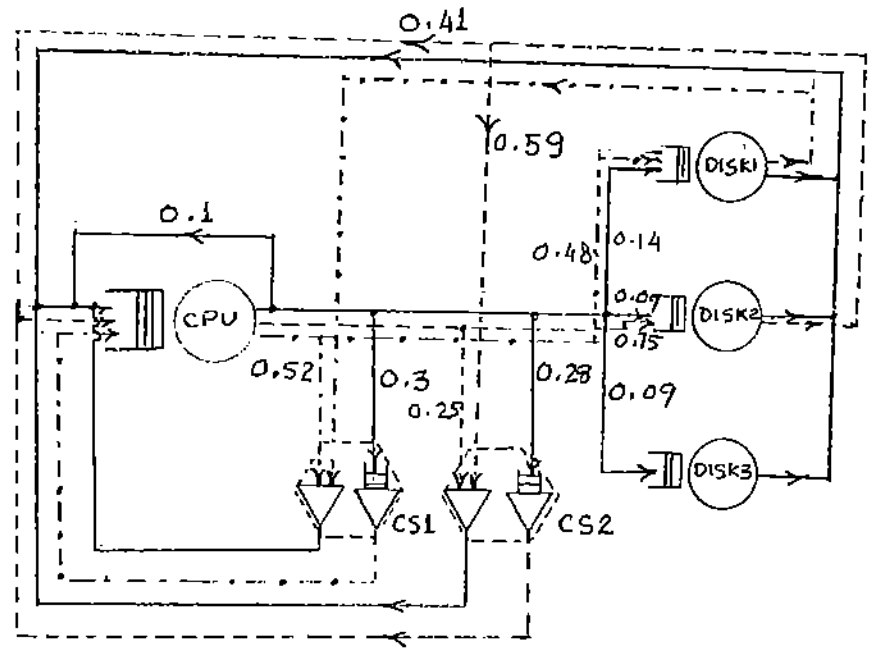
stretched out service times were negative, and were about 1.5-2.0 times, in magnitude, the error in throughput. These lower stretched out service time requirements result in higher throughput, but error in over all throughput is much less than the error in individual stretched out service times. This is due to a "negative feedback effect": increased throughput results in more contention and therefore greater delays, and hence reduces throughput. We also used the stretched out service times obtained from global balance steady state solution to construct a product form aggregate server model and solved it using conventional techniques (i.e., we take the service time adjustment factors obtained from the global balance solution to be the "correct" ones and do not iterate). This model's throughput with 4 customers in the network was 7.1% less than the correct throughput. (Recall, aggregate server method yielded 6.0% higher throughput.) As an aside, with these initial adjustment factors, the iteration converges to the same values of adjustment factors as obtained with initial guess of 1.0.

To demonstrate that proper modeling of serialized phases when serialization delays are significant is essential and that the aggregate server method provides a good approximation, we once again consider the representative case considered above. This time we model this situation using two other techniques: (a) **NOCS**: do not represent serialized phases explicitly, i.e., the service requirement at a device is the sum of serialized and non-serialized service requirement at that device; and (b) **H1**: use the crude model developed in the Section 2, i.e., the service time adjustment factor H is uniformly considered to be 1.0. The results are shown in the Table 5. This data shows that H1 is a rather crude model (in fact, it is worse than ignoring the serialized phases completely). The aggregate server method performs significantly better than the other two methods.

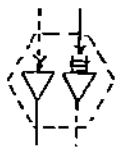
5.3. Source(s) of Error

To investigate the source of error further, it is necessary to look more closely at the process of approximating the effect of contention. So far, for each phase of

Figure 7: Representative Case Description



LEGEND



serializer with
- as entry point - queueing can occur
- as exit point - no queueing occurs

——— NCS : non serialized processing

—·—·— CS1 : critical section (serialization phase) 1

– – – – – CS2 : critical section (serialization phase) 2

SERVER SPECIFICATIONS		
Server	Speed	Service Time per Visit
CPU	50.0	0.02
DISK1	25.0	0.04
DISK2	25.0	0.04
DISK2	25.0	0.04

VISITS			
Server	Processing Phase		
	NCS	CS1	CS2
CPU	10.0	1.00 * 3.0 = 3.00	1.444 * 2.8 = 4.04
DISK1	1.4	0.48 * 3.0 = 1.44	0.00
DISK2	0.9	0.00	1.083 * 2.8 = 3.03
DISK3	0.9	0.00	0.00
CS1	3.0	--	--
CS2	2.8	--	--

