Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

A PERFORMANCE MODEL FOR PARALLEL SIMULATION

Phillip M. Dickens
Paul F. Reynolds, Jr.

Institute for Parallel Computation
School of Engineering and Applied Science
University of Virginia
Charlottesville, VA 22903

ABSTRACT

Widowing algorithms are an important class of synchronization algorithms for parallel discrete event simulation. In these algorithms, a simulation window is chosen such that all events within the window can be executed concurrently without the possibility of a causality error. Using the terminology of Chandy and Sherman (1989), these are *unconditional* events. Windowing algorithms, as all non-aggressive algorithms, have been criticized for not allowing a computation to proceed because there exists the *possibility* of a causality error. We are interested in the impact of extending the simulation window in order to allow the computation of *conditional* events, that is, those events that may cause an error. In this paper we develop a model to investigate the probability of a causality error occurring when the simulation window is extended to allow conditional events into the computation stream. Also we give results from simulation studies which validate our model.

1 INTRODUCTION

Most of the protocols developed for parallel discrete event simulation fall into two basic categories. One category (using the terminology developed by Reynolds 1988) is protocols that are *accurate, non-aggressive* and *without risk* (also known as "conservative" protocols, e.g. Chandy and Misra 1979, Lubachevsky 1988, Nicol 1991 and Peacock, Manning and Wong 1978). The second category is protocols that are *accurate, aggressive* and *with risk,* also known as "optimistic", e.g. Time Warp (Jefferson 1985). Protocols that are non-aggressive and without risk do not allow a *logical process* (LP) to process a message with timestamp *t* if it is possible that it will receive another message with a timestamp less than *t* at some point in the future. Protocols that are aggressive allow LPs to process any events received, and any causality errors that result from the aggressive processing are corrected through a *rollback* mechanism.

At least two researchers are investigating the benefits of adding aggressiveness to existing non-aggressive protocols (Lubachevsky et al. 1989c, Dickens 1990). This paper lays the groundwork for an analytical investigation into the benefits of adding aggressiveness to a very important class of non-aggressive protocols: *synchronous windowing algorithms* (Nicol 1991, Lubachevsky 1988, Chandy and Sherman 1989, Ayani 1989).

The windowing algorithms under consideration proceed in three distinct phases, each separated by barrier synchronization. In the first phase, LPs determine the simulation window cooperatively. The floor of the window is the minimum timestamp in the system. The ceiling of the window is chosen such that all events with timestamps falling within the window can be executed concurrently without the possibility of any causality errors. Using the terminology of Chandy and Sherman (1989) events falling within the window are *unconditional* events. That is, their execution cannot be affected by any other event in the system. Events with timestamps outside the window boundary are *conditional* events: Their computation may be affected by some other event in the system.

The second phase of a windowing algorithm consists of the concurrent execution of all events having timestamps within the window. In the third phase, events generated as a result of the processing in the second phase are passed on to other LPs. For the purposes of this discussion, the mechanism used to determine the simulation window is unimportant. The primary difference among the various windowing algorithms is the mechanism to determine which events can be processed concurrently without causality errors.

As discussed by Lubachevsky (1989a), a very important issue is whether the overhead associated with the synchronization mechanism grows rapidly as the size of the simulation grows. This question relates to the *scalability* of a protocol. Both Time Warp (Jefferson 1985) and the Null Message Protocol (Chandy and Misra 1979) have the potential for explosive overhead costs: Time Warp because of the potential for cascading rollbacks and increasing state saving costs, and the Null Message Protocol because of the potential for an explosion of null messages (Lubachevsky 1989a). The windowing algorithms represent an important class of protocols because they are the only mechanisms for which scalability results have been proven.

Windowing algorithms, as all non-aggressive protocols, have been criticized for not fully exploiting all of the parallelism available in the simulation application (Fujimoto 1990). This is because an event will not be executed if it is *possible* that some other event in the system can affect its execution. Thus if it is possible that one event can affect another event's execution, but rarely does so, then the two events will be executed sequentially even though most of the time they could be executed concurrently.

One way to gain more parallelism in a windowing algorithm is to extend the window boundary and allow the execution of conditional events. The benefit of this approach is that events that may affect each other, but generally do not, can be executed concurrently. Such events would be excluded from concurrent execution by the basic algorithm because of the *possibility* of an error. The disadvantage is that there must be some state saving and rollback mechanism to correct causality errors that do occur as a result of the processing of conditional events.

