Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

ON DETERMINING THE DISTRIBUTION OF SOFTWARE RESPONSE TIMES

Charles M. Shub
Associate Professor
Computer Science Department
University of Colorado at Colorado Springs
P.O. Box 7150, Colorado Springs CO 80933

1. ABSTRACT

This paper reports on a simulation model for predicting software response time in symmetric distributed computing systems. This model extends previous work by considering a collection of similar processing elements connected together by some interconnection mechanism. The model also extends previous work by considering stimuli that require processing at multiple priority levels.

The paper begins by describing the nature of the problem. In this section, the overall problem is decomposed so that attention may be focused on the software response, the topic of this report. Next a model is formulated. The coding of the model in GPSS/H is described. Finally, sample results are presented.

2. THE PROBLEM

In many transaction based systems, including but not limited to, data base systems, real time systems, there is concern for predicting response time. In this context, response time is defined as the time between when a stimulus is sent from an external device to the time that a response is received by that device. For the purposes of this problem, it is assumed that there is no peripheral I/O other than message communication so response to all stimuli is processing and communication intensive.

A typical scenario proceeds as follows:

1. A user generates a stimulus by depressing a key on a terminal.
2. The terminal generates a message encapsulating the key depression.
3. The message flows over some medium and arrives at a processing element.
4. The message is now ready for processing and exists competing for the processor to receive service.
5. After receiving processing, the message may be forwarded over some medium to receive additional processing from another physical processor. This may happen several times.
6. A response message flows over some medium and arrives at the terminal.
7. The terminal displays the response.

It is furthermore assumed that for any given stimulus there is a single response, and that the displaying of the response by the terminal occurs instantaneously. This definition is compatible with the usual definition of response time as being from the point when the "carriage return" key is depressed until the time when the response message first starts appearing on the screen. Moreover, it is certainly the case that processing may continue after the response point, so there can be overlap between "clean up" processing and what is normally thought of as think time.

From the above description, it is clear that there are several components making up the response delay. These include:

1. The delay between the stimulus and the arrival of the message at a processing element for processing.
2. The delay between the transmission of the response message from a processing element to the terminal and the ensuing display.
3. The communication delay when a message is forwarded from one processing element to another.
4. The processing service received.
5. The time spent contending for a processor.

The crucial point in the above analysis is that the message transmission delays are INDEPENDENT of the processing and contention delays. The message transmission delays, of course, consist of contending for the message transmission medium at the endpoints, the transmission latency, and any message processing at an intermediate network point.

Clearly the first three items are highly dependent on the terminal and transmission media. They are thus not included in the model described below. In terms of application, these delays can be modeled by stochastic variables to be convolved with the distribution described below to determine an overall response time distribution.

Thus, the overall response distribution is composed of the five components described above and can be expressed as the convolution of:

1. The initial arrival delay distribution.
2. The final departure delay distribution.

3. The n-fold convolution of the transmission delay distribution, where n is the known (for a particular stimulus) number of inter processor transmissions.

4. The processing time distribution.

5. The processor contention time distribution.

The remainder of this paper describes the model used to generate an approximation to the distribution of the software delays.

3. THE MODEL

The model described in this section is designed to predict the software latency in the stimulus response problem described above. In turn, we shall describe the operating system environment; how stimuli are processed; the relation among processes, processors, and stimuli; the model for contention; the handling of multiple priority levels; the model for competing work; and the integration of all these components into the overall model.

3.1 The Operating System

The following assumptions are made about the operating system support for software running in this system:

1. The software processing is done in an environment where the operating system supports several software processes, each of which can perform some service in processing stimuli to generate responses.

2. Processes can send messages to other processes. After sending a message, a process can continue execution.

3. Processes can become blocked when they attempt to receive a message that has not arrived yet.

4. Processes have static priorities associated with them, and the operating system assures that the highest priority process on any given processor has control of the processor. In other words, if a message arrives for a high priority process, a currently running low priority process will be preempted until no high priority process is unblocked. The preempted process will then continue from the point of interruption.

5. Processes can block themselves waiting for a specific message even though there may be other messages waiting for service by that process.

6. The operating system maintains ready lists in a first come first served arrangement at each priority level.

In summary, then, we have a multi programming environment with preemptive resume scheduling and a complex processor allocation algorithm.

3.2 Stimulus Processing

In addition, we must now consider the type of processing a stimulus receives. The following properties hold:

1. For any specific stimulus, the service required is well known. That is to say that whenever a particular stimulus arrives, the sequence of processes that must process that stimulus is known. Moreover, the work to be done by each process in the chain is known. Thus, for any particular stimulus, we have a predetermined route through the system and the work required at each stop on the route is known. This information can be obtained in many fashions, including traces, walkthroughs, measurement, and the like. The details of how this information is obtained is outside the scope of this paper.

