Performance Analysis of Time Warp with Homogeneous Processors and Exponential Task Times*

Anurag Gupta *Ian F. Akyildiz* *Richard M. Fujimoto*

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

Abstract

The behavior of n interacting processors [1] synchronized by the "Time Warp" protocol is analyzed using a discrete state continuous time Markov chain model. The performance and dynamics of the processes are analyzed under the following assumptions: exponential task times and timestamp increments on messag 3, each event message generates one new message that is sent to a randomly selected process, negligible rollback, state saving, and communication delay, unbounded message buffers, and homogeneous processors that are never idle. We determine the fraction of processed events that commit, speedup, rollback probability, expected length of rollback, the probability mass function for the number of uncommitted processed events, and the probability distribution function for the virtual time of a process. The analysis is approximate, so the results have been validated through performance measurements of a Time Warp testbed (PHOLD workload model) executing on a shared memory multiprocessor.

1 Introduction

Over the last several years, research in synchronization mechanisms for parallel discrete event simulation programs has progressed along two fronts — *conservative* [1] [20] and *optimistic* [7] [11] approaches. Conservative schemes do not allow an event with timestamp t to be processed if there is a chance that an event with timestamp s, where $s < t$, may arrive. On the other hand, Time Warp [11], an optimistic scheme, allows computation of an event with timestamp t to proceed without regard to the possibility of the arrival of another event with a lower timestamp. If an event with a lower timestamp does arrive, the Time Warp scheme "rolls back" to the most recently saved state with virtual time earlier than the timestamp of the arriving event. The relative advantages and disadvantages of these schemes have been extensively debated [16].

The Time Warp scheme allows the processes to have different virtual times in their local clocks. It attempts to enforce a partial ordering of events [14]. Processes communicate only by exchanging messages. Each message is stamped with a *virtual send time*, and a *virtual receive time*. Virtual send time is the local time of the sender when the message is sent. Likewise, virtual receive time is the virtual time at which the message is to be processed by the receiving processor, and is also referred to as its *timestamp*. The receiver compares the timestamp of the message with its virtual time. If the message is "in the future" it is queued for later processing; if the message is "in the past" the computation must be rolled back.

Our research focusses on determining the performance measures that characterize the dynamics of the Time Warp program. The analytical solutions for these performance measures are validated against performance measurements of an implementation of Time Warp executing on a shared memory multiprocessor [6].

To our knowledge, this is the first analytic performance model for Time Warp that has been compared with measurements of an operational Time Warp system. Many analyses have dealt with the two-processor case, and could not be extended due to complexity of the problem. Lavenberg, Muntz, and Samadi [15] have obtained an approximate solution for two processors with very low interaction. Mitra and Mitrani [21] have obtained exact solutions for the two-processor case under more general conditions. Kleinrock [13] has considered a discrete state, continuous time model for the two-processor case. Felderman and Kleinrock [4] have considered a discrete time discrete state Markov model, and by taking limits have provided a unifying framework for previous work on two processor Time Warp. Plateau and Tripathi [23] have obtained numerical results for the rate of message exchange, and blocking probabilities for two communicating processors. They have employed tensor algebra to handle the three-dimensional Markov chain model of the system. Jefferson and Witkowski [12] have proposed a new stochastic process - linear Poisson process - to model timestamp driven schemes. Performance analyses of the general n process case have appeared recently. Madisetti, Walrand, and Messerschmitt [19] have derived an analytical estimate for the progress of distributed computation for the two processor case. They have also investigated different synchronization schemes for the general n process case. Felderman and Kleinrock [2] give an upper bound on the gain in speedup that n asynchronous pro-

*This work was supported by Innovative Science and Technology contract number DASG60-90-C-0147 provided by the Strategic Defense Initiative Office and managed through the Strategic Defense Command Advanced Technology Directorate Processing Division, and by NSF grant number CCR-8902362.

[1] Each process executes on a distinct processor. Hence the words "process" and "processor" are used interchangeably.

cesses can achieve relative to n synchronous lock-step processes for large n. Nicol [22] derives an upper bound on Time Warp's performance for the general n processor case. Nicol considers a self-initiating model which schedules its own state re-evaluation time as opposed to our model where a process's state is affected when messages are received from other processes. Further, his analysis of Time Warp ignores effects due to rollback propagation. Greenberg et. al. [9] present methods for parallel discrete event simulation when the number of processors is larger than the number of simulated objects. Lin and Lazowska [16] show that Time Warp always performs at least as well as any conservative mechanism (and possibly better) under certain conditions; like our analysis, they assume the overheads for state saving and rollback are negligible. Lipton and Mizell [17] demonstrate that while Time Warp may outperform the Chandy/Misra algorithms (also known as the conservative algorithms) [1][20] by an arbitrary large amount, the opposite is not true, i.e., Chandy/Misra algorithm can only outperform Time Warp by a constant factor. Lubachevsky et. al [18] present a tunable "filter" to bound the lag so that their algorithm is conservative at one extreme and is optimistic at the other extreme.

The paper is organized as follows. The model is described in section 2. Related equations are derived in section 3. The analytical results are compared with performance measurements of the Time Warp system in section 4. Finally, conclusions and suggestions for future research are given in section 5.

2 The Model

Assume there is a job which is partitioned into n processes, each of which is executed on a separate, but identical processor. Each processor is assumed to have an unbounded local buffer to store received messages. It is assumed that processors are never idle, i.e., every processor has at least one unprocessed message throughout the entire simulation. Further, we assume that the processing time of events is exponentially distributed and that upon processing a message, each event produces a single new event with a timestamp increment (i.e., receive time minus send time) that is selected from an exponential distribution. The new event message is equally likely to be sent to any other process. Thus, in our system, the number of unprocessessed messages (message population) remains constant at say, m, throughout the simulation. Most real simulations stabilize at a message population after the transient phase with minor fluctuations (otherwise, the simulation is probably unbounded).

Each processor is equipped with a virtual local clock that indicates the virtual time of that processor. Virtual times are real values totally ordered by the relation $<$. As the events are processed, the virtual local clocks in different processors move to higher virtual times, though they occasionally jump backwards when a rollback occurs. It is assumed that the time to rollback is negligible. This assumption is realistic when the computation granularity (processing time) of an event is large. We also assume a non-preemptive rollback, i.e., if a message in the past is received while an event is being processed, the rollback does not take effect until we finish processing the current event. If more than one message with a timestamp "in the past" arrive during processing, the effect is the same as if only the message with the least timestamp had arrived. Also, assume that state is saved after processing *every* event. This ensures that a process will rollback to the event with timestamp immediately less than the timestamp of the incoming late arrival, rather than to an earlier state, viz., the last saved state. The processing of an event involves the following operations:

i) receive a message with timestamp t.
ii) compare the timestamp t with the receiver's virtual clock time, s.
iii) if $t < s$, roll back to time t.
iv) - set virtual clock to t.