When the window is extended to allow the computation of conditional events, the amount of progress made by the system can be viewed as consisting of two components. The first component, the *unconditional* progress, is the amount of progress made as a result of processing unconditional messages. The second component, the *aggressive* progress, is the amount of progress made as a result of processing conditional events.

Nicol (1991) derives the expected amount of unconditional progress made by a system synchronized with his windowing protocol. We are interested in investigating the *aggressive* progress made by a system synchronized by a protocol similar to the one developed by Nicol (1991). There are three primary issues to investigate. First, what is the probability of a causality error occurring given that conditional events are admitted into the computation stream? Second, what is the amount of aggressive progress made by a system before the first causality error occurs? Third, how much useful work can be identified and maintained given that causality errors have occurred?

In this paper we present the results of the first phase of our investigation into the aggressive progress of a system. We define the aggressive progress of a system as a sequence of "steps", and determine the probability of a causality error occurring at a given step. Also we report on simulation studies that validate our model. The rest of the paper is organized as follows. In section 2 we discuss other analytic results that have been developed to examine the behavior of various synchronization mechanisms. In section 3 we develop our model. In section 4 we derive the probability of a causality error at a given step. In section 5 the analytic results are compared to simulation studies and in section 6 we give our conclusions and discuss our future work.

2 BACKGROUND

As noted, windowing algorithms represent an important class of synchronization mechanisms because they are the only protocols which have been proven to have scalability properties. Lubachevsky (1989a) addresses the issue of the scalability of the Bounded Lag Algorithm. It is shown that if it takes time T to complete a simulation on one processor, then to simulate a problem K times larger using K processors will take $O(T\log K)$ time to complete. Thus the proof is addressing the situation where both the problem size and the size of the architecture grow proportionately while other system parameters such as event density remain constant.

Nicol (1991) also proves the scalability of his approach, but in the context of a fixed size architecture. It is shown that as the problem size grows relative to a fixed architecture, the per event overhead approaches O (log T) where T is the total number of events in the system. The event overhead includes synchronization costs, event list manipulation, lookahead calculation and idle time due to synchronization constraints. Thus the overhead of the method approaches the overhead of a sequential simulation as the problem size increases.

Also Nicol shows that as the problem size grows, the number of events available for execution in a given iteration grows. It is shown that given some constant minimum delay greater than zero, the average number of events processed per window grows at least linearly as the total event rate in the system increases. As pointed out by Nicol, this shows that for large problems executed on medium size machines, there will generally be enough work to keep most processors from being idle.

Most other analytic studies have compared Time Warp to a system synchronized by some other approach, primarily the Chandy/Misra Null Message Protocol (Chandy and Misra 1979). Lin and Lazowska (1989b, 1990b) develop a model comparing Time Warp and the Chandy Misra Null Message protocol. State saving and rollback costs are assumed to be zero. They show that as long as "correct" computation is never rolled back by "incorrect" computation, that Time Warp always performs at least as well as the null message approach. Lipton and Mizell (1990) perform a worst case analysis between Time Warp and the Null Message protocol. They assume the state of an LP is saved after every event, but do not consider state saving costs. The authors show that there exists a simulation such that Time Warp can arbitrarily out perform Chandy/Misra. The authors then prove the converse is not true: There is no simulation problem such that Chandy/Misra arbitrarily outperforms Time Warp. But Time Warp can be worse by a factor equal to the cost of a rollback.

Felderman and Kleinrock (1990) derive an upper bound on the improvement of Time Warp over a time stepped simulation. It is assumed that Time Warp incurs no state saving or rollback costs, and that in the timestepped approach each LP has an event to process at each time increment. The authors show that when the event computation time is exponentially distributed, the maximum improvement of Time Warp over time stepped simulation is ln (P) with P processors.

Two researchers have developed models that study the behavior of a system synchronized by Time Warp. Madisetti, Walrand and Messerschmitt (1990) develop a model to derive the average rate of progress of a system under Time Warp. In this model LPs are categorized as either fast or slow, where the fast LPs are at least twice as fast as the slow LPs. They assume rollback costs are constant and independent of rollback distance. Using this model they investigate the relative advantages of different rollback schemes.

Gupta, Akyildiz and Fujimoto (1991) study the behavior of a Time Warp system under the assumption of no costs for state saving or rollback. They investigate the probability of a rollback, the fraction of events that are committed, and the expected amount of the rollback distance. They then compare the analytic results with a Time Warp system.

