State Saving and Rollback Costs for an Aggressive Global Windowing Algorithm

Phillip M. Dickens
Paul F. Reynolds, Jr.
J. M. Duva

Computer Science Report No. TR-92-18
July 1, 1992

State Saving and Rollback Costs for an Aggressive Global Windowing Algorithm

Phillip M. Dickens
Paul F. Reynolds, Jr.

Department of Computer Science
University of Virginia
Charlottesville, Va. 22903
(pmd@koa.cs.virginia.edu)
(pfr@virginia.edu)

J. M. Duva
Department of Applied Mathematics
University of Virginia
Charlottesville, Va. 22903
(duva@amsun17.apma.virginia.edu)

Abstract Windowing algorithms represent an important class of synchronization protocols 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. In Dickens, Reynolds and Duva (1992) we investigated 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. We demonstrated analytically and empirically that significant performance gains are made possible as a result of this aggressive processing.

In this paper we propose a simple state saving and rollback protocol to correct errors that occur as a result of aggressive processing. We extend our analytic model to include the costs associated with aggressive processing. As we discuss, the two primary costs of aggressive processing are saving state and the possibility of cascading rollbacks. We demonstrate analytically that the probability of cascading rollbacks developing is negligible. Also we show that excellent performance gains are made possible by aggressive processing *even when the costs of saving state are included in the model.*

1. Introduction

The sheer magnitude of many simulation problems makes the use of conventional sequential techniques impractical. For this reason researchers have turned to the use of multiprocessor architectures to provide the power necessary to simulate these large systems. In a parallel discrete event simulation (PDES) the physical system is partitioned into a set of physical processes (PP) and each PP is modeled by a corresponding logical process (LP). Logical processes communicate through the use of timestamped messages where each message represents a change to the state of the system being simulated. The timestamp of the message represents the time the system state is changed.

Parallel discrete event simulation poses very difficult synchronization problems due to the underlying sense of logical time. Each LP maintains its own logical clock representing the time up to which the corresponding PP has been simulated. The fundamental synchronization problem in PDES is in the determination of when it is permissible for an LP to advance its logical clock. If an LP advances its logical clock too far ahead of any other LP in the system it may receive a message with a timestamp in its logical past. When an LP receives a message with a timestamp in its logical past it is termed a *causality error* or *fault* and may lead to incorrect results.

Most of the synchronization 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 an 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 an LP to process any event it receives, and any causality errors that result from this aggressive processing are corrected through a *rollback* mechanism.

Non-aggressive protocols are criticized for not allowing a computation to proceed because there exits the *possibility* of a causality error. Thus computations that can possibly result in an error, but

generally do not, will not be allowed to proceed. Aggressive protocols are criticized for the high overhead costs associated with state saving and rollback and because of the possibility of cascading rollbacks. A cascading rollback occurs when one rollback causes a chain of rollbacks and the number of participants increases without bounds (Lubachevsky *et al.* 1989).

At least two researchers are investigating the benefits of adding aggressiveness to existing non-aggressive protocols (Lubachevsky *et al.* 1989, Dickens and Reynolds 1990, Dickens and Reynolds 1991, Dickens, Reynolds and Duva 1992). This investigation seeks to maximize the benefits and minimize the costs of both approaches. A very important class of non-aggressive protocols to consider for this new approach is the *Global Windowing Algorithms*. These algorithms are important because they are the only class of protocols for which important scalability properties have been proven (Lubachevsky 1989a, Nicol 1991).

In Dickens, Reynolds and Duva (1992), we develop a model to investigate the benefits of adding aggressiveness to a non-aggressive Global Windowing Algorithm. We show that significant performance improvement can be attained using this approach. Further, we show that this approach maintains the very important scalability properties of the Global Windowing Algorithms: The probability of a causality error (at a given LP) does not increase *as the number of LPs goes to infinity*. Also, we are able to determine the probability of a causality error *as a function of the level of aggressiveness*. This is the first time the relationship between the probability of a causality error and the level of aggressiveness has been established.