2. Since priorities are associated with processes, and the routing of a stimulus is to a sequence of processes, it is the case that a particular stimulus can receive different portions of its processing at distinct priority levels.

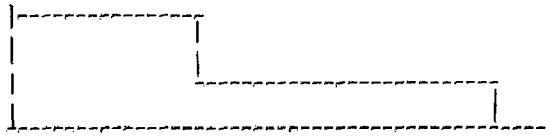
3. The distribution of all work to be performed by the processor can be approximated. The technique is rather simple. Given an overall stimulus arrival rate, and a distribution of stimulus types, the known data about service for stimuli can be used to model the processor occupancy at any given priority level. Moreover, from the scenario traces one can derive distributions of the service times.

4. What is unknown is exactly which stimuli are being processed when any particular stimulus arrives for processing.

We divide the total work distribution on a processor into two pieces parts. One is the work requirement for the specific stimulus in question whose response distribution is to be determined. The other is the work requirement for all stimuli in competition with the selected stimulus. We use the term "competing work" to describe this second component.

Thus, the actual work and routing for a particular stimulus is well known, but the work requests for competing stimuli are not exactly known. However, the distribution of the competing requests are known, so the competing work can be characterized by a stochastic process. This competing work causes a specific stimulus to encounter delays in its progress through the system due to contention. It is the purpose of this model to characterize the distribution of these contention delays.

The service requirement distribution selected for the competing work takes the form shown below.



This model of service requirements for competing work is intended to represent a uniform distribution with a tail portion. It was chosen because the majority of the service times fall in a range, and there are a few outliers outside the range. Although we use overall average, maximum value, and percentage in the tail, almost any three of the following parameters are sufficient to characterize the distribution.

The percent of area in the tail
The average value
The maximum value
The ratio of the tail length to the majority length
The maximum non-tail value

3.3 Processors, Processes, and Stimuli

Stimuli are served by several processes in succession. These processes may be distributed among several physical processors. On any given physical processor, only one process can have control of that processor at a given time. Thus, there is contention among processes for processors. Moreover, at any given time, several stimuli may be competing for service from a process. None of these stimuli can receive service until the process both has control of the processor and decides to perform work on a particular stimulus.

In terms of contention algorithms, stimuli are served by processes on a first come first served basis. When a stimulus is sent to a process, the stimulus is placed at the end of the queue of stimuli awaiting service by the process. In addition, if the process is not already waiting for the processor, the process is placed at the end of the queue of processes waiting for the processor. This means that stimuli are not necessarily served in first come first served sequence.

Consider a case where there are three processes, P, Q, and R, all resident on the same physical processor and all assigned the same priority. Suppose process P is using the processor and two stimuli, A and B waiting for service from process P. Suppose that process Q is waiting for the processor to provide service to stimulus C and process R is not waiting for the processor. Suppose further that process P decides that stimulus A needs additional work from process R. Process R will then get in line behind process Q. If process P then decides that stimulus B needs service from process Q, the physical processor will eventually pass control to process Q before process R so stimulus B will receive another portion of service BEFORE stimulus A does. Thus stimulus B has effectively cut in front of stimulus A. The sequence is shown below:

A receives service from P
B receives service from P
C receives service from Q
B receives service from Q
A receives service from R

3.4 Contention

We now address the circumstances under which a stimulus can be delayed in receiving service. These are enumerated below.

1. A stimulus can be receiving service from a process, and that process may be preempted by the arrival of a message to a higher priority process.

2. A process, while providing service to a stimulus, may decide the particular stimulus needs additional service from another process and may forward the stimulus to that other process for additional service. This can cause delay in two fashions.

   a. The sending process has control of the processor and will continue to work, probably on another stimulus, thus causing the original stimulus to wait until the receiving process can get control of the processor.

   b. Even though the receiving process may get control of the processor, the stimulus message may be queued behind other messages awaiting service by the receiving process.

3. When a stimulus initially arrives at a processor, the processor may be doing something else. In this case, the stimulus must wait its turn.

We have enumerated the three causes of delay for processing a stimulus. The existence of higher priority work can cause delays, and so can the existence of contention among competing stimuli receiving attention at the same priority level. This second form of delay can occur only when a stimulus is routed from one process to another for additional processing. Finally, we have delineated the initial arrival at a processor as another potential contention point.