The model developed here is similar to the one developed by Gupta, Akyildiz and Fujimoto (1991). There are two primary differences. First, their model assumes the system is in a steady state while ours does not. Second, their model assumes that each LP has at least one message to process at all times. Our model handles the case where an LP has no messages to process.

In the following sections we analyze the benefits of adding aggressiveness to a non-aggressive protocol.

3 MODEL

The system being modeled is synchronized by a windowing algorithm, where the ceiling of the window is extended to allow the computation of conditional events. As noted, we are interested in the amount of aggressive progress made by the system. Thus we are interested in the amount of logical time advanced past the window ceiling without a causality error. Note that we use the term causality error and fault interchangeably. For simplicity, our analysis starts the system at time $t = 0$. The analysis can be generalized to begin at time $W =$ *Window Ceiling* without difficulty.

The system consists of N LPs communicating through time-stamped messages. For the purposes of this analysis it is assumed that each LP begins the simulation with M=1 messages. The number of messages with which each LP begins the simulation can be changed without difficulty. Messages are neither created nor destroyed in the course of the simulation, thus there will be N messages in the system throughout the simulation.

Processing a message consists of adding a service time to the timestamp of the message. All service times are drawn from independent, identically distributed (iid) exponential distributions with parameter λ. After processing a message, the LP randomly selects one of the N LPs in the system, and sends it the message. Each LP is equally likely to be selected. Note that an LP can send a message to itself. It is assumed that the timestamps of all messages in the system can be treated as statistically independent. This is strictly true only if once a message visits an LP it does not return. However assuming independence is reasonable if there are a large number of LPs in the system. Thus we do not preclude message cycles, we just assume it rarely happens and does not have a significant impact upon the analysis.

We assume the existence of a synchronization mechanism that advances the system in a manner analogous to a time stepped simulation. In a time stepped simulation, a simulation step involves processing all of the messages falling within a time increment. In our model, a simulation step involves processing *all* of the messages in the system. At the beginning of step $j>1$, an LP has some number $0 \leq M \leq N$ messages in its queue as a result of the processing of the previous step. We derive the distribution for M below. As noted, we assume M=1 at the beginning of step 1.

The M messages are sorted according to timestamp order and processed. After each message is processed, the LP randomly selects one of the LPs in the system and sends it the message. Only those messages in the queue at the beginning of the step are processed in the current step. Any messages received as a result of processing in the current step are placed in the input queue to be processed in the next step. Once each LP has processed all of its messages, the step is completed and the system advances to the next step.

Note that the "timestepped" model assumes that all messages are processed at each step of the simulation. In terms of the windowing algorithms we are investigating, this is equivalent to assuming a simulation window that is large enough to admit *all* conditional events to be processed. This is clearly the extreme case of extending the simulation window in order to compute conditional events. As we discuss below, current research is aimed at modeling the system when the window is extended to allow some, but not all, conditional events into the computation stream.

As noted, the assumption of the step synchronization mechanism implies that each message is processed at each step. Thus at the end of step 1, the timestamp of any message in the system will be the sum of one exponential random variable. At the end of step 2, each message in the system will have a timestamp that is the sum of two iid exponential random variables. At the end of step *i*, the timestamp of each message in the system will be the sum of *i* iid exponential random variables. This simplifies the calculation of the Clock random variable defined below.

The Clock value of a given LP_j at step *i* ($Clock_{i,j}$) is a random variable defined to be the maximum timestamp of all messages received by LP_j at that step. At step 0, $Clock_{0,j} = 0$ as all initial messages are assumed to have a timestamp of zero. At step *i*,

$$Clock_{i,j} = Max\ [OverM\ Gamma_i]$$

where *M* is the number of messages received by LP_j during step *i*, and $Gamma_i$ is the sum of *i* iid exponentials. A fault occurs if LP_j receives a message at step *i* with a timestamp less than $Clock_{i-1,j}$. In order to derive the CDF of $Clock_{i,j}$, we first derive the probability distribution for M, the number of messages received by an LP at a given step.

3.1 Probability Distribution for the Number of Messages Received

Consider the probability that some LP_i receives M messages during a given step. We first need to determine the probability that some LP (LP_j) sends a message to some LP_i under consideration. LP_j has N LPs to which it can send a message, and each LP is selected with equal likelihood. Thus the probability that it will select LP_i is $1/N$. The probability that it will not select LP_i is $(N-1)/N$.