Table 2: Service Time Parameters for Typical Case Example

Typical Case Example - Service Time Requirements *				
SERVER	NCS	CS1	CS2	TOTAL
CPU	0.200	0.060	0.081	0.341
DISK1	0.056	0.058	0.000	0.114
DISK2	0.036	0.000	0.121	0.157
DISK3	0.036	0.000	0.000	0.036
TOTAL	0.328	0.118	0.202	0.648

* NCS: nonserialized phase
CS1: serialized phase 1
CS2: serialized phase 3

Table 3: Load Distribution for the Typical Case

distribution of load amongst phases for each device				
SERVER	NCS	CS1	CS2	TOTAL
CPU	58.7	17.6	23.8	100.0
DISK1	49.1	50.9	0.0	100.0
DISK2	22.9	0.0	77.1	100.0
DISK3	100.0	0.0	0.0	100.0
TOTAL	50.6	18.2	31.2	100.0

processing, and each server, we are averaging the effect of contention over the whole period during which there is a customer of that phase present at the server. A moment's reflection will show that as the number of customers present at the server increases, the contention among them increases, and thereby affecting the fraction of server capacity used by customers of individual phases. Thus, from the point of the view of an individual customer, the server behaves in a load dependent fashion. The instantaneous load dependent relative rate of the server i, as perceived by phase z customers is equal to $n_{zi} / \sum_{l=0}^{Z} n_{li}$. Since this representation carries more information, we expect this method to yield better approximation.

To examine this hypothesis, using the "exact" steady state solution, we computed

Table 4: Typical Case System Solution - a Performance Measures' Summary

Typical Case - Device Utilizations and Throughput										
Pop.	No. of ASM Iterations*	Exact Utilizations**						Throughput		% ERROR
		CPU	DISK1	DISK2	DISK3	CS1	CS2	GBS	ASM	
1	1	0.53	0.18	0.24	0.06	0.18	0.31	1.54	1.54	0.00
2	7	0.72	0.24	0.33	0.08	0.32	0.52	2.11	2.21	4.60
3	10	0.81	0.27	0.37	0.09	0.43	0.68	2.37	2.51	5.89
4	14	0.85	0.28	0.39	0.09	0.51	0.79	2.49	2.64	5.99
5	20	0.87	0.29	0.41	0.09	0.57	0.86	2.56	2.71	5.87
6	24***	0.88	0.29	0.41	0.09	0.61	0.91	2.60	2.75	5.78
7#	24***	0.89	0.30	0.41	0.09	0.63	0.95	2.62	2.77	5.75
8	24***	0.90	0.30	0.41	0.09	0.65	0.97	2.63	2.78	5.77

ASM - Aggregate Server Method
GBS - Global Balance Solution
This case was run on a VAX 11/780. Other cases were run on a DEC 20. Different precision of the machines explains the slight deviation from the trend in the error.
* Maximum change between successive $H_{zi} \leq 10^{-6}$.
** From the steady state global balance solution
*** New estimate of H_{zi}'s is average of old value and the one obtained from using the equations 3 & 4. This averaging was done to speed up the convergence by taking advantage of its oscillatory nature.

Table 5: Comparison of Aggregate Server Method With Other Methods

Does it make sense to use Aggregate Server Method?							
Pop.	Throughput				% Error		
	GBS	ASM	H1	NOCS	ASM	H1	NOCS
1	1.54	1.54	1.54	1.54	0.0	0.0	0.0
2	2.11	2.21	2.49	2.25	4.6	18.0	6.6
3	2.37	2.51	3.10	2.60	5.9	30.8	9.7
4	2.49	2.64	3.51	2.77	6.0	41.0	11.2
5	2.56	2.71	3.79	2.85	5.9	48.0	11.3
6	2.60	2.75	3.99	2.89	5.8	53.5	11.2
7	2.62	2.77	4.14	2.92	5.8	58.1	11.4
8	2.63	2.78	4.25	2.92	5.8	61.6	11.0