The analysis in Dickens, Reynolds and Duva (1992) lays the theoretical groundwork for the investigation of adding aggressiveness to an existing non-aggressive protocol. It is however incomplete. First, the predicted improvement in performance is an *upper bound* on the performance that can be attained in practice. This is because the model does not account for the costs associated with aggressive processing. Second, while we show that the probability of a causality error is very small over the range of aggressiveness that we consider, it is still greater than zero. Thus we need some mechanism to correct the few causality errors that do occur.

In this paper we address both of these issues. We propose a simple state saving and rollback scheme to correct the few causality errors that occur as a result of aggressive processing. Also, we develop a model to capture the costs associated with this protocol. Further, we modify our original model to include the costs of this protocol. The result of this work is a model that gives a much clearer picture of the costs and benefits of adding aggressiveness to an existing non-aggressive protocol.

The rest of the paper is organized as follows. In section 2 we briefly review the non-aggressive version of the Global Windowing Algorithms. We then introduce our modifications that add aggressiveness to these protocols. In section 3 we develop our model to capture the costs of this approach. In section 4 we modify our previous model to include the costs of aggressive processing. In section 5 we give our conclusions and directions for future research.

2. Proposed Aggressive Global Windowing Algorithm

The non-aggressive Global Windowing Algorithms under discussion generally proceed in three phases. In the first phase, the LPs cooperatively determine the synchronization window. The floor of the window is the minimum timestamp over all unprocessed messages in the system. The ceiling of the window is chosen such that all messages within the window can be executed concurrently without any possibility of a causality error. We term this simulation window defined by the protocol the *Lookahead* window. In the second phase, each LP executes all of its messages with timestamps falling within the Lookahead window. In the third phase, the events generated as a result of execution within the Lookahead window are exchanged. Each phase is separated by a barrier synchronization. The primary difference among the various windowing protocols is the mechanism used to determine the Lookahead window. Note that only those messages with timestamps falling within the Lookahead window (where there is no possibility of a causality error) are considered for execution.

Our modified algorithm extends the simulation window past the Lookahead window established by the non-aggressive protocol. Assume the system is synchronized at logical time T where T is the current window floor. The non-aggressive windowing algorithm defines the Lookahead window from logical time T to logical time T+L, where L is the length of the Lookahead window. Our modified algorithm

defines an extended simulation window from the upper bound of the Lookahead window (logical time T+L) to logical time T+L+A. We term this extension to the Lookahead window the *aggressive* window and note that it has a length of A logical time units. In our aggressive algorithm we allow messages with timestamps falling within the aggressive window to be processed as well as those messages with timestamps falling within the Lookahead window. This is shown in Figure 1.

As noted all processing within the Lookahead window is guaranteed to be correct. Processing within the aggressive window (we term this *aggressive* processing) may lead to causality errors. Aggressive processing that does not result in a causality error is termed *successful aggressive processing*. Aggressive processing that does result in causality errors requires some mechanism to correct those errors.

The approach we choose to correct the causality errors that do occur is a simple state saving and rollback mechanism. Note that other approaches (such as some type of iterative technique) are also possible. We choose a state saving and rollback mechanism for two reasons. First, it is a widely accepted approach with significant development of hardware support (Reynolds *et al.* 1992, Buzzel *et al.* 1990). Second, we are able to use the results of our previous model (Dickens, Reynolds and Duva 1992) to derive the expected costs of using a state saving and rollback mechanism. As will be seen we are able to derive the expected number of states saved, the expected number of messages reprocessed, the probability of receiving an anti-message, and the probability of cascading rollbacks developing.