Now consider the probability that LP_i receives exactly one message during a given step. In order for this to occur, one LP would have to send LP_i its message, and all of the other LPs would have to select another LP. The probability of an LP sending LP_i a message is $1/N$. The probability that all other LPs select another LP is $((N-1)/N)^{N-1}$. This is because there are $N-1$ LPs that must not select LP_i, and each LP has a probability of $(N-1)/N$ of not selecting it. There are also *N* ways that LP_i can receive exactly one message. Namely, any of the *N* LPs can be the LP to send it the message. Without further elaboration, the formula for the probability of receiving *M* messages during a step is given below.

$$P(M) = \left[\frac{1}{N}\right]^M \left[\frac{(N-1)}{N}\right]^{N-M} C(N,M). \qquad (1)$$

The $C(N,M)$ term is the number of *M* combinations out of *N* objects. This term quantifies the number of ways that an LP can receive exactly *M* messages out of *N* LPs. Note that this is the binomial distribution.

Now that the probability distribution for receiving M messages at a given step is established, the density function for the clock can be computed.

3.2 Density Function for the Clock

Recall the Clock for LP_j at a given step ($Clock_{i,j}$) is defined to be the maximum timestamp of all messages received by LP_j during step *i*. Consider $Clock_{1,j}$. In step 1, LP_j receives messages with timestamps that are exponentially distributed. Assume LP_j receives $M \geq 1$ messages. In order for the maximum of the *M* messages to be less than some time *c*, all *M* messages must be less than *c*. The probability that one exponentially distributed random variable is less than some time *c* is $P(C < c) = 1-e^{-\lambda c}$. The probability that *M* iid exponentially distributed random variables are all less than some time *c* is $(1-e^{-\lambda c})^M$. Note that this expression gives the probability that the maximum of *M* iid exponentially distributed random variables is less than some time *c*. The Cumulative Distribution Function of $Clock_{1,j}$ with $M \geq 1$ is:

$$CDF\ Clock_{1,j} = (1-e^{-\lambda c})^M \quad M=1,2,\ldots N. \qquad (2)$$

As can be seen, the CDF of $Clock_{1,j}$ is dependent upon the number of messages received by LP_j during step 1. In order to calculate the unconditioned CDF, we need to calculate the CDF given $M=m$ messages are received by LP_j, times the probability that $M=m$ messages are received, over all possible values of $M=m$. This is done below for $(M=m) \geq 1$.

$$CDF\ Clock_{1,j}(m \geq 1) = \sum_{m=1}^{N} (1-e^{-\lambda c})^m\ P(M=m). \qquad (3)$$

In the above Equation $P(M=m)$ is the probability of receiving $M=m$ messages in a given step which, as shown above, is binomially distributed.

Equation (3) defines the CDF of $Clock_{i,j}$ when the number of messages received is greater than or equal to one. It is slightly more complex when LP_j receives zero messages with probability $P(0)$. In this case the $Clock_{1,j}$ has a value of zero with probability one. That is, with probability one, $Clock_{1,j}$ has not advanced. The CDF for $Clock_{1,j}$ is therefore mixed. It is discrete when $m=0$, and it is continuous when $m \geq 1$. We need to describe the CDF of $Clock_{1,j}$ when $m=0$.

Recall that the CDF is defined to be the probability that a random variable is less than some given value ($F(c) = P(C < c)$). With m=0, there is zero probability that $Clock_{1,j}$ is in the range of minus infinity to zero minus epsilon, for any positive epsilon ($F(c) = 0,\ c < 0$). As the probability that $Clock_{1,j}$ equals zero is one for m=0, it follows that the probability that it is in the range of zero to infinity is also one. Therefore $F(c) = 1,\ c \geq 0$.

The Heaviside function, H(c), is defined to be zero when c is in the range of minus infinity to zero minus epsilon, one when c equals zero, and one for c equals zero to infinity. The CDF of $Clock_{1,j}$ when $m=0$ is therefore described by the Heaviside function H(c).