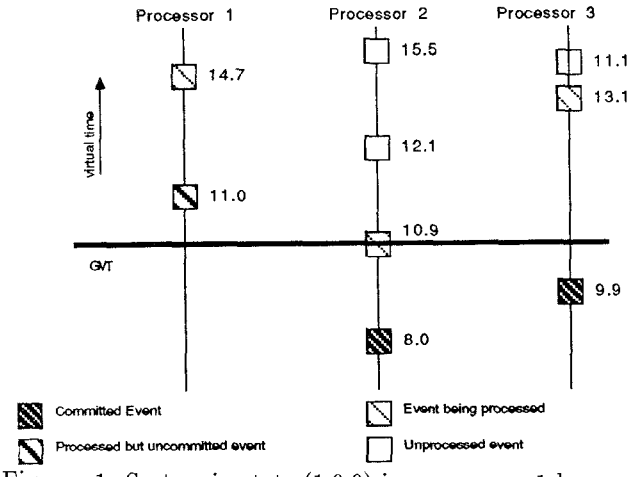

Figure 1: System in state (1,0,0) i.e. processor 1 has one processed uncommitted event, and processors 2 and 3 have none.

- read contents of message.
- update state variables.
- send a message with timestamp $t+\xi$ (where ξ is an exponentially distributed random variable.)

Communication delay for the message is assumed to be negligible. The local clock does not change during the time an event is processed; it changes only between events, and then only to the timestamp of the next message to be processed.

Jefferson [11] defines the notion of *global virtual time* (GVT). GVT serves as a floor (lower limit) for the virtual time to which a process will ever roll back. The state of the system can then be defined as $(k_1, k_2, \ldots, k_n)$ where k_i, $1 \leq i \leq n$, is the number of events that have been processed at process i that have a timestamp greater than the GVT. Figure 1 shows the case where there are three processors. However, our analysis holds for the general n-processor case. An event that contains a timestamp that is less than GVT is called a *committed* event, and others are called *uncommitted* events. Any event with timestamp less than the GVT cannot be rolled back and can be committed safely. Figure 1 shows processor 1 with one uncommitted (but processed) event, and another event being processed. For the range of virtual times that are shown next to the events, processor 1 has no committed events while processors 2 and 3 each have one. It may be noted that process 3 has an unprocessed event with timestamp *less* than the timestamp on the one being processed. Due to the non-preemptive nature of the model, the arrival of event with timestamp 11.1 is not observed by the process. This is because the event with timestamp 13.1 arrived earlier than the event with timestamp 11.1, and the former was being processed when the latter arrived.

The virtual time of a processor is the timestamp of the event being processed. GVT is the minimum of all virtual processor times (in this case, the virtual time of processor 2) and the timestamps of all unprocessed messages. The latter is necessary to account for received messages that have not yet caused a rollback due to the non-preemptive nature of the model. In such a case, a message with lower timestamp will be waiting in the input queue while a message with higher timestamp is being processed. The integer values in the state tuple are measures of the amount of work done (number of processed uncommitted events) in the process which has not yet been committed. This characterization of state is different from earlier studies that measure the progress of a process with respect to another process. The GVT-based characterization allows us to only consider interaction of a process with GVT rather than with all of the processes. The rate of progress of GVT can be considered to be a measure of system progress.

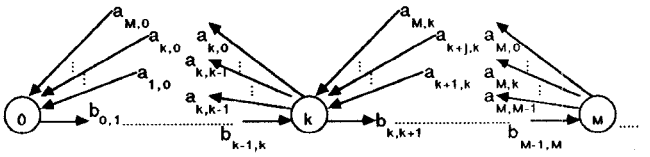

Figure 2: The Markov chain model.

In contrast to earlier studies mentioned in the previous section which have analyzed the system from the sender process point of view, our analysis is focussed on the receiver process's point of view. Cascaded rollbacks are difficult to analyze from the sender process but are accounted for easily when analyzed at the receiver process because, at any time, we need to be concerned only with the rollback at the process under consideration. Appealing to symmetry and homogeneity arguments, we observe that it is sufficient to analyze one process and its interaction with GVT. The dynamics at the other processes are identical. For our analysis, we observe that it is sufficient to classify messages as future messages and past messages rather than the usual classification into true messages (normal messages), false messages (anachronous or pre-mature execution of messages), and antimessages (to annihilate false messages). This is because the length of rollback depends on the timestamp of the message only and not on its being a true or false message, or an antimessage. Future (past) messages are those messages that are received by a process in the "future" ("past") virtual time.

Lastly, if the timestamp increment is exponential (timestamps of messages being sent out are Poisson distributed in virtual time) the committed messages at all the receiving processors are also Poisson distributed. This is the well-known Markov implies Markov (M -> M) property. We note that the uncommitted messages will *not* be Poisson distributed. This is because these events include events which will be later cancelled and exclude true messages that have not yet arrived.

3 Analysis of the Model

In this section, we present an approximate analysis of the performance of multiprocessor Time Warp which we have modeled as a Markov chain. We discuss the state space of the Markov model, derive transition rates and solve the model, determine performance measures, and finally present an algorithm for numerical solution of the model.

3.1 State Description of the Markov Model

We analyze the behavior of n coupled processes which we have modeled as a one-dimensional discrete state continuous time Markov chain as shown in Figure 2. Since we have assumed an unbounded buffer at each process, we could theoretically have an unbounded number of processed uncommitted events at a process. This is because a process can be arbitrarily far ahead of another. This implies that the associated Markov chain has an infinite number of states. However, to solve the system numerically we desire a finite number of states. We observed from subsequent characterizations of the transition rates that the greater the number of processed uncommitted events at a process, the greater the probability of the process being rolled back. Thus, if a process goes far ahead, there is a tendency for it to be pulled back. This indicates the existence of an equilibrium whence we assume that the states in the Markov chain model are not transient. In such a case, given a tolerance ϵ, we can find a finite integer M (dependent on ϵ) such that the difference between performance measures obtained by truncating the Markov chain at M and $M+1$ states is less than ϵ. Essentially, we are approximating an infinite buffer space by a finite M such that the results obtained are within a tolerance ϵ of the actual results.

To facilitate subsequent discussion, we introduce the notion of GVT-regulator. A process is a GVT-regulator if it has the event with the minimum timestamp among all the uncommitted events. Since communication delay is assumed to be zero, an event is guaranteed to be associated with a unique process - either the sender, or the receiver. It may be noted that, owing to the non-preemptive assumption, it is possible for the GVT-regulator to be processing an event with a timestamp higher than the GVT event because the latter may be waiting to be processed in the GVT-regulator's input queue.

From a state k, a process can make the following transitions (Figure 2):

- rollback to state l_1, for $0 \leq l_1 \leq k$ (rollback).
- come down by l_2 states, $0 \leq l_2 \leq k$ (GVT advancement).
- move to state $k+1$ after processing the current event (forward movement).

A state change occurs either when the process under consideration (receiver) completes processing of an event or the GVT-regulator completes its event. A rollback to state l_1 occurs when the receiver process, upon completion of the current event ($k+1^{th}$), observes an event in the past with a timestamp between the timestamps of the l_1^{th} and l_1+1^{th} events. A process comes down by l_2 states (moves to state $k-l_2$) when the GVT-regulator completes its event and GVT of the system moves past l_2 events while the current event is being processed. This is because the process now has l_2 fewer uncommitted processed events after its l_2 events were committed by GVT advancement. A process moves to state $k+1$ if neither an arrival in the "past" nor GVT advance occurs prior to completion of its current event.

To solve the model we need to derive the transition rates (Figure 2).

3.2 Determining Transition Rates

To determine the transition rates we will deal with sums of independent and identically distributed (iid) exponential random variables. For this purpose, in Appendix A we develop $L_{j,i}$ - the probability that the sum of j independent random variables is less than the sum of i independent random variables, all of which are exponentially distributed with the same rate. In Appendix B we derive $C_{j,i}$ - the probability that the sum of j random variables with rate α and a random variable with rate β is less than the sum of $i+1$ random variables with rate α. All of the random variables are independent and exponentially distributed.