Our modified algorithm proceeds in two phases as follows. Assume the system is synchronized at logical time T. In the first phase of the algorithm the LPs cooperatively define the simulation window. As

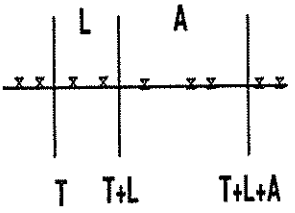


Figure 1.

in the non-aggressive version, the Lookahead window is defined such that all events within the window can be processed concurrently without the possibility of a causality error. In the aggressive version the simulation window is extended from logical time T+L to logical time T+L+A where A is the length of the aggressive window. In the second phase of the modified algorithm the LPs concurrently execute all of their events within the extended simulation window. We now elaborate on this second phase of the algorithm.

We assume state is saved before each message processed in the aggressive window. If, during the course of processing within the aggressive window, an LP receives a message in its logical past it must perform a rollback. This involves restoring the state of the LP immediately before the timestamp of the message causing the rollback. Any messages sent as a result of processing that has been rolled back is potentially invalid and is cancelled through the use of an *anti-message*. Note that this corresponds to the *aggressive cancellation policy* in Time Warp (Reiher *et al.* 1990).

In the modified algorithm the LPs compute in the extended simulation window without synchronization. Any messages generated as a result of this processing are sent as soon as they are generated. Thus the LPs do not synchronize to send messages as they do in the non-aggressive version. Once an LP reaches the ceiling of the extended simulation window it blocks waiting for all of the other LPs to similarly complete. We require that all messages generated with timestamps in the simulation window be processed before the LPs establish another simulation window. This ensures that once a new simulation window is established that no LP will ever have to roll back beyond this point. To guarantee that all messages generated within the aggressive window are processed requires two assumptions. First we assume that all messages sent are eventually received. Second, we assume there is some mechanism that allows the LPs to determine when this condition has been met. One such mechanism is to use the synchronization hardware as proposed by Reynolds *et al.* (1992). In this approach a global count of the number of outstanding messages is maintained. Each time a message is sent the counter is incremented. Each time a message is received the counter is decremented. When the count reaches zero there are no outstanding messages. When the message count reaches zero, and all LPs have reached the ceiling of extended simulation window, a new simulation window is determined.

Thus our modifications to the non-aggressive version are minimal. Note that our modified algorithm guarantees that an LP will never have to roll back past the floor of the aggressive window established in the first phase of the algorithm. This implies that state only needs to be saved when processing in the aggressive window and that it can be discarded once a new simulation window is chosen. Thus the memory requirements are much less than in a fully aggressive approach.

Note also that the mechanism we propose is essentially Time Warp (Jeferson 1985) with limited aggressiveness. It is also similar to the filtered rollback algorithm proposed by Lubachevsky *et al.* (1989) in which he adds aggressiveness to his Bounded Lag Algorithm (Lubachevsky 1988). There are two primary differences between the research presented here and that presented by Lubachevsky. The first difference is in the analysis of the algorithms. The second difference is in the level of aggressiveness allowed by the protocols. We discuss these two issues in turn.

The analysis presented in this paper is significantly different from the analysis presented by Lubachevsky in the following way. Lubachevsky derives enough information to show that the aggressive version of his Bounded Lag Algorithm maintains the scalability properties of the non-aggressive version. To accomplish this he derives the conditions under which cascading rollbacks will not develop. Also he derives an upper bound on the number of events processed in the simulation including events processed more than once due to rollbacks. Our model does not derive the conditions under which cascading rollbacks will not occur. Rather we derive explicit estimates as to the number of first generation rollbacks, the number of second generation rollbacks, etc. Using these derived probabilities we show that the probability of cascading rollbacks is negligible. Further, we do not provide an upper bound on the number of events processed in the simulation but rather derive the more precise measurement of the expected number of messages that will have to be reprocessed due to rollbacks. Finally, we derive the expected improvement in performance *as a function of the level of aggressiveness including the costs of state saving and rollbacks*. Lubachevsky does not address this issue.