The final equation for the CDF of $Clock_{1,j}$ combines Equation (3) which gives the CDF when m>=1, and the Heaviside function H(c) which gives the CDF when m=0.

$$CDF\ Clock_{1,j} = \sum_{m=1}^{N} (1-e^{-\lambda t})^m \, P(M=m)$$

$$+H(c)P(M=0). \tag{4}$$

The density function for $Clock_{1,j}$ is the derivative of the CDF. This is straight-forward except with regard to the derivative of the Heaviside function. The derivative of the Heaviside function is the Dirac Delta function. For this analysis, the most important aspect of the Delta function is that it is defined such that it has a unit pulse at $c=0$. That is, the integral over any region that includes zero is one. The integral over any region that does not include zero is zero. It is also important to note that the integral of the Dirac Delta function times any other function $f(c)$, evaluated from negative to positive infinity, is $f(0)$.

$$\int_{-\infty}^{\infty} \delta(c)dc = 1, \quad \int_{-\infty}^{0-\varepsilon} \delta(c)dc = 0, \quad \int_{0+\varepsilon}^{\infty} \delta(c)dc = 0, \quad \int_{0-\varepsilon}^{0+\varepsilon} \delta(c)dc = 1,$$

$$\int_{-\infty}^{\infty} \delta(c)\, f(c)dc = f(0)$$

Now consider the CDF of $Clock_{2,j}$ when the number of messages received is greater than or equal to one. At step 2, each of the $M=m$ messages received by LP_j has a timestamp that is the sum of two exponentials, and $Clock_{2,j}$ is the maximum timestamp over all messages received during the step. The sum of two exponentials has a gamma distribution. Consider the maximum of $M=m$ $Gamma_2$.

In order for the maximum of $M=m$ $Gamma_2$ to be less than some time c, each of them must be less than c. The probability that one $Gamma_2$ is less than some time c is the integral from 0 to c over the density function.

$$P(Gamma_2 < c) = \int_0^c \lambda^2 c\, e^{-\lambda c}\, dc \ = \ 1-e^{-\lambda c}-\lambda c e^{-\lambda c}.$$

The probability that all $M=m$ (independent) $Gamma_2$ are less than some time c is thus

$$[1-e^{-\lambda c}-\lambda c e^{-\lambda c}]^m \quad m \geq 1.$$

As this is the probability that the maximum message received at step 2 is less than some time c, it is the CDF of $Clock_{2,j}$.

A generalized formula for the probability that one $Gamma_i$ is less than some time c is given below.

$$P(Gamma_i < c) = 1 - e^{-\lambda c} \sum_{n=0}^{n=i-1} \frac{(\lambda c)^n}{n!} \tag{5}$$

The right hand side of Equation (5) raised to power $M=m$ gives the probability that the maximum of m $Gamma_i$ is less than some time c, and is thus the CDF of $Clock_{i,j}$ with $m,i \geq 1$.

Now consider the CDF of $Clock_{i,j}$ with $i>1$ when $M=0$ messages are received. In this case the CDF of $Clock_{i,j}$ is unchanged from the previous step. Thus the CDF of $Clock_{i,j}$ with $i>1$ is equal to the CDF at step $i-1$ with probability $P(M=0)$. The CDF of $Clock_{i,j}$ is given below.

$$CDF\ Clock_{i,j} = \begin{cases} \displaystyle\sum_{m=1}^{N} (1-e^{-\lambda c})^m \, P(M=m) + H(c)\, P(M=0) & \text{if } i = 1 \\ \displaystyle\sum_{m=1}^{N} [1 - e^{-\lambda c} \sum_{n=0}^{i-1} (\lambda c)^n]^m \ * & \\ P(M=m) + CDF\ Clock_{i-1,j}\, P(M=0) & \text{if } i > 1 \end{cases} \tag{6}$$

4 PROBABILITY OF A FAULT

Now that the CDF of $Clock_{i,j}$ is known, the probability of a fault at a given step can be computed. A fault occurs if any message received by LP_j at step i is less than $Clock_{i-1,j}$. In order for LP_j to progress safely past step i, the timestamps of all messages received at step i must be greater than $Clock_{i-1,j}$. We want to compute this probability.

As noted, the timestamp of a message at step i will be the sum of i iid exponentials, and will thus have a gamma density function with parameters i and λ. Consider one such $Gamma_i$. We want the probability that $Gamma_i$ is greater than some particular $Clock_{i,j}$ value $Clock_{i,j}=c$. In Equation (5), we gave the probability that one $Gamma_i$ is less than some particular value $Clock_{i,j}=c$. One minus this is the probability that one $Gamma_i > Clock_{i,j}=c$.

$$P(Gamma_i > Clock_{i,j} \mid Clock_{i,j}=c) = e^{-\lambda c} \sum_{n=0}^{i-1} \frac{(\lambda c)^n}{n!} \quad (7)$$

Due to the (assumption of) independence of the messages received by an LP during a given step, the probability that $M=m > 1$ messages have timestamps greater than $Clock_{i,j}=c$ is Equation (7) raised to the power m.

$$P(m \;\; Gamma_i > Clock_{i,j} \mid Clock_{i,j} = c) =$$

$$[e^{-\lambda c} \sum_{n=0}^{i-1} \frac{(\lambda c)^n}{n!}]^m \quad m \geq 1 \quad (8)$$

Equation (8) gives the probability of not faulting given some particular value $M=m$ and a particular Clock value $Clock_{i,j}=c$. The analysis is complicated by the fact that both the number of messages received and the Clock value are random variables. The probability of not faulting is thus conditional, dependent upon two random variables.