ASM - Aggregate Server Method
GBS - Global Balance Solution
H1 - Contention among serialized and nonserialized customers is not represented, i.e., $H_{zi}=1.0$.
NOCS- Critical Sections are not explicitly represented

the effective, relative rate of the servers as shown below. Average relative rate of server i when there are n nonserialized customers at device i is:

$$Rate_i(n) = \frac{\sum_{k=0}^{\min(N-n,Z)} \frac{n}{n+k} p(n_{0i}=n \wedge \sum_{q=1}^{C} n_{qi}=k)}{p(n_{0i}=n)} \quad (7)$$

Similarly, average relative rate of the aggregated serialized phase server $z+K$, $z \geq 1$ can be given as

$$Rate_{K+z}(n) = \frac{\left\{\sum_{i=1}^{K} \sum_{k=1}^{\min(Z,N-n+1)N-k} \sum_{c=1} \frac{1}{c+k} p(n_z=n \wedge n_{0i}=c \wedge n_{zi}=1 \wedge \sum_{l=1}^{Z} n_{li}=k)\mu_i q_{zi0}\right\} \sum_{i=1}^{K} D_{zi}}{p(n_z=n)} \quad (8)$$

In equation 8 above, μ_i is the processing rate of the server i in terms of number of customer-visits per second, and q_{zi0} is the probability that a serialized phase z customer will depart from the serialized phase (for that visit only) after receiving service at server i. Note that $\sum_{i=1}^{K} D_{zi}/V_z$ is the nominal service time requirement per visit to the serialized phase z. That is, $1/(\sum_{i=1}^{K} D_{zi}/V_z)$ is the nominal processing rate of the serialized phase in terms of customer-visits per second.

Our test case is the representative system considered above with 4 customers in the network. (Note that using the aggregate server technique, maximum relative error occurs for this network population.) Parameters of this model are shown in Table 6.

Table 6: An aggregate server model with load dependent service rates obtained from exact solution

Number of Customers in the Network =					4
SERVER	SERVICE TIME	RATE(n)			
		1	2	3	4
CPU	0.200	0.602	0.709	0.822	1.000
DISK1	0.056	0.878	0.905	0.939	1.000
DISK2	0.036	0.764	0.825	0.890	1.000
DISK3	0.036	1.000	1.000	1.000	1.000
CS1	0.118	0.522	0.612	0.751	0.989
CS2	0.202	0.508	0.623	0.775	1.022

As before the refined approximation is assumed to satisfy the conditions for product form solution. The refined model proved to be remarkably accurate: approximate throughput was within 0.5% of actual throughput, and waiting times at individual

servers were within 1.5%. This small residual error, we believe, results from the assumption of product form - we have considered the rates to be a function of local queue lengths only.

This refined analysis illustrates the power of the aggregate server method. The use of mean stretched out service times is for computational ease only. If we use load dependent rates, significantly better results can be obtained. It is possible to develop formulas to compute these load dependent rates in aggregate server setting. However, the cost of computation (especially that of solving the resulting model which will consist solely of load dependent servers) may be prohibitive. Convergence may also be a problem. In most cases the additional computational effort may not be justified by the resulting gain in accuracy.

To summarize, our empirical investigations show the aggregate server method to be reasonably accurate. The errorr is dependent on the customer population and distribution of load amongst various phases and devices. Main source of error appears to be the use of mean stretched out service times, rather than load dependent service rates. If accuracy is of utmost importance, one may revise the aggregate server method to incorporate load dependent servers.

6. Computational Complexity and Convergence

As for any iterative method in queueing network modeling, it is very difficult to prove convergence, and even more difficult to say anything about the point of convergence. However, in all but one of the cases that we solved, the algorithm converged (see next paragraph). The convergence was oscillatory in nature, i.e, the service time adjustment factors oscillated back and forth around the value to which they finally converged.

However, in one case, we noticed oscillatory divergence. Iteration did converge when we took the average of old H_{zi}'s (input for iteration) and new H_{zi}'s (algorithm

output) as the H_{zi}'s for next iteration. Divergence can be detected by monitoring maximum change in the H_{zi}'s during successive iterations. If this increases, then the iteration is likely to diverge. As already noted, averaging of successive values of the H_{zi}'s appears to be effective in eliminating the divergence.

We also cannot say anything about number of iterations required except that it is a complicated function of network population, number of servers and serialized phases, and distribution of load. However, empirically we have noted that the iteration converges linearly. Thus, we can apply some well-known techniques such as Aitken's Δ^2 process [9] to speed up the convergence.