The second difference is in the level of aggressiveness allowed by the two protocols. Lubachevsky allows an arbitrary bound on the amount of aggressiveness. Thus at one extreme it can be a fully aggressive algorithm. As discussed in Dickens, Reynolds and Duva (1992), we allow only a limited amount of

aggressiveness and our analysis is not applicable to a fully aggressive system.

Also we have derived many of the same measurements as Gupta *et al.* (1991) in their investigation of Time Warp. They are however studying Time Warp which is a fully aggressive protocol. Also we are interested in the expected improvement in the number of messages processed between global synchronization points due to our modified algorithm. This is not the same issue as investigated by Gupta *et al.* Further, in terms of a queueing model they assume infinite servers while our model assumes one server per LP and therefore imposes some kind of queueing. Finally they assume state saving costs are negligible and we include these costs in our model.

Now that we have described our mechanism for correcting causality errors we can develop our model for the costs of this aggressiveness. We do this in the following sections.

3. Model

We are interested in capturing the costs associated with aggressive processing. Many of the equations we need to investigate this issue are derived in Dickens, Reynolds and Duva (1992). In this section we begin by reviewing the model as developed in Dickens, Reynolds and Duva (1992). After this brief review we give the major results established in that paper that are needed for our current investigation. The interested reader is directed to Dickens, Reynolds and Duva (1992) for the derivation of these results.

Our model is closely related to the one developed by Nicol (1991) although he is not investigating processing outside of the Lookahead window. Also his model is more general than the one presented here in that his captures general timestamp increment distributions.

We model our protocol as a collection of servers where *activities* occur. There are N LPs and one server per LP. An activity begins, ends and upon its completion causes other activities. In our model each completion causes exactly one other activity. We assume the completion causes an activity at a server that is picked at random where each server is equally likely to be chosen. Note that the assumption that each server is equally likely to receive an activity is a common assumption in the parallel simulation community (Akylidiz *et al.* 1992, Felderman and Kleinrock 1992, Gupta *et. al.* 1991, Nicol 1991).

It is chosen in order to make the analysis tractable.

The delay in simulation time between when an activity begins and ends is called the *duration* of the activity. We assume the duration of an activity is drawn from independent, identically distributed exponential random variables with mean $1/\lambda$.

Our model assumes a closed queueing system. Also we assume the system is heavily loaded such that the probability of a server being idle is very close to zero. In another paper (Dickens and Reynolds 1992a) we relax the assumption of a closed system and model an open system with external Poisson arrival streams.

The width of the Lookahead window (L) at a given iteration of the protocol is a random variable. In this analysis we treat L as a constant rather than as a random variable. In particular, we use the expected value of L in our analysis. This expected value can be obtained analytically or through sample simulation runs. In Dickens, Reynolds and Duva (1992) we demonstrate analytically that using the expected value of L in our equations is very reasonable. Simulation studies validate the use of the expected value of L in our equations.

Our approach to modeling the costs/benefits of aggressive processing is as follows. We first determine the distribution for the number of messages with timestamps within the Lookahead window. These are the messages processed in the non-aggressive version of the algorithm. Next we determine the distribution for the number of aggressive messages. These messages represent one source of messages processed in the aggressive version not processed in the non-aggressive version. The other messages processed in the aggressive version are *arrival messages*. These are the messages that arrive into the aggressive window as a result of processing within the Lookahead window or the aggressive window. We discuss these messages in detail below. We note that the arrival messages are important because they are the messages that may invalidate an already processed aggressive message. Finally, we need the timestamp distribution for the aggressive messages and the arrival messages. These distributions determine the probability that an arrival message will invalidate an aggressive message.

We define the improvement due to aggressive processing as the number of messages processed between global synchronization points in the aggressive version, minus the costs of this aggressive processing, compared to the number of messages processed between global synchronization points in the non-aggressive version. The costs of aggressive processing include saving state, reprocessing messages due to rollbacks, the cost of performing a rollback and cascading rollbacks. In order to achieve compatibility with our original model we express these costs in terms of the cost of processing a message. For example, we model the costs of saving state as some percentage of the costs of processing a message. We then determine the expected improvement as a function of the cost of saving state. Note that we assume an approximately constant message processing cost.