We recall that the message population is m. This is the number of "threads" in the parallel simulation. We define message density to be the ratio of message population m to the number of processors n. If the timestamp increment is exponentially distributed with rate λ one might expect the messages at a processor to be Poisson distributed in virtual time with rate $m\lambda/n$ (merge m Poisson streams and then split them over n processes). While this is certainly true of the events that have committed and no longer influence computation, this is not true of uncommitted events. The effective message density for processed uncommitted events is less than the above value because lagging processors have not yet sent out their share of messages. In reality, the nearer an event is to the GVT, the smaller will be the distance between uncommitted events. This is because events far ahead of GVT would not have been generated yet (we call these yet to be generated events "holes") - there will be fewer such holes near GVT. In our analysis we make the following approximation: in computing $C_{j,i}$ we have added j random variables, each of which is identical and represents the average distance between two uncommitted events. In

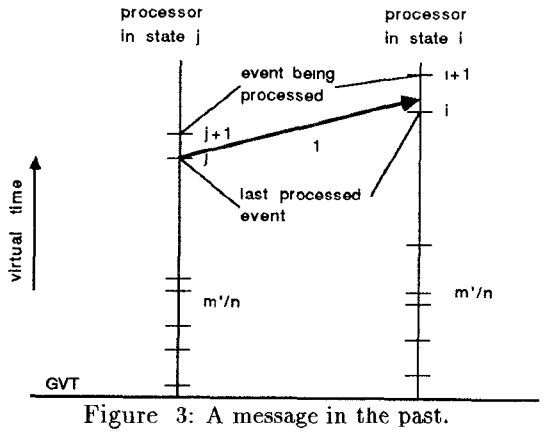

Figure 3: A message in the past.

reality, the distribution for uncommitted events will only be "pseudo" Poisson in that although the interdistance between consecutive events will be exponentially distributed, the rates for two consecutive pairs will differ slightly. We approximate this by a Poisson process with rate $\rho\lambda = m'\lambda/n$, where ρ is the effective message density and is yet to be determined. Thus, although m' does take into account the number of holes, we do not consider the fact that the virtual time between events is smaller for those near GVT than for those that are far into the (virtual time) future.

Let Q_k be the probability that a message arriving at a process in state k is "in the past". Let P_j be the steady state probability of the process being in state j. We make the following observation in Figure 3. When a receiver in state i receives a message from a sender in state j "in the past," it implies that the sum of j random variables with rate $m'\lambda/n$ and one random variable with rate λ is less than the sum of $i+1$ random variables with rate $m'\lambda/n$. The j random variables correspond to the interdistance between j events "in the past". Each of these interdistances is assumed to be exponentially distributed with rate $\rho\lambda$. In this context, $\alpha = \rho\lambda$ and $\beta = \lambda$ (see Appendix B) because the processed uncommitted events are distributed with rate $\rho\lambda$ and the timestamp increment is distributed with rate λ. Hence we set $\gamma = \alpha/\beta = \rho\lambda/\lambda = \rho$ in the expression for $C_{j,i}$. $C_{j,i}$ can now be interpreted as the probability that a message will be "in the past" of the receiver if the sender process is in state j and the receiver process is in state i. Further, no process can send a message to itself "in the past" (hence the factor $(n-1)/n$). Owing to the non-preemptive nature of the model, it is possible that none of the processes is in state 0. In the following expression we assume that the steady state distributions at all the processes are independent. This product form assumption is also implicit in the derivation of some of the other expressions later on. Hence,

$$Q_k = \frac{n-1}{n}\sum_{j=0}^{M} P_j C_{j,k} \qquad \text{for } k = 0, 1, \ldots, M \quad (1)$$

where M is the number of states. The summation represents the probability that a message arriving at a process in state k is "in the past" given that the message is from another process.

3.2.1 Transition Out of State k

Now we determine the probabilities of rollback (R_k), GVT advancement (G_k), and forward movement of a process (F_k) while a process is in state k. Processing times of events at all of the processes are assumed to be identical and exponentially distributed. The mean processing time is independent of the mean timestamp increment.[2] Due to the non-preemption assumption, it is possible for the GVT-regulator (process having the unprocessed event with least timestamp) to be in state k, $k > 0$. However, we observe that the higher the value of k the lower is the probability that the processor is GVT-regulator. This is later verified in (8). Intuitively, a process is more likely to roll back to immediate past than it is to roll back to distant past. This issue of locality has also been discussed in [11]. In deriving equations (2- 7), we ignore the possibility that a process in state k, $k > 0$ is the GVT-regulator. When a process is in state k, $k > 0$ (process is not the GVT-regulator), one of the following can occur:

(i) the process completes execution of its current event and observes a message "in the past" (rollback with probability R_k).
(ii) the GVT-regulator completes execution of its current event and observes a message "in the past".
(iii) the process completes execution of its current event and does not observe a message "in the past" (forward movement with probability F_k).
(iv) the GVT-regulator completes execution of its current event and does not observe a message "in the past" (GVT advancement with probability G_k).

It is possible for the GVT-regulator in state 0 to observe an event "in the past" due to the non-preemptive rollback assumption. Now, consider two processes - one is the GVT-regulator, and the other is the process under consideration (receiver process). As the processing times of the processes are iid exponential each of the process is equally likely (i.e. probability 1/2) to complete execution of its current event first. Among the above four actions, only the second one does not affect the state of the process (GVT-regulator finishes before the process with probability 1/2, and observes an event in the past with probability Q_0). Hence the normalization factor $(1 - Q_0/2)$ is used in the denominator of expressions (2 - 4). A rollback occurs when the process completes execution of its current event before the GVT-regulator (probability 1/2), and observes an event in the past (Q_k):

$$R_k = \frac{1}{2}Q_k/(1 - \frac{Q_0}{2}) \qquad k = 1, 2, \cdots \quad (2)$$

The process moves forward when it completes execution of its current event before the GVT-regulator (probability 1/2), and does not observe an event "in the past" (probability $1 - Q_k$):

$$F_k = \frac{1}{2}(1 - Q_k)/(1 - \frac{Q_0}{2}) \qquad k = 1, 2, \cdots \quad (3)$$

GVT advances when the GVT-regulator finishes first (probability 1/2), and does not observe an event "in the past" (probability $1 - Q_0$):

$$G_k = \frac{1}{2}(1 - Q_0)/(1 - \frac{Q_0}{2}) \qquad k = 1, 2, \cdots \quad (4)$$

When the considered process is in state 0, we must take into account the possibility of its being the GVT-regulator. Here we assume that the GVT-regulator is identical to all

[2] The value of this mean does not enter our analysis because the overhead costs such as rollback, state saving, communication, etc. have been assumed to be negligible in our model and all the phenomena at the uniprocessor and the multi-processor are scaled in the same proportion, leaving the performance measures unchanged.

the processes in state 0. Specifically, whether a process is GVT-regulator (with probability $1/i$) or not (with probability $(i-1)/i$) is assumed to be independent of the probability of observing an event in the "past" upon completion (Q_0). This is not strictly true because GVT-regulator is known to have the unprocessed event with least timestamp (either in the input queue or being processed), so it is less likely to receive another event with a timestamp lower than the timestamp of the event being processed. This assumption is implicit in equations (5 - 7).