To determine the unconditioned probability of not faulting, we need to determine the probability of not faulting given a particular value for $M=m$ and $Clock_{i,j}=c$, over all possible values for $M=m$ and $Clock_{i,j}=c$, times the probability that $M=m$ and $Clock_{i,j}=c$. Equation (8) gives the probability of LP_j not faulting at step i given $M=m$ is greater than zero, and a particular value of $Clock_{i,j}=c$. Below we give the unconditioned probability of not faulting at step i for $M=m \geq 1$.

$$P(\bar{F})_{i,j} = \sum_{m=1}^{N} \int_0^{\infty} [e^{-\lambda c} \sum_{n=0}^{i-1} \frac{(\lambda c)^n}{n!}]^m \,*$$

$$P(M=m) \; f(c)_{i-1,j} \; dc. \;\; m \geq 1 \quad (9)$$

The $f(c)_{i-1,j}$ term is the density function of $Clock_{i-1,j}$, which is the derivative of the CDF $Clock_{i,j}$ given in Equation (6). The

$$[e^{-\lambda c} \sum_{n=0}^{i-1} \frac{(\lambda c)^n}{n!}]^m$$

term is the probability of not faulting given a particular value of $M=m$ and $C_{i,j}=c$. This is summed over all possible values of $M=m$ and multiplied times the probability that $M=m$. For each value of $M=m$, all possible values for $C_{i,j}=c$ (or more precisely being in a small range around c), times the probability that $C_{i,j}=c$, are considered. This is done by integrating the density function of $Clock_{i,j}$ over the range of zero to infinity.

The probability of not faulting when zero messages are received is slightly different. In this case, the probability of not faulting is exactly the same as in the last step. That is, the probability of making it to step i is exactly the same as the probability of making it to step $i-1$ if zero messages are received during step i. The base case is step 1, where the probability of not faulting is one. This is because $Clock_{0,j}$ is zero, and there is no possibility of receiving a message with a timestamp less than zero during step 1. Equation (10) gives the final formula for the probability of not faulting at step i.

$$P(\bar{F})_{i,j} = P(\bar{F})_{i-1,j} \; P(M=0) +$$

$$\sum_{m=1}^{N} \int_0^{\infty} [e^{-\lambda c} \sum_{n=0}^{i-1} \frac{(\lambda c)^n}{n!}]^m \; P(M=m) \; f(c)_{i-1,j} \; dc. \quad (10)$$

5 VALIDATION OF MODEL

Our model predicts the behavior of a "typical" LP in the system. Simulation studies show the model predicts the behavior of a "typical" LP in the system almost perfectly. The difference between predicted and observed behavior is less than one half of one percent. The obvious question is how well can system performance be extrapolated from the behavior of a "typical" LP. One approach to predicting the performance of a system with N LPs is to take the probability of a "typical" LP making it safely to a given step, and raising this probability to power N. This approach assumes the faulting behavior of the LPs in the system is independent. To determine the feasibility of this approach, we predicted the probability of the system making it to step 2 without a fault for N=M=(4,6,8,10,12) LPs. We then ran simulation studies to determine the actual faulting behavior. The results of this study are shown in Figure 1.

As can be seen, the predicted system performance based on the independence assumption overestimates the actual probability of the system making it to a given step safely. In fact, the prediction based on the independence assumption can be considered an upper bound on the actual system performance as illustrated in the following example.

Assume a simple system with three LPs (A,B and C) and three initial messages (M0, M1, M2). Figure 2 shows the possible paths the three messages may follow from step 0 to step 2. Assume the following path for message M0. LP_A begins the simulation with message M0, and after processing the message sends it to LP_B. In step 1 LP_B processes M0 and then sends it to LP_C. Thus the path of message M0 is A-B-C. Note that at step 1 M0 will have a timestamp that is exponentially distributed, and that in step 2 its timestamp will be the sum of two exponentials. Assume its timestamp is α_0 in step 1, and α_0+α_1 in step 2. Similarly, assume message M1 follows path B-C-A, and has a timestamp of β_0 in step 1 and β_0+β_1 in step 2. Further assume message M2 follows path C-A-B, and has a timestamp of γ_0 in step 1 and γ_0+γ_1 in step 2. Now consider the necessary conditions for the system to make it safely to step 2.