Our model includes the costs associated with saving state and reprocessing messages. As discussed above, we represent state saving costs as some percentage of the costs of processing a message. We represent the costs of reprocessing messages by subtracting the number of messages reprocessed from the number of messages processed aggressively. The primary cost of aggressive processing is the potential for cascading rollbacks. We show that the probability of cascading rollbacks is negligible and can therefore be excluded from the model. We assume the cost of performing a rollback is negligible when compared with the costs of state saving and do not include it in our model.

Our model derives the expected improvement in the number of messages processed between global synchronization points when aggressiveness is added to the non-aggressive version of the protocol. As will be seen, our model predicts a significant improvement for a very reasonable range of state saving costs. Note however that our model does not address the question of whether the aggressive version will necessarily complete in less time than the non-aggressive version. Situations can be contrived where significantly more messages are processed between global synchronization points in our aggressive version of the algorithm and yet its completion time is worse than the non-aggressive version. We believe however that if our model predicts significantly more messages are processed between global synchronization points (after accounting for the costs associated with aggressive processing), that finishing time can be expected to be reduced.

We now give a summary of the model and the results established in Dickens, Reynolds and Duva (1992). The interested reader is directed to this paper for the derivations.

3.1. Distributions of Interest

In the equations that follow L is the expected value of the width of the Lookahead window, A is the size of the aggressive window and $1/\lambda$ is the mean of the exponential service time distribution. In Equation (1) we give the probability of processing $K \geq 1$ messages in the Lookahead window. These are the messages that will be processed between global synchronization points in the non-aggressive version of the protocol.

$$P(K,K\geq 1) = \frac{(1-e^{-\lambda L})^K}{K!}\, e^{-(1-e^{-\lambda L})}\, e^{-\lambda L} + \frac{(1-e^{-\lambda L})^{K-1}}{K-1!}\, e^{-(1-e^{-\lambda L})}\, (1-e^{-\lambda L}). \tag{1}$$

In Equation (2) we give the probability of K messages in the aggressive window given that $K \geq 1$. As discussed above these are the messages that will be processed aggressively.

$$P(K,K>0) = \frac{(e^{-\lambda L}-e^{-\lambda(L+A)})^{K-1}}{K-1!}\, e^{-(e^{-\lambda L}-e^{-\lambda(L+A)})}\, e^{-\lambda L}-e^{-\lambda(L+A)} \tag{2}$$

$$+ \frac{(e^{-\lambda L}-e^{-\lambda(L+A)})^{K}}{K!}\, e^{-(e^{-\lambda L}-e^{-\lambda(L+A)})}(1-e^{-\lambda L}-e^{-\lambda(L+A)}).$$

In Equation (3) we give the probability that an LP will receive K arrival messages. These are the messages that can potentially result in a causality error.

$$P(K) = \frac{(\lambda A-(e^{-\lambda L}- e^{-\lambda(L+A)}))^K}{K!}\, e^{-(\lambda A-(e^{-\lambda L}- e^{-\lambda(L+A)}))} \tag{3}$$

Note that this is the Poisson distribution with rate $\lambda A-(e^{-\lambda L}-e^{-\lambda(L+A)})$.

In Equation (4) we give the timestamp distribution for the messages in the aggressive window at the synchronization point.

$$pdf\,(x \mid L \leq x \leq L+A) = \frac{\lambda e^{-\lambda x}}{e^{-\lambda L}-e^{-\lambda(L+A)}} \tag{4}$$

Note that this is the exponential distribution conditioned on being within the aggressive window.