Since the considered process is in state 0, we have at least one process in state 0. Given that at least one process is in state 0 (hence the normalization factor $1-(1-P_0)^n$), the probability that exactly i processes are in state 0 is

$$Z_i = \frac{\binom{n}{i} P_0{}^i (1-P_0)^{n-i}}{1-(1-P_0)^n} \qquad \text{for } i = 1, 2, \cdots, n$$

where P_0 is the steady state probability that a process is in state 0.

The state of the process is not affected (i.e., the number of processed uncommitted events at the process remains unchanged) if (i) the process is not the GVT-regulator (probability $(i-1)/i$), (ii) GVT-regulator completes its event before the considered process (probability 1/2), and (iii) GVT-regulator rolls back (Q_0). Hence the normalization factor $(1-\frac{(i-1)Q_0}{2i})$ occurs in the denominator of expressions (5-7). Only one of the above i processes is the GVT-regulator. A rollback occurs when the process is the GVT-regulator (probability $1/i$) and, upon completion, observes an event in the past (probability Q_0), or if it is not the GVT-regulator (probability $(1-1/i = (i-1)/i)$), completes its event before the GVT-regulator (probability 1/2), and observes an event in the past (Q_0):

$$\begin{aligned} R_0 &= \sum_{i=1}^{n} Z_i \frac{\frac{Q_0}{i} + \frac{(i-1)Q_0}{2i}}{1-\frac{(i-1)Q_0}{2i}} \\ &= \sum_{i=1}^{n} Z_i \frac{Q_0(i+1)}{2i-(i-1)Q_0} \end{aligned} \tag{5}$$

The forward movement of a process in state 0, F_0, occurs if it is not the GVT-regulator (probability $(i-1)/i$), completes its event before the GVT-regulator(probability 1/2), and does not observe an event in the past $(1-Q_0)$:

$$\begin{aligned} F_0 &= \sum_{i=1}^{n} Z_i \frac{\frac{(i-1)(1-Q_0)}{2i}}{1-\frac{(i-1)Q_0}{2i}} \\ &= \sum_{i=1}^{n} Z_i \frac{(i-1)(1-Q_0)}{2i-(i-1)Q_0} \end{aligned} \tag{6}$$

GVT advancement for a process in state 0 occurs when either the process is GVT-regulator (probability $1/i$) and does not observe an event in the past upon completion $(1-Q_0)$, or it is not the GVT-regulator (probability $(i-1)/i$), the latter completes its event before (1/2), and does not observe an event in the past $(1-Q_0)$:

$$\begin{aligned} G_0 &= \sum_{i=1}^{n} Z_i \frac{\frac{1-Q_0}{i} + \frac{(i-1)(1-Q_0)}{2i}}{1-\frac{(i-1)Q_0}{2i}} \\ &= \sum_{i=1}^{n} Z_i \frac{(i+1)(1-Q_0)}{2i-(i-1)Q_0} \end{aligned} \tag{7}$$

3.2.2 Transition Out of State k Into State j

After determining the collective probabilities, we now determine the individual transition rates from state k to state j. The virtual time distribution of uncommitted events at a processor has earlier been approximated by a Poisson distribution with rate $\rho\lambda$. This excludes events which are yet to be generated (holes), and includes events that will ultimately be cancelled. The virtual time distribution of committed events is Poisson $(m\lambda/n)$. The difference can be accounted for by a virtual time Poisson distribution (corresponding to events that cause a rollback) with rate $\rho_t + \rho_a$ where the holes are Poisson distributed in virtual time with rate ρ_t (true messages arriving out of sequence in virtual time), and the antimessages (messages that will annihilate "false" events that currently exist) are Poisson distributed in virtual time with rate ρ_a. In addition, there will be Poisson distributed streams in virtual time with rate ρ_f for messages and corresponding antimessages that will arrive in the future (real time). These are the messages which have not yet arrived, but will arrive at a future time, and subsequently become annihilated. The messages that arrive "in the past" (and hence cause a rollback) will be Poisson distributed in virtual time with rate $(\rho_t + \rho_a + \rho_f)$. Since the uncommitted events are Poisson distributed with rate ρ, and the incoming messages "in the past" are Poisson distributed with rate $\rho_t + \rho_a + \rho_f$, the length of rollback is geometrically distributed with parameter, say,

$$p = \frac{\rho_t + \rho_a + \rho_f}{\rho}$$

Since ρ_t, ρ_a, and ρ_f are unknown, the above expression is of little use. However, it does establish that rollback length is geometrically distributed - a fact that we will use shortly. It may be noted that since we have used M as an approximation to an infinite number of states, the geometric distribution for rollback length is truncated.

Let $r_{k,j}$ be the probability of rollback from state k to state j. We note that a rollback of length 1 does not cause a change of state since only the event being processed is rolled back and the number of processed uncommitted events remains unchanged. This occurs when the arriving message has a timestamp lower than the current event ($k+1^{th}$ event) being processed but greater than the last processed event. Thus, a process rolls back from state k to state k with probability p (recall the definition of p from the previous unlabeled expression). A rollback stops at a state with probability p, and continues beyond that state with probability $q = 1-p$. Further, no arriving message will have a timestamp less than GVT. Hence, the maximum length of rollback at state k is $k+1$. Then,

$$r_{k,j} = \begin{cases} R_k q^{k-j} p & \text{for } 0 < j \le k \\ R_k q^k & \text{for } k > 0 \text{ and } j = 0 \\ R_0 & k = 0 \text{ and } j = 0 \\ 0 & \text{for } k < j \end{cases} \tag{8}$$

R_k is the probability of rollback from state k. This probability is geometrically distributed with respect to the length of rollback.

We derive $g_{k,i}$ the probability that the GVT advances by exactly i events when the process is in state k. The timestamp increments at the processes are identically and exponentially distributed. For $k > i$ and $k > 0$, we observe that the GVT can advance in two ways (see Figure 1):

(1) the GVT moves up (probability G_k), its next event has a timestamp that lies between the timestamps of the i^{th} and $i+1^{th}$ events of the considered process ($(\frac{1}{2})^{i+1}$ because the timestamp increments at the two processes are iid exponential), and all the remaining $n-2$ processes have their virtual clock time greater than the timestamp of the i^{th} event of the considered process. $L_{i,l+1}$ (Appendix A) is the probability that a process with l uncommitted events

and processing the $l+1^{th}$ event has a virtual time higher than the timestamp of the i^{th} event on another process.

$$G'_{k,i} = G_k(\frac{1}{2})^{i+1}(\sum_{l=0}^{M} P_l L_{i,l+1})^{n-2} \qquad k > 0$$

(2) the GVT moves up (probability G_k), its next event has a timestamp greater than the timestamp of the i^{th} event of the process $(\frac{1}{2}^i)$, and at least one of the remaining $(n-2)$ processes has a virtual clock time that lies between the i^{th} and $i+1^{th}$ event of the process.

$$G''_{k,i} = G_k(\frac{1}{2})^i \left[(\sum_{l=0}^{M} P_l L_{i,l+1})^{n-2} - (\sum_{l=0}^{M} P_l L_{i+1,l+1})^{n-2}\right] \qquad k > 0$$