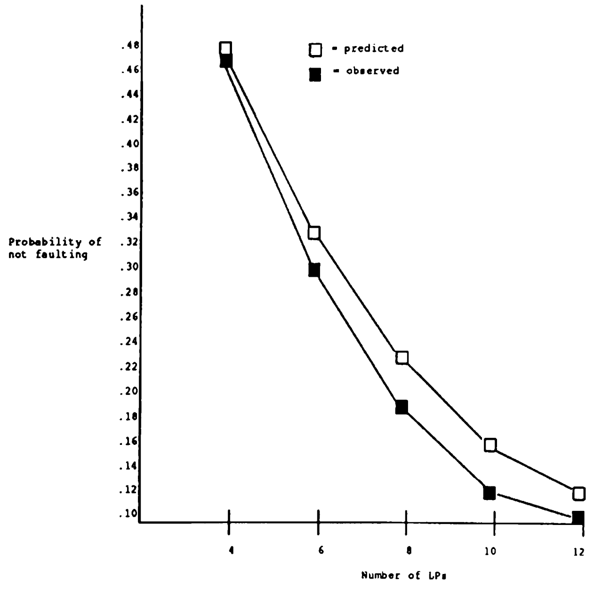

Figure 1: Predicted Versus Observed Performance

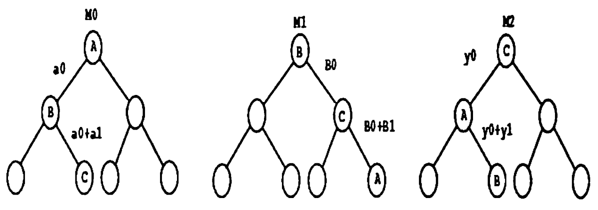

Figure 2: Possible Message Paths

In step 2, LP_C receives message M0, having a timestamp of $\alpha_0 + \alpha_1$. Note that in step 1, LP_C received message M1 with timestamp β_0. Thus for there to be no fault at LP_C, $\alpha_0+\alpha_1 > \beta_0$. Similarly, for there to be no fault at LP_B in step 2, it must be the case that $\beta_0+\beta_1>\gamma_0$. Finally, for there to be no fault at LP_A in step 2, $\gamma_0+\gamma_1>\alpha_0$. Thus we have the following three conditions which must all be true for the system to progress to step 2 without a fault.

$$A:\ \alpha_0+\alpha_1 > \beta_0$$
$$B:\ \beta_0+\beta_1 > \gamma_0$$
$$C:\ \gamma_0+\gamma_1 > \alpha_0$$

If the three events are independent, then by symmetry their individual probabilities are equal. In this case the probability of all three events occurring is the product of their individual probabilities, or equivalently, one probability raised to the third power. However the three events are not independent, and the dependence among them restricts the number of ways in which all three can occur simultaneously. That is, due to the dependence among the events the range of values for which all three events is true is narrower than the range of values if they were independent. Thus the probability that all three events occur given that they are not independent is less than the probability of one event occurring raised to the third power.

Currently our model ignores this dependence among message timestamps in the system, and the influence this has upon the faulting behavior. For this reason, it consistently over estimates the probability of the system progressing to a given step safely. Analytically capturing this dependence, and how it influences the faulting behavior of the system, is a difficult task and is the focus of current research.

6 DISCUSSION

We have a developed a model to study the behavior of a system synchronized by a windowing protocol when conditional events are allowed into the computation stream. Our model accurately predicts the probability that a "typical" LP in the system will make it to a given step without faulting. Using this probability, and extrapolating system performance, gives what we expect to be an upper bound on system performance.

Extrapolating system performance from the behavior of a "typical" LP requires the assumption of independence among the LPs. We are currently investigating ways to relax this assumption and capture the dependence among the LPs in the prediction of system performance.

The model gives the probability of a causality error at a given step. We are in the process of determining the expected amount of progress made by the system before the first causality error, and from this deriving the amount of useful work performed before the first error. Also we are investigating ways to model the amount of useful work that is performed *after* a causality error has occurred.