It is more complicated to derive the timestamp distribution for arrival messages. This is because the arrival messages can be generated either as a result of processing within the Lookahead window or processing within the aggressive window. The arrival messages will have different timestamp distributions

depending on where they are generated. In Dickens, Reynolds and Duva (1992) we show that the messages that arrive as a result of processing within the Lookahead window have the same distribution as given in Equation (4). We also show that we can restrict the size of the aggressive window such that the probability of an arrival message being generated as a result of processing within the aggressive window is very small. Thus we can use Equation (4) as an approximation to the timestamp distribution of the arrival messages. Note that the consequence of this approximation is that for larger aggressive window sizes our model will slightly over-predict the probability of a causality error. Further note that due to this assumption we restrict the aggressive window size to be no greater than the mean of the service time distribution ($0 \leq A \leq 1/\lambda$).

Equations (1) through (4) are needed to determine the costs associated with aggressive processing. We do this in the next section.

4. Costs of Aggressive Processing

There are two primary costs associated with aggressive processing. First is the cost of state saving. As discussed by Fujimoto (1990) large state saving costs can significantly impact the performance of aggressive processing. However if the granularity of event processing is significantly larger than state saving overhead, or if hardware support is used (Buzzel *et al.* 1990), then good performance can be achieved with aggressive processing (Fujimoto 1990). In this section we derive the expected number of states that must be saved in order to allow aggressive processing.

The second primary cost associated with aggressive processing is the possibility of *echoing* or *cascading rollbacks*. Echoing occurs when one rollback causes a chain of rollbacks and the amplitude of the rollbacks increase without bound (Lubachevsky *et al.* 1989). Cascading occurs when a rollback chain develops and the number of participants increases without bound (Lubachevsky *et al.* 1989). Echoing is not a concern in our protocol as an LP can never roll back beyond the ceiling of the Lookahead window. It is not immediately obvious that cascading rollbacks cannot develop. In this section we derive the probability of one rollback causing another rollback and show that the probability of cascading is negligible.

The other costs we measure are the expected number of messages that must be reprocessed due to rollbacks. We begin by computing the probability of generating an anti-message.

4.1. Probability of Generating an Anti-Message

The particular windowing algorithm we assume is the one developed by Nicol (1991). In order to derive the probability of generating an anti-message it is necessary to briefly review one aspect of his protocol.

In Nicol's algorithm each LP "pre-sends" activity arrivals. That is, the completion time of an activity, and the LP to receive this activity upon its completion, are both calculated at the time the activity begins. The LP to subsequently receive the activity is notified of this reception *at the time the activity begins*. We term these messages sent to inform an LP of a future arrival a *"pre-sent" completion message*. Note that the ability to "pre-send" completion messages assumes powerful lookahead capabilities. In particular, it assumes non-preemptive service and that the routing of the activity is unaffected by the load on the network at service completion. Also note that these "pre-sent" completion messages are the only messages exchanged by the LPs and are the source of the arrival messages discussed above.

When a server places an activity into service it also schedules a message for itself at the service completion time. We term this message the *"complete_service"* message to differentiate it from an arrival message. Processing of the "complete_service" message consists of giving service to the next scheduled activity as well as any statistics gathering required for the simulation. Thus for every server that is busy there is one "pre-sent" completion message, sent to the receiving LP, and one "complete_service" message scheduled on its own event list. In Dickens, Reynolds and Duva (1992) we show that the timestamp of a "complete_service" message falling within the aggressive window is exponentially distributed conditioned on being within the aggressive window. Thus it will have the same timestamp distribution given in Equation (4).

Recall our assumption that the system is heavily loaded and that the probability of an idle server is very low. This implies that with high probability each LP will have a "complete_service" message on its event list scheduled for some time in the future. It also implies that with high probability any arrival mes-

sage received will not go into service immediately. Therefore it is with high probability that it is only the processing of a "complete_service" message that will cause another activity to be placed into service. Given that it is only the placing of an activity into service that results in a message being sent to another LP (the "pre-sent" completion message), and given our assumption of a heavily loaded system, we conclude that it is only the processing of a "complete_service" message that will result in a message being sent to another LP. For the purposes of our analysis we assume this is always the case.