Note that these two probabilities are not mutually exclusive. It is possible that the GVT moves up (G_k), the next event of the GVT-regulator lies between the i^{th} and $i+1^{th}$ events of the process $((\frac{1}{2})^{i+1})$, and at least one of the remaining $n-2$ processes has a virtual clock time between the i^{th} and $i+1^{th}$ events. Hence,

$$g_{k,i} = G'_{k,i} + G''_{k,i} - G_k(\frac{1}{2})^{i+1}\left[(\sum_{l=0}^{M} P_l L_{i,l+1})^{n-2} - (\sum_{l=0}^{M} P_l L_{i+1,l+1})^{n-2}\right] \qquad k > i \geq 0 \tag{9}$$

If the considered process drops from state k to 0 ($k = i > 0$), it must be that the timestamp of the next event of the GVT-regulator and the virtual clock time of all the remaining processes is greater than the timestamp of the k^{th} event of the process.

$$g_{k,i} = \begin{cases} G_k(\frac{1}{2})^k(\sum_{l=0}^{M} P_l L_{k,l+1})^{n-2} & k > 0 \text{ and } i = k \\ G_0 & k = i = 0 \\ 0 & k < i \end{cases} \tag{10}$$

Now we are ready to set up the transition rates in the Markov chain model of Figure 2.

$$a_{k,j} = r_{k,j} + g_{k,k-j} \qquad 0 \leq j \leq k \leq M \tag{11}$$

$$b_{k,k+1} = F_k \qquad 0 \leq k < M \tag{12}$$

Informally, $r_{k,j}$ is the rate of transition out of state k and into state j due to rollback. Similarly, $g_{k,k-j}$ is the transition from state k to state j when the GVT advances by $k-j$ events. Thus, $a_{k,j}$ is the rate of transition from state k to state j when $0 \leq j \leq k \leq M$. Upon forward movement (F_k) a process moves from state k to state $k+1$.

The following balance equations are obtained from Figure 2.

$$P_k(b_{k,k+1} + \sum_{j=0}^{k-1} a_{k,j}) = P_{k-1}b_{k-1,k} + \sum_{j=k+1}^{M} P_j a_{j,k} \qquad 0 < k < M \tag{13}$$

$$P_M \sum_{j=0}^{M-1} a_{M,j} = P_{M-1}b_{M-1,M} \tag{14}$$

$$P_0 b_{0,1} = \sum_{j=1}^{M} P_j a_{j,0} \tag{15}$$

Note that the following is valid

$$\sum_{j=0}^{M} P_j = 1 \tag{16}$$

3.3 Performance Measures

In this subsection we determine the expected length of rollback, expected number of processed uncommitted events, expected number of processed events above the GVT, effective message density, probability of rollback, expected number of wasted events, expected fraction of committed events, speedup assuming processors are never idle, and an approximation to the probability distribution function of the process's virtual time with origin at GVT.

The *expected length of rollback* is given by $1/p$ since rollback length is geometrically distributed with rate p.[3] Alternatively, it can be computed directly by multiplying rollback lengths with their corresponding probabilities. Equating the two expressions for expected length of rollback, we have

$$1/p = \frac{\sum_{i=1}^{M+1} i[P_{i-1}Q_{i-1}q^{i-1} + \sum_{j=i}^{M} P_j Q_j q^{i-1}p]}{\sum_{i=0}^{i=M} P_i Q_i} \tag{17}$$

The *expected number of processed uncommitted events* is

$$U = \sum_{k=1}^{M} kP_k \tag{18}$$

The *expected number of processed events above the GVT* is

$$T_{processed} = U + 1 \tag{19}$$

because the current event being processed will count as processed irrespective of whether or not a rollback occurs. This is due to the non-preemptive rollback assumption.

We now determine ρ, the *effective message density* of processed uncommitted events. It may be noted that although the *unprocessed* message population is greater than m due to the presence of antimessages, we require the message density of *processed* uncommitted events since these are the ones that are liable to roll back. We recall that the effective message density of processed uncommitted events is less than m/n since the lagging processors have not yet contributed their share of messages. Let us consider a "typical" processor. Such a processor would have half the processors ahead of it, and the other half lagging behind it. The processors that are ahead have contributed their share of messages to the typical processor. The processors that are lagging have only contributed a fraction of their share. If we ignore the messages sent by the lagging processors into "the future" of the "typical" processor, we observe that the set of messages sent out by the lagging processors to the 'typical" processor is also the set of messages that caused a rollback in the latter. A "typical" rollback message is received $1/p$ events earlier than the virtual time of the "typical" processor. Since the lagging processors (half the total number of processors, approximately) have not yet contributed $\frac{1}{p}/T_{processed}$ fraction of messages, the effective message density is

$$\rho = \frac{m}{n}(1 - \frac{\frac{1}{2p}}{T_{processed}}) \tag{20}$$

[3] It is truncated geometric for which the mean is not $1/p$. However, the choice of a large enough M ensures that the error due to this is less than the specified tolerance ϵ.

The equations for $C_{j,t}$'s (Appendix A), Q_k's (1), R_k's (2, 5), F_k's (3, 6), G_k's (4, 7), $r_{k,j}$'s (8), $g_{k,t}$'s (9, 10), transition rates (11, 12), balance equations (13 - 16), average rollback length $1/p$ (17), expected number of uncommitted events (18), expected number of processed events $T_{processed}$ (19), and effective message density ρ (20) can be solved numerically. In practice, the solution requires 3 - 4 iterations on ρ. More details on the solution procedure are given later.

Once the stationary probabilities are determined, other performance measures can also be computed.

The *probability of rollback* is

$$Q = \sum_{t=0}^{M} P_t R_t \qquad (21)$$

On an average, one will find $T_{processed}$ processed events at a processor above GVT. Some of these events will be rolled back ("wasted"), while the others will be committed ("useful"). The expected number of rollbacks for this set of processed events is $QT_{processed}$. The average length of rollback is $1/p$. The expected number of "wasted" events is

$$waste = QT_{processed}/p.$$

The *fraction of events expected to commit* is

$$\eta = (T_{processed} - waste)/T_{processed}. \qquad (22)$$

Speedup is defined to be the ratio of useful events processed by n processors to the number of events processed by a single processor per unit *real* time. Since the processors are assumed to be never idle, the speedup is

$$S = n\eta \qquad (23)$$

The *distribution of processed uncommitted events in virtual time* has been approximated by a Poisson distribution with rate ρ. The virtual time of a process is the timestamp of the event it is currently processing. The virtual time of a process in state j (j processed uncommitted events above GVT) is the timestamp of the $j+1^{th}$ event being processed. With GVT as origin, the virtual time of a process in state j is the sum of $j+1$ iid exponential random variables with rate ρ. The probability distribution function (pdf) of a process's virtual time with GVT as origin can be interpreted as a random sum of random variables and is given by ($\mathcal{L}^{-1}$ is the inverse Laplace transform)

$$f(t) = \mathcal{L}^{-1}\left(\sum_{j=0}^{M} P_j \left(\frac{\rho\lambda}{\rho\lambda + s}\right)^{j+1}\right) \qquad t \geq 0$$

$$= P_0 \rho\lambda e^{-\rho\lambda t} + \sum_{j=1}^{M} P_j \frac{(\rho\lambda)^{j+1} t^j e^{-\rho\lambda t}}{j!} \qquad (24)$$

where ρ, given by equation (20), is the effective message density and $1/\lambda$ is the mean of the exponential distribution for timestamp increments.