Computational requirements for the method using the convolution algorithm to compute the service time adjustment factors are $O(KZ_{avg}^3+5NK)$ computations per iteration after the model has been solved. Z_{avg} is the mean number of phases in which a customer may visit a given device, i.e, mean number of phases sharing a device.

7. Extensions

We have presented a technique for modeling of serialization delays in queueing network models of computer systems. We discussed and validated the technique assuming a processor sharing scheduling discipline and load independent servers. Since, in a product form network PS, FCFS, and LCFS scheduling disciplines yield the same steady state solution, we believe that same relations will hold for them. For IS discipline (infinite servers) the H_{zi} are always equal to 1 for all phases.

Variable rate servers do not pose any problem. Expressions for H_{zi}'s can be derived by appropriately modifying equations 3 and 4 in section 4. Of course, computation of H_{zi} will be more expensive in this case.

Extension to multi-class networks in which various classes do not share a serialized phase is straightforward. The service time adjustment factors can be computed relatively efficiently, by precomputing a table of numbers, which is independent of ser-

vice time parameters of the network.

Though the underlying principle remains unchanged, extension to multiclass networks with shared serialized phases is computationally difficult.

8. Conclusion

We have presented the aggregate server method for modeling serialization delays in computer systems. Examples of serialization delays include waiting for entry into a critical section or a lock, executing non-reentrant subroutines, etc. These delays are normally encountered in operating system calls, database enquiry and updates, process synchronization etc. The aggregate server method involves including an additional server for each source of serialization delay. Service time requirements at these serialization delay servers and physical servers are suitably stretched out to account for the contention for service at a server amongst jobs inside and outside some serialization phase. Stretchout factors are iteratively obtained. For typical systems, the accuracy appears to be satisfactory. We found that the accuracy of the method depends heavily on the evenness of distribution of load amongst various phases of processing. Uniform distribution of load results in poor accuracy. Fortunately, most real systems are unbalanced.

In this paper, we have provided an intuitive explanation of the rationale behind the aggregate server method. Rigorous development of the method in the framework of meta-modeling is provided by Buzen and Agrawal [6].

We have developed the method for single class closed model consisting solely of load independent, PS scheduled single servers. It can be readily extended to include load dependent rates, other scheduling disciplines, as well as, multi-class closed networks.

Acknowledgement

We wish to thank Peter Denning, Andre Bondi and Teemu Kerola for their constructive comments which have improved this paper considerably.

APPENDIX A

Service Time Adjustment Factors

In Section 3.1 (equations 3 and 4), we saw that service time adjustment factors, H_{zi}'s, for $z=1,2,\ldots,Z$ and $i=1,2,\ldots,K$, are given by following expressions:

$$H_{0i}=\frac{\sum_{c=1}^{N}\sum_{k=0}^{\min(Z,N-c)}\frac{c}{c+k}p\left(n_{0i}=c\wedge\sum_{q=1}^{Z}n_{qi}=k\right)}{p\left(n_{0i}\geq 1\right)} \tag{A1}$$

and

$$H_{zi}=\frac{\sum_{k=1}^{\min(Z,N)}\sum_{c=0}^{N-k}\frac{1}{c+k}p\left(n_{zi}=1\wedge n_{0i}=c\wedge\sum_{q=1}^{Z}n_{qi}=k\right)}{p\left(n_{zi}\geq 1\right)}. \tag{A2}$$

In this appendix, we will develop a relation for an H_{zi}, $z\geq 1$. Development of the relation for H_{0i}'s follows similar line of argument and will not be provided. Our notation was summarized in Table 1.

In the expression for H_{zi}, the basic term is $p\left(n_{zi}=1\wedge n_{0i}=c\wedge\sum_{q=1}^{Z}n_{qi}=k\right)$, which is the probability that there are c non-serialized and k serialized customers present at device i, and one of serialized customers present is in serialized phase z. Since $n_{qi}\in\{0,1\}, q=1,2,\ldots,Z$, we see that this probability is simply the sum of all probabilities

$p(n_{0i}=c\wedge n_{zi}=1\wedge$ some other $k-1$ serialized customers are at device $i\wedge$ remaining $Z-k$ serialized customers (if any) are not at device i).