3.5 Priorities

Priorities can make the situation more complex. We use the following scheme to alleviate the confusion caused by multiple priorities. First, we consider only the highest priority level. This is strictly a one priority level system and can be modeled easily.

Next, we consider ONLY the second highest priority level. All the traffic at this level is a one priority level system, and can easily be analyzed or modeled.

We then ask, what is the effect of the higher priority work being done on response at this second priority level. The work at the higher priority level introduces periods of unavailability that increase the response time at the second priority level. With analytic models, this complicates matters considerably. With simulation, we can easily model these preemption delays via preemption capabilities in the simulation language. In fact, it is this difficulty that leads to using simulation as a technique for solving this problem.

Going on to additional levels of priority, similar techniques can be used to model the effects of more priority levels.

3.6 Competing Work

We have already discussed the notion of competing work. Since we know the processing of stimuli exactly, we can easily derive a distribution of service quanta at any priority level. Given a system load, we can also derive a processor occupancy for any priority level. Knowing both the service distribution and the occupancy at a given priority level, it is easy to derive the average interarrival time. The only remaining issue is that of the distribution to be assigned to competing work.

We know that stimuli arrive independently, and the stimuli arrive according to a Poisson process. However, stimuli require a sequence of quanta for service. One can argue, and justifiably so, that the arrival of requests for quanta are highly correlated and thus do not admit to being modeled by a Poisson process. To put this another way, the system might erroneously be viewed as an assembly line with several tandem stations. In that situation, the arrival at any station after the first is highly correlated with the service at the prior station. Below, we explain why this does not happen.

There are two effects that counter this argument. One is the changes in priority level. The changes in priority level are independent from stimulus to stimulus, and these priority changes reduce the correlation. The other factor that reduces the correlation among requests for quanta is the multiple processor configuration of the system. In this multiple processor configuration, the association between stimuli and physical processors is independent of when stimuli arrive. Thus, requests for quanta on a particular processor are equally likely to come from processes residing on any of the processors and are equally likely to be routed to processes residing on different processors. These two effects are used to justify using an exponential arrival pattern for the competing quantum arrival process.

In summary, we have used the independence of stimulus arrivals, the routing independence, and the multiple priority levels to justify using a Poisson process to model the arrival of competing service quanta. We recognize this as being somewhat optimistic. The worst case would involve perfect correlation between quanta. That case can be handled analytically by taking the n-fold product of a single server single queue contention system. This situation is more realistic and does not lend itself to analytical modeling.

3.7 The Integrated Model

The model should be clear at this point. Given a particular stimulus response trace, we merely repeatedly simulate that trace allowing the transaction to compete with competing work generated as part of the simulation model. We observe the transit times and present them in a table. The GPSS/H implementation of this model is described in the next section.

4. GPSS IMPLEMENTATION

GPSS (SCHR74) was chosen for the programming. The specific version selected was GPSS/H (HENR83) for reasons detailed below. This choice allows the model to be data driven. We first describe the form of the data. Next we describe the preliminary processing by the GPSS/H program. We then describe the simulation process. Finally, the verification and validation of the program are delineated.

4.1 Data

The GPSS/H program is data driven to allow runs for various stimuli without reprogramming the model. The data file contains the number of passes for the specific stimulus to make. The next two lines of data characterize the competing load by giving the average quantum, the maximum quantum, and the percent of the quanta in the tail. The fourth line of data defines the occupancy at low and medium priorities. The fifth line contains the number of quanta in the specific stimulus under consideration, and the rest of the lines characterize each quanta of the special processing by providing the length and priority.

4.2 Preliminary Processing

After reading in the data, some initial processing is done. The values of GPSS/H variables used to generate competing quanta are calculated from the input values. The total work for the special stimulus is also computed by adding all input values. A first cut approximation of delay is made using the formula that the expected delay will be:

```
work * occupancy / ( 1 - occupancy )
```

A delay parameter is set to control the cycling rate of the special stimulus so it will not always be contending for resources. Since the occupancies and average service times are known for the competing work and the special stimulus, the average interarrival time parameter is then calculated for each priority level.

4.3 Simulation

Low priority transactions arrive at the system according to the distribution detailed above. A service time is assigned using the input distribution. The transaction then seizes a processor for its quantum before leaving. There are also appropriate bookkeeping blocks to tabulate statistics about transit times.

Medium priority transactions arrive according to their distribution detailed above. Service times are assigned to the transaction in a similar fashion. The transaction then preempts the processor for its quantum. Clearly, if there are two medium priority transactions in the system at the same time, the second one must wait until the first has completed its quantum because the preemption is done on a priority basis. There are also bookkeeping blocks to keep track of occupancy, contention, and preempted intervals.