3.4 Algorithm

In the following we outline the procedure to solve the system of equations. Even though the highly inter-dependent set of coupled non-linear equations appear formidable, the following procedure computes the performance measures efficiently by utilizing the mutual dependence and structure of the expressions derived in the earlier subsections. The number of equations is the same as the number of unknown variables. Since some of the equations are non-linear, the possibility that the system has more than one solution is not ruled out. However, the procedure is iterative and our tests showed that numerical solutions converged to the same value irrespective of initial values assumed. Although the intermediate expressions are not listed, all labelled equations except equation 24 are necessary for solution of the system. The procedure is presented below.

PROCEDURE

input number of processes (n), tolerance (ϵ),
message population (m)
set ρ = m/n. /* equation (20) */
set p to some initial value; q = 1 - p.
/* equation (17) */
set P_t's to any initial value such that sum is 1.
/* equations (13- 16) */
compute L /* Appendix A */
set$\eta_{old} = \eta_{new} = 0$ /* fraction of committed events */
set $S_{old} = S_{new} = 0$ /* speedup */
do {
$S_{old} = S_{new}$
$\eta_{old} = \eta_{new}$
compute Q* Appendix B, makes use of ρ */
/* $\gamma = \rho$ is the effective
message density at this iteration */
do {
compute Q_k's
/* equation (1) */
compute $a_{k,j}$ and $b_{k,k+1}$
/* equation (11- 12 */
update P_t's
/* equations 13 - 16 */
update p /* equation 17 */
} **while** difference between any P_t's is greater
than ϵ
compute η_{new}, S_{new} /* equations (22, 23) */
update ρ /* equation (20) */
}**while** S and η not within desired accuracy
print S, η, and rollback probability

In the experiments the tolerance was set to 0.001. The final performance measures are accurate to at least two significant figures. The solution required 3 - 4 iterations of the outer loop. Each iteration of the outer loop corresponds to approximately 100 iterations of the inner loop. Computation of $C_{j,t}$ is the more expensive portion which, fortunately, is in the outer loop. This procedure must be run for a large enough value of M so that the error in accuracy of the results is within tolerance. For a tolerance of 0.001, a message density of 1, 4, and 16 required approximately (depending on the number of processors) M to be 10, 30, and 60 respectively. The complexity of the procedure depends most strongly on the value of the parameter M.

4 Numerical Results and Validation

The analytic model makes a number of simplifying assumptions in order to make the analysis tractable: zero communication delays, unbounded buffers, etc. Measurements of a Time Warp program running on a shared memory multiprocessor (specifically, a GP-1000 BBN Butterfly) were made, and compared with the performance predicted by the analytic model, in order to test the significance of these assumptions, and the validity of the approximations underlying the analysis. The Time Warp program is described in more detail in [5].

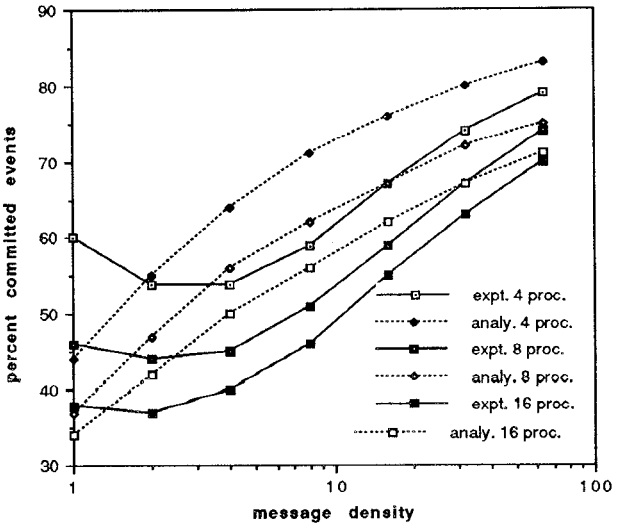

Figure 4: Analytical and experimental results for η.

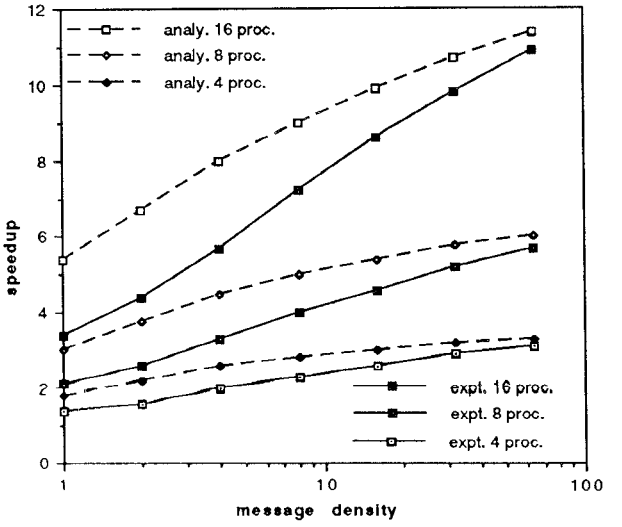

Figure 5: Analytical and experimental results for speedup.

The assumptions used in the analytic model pertaining to the application program (exponential time per event, exponential timestamp increment, fixed message population, etc.) correspond to a specific instance of the parallel HOLD (PHOLD) workload model [6]. PHOLD was used in the experiments performed here. While this satisfies certain assumptions about the Time Warp program (e.g., exponential timestamp increment, random message routing, fixed message population, and others), assumptions such as non-idle processors, zero rollback time and communication delay still do not hold.

Figure 4 compares the analytical estimate for fraction of messages that are eventually committed (useful messages) η with experimentally obtained values. We observe that for very low message densities (around message density = 1), the fraction of useful messages predicted by the analytical model is less than that which was observed experimentally. This can be attributed to the non-idle processor assumption. Consider the following: With message population of 1 (message density $1/n$), the efficiency must be 100% since no rollback can occur. In this case, the analytic model assumes that none of the processors are idle when, in fact, $n-1$ of them are idle. In the analytic model, rollback can still occur and the predicted fraction of committed messages is less than what is observed. Initially, as the message density is increased the number of rollbacks increases which reduces the fraction of committed events. Later, however, other effects come into play which cause the fraction of committed events to increase with message density. These effects are discussed shortly. For this reason, we observe a "dip" in the observed fraction of committed events when the message density is very low (around message density = 2).

For higher message densities (Figure 4), processors are busy a larger fraction of the time, so the non-idle assumption is less of a problem. We observe that as the message density is increased, generally better agreement is obtained between the predicted and measured fraction of committed events. At moderate to high message densities, the analytic model overestimates performance. We believe that the principle cause of inaccuracy is the assumption that rollback and message cancellation requires negligible time. Non-zero rollback cost causes overly optimistic processes to advance further into the future, processing more incorrect events than they would have had rollback required no time. This is because the rollback "wave" takes a non-zero time to propagate forward and "catch up" with the incorrect messages. Another assumption that contributes to the discrepancy is negligible communication delay. Since the communication delay is non-zero, a message might be in the future of the receiving processor when it was sent but in the past of the receiving processor when it arrives. This contributes to increased rollback, and hence a reduced fraction of committed events. The model assumes these overheads to be negligible, so it predicts better than actual performance. These effects are less prevalent for low message populations because, as noted earlier, processes tend to become idle rather than incorrectly advancing forward, and message traffic is lower.