Let $S(Z,k,|0,z)$ be the set of all tuples $(t_0,t_1,\ldots,t_Z)$ such that $t_0=0$, $t_1=z$, and $t_2,\ldots,t_k$ are other $k-1$ serialization phase indices out of $\{1,2,\ldots,Z\}-\{z\}$ and remaining $t_{k+1},\ldots,t_Z$ are other $Z-k$ indices. We will also denote the set $\{t_l,\ldots,t_m\}$ by τ_{lm} or $\tau_{l,m}$. Then, we can express this relationship very precisely as

$$p\left(n_{0i}=c \wedge \sum_{q=1}^{Z} n_{qi}=k \wedge n_{zi}=1\right)$$

$$= \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2k}} n_{qi}=1 \bigwedge_{q\in\tau_{k+1,Z}} n_{qi}=0\right) \qquad \text{(A3)}$$

We also note that given $(t_0,\ldots,t_Z)$,

$$p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1 \bigwedge_{q\in\tau_{k+1,Z}} n_{qi}=0\right) =$$

$$p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1\right)$$

$$- \sum_{m=k+1}^{Z} \sum_{\substack{(u_0,\ldots,u_Z)\in \\ S(Z,m|0,z,t_2,\cdots,t_k)}} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{\{u_2,\cdots,u_m\}} n_{qi}=1 \bigwedge_{\{u_{m+1},\ldots,u_Z\}} n_{qi}=0\right) \qquad \text{(A4)}$$

where $(u_0,\ldots,u_Z)\in S(Z,m|t_0,t_1,t_2,\cdots,t_k)$ satisfies $u_l=t_l, l=0,1,\ldots,k$ and $u_l\in\tau_{k+1,Z}$, $l=k+1,\ldots,Z$, and that $u_l\neq u_m, l\neq m$. Thus, using A3, we have,

$$H_{zi} = \frac{\sum_{k=1}^{\min(Z,N)} \sum_{c=0}^{N-k} \frac{1}{c+k} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1 \bigwedge_{q\in\tau_{k+1,Z}} n_{qi}=0\right)}{p\left(n_{zi}\geq 1\right)} \qquad \text{(A5)}$$

Now, the probability terms in the double summation in the right hand side of equation A4 above also occur in equation A5 for larger values of k. We will use this observation to obtain following lemma.

Lemma A1: Using equation A4 to expand probability terms in the innermost sum in equation A5 for $k=1,2,\ldots,a$ (in that order), we get,

$$H_{zi} = \left\{ \sum_{k=1}^{a} \sum_{c=0}^{N-k} \frac{(-1)^{k-1}(k-1)!}{\prod_{l=1}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1\right) \right.$$

$$\left. + \sum_{m=a+1}^{\min(Z,N)} \sum_{c=0}^{N-m} \frac{\prod_{l=1}^{a}(l-m)}{(c+m)\prod_{l=1}^{a}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,m|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,m}} n_{qi}=1 \bigwedge_{q\in\tau_{m+1,Z}} n_{qi}=0\right) \right\}$$

$$\Big/\; p\left(n_{zi}\geq 1\right)$$

Proof:

This lemma can be proved by induction, beginning at a=1. It involves little combinatorial algebra. We will merely outline the procedure and leave the details to the reader. We first note that

$$\left|\left|S(Z,m|t_0,t_1,\cdots,t_k)\right|\right| = \binom{Z-k}{m-k} \tag{A6}$$

In particular, for $m=k$,

$$\left|\left|S(Z,k|0,z)\right|\right| = \binom{Z-1}{k-1} \tag{A7}$$

Equation A6 implies that expansion of probability term on the left hand side of A4 yields $\binom{Z-k}{m-k}$ probability terms on the right hand side for given m and k, $m>k$. Let us now focus our attention on the innermost sum in the equation A5 for some k. When the probability terms are expanded, due to symmetry, we get

$$\sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1 \bigwedge_{q\in\tau_{k+1,Z}} n_{qi}=0\right)$$

$$= \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1\right) -$$

$$\sum_{m=k+1}^{Z} \Theta_{mk} \sum_{(t_0,\ldots,t_Z)\in S(Z,m|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2m}} n_{qi}=1 \bigwedge_{q\in\tau_{m+1,Z}} n_{qi}=0\right)$$

where

$$\Theta_{mk} = \frac{\binom{Z-k}{m-k}\binom{Z-1}{k-1}}{\binom{Z-1}{m-1}} = \binom{m-1}{k-1}.$$