A single transaction represents the special stimulus. This transaction makes the requisite number of passes through the system. For each pass, the scenario is to save the current time, request each of the quanta in the list of quanta, and

record in a GPSS/H table the pass transit time. The transaction then waits for a calculated period before making another pass. For each quantum in the pass, the service time and priority level is selected from the appropriate table entry. GPSS/H conditional branch statements route the transaction to code that is similar to that for competing transactions.

4.4 Verification and Validation

Verification and validation were done in traditional fashions. The trace facilities of GPSS/H assured correct transaction flow for all logical paths through the model. Interactive tracing assured that transactions were being preempted properly, waiting was handled as intended by the model designer, and that simultaneous events were handled in the proper fashion.

Initially, tables were kept of service distributions to assure GPSS/H was generating service quanta according to the input distributions. GPSS/H constructs were used to keep track of occupancy at the different priority levels. As this was the author's first use of this particular dialect of GPSS, he was delighted with the ease of this process. The GPSS/H program is 233 lines long including comments, and was designed and debugged in less than a week of full time equivalent time. The particular implementation is of GPSS/H running on a VAX using the VMS 3.7 operating system. The system is highly loaded with typical University academic and research use, and the GPSS/H response was as good as anything during the development phase.

In addition, the model was run in a single priority mode, and the results were compared with numerical results generated (JAGE82) by a different method. The results obtained by the two methods did not differ significantly.

5. RESULTS

The results presented here show the difference between using Poisson arrivals and Uniform arrivals for the competing load.

With uniform arrivals of the competitive load, the results of the transit of the special stimulus change little. There are significant decreases, however, in the waiting of competitive stimuli. This is to be expected with uniform arrivals rather than Poisson arrivals because the competitive stimuli are less likely to compete with one another.

Of interest is the distribution of the tail of the special transit response distribution. This is shown below.

It can be seen that there is very little difference between the tail distributions.

The model has been used in other circumstances not reported here, and the results have been equally as good.

6. References

(HENR83) Henriksen, J. O. and Crain, R. C., "GPSS/H User's Manual," Wolverine Software, Annandale, VA, 1983.

(JAGE82) Jagerman, D. L., "An Inversion Technique for the Laplace Transform," The Bell System Technical Journal, Volume 61, No. 8, October, 1982, pages 1195-2002.

(SCHR74) Schreiber, T. J., "Simulation Using GPSS," Wiley, 1974.

Item	Poisson	Uniform	Difference
Low priority quanta	48297	48418	0.25 %
Med priority quanta	23789	24145	1.5 %
Special Transit	15.63	15.82	1.2 %
Low average wait	1.837	1.467	20 %
percent waiting	30.5	25.4	17 %
Low non-zero avg	6.017	5.773	4.05 %
Medium avg wait	0.326	0.260	20 %
percent waiting	10.1	8.2	19 %
Medium non-zero	3.228	3.156	2.2 %
Preemptions	9176	9569	4.3 %
Average Preempt	4.695	4.818	2.6 %

Time	Poisson	Cumulative	Uniform	Cumulative
below 20	88.41 %	88.41 %	87.19 %	87.19 %
20 to 30	9.89 %	98.30 %	10.54 %	97.73 %
30 to 40	1.10 %	99.40 %	1.49 %	99.22 %
40 to 50	0.46 %	99.86 %	0.57 %	99.79 %
50 to 60	0.14 %	100.00 %	0.19 %	99.98 %
60 to 70			0.02 %	100.00 %

DR. CHARLES M. SHUB

Dr. Charles M. Shub, Associate Professor of Computer Science at the University of Colorado at Colorado Springs, received his BSEE and MSEE from the University of Maryland in 1967 and 1968. He received a Ph. D. in Computer Science from the University of Kansas in 1974. In addition to his current appointment, he has been on the Computer Science faculty at the Universities of Wyoming and Vermont. He has also been a researcher for A. T. and T. Information Systems (nee Bell Laboratories) working on the performance prediction of large scale distributed systems. He is a member of ACM and SIGSIM, serving as vice chairman of SIGSIM from 1978 to 1980. He is also a member of the Society for Computer Simulation, having served on their board of governors since 1982. He has published several papers and participated in panel discussions on Simulation, Performance Evaluation, and Operating Systems.

Associate Professor
Computer Science Department
University of Colorado-Colorado Springs
P.O. Box 7150
Colorado Springs CO 80933-7150
(303) 593-3492