Analytical and experimental results for speedup are shown in Figure 5. Due to the non-idle processor assumption, the analytical model processes more events than the simulation program. When the processors in the simulation are idle, the analytical model continues to process events, some of which are useful and contribute to increased speedup in the analytical model. For example, if a processor rolls back 40% of the events it processes, and is idle for 100 milliseconds (assuming 10 milliseconds per event), the analytic model will roll back 4 of the 10 events processed but it will still add 6 committed events to its tally while the simulation program does not add anything during this time. As the message density is increased the experimental values for speedup are observed to approach the analytically predicted values.

Analytical estimate for the fraction of committed events is shown as a function of the number of processors in Figure 6. The larger the message density, the higher the inherent parallelism leading to better performance. For a constant message density and increasing number of processors, the fraction of committed events drops rapidly at first, and then stabilizes. Initially, adding another processor increases the probability of rollback sharply but later the effect is only marginal.

Analytical estimate for the dependence of speedup on the number of processors is shown in Figure 7. The larger the message density, the larger the inherent parallelism which leads to higher speedup. As the message density is increased, the speedup plot approaches the maximum speedup possible, viz. speedup = n - the number of processors.

5 Conclusions and Future Work

We have derived and evaluated a discrete state continuous time Markov model for Time Warp. We have determined the dependence of fraction of useful events and speedup on the message population when n processes limit each others' rate of progress by passing messages between each other.

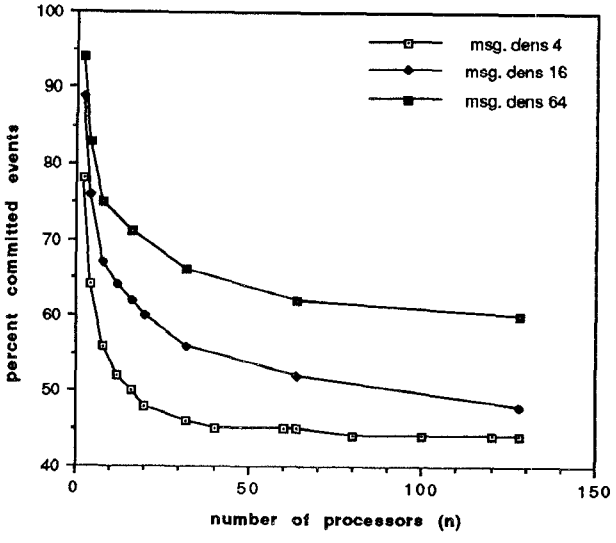

Figure 6: Analytical Estimate for the Fraction of Committed Events.

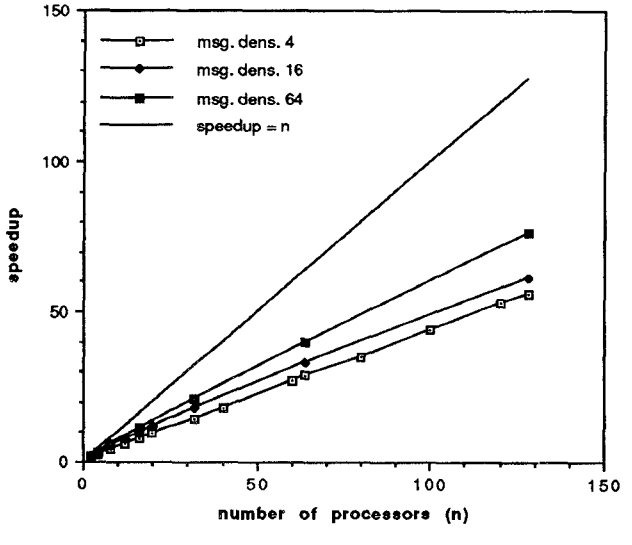

Figure 7: Analytical Estimate for Speedup.

These messages may cause a rollback and waste some of the work that a process has accomplished. In addition, we have derived expressions for several measures that characterize the dynamics of Time Warp such as expected number of processed uncommitted events at a process, the expected number of these events that will be rolled back, rollback probability as a function of number of processed uncommitted events, pdf of the process's virtual clock time, and pmf for the number of processed uncommitted events at a process. These have been determined for exponential task times, negligible rollback time and communication delay, and unbounded buffer, homogeneous, processes.

The principal contribution is the analysis of the n process case with a straight-forward Markovian model. The close agreement between the analytical and experimental results demonstrate the validity of the approximations. This provides a basis that more realistic models for Time Warp can be analyzed. In particular, our model has successfully analyzed cascaded rollbacks where complex earlier analyses had stumbled. The difficulty in analyzing cascaded rollbacks was perhaps the single most important factor that prevented extensions to the earlier two-processor analyses. In addition, to our knowledge, this is the first analysis to be backed by performance measurements of a Time Warp prototype.

There are several interesting avenues for extension. One is to investigate the effect of idle processors on the performance of Time Warp. We have estimated this, and it has been discussed in [10]. Other possible generalizations include the following.

- Extending the model to heterogeneous processors.
- Extending the analysis to general (non-exponential) timestamp distributions.

Acknowledgements

We would like to thank D. Nicol, L. Kleinrock, R.E. Felderman, and the anonymous reviewers for their insightful comments and criticisms on an earlier draft of this paper.

References

[1] K.M. Chandy and J. Misra. Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Communications of the ACM*, Vol. 24(11), pp. 198-206, November 1981.

[2] R. E. Felderman and L. Kleinrock. An Upper Bound on the Improvement of Asynchronous versus Synchronous Distributed Processing. In *Proceedings of the SCS Multiconference on Distributed Simulation*, Vol. 22(1), pp. 131-136, January 1990.

[3] R.E. Felderman. Speedup for Large Number of Processors. Personal Communication, December 1990.

[4] R.E. Felderman and L. Kleinrock. Two Processor Time Warp Analysis: Some Results on a Unifying Approach. To appear in *1991 SCS Parallel and Distributed Simulation Workshop*, January 1991.

[5] R.M. Fujimoto. Time Warp on a Shared Memory Multiprocessor. *Trans. Soc. for Comput. Simul.*, Vol. 6(3), pp. 211-239, July 1989.

[6] R. M. Fujimoto. Performance of Time Warp under Synthetic Workloads. In *Proceedings of the SCS Multiconference on Distributed Simulation*, Vol. 22(1), pp. 23-28, January 1990.

[7] R. M. Fujimoto. Parallel Discrete Event Simulation. In *Communications of the ACM*, Vol. 33(10), pp. 30-53, October 1990.

[8] E. Gelenbe. Product Form Networks with Negative and Positive Customers. To appear in *Journal of Applied Probability*

[9] A. G. Greenberg, B. D. Lubachevsky, and I. Mitrani. Unboundedly Parallel Simulations Via Recurrence Relations. In *Proc. of the 1990 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, Vol. 18(1), pp. 11-12, May 1990.

[10] A. Gupta, I.F. Akyildiz, and R.M. Fujimoto. Performance Analysis of Time Warp with Multiple Homogeneous Processors. Submitted for publication.

[11] D. R. Jefferson. Virtual Time. In *ACM Transactions on Programming Languages and Systems*, Vol. 7, pp. 404-425, 1985.

[12] D. Jefferson and A. Witkowski. An Approach to Performance Analysis of Timestamp-driven Synchronization Mechanisms. In *Proceedings of the 3rd ACM Annual Symposium on Principles of Distributed Computing*, 1984.

[13] L. Kleinrock. On Distributed Systems Performance. In *Computer Networks and ISDN Journal*, Vol. 20(1-5), pp. 209-216, December 1990.