Completion of this expansion yields, for $z=1,2,\ldots,Z$,

$$H_{zi} = \frac{\sum_{k=1}^{\min(N,Z)} \sum_{c=0}^{N-k} \frac{(-1)^{k-1}(k-1)!}{\prod_{l=1}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}=c \wedge n_{zi}=1 \bigwedge_{q\in\tau_{2,k}} n_{qi}=1\right)}{p(n_{zi}\geq 1)} \tag{A8}$$

Noting that

$$p\left(n_{0i}=c \bigwedge_{q\in\tau_{lm}} n_{qi}=1\right) = p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{lm}} n_{qi}=1\right) - p\left(n_{0i}\geq c+1 \bigwedge_{q\in\tau_{lm}} n_{qi}=1\right)$$

and physical constraint that number of customers in the network is N, we have the following theorem.

Theorem A1: The service adjustment factor for serialization phase z, $z=1,\ldots,Z$ at server i, $i=1,2,\ldots,K$ is:

$$H_{zi} = \sum_{k=1}^{\min(N,Z)} \left\{ \frac{(-1)^{k-1}}{k} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}\geq 0 \bigwedge_{q\in\tau_{1,k}} n_{qi}=1\right) + \sum_{c=1}^{N-k} \frac{(-1)^k\ k!}{\prod_{l=0}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{1,k}} n_{qi}=1\right) \right\} \Big/ p\left(n_{zi}\geq 1\right) \tag{A9}$$

Similarly, for the non-serialized phase, we can have,

Theorem A2: The service adjustment factor for non-serialized activity at server i, $i=1,2,\ldots,K$ is:

$$H_{0i} = \sum_{k=0}^{\min(N-1,Z)} \left\{ (-1)^k \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} p\left(n_{0i}\geq 1 \bigwedge_{q\in\tau_{1,k}} n_{qi}=1\right) + \sum_{c=1}^{N-k} \frac{(-1)^{k+1}\ k\ k!}{\prod_{l=0}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{1,k}} n_{qi}=1\right) \right\} \Big/ p\left(n_{0i}\geq 1\right)$$

or,

$$H_{0i} = \left\{ \sum_{k=0}^{\min(N-1,Z)} (-1)^k \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} p\left(n_{0i}\geq 1 \bigwedge_{q\in\tau_{1,k}} n_{qi}=1\right) + \sum_{c=1}^{N-1} \sum_{k=1}^{\min(N-c,Z)} \frac{(-1)^{k+1}\ k\ k!}{\prod_{l=0}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{1,k}} n_{qi}=1\right) \right\} \Big/ p\left(n_{0i}\geq 1\right) \tag{A10}$$

APPENDIX B

Obtaining Approximate Service Time Adjustment Factors from Product Form Solution of Aggregate Server Model

In order to be able to compute service time adjustment factors from the solution of a product form model, we use the following homogeneity assumption (equation 6):

$$p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{2k}} n_{qi}=1 | n_{l_1 i}=1\right) = p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{2k}} n_{qi}=1 | n_{l_1}\geq 1\right).$$

Thus, using equations 5, A9 and A10, we get, for $z=1,2,\ldots,Z$ and $i=1,2,\ldots,K$,

$$H_{zi} = \sum_{k=1}^{\min(N,Z)} \left\{ \frac{(-1)^{k-1}}{k} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} \left(\prod_{q\in\tau_{2k}} F_{qi} \right) p\left(n_{0i}\geq 0 \bigwedge_{q\in\tau_{1k}} n_q \geq 1\right) + \sum_{c=1}^{N-k} \frac{(-1)^k\, k!}{\prod_{l=0}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} \left(\prod_{q\in\tau_{2k}} F_{qi} \right) p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{1k}} n_q \geq 1\right) \right\} \Big/ p(n_z \geq 1) \quad \text{(B1)}$$

and

$$H_{0i} = \left\{ \sum_{k=0}^{\min(N-1,Z)} (-1)^k \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} \left(\prod_{q\in\tau_{1k}} F_{qi} \right) p\left(n_{0i}\geq 1 \bigwedge_{q\in\tau_{1k}} n_q \geq 1\right) + \sum_{c=1}^{N-1} \sum_{k=1}^{\min(N-c,Z)} \frac{(-1)^{k+1}\, k\, k!}{\prod_{l=0}^{k}(c+l)} \sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} \left(\prod_{q\in\tau_{1k}} F_{qi} \right) p\left(n_{0i}\geq c \bigwedge_{q\in\tau_{1k}} n_q \geq 1\right) \right\} \Big/ p(n_{0i} \geq 1) \quad \text{(B2)}$$