As noted, these results are obtained under the assumption that the simulation window is large enough to allow all messages to be processed at each step. This is essentially the same as assuming an infinite simulation window. We would like to modify this assumption of an infinite window, and examine the behavior of the system when the window is extended to allow some, but not all, conditional events to be processed. This is also the focus of current research.

ACKNOWLEDGMENTS

The authors wish to thank David Nicol and Marc Duva for their valuable input into the development of this paper.

REFERENCES

Ayani, Rassul 1989. A Parallel Simulation Scheme Based on Distances Between Objects. *Proceedings of the 1989 SCS Multiconference on Distributed Simulation*, Volume 21 Number 2, 113-118. Society for Computer Simulation, March 1989.

Chandy, K.M. and J. Misra 1979. A Case Study in the Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering* SE-5,5 May 1979, 440-452.

Chandy, K.M. and R. Sherman 1989. The Conditional Event Approach to Distributed Simulation. *Proceedings of the 1989 SCS Multiconference on Distributed Simulation*, January, 1989, 93-99.

Dickens, P.M. and P.F. Reynolds 1990. SRADS with Local Rollback. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, January, 1990, 161-164.

Felderman, R.E. and L. Kleinrock 1990. An Upper Bound on the Improvement of Asynchronous vs. Synchronous Distributed Processing. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, January, 1990, 131-136.

R. M. Fujimoto 1990. Parallel Discrete Event Simulation. *Communications of the ACM*, Volume 33, Number 10, October 1990, 30-53.

Gupta, A., I. Akyildiz and R. FUjimoto 1991. Performance Analysis of "Time Warp" with Homogeneous Processors and Exponential Task Times. *Sigmetrics Conference*, May 1991.

Jefferson, D.R. 1985. Virtual Time. *ACM Transactions on Programming Languages and Systems*, 7,3 (1985), 404-425.

Lin Y.B. and E.D. Lazowska 1989. Optimality Considerations for Time Warp Parallel Simulation. Technical Report 89-07-05, University of Washington, July, 1989.

Lin Y.B. and E.D. Lazowska 1990. Optimality Considerations for Time Warp Parallel Simulation. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, January, 1990,29-34.

Lubachevsky B. 1988. Bounded Lag Distributed Discrete Event Simulation. *Proceedings of the 1988 SCS Multiconference on Distributed Simulation*, January, 1988,183-191.

Lubachevsky B., A. Shwartz and A. Weiss 1989c. Rollback Sometimes Works... If Filtered. *Proceedings of the 1989 Winter Simulation Conference*. December, 1989, 630-639.

Lubachevsky, B. 1989a. Scalability of the Bounded Lag Distributed Event Simulation. *Proceedings of the 1989 SCS Multiconference on Distributed Simulation*, January, 1989, 100-105.

Madisetti V. and J. Walrand 1990. Synchronization in Message-Passing Computers. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, January, 1990, 35-48.

Mizel D. and R. Lipton 1990. Time Warp vs. Chandy-Misra: A Worst Case Comparison. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, January, 1990, 137-143.

Nicol D.M. 1991. The Cost of Conservative

Synchronization in Parallel Discrete Event Simulation. *JACM*, to appear, 1991.

Peacock, J.K, E.G. Manning and J.W.Wong 1979. Distributed Simulation Using a Network of Processors. *Computer Networks* 3 (1979), 44-56, North-Holland Publishing.

Reynolds, P.F. 1988. A Spectrum of Options for Parallel Simulation. *Proceedings of the 1988 Winter Simulation Conference*, December 12-14, San Diego, California, 325-332.

AUTHOR BIOGRAPHIES

PHILLIP M. DICKENS received his M.S. degree in Computer Science in January of 1986 from the University of Virginia. His research interests include algorithms for distributed simulation and parallel computation, parallel languages and compiler design. He is currently completing his dissertation involving an investigation into analytic models for parallel simulation.

PAUL F. REYNOLDS, JR., Ph.D., University of Texas at Austin, '79, is an Associate Professor of Computer Science at the University of Virginia. He has been a member of the faculty at UVa since 1980. He has published widely in the area of parallel computation, specifically in parallel simulation, and parallel language and algorithm design. He has served on a number of national committees and advisory groups as an expert on parallel computation, and more specifically as an expert on parallel simulation. He has been a consultant to numerous corporations and government agencies in the systems and simulation areas, and he has been a Research Associate at NASA, Langley, in Hampton Virginia since 1985.