[14] L. Lamport. Times, Clocks, and the Ordering of Events in a Distributed System. In *Communications of the ACM*, Vol. 21, No. 7, 1978.

[15] S. Lavenberg, R. Muntz, and B. Samadi. Performance Analysis of a Rollback Method for Distributed Simulation. In *Performance '83*, North-Holland Publishing Company, pp. 117-132, 1983.

[16] Y. B. Lin, and E. D. Lazowska. Optimality Considerations for "Time Warp" Parallel Simulation. In *Proc. 1990 SCS Multiconference on Distributed Simulation*, pp. 29-34, January 1990.

[17] R. J. Lipton and D. W. Mizell. Time Warp vs. Chandy-Misra : A Worst-Case Comparison. In *Proceedings of the SCS Multiconference on Distributed Simulation*, Vol. 22(1), pp. 137-143, January 1990.

[18] B. Lubachevsky, A. Shwartz, and A. Weiss. Rollback Sometimes Works ... if Filtered. In *Proc. of the 1989 Winter Simulation Conf.*, pp. 630-639, 1989.

[19] V. Madisetti, J. Walrand, and D. Messerschmitt. Synchronization in Message Passing Computers: Models, Algorithms, and Analysis. In *Distributed Simulation 1990*, pp. 35-48, 1990.

[20] J. Misra. Distributed Discrete-Event Simulation. In *Computing Surveys*, 18(1), pp. 39-65, 1986.

[21] D. Mitra and I. Mitrani. Analysis and Optimum Performance of Two Message-Passing Parallel Processors Synchronized by Rollback. In *Performance Evaluation 7*, Elsevier Science Publishers B.V. (North-Holland), pp. 111-124, 1987.

[22] D. M. Nicol. Performance Bounds on Parallel Self-Initiating Discrete-Event Simulations. Technical report ICASE 90-21, March 1990.

[23] B. D. Plateau and S. K. Tripathi. Performance Analysis of Synchronization for Two Communicating Processes. In *Performance Evaluation 8*, Elsevier Science Publishers B.V. (North-Holland), pp. 305-320, 1988.

A Derivation of $L_{j,i}$

In this appendix we derive $L_{j,i}$ - the probability that the sum of j independent random variables is less than the sum of i independent random variables, all of which are exponentially distributed with the same rate λ. Alternatively, $L_{j,i}$ can also be viewed as the probability that Erlang(j) is less than Erlang(i). Let t_1 (t_2) denote the random sum of j (i) random variables. Then, ($\mathcal{L}^{-1}$ is the inverse Laplace transform)

$$L_{0,i} = 1 \qquad i = 1, 2, \cdots \tag{25}$$

$$L_{1,i} = 1 - 1/2^i \qquad i = 1, 2, \cdots \tag{26}$$

$$\begin{aligned} L_{j,i} &= \int_0^\infty \mathcal{L}^{-1}\left(\frac{\lambda}{\lambda+s}\right)^i \left[\int_0^{t_2} \left(\mathcal{L}^{-1}\left(\frac{\lambda}{\lambda+s}\right)^j\right) dt_1\right] dt_2 \\ &\qquad j, i = 1, 2, \cdots \\ &= \int_0^\infty \frac{\lambda^i\, {t_2}^{i-1}\, e^{-\lambda t_2}}{(i-1)!} \left[\int_0^{t_2} \frac{\lambda^j\, {t_1}^{j-1}\, e^{-\lambda t_1}}{(j-1)!}\, dt_1\right] dt_2 \end{aligned}$$

$$\begin{aligned} L_{j,i} &= 1 - \sum_{k=1}^{j} \binom{i+k-2}{k-1} \left(\frac{1}{2}^{i+k-1}\right) \\ &\qquad \text{for } j, i = 1, 2 \cdots \end{aligned} \tag{27}$$

B Derivation of $C_{j,i}$

In this appendix we derive $C_{j,i}$ - the probability that the sum of j random variables with rate α and a random variable with rate β is less than the sum of $i+1$ random variables with rate α. All the random variables are independent and exponentially distributed. Let t denote the random sum of j random variables with rate α and one random variable with rate β. Similarly, let t' denote the random sum of $i+1$ random variables with rate α. Then,

$$C_{j,i} = L_{j+1,i+1} \qquad \text{for } \alpha = \beta \tag{28}$$

$$\begin{aligned} C_{0,i} &= \int_0^\infty \frac{\alpha^{i+1}\, {t'}^i\, e^{-\alpha t'}}{i!} \left[\int_0^{t'} \beta\, e^{-\beta t}\, dt\right] dt' \\ &= 1 - \left(\frac{\gamma}{\gamma+1}\right)^{i+1} \qquad \text{for } i = 0, 1, 2, \cdots \end{aligned} \tag{29}$$

where $\gamma = \alpha/\beta$.

$$\begin{aligned} C_{j,i} &= \int_0^\infty \mathcal{L}^{-1}\left(\frac{\alpha}{\alpha+s}\right)^{i+1} \left[\int_0^{t'} \left(\mathcal{L}^{-1}\left(\left(\frac{\alpha}{\alpha+s}\right)^j \frac{\beta}{\beta+s}\right)\right) dt\right] dt' \qquad \text{for } \alpha \neq \beta \\ &= \int_{t_1=0}^\infty \int_{t_2=0}^\infty \int_{t_3=t_1+t_2}^\infty \frac{\alpha}{i!}(\alpha t_3)^i e^{-\alpha t_3} \beta e^{-\beta t_2} \frac{\alpha}{(j-1)!}(\alpha t_1)^{j-1} e^{-\alpha t_1}\, dt_3\, dt_2\, dt_1 \\ &= \int_{t_1=0}^\infty \int_{t_2=0}^\infty \left(\sum_{k=0}^{i} \frac{\alpha^k (t_1+t_2)^k}{k!} e^{-\alpha(t_1+t_2)}\right) \beta e^{-\beta t_2} \frac{\alpha}{(j-1)!}(\alpha t_1)^{j-1} e^{-\alpha t_1}\, dt_2\, dt_1 \\ &= \sum_{k=0}^{i} \int_{t_1=0}^\infty \frac{\alpha^{j+k}\beta}{(j-1)!} {t_1}^{j-1} e^{-2\alpha t_1} \int_{t_2=0}^\infty \frac{(t_1+t_2)^k}{k!} e^{-(\alpha+\beta)t_2}\, dt_2\, dt_1 \\ &= \sum_{k=0}^{i} \int_{t_1=0}^\infty \frac{\alpha^{j+k}\beta}{(j-1)!} {t_1}^{j-1} e^{-2\alpha t_1} \left\{\frac{1}{(\alpha+\beta)^{k+1}} \sum_{l=0}^{k} \frac{[(\alpha+\beta)t_1]^l}{l!}\right\} dt_1 \\ &= \sum_{k=0}^{i} \sum_{l=0}^{k} \frac{\alpha^{j+k}\beta(j+l-1)!(\alpha+\beta)^l}{(j-1)!l!(\alpha+\beta)^{k+1}(2\alpha)^{j+l}} \\ &= \sum_{k=0}^{i} \sum_{l=0}^{k} \binom{j+l-1}{j-1} \left(\frac{1}{2}\right)^{j+l} \frac{\gamma^{k-l}}{(1+\gamma)^{k-l+1}} \qquad \text{where } \gamma = \alpha/\beta \end{aligned} \tag{32}$$