The necessary probabilities required in the expressions above can be using the conventional solution techniques for product form network, e.g. by using the G vector in convolution[3], or by using prime-recursion formulas in the mean value analysis setting[7]. Relations for H_{zi}'s using convolution algorithm are given below.

$$H_{zi} = \sum_{k=1}^{\min(N,Z)} \left\{ (-1)^{k-1} \left(\sum_{(t_0,\ldots,t_Z)\in S(Z,k|0,z)} \prod_{q\in\tau_{2k}} Y_{qi} \right) * \left(\frac{G(N-k)}{k} - k! \sum_{c=1}^{N-k} \frac{Y_{0i}^{c}}{\prod_{l=0}^{k}(c+l)} G(N-c-k) \right) \right\} \Big/ G(N-1) \quad \text{(B3)}$$

and

$$H_{0i} = \left\{ G(N-1) + \sum_{k=1}^{\min(N-1,Z)} (-1)^k \left(\sum_{(t_0,\ldots,t_Z)\in S(Z,k|0)} \prod_{q\in\tau_{1k}} Y_{qi} \right) * \left(G(N-k-1) - \frac{k\, k!}{Y_{0i}} \sum_{c=1}^{N-k} \frac{Y_{0i}^{c}}{\prod_{l=0}^{k}(c+l)} G(N-c-k) \right) \right\} \Big/ G(N-1) \quad \text{(B4)}$$

References

1. *BEST/1 User's Guide*, BGS Systems, Inc., Waltham, MA. (1980).

2. Bard, Y., "A Model of Shared DASD and Multipathing," *Communications of the ACM* **23**(10) pp. 564-583 (1980).

3. Bruell, S.C. and Balbo, G., *Computational Algorithms for Single and Multiple Class Closed Queueing Network Models*, Series on Programming and Operating Systems, Elsevier/North-Holland Publishing Co., New York (1980).

4. Buzen, J.P. et al., "BEST/1 - Design of a Tool for Computer System Capacity Planning," pp. 447-455 in *Proc. 1978 AFIPS National Computer Conference*, AFIPS Press, Montvale, N.J. .

5. Buzen, J. P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the A.C.M.* **16**(9) pp. 527-531 (1973).

6. Buzen, J.P. and Agrawal, S.C., "Meta-Modeling: A Theory of Models and the Modeling Process," *CSD-TR-385*, Purdue University, (To appear).

7. Buzen, J.P. and Denning, P.J., "Measuring and Calculating Queue Length Distributions," *Computer* **13**(4) pp. 33-44 (April 1980).

8. Buzen, J.P., Liu, M.H., and Shum, A.W., "Aggregate Servers and Their Representation in BEST/1," *BGS Systems' Internal Report*, BGS Systems, Inc., (1981).

9. Conte, S.D. and de Boor, Carl, *Elementry Numerical Analysis: An Algorithmic Approach*, McGraw Hill Book Co., New York (1972).

10. Denning, P.J. and Buzen, J.P., "The Operational Analysis of Queueing Network Models," *Computing Surveys* **10**(3) pp. 225-261 (Sept. 1978).

11. Dijkstra, E.W., "The Structure of THE Multiprogramming System," *Comm. ACM* **11**(5) pp. 341-346 (May 1968).

12. Jacobson, P.A. and Lazowska, E.D., "The Method of Surrogate Delays: Simultaneous Resource Possession in Analytical Models of Computer Systems," *Performance Evaluation Review: Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, (September 1981).

13. Kumar, B. and Gonsalves, T.A., "Modelling of Analysis of Distributed Software Systems," *Proceedings of ACM-SIGOPS Seventh Symposium on Operating Systems Principles*, pp. 2-8 (Dec. 1979).

14. Potier, D. and Leblanc, Ph., "Analysis of Locking Policies in Data Base Management Systems." *Communications of the ACM* **23**(10) pp. 584-572 (1980).

15. Smith, C.U. and Browne, J.C., "Aspects of Software Design Analysis: Concurrency and Blocking," *Performance Evaluation Review: Proceedings of Performance '80* 9(2) pp. 245-253 (Summer 1980).

16. Stewart, W.J., "A Comparison of Numerical Techniques in Markov Modeling," *Communications of the ACM* **21**(2) pp. 144-152 (February 1978).