Note that it is not in fact the processing of the "complete_service" message that causes the "pre-sent" completion message to be sent. Rather, it is the placing of the next job into service that causes this message to be generated. Since the next job will not enter into service until the "complete_service" message is processed (because we assume with high probability there will always be an activity receiving service), we state it as if it is the processing of the "complete_service" message that causes the LP to send a "pre-sent" completion message.

If a "complete_service" message is processed aggressively, and the LP receives an arrival message with a timestamp less than the timestamp of the "complete_service" message, the message generated as a result of this processing *may* be invalid. Note that it is not necessarily the case that it will be invalidated but for the purposes of this analysis we assume it always will. Thus if a "complete_service" message is processed aggressively, and the LP receives an arrival message with a timestamp less than the timestamp of the "complete_service" message, we assume the generated message is invalid. The LP must therefore send an anti-message to cancel this message.

By definition the timestamp of the anti-message will have the same timestamp as the message being cancelled. If this timestamp falls outside of the aggressive window it will do no serious damage as the message it will cancel has not yet been processed. If however if falls within the aggressive window it can cause other aggressive messages to be rolled back and possibly cause another (second generation) anti-message to be generated. We now determine the probability that an LP generates an anti-message with a timestamp that falls within the aggressive window.

There are three conditions required for an LP to generate an anti-message with a timestamp within the aggressive window. First, the LP must process the "complete_service" message aggressively.

Second, the processing of the "complete_service" message must be invalidated. Third, the message generated as a result of processing the "complete_service" message must have a timestamp that falls within the aggressive window. We now calculate the probability of each of these events.

The first condition is that the LP must process the "complete_service" message aggressively. In order for this to occur the timestamp of the "complete_service" message must fall within the aggressive window. Let (C) be the event that the "complete_service" message has a timestamp that falls within the aggressive window. As shown in Dickens, Reynolds and Duva (1992), the probability of this event is

$$P(C) = e^{-\lambda L} - e^{-\lambda(L+A)}. \tag{5}$$

The second condition required to generate an anti-message is that the LP receive at least one arrival message with a timestamp less than the timestamp of the "complete_service" message. Note that an anti-message can also invalidate the processing of the "complete_service" message. We ignore this issue for the moment and discuss it in detail below. To determine the probability that an arrival message invalidates the "complete_service" message we first need the distribution for the number of arrival messages that fall within the aggressive window. As shown above, the probability of receiving K arrival messages with timestamps within the aggressive window is Poisson distributed with rate $\lambda A - (e^{-\lambda L} - e^{-\lambda(L+A)})$. Also we must know the distribution of the timestamps of both the "complete_service" message and the arrival messages. As discussed above, the timestamp distribution for arrival messages that fall within the aggressive window is exponential conditioned on being within the aggressive window. As noted above this is also the timestamp distribution for a "complete_service" message that falls within the aggressive window.

To determine the probability that a generated anti-message has a timstamp within the aggressive window we must determine the timestamp of the message generated as a result of processing the "complete_service" message. Recall that the service time distribution is iid exponential with mean $1/\lambda$. Thus the message generated by processing the "complete_service" message will be the sum of the timestamp of the "complete_service" message and an exponential increment of mean $1/\lambda$. Thus we need to determine the probability that an exponential increment from the timestamp of the "complete_service" message is less than the length of the aggressive window.

Recall that the aggressive window has a length of A logical time units. Thus given a "complete_service" message with timestamp S=s within the aggressive window we seek the probability that the new timestamp $s+\xi$ is less than A. For S=s this probability is

$$P(s+\xi < A \mid S=s) = 1-e^{\lambda(A-s)}.$$

We compute the unconditional probability by integrating the above equation over all possible values of $S=s$ times the probability of $S=s$. Recall that the distribution for S is given in Equation (4). Let $(2nd)$ be the event that processing the "complete_service" message aggressively causes a message to be generated with a timestamp that falls within the aggressive window. The probability of this event is

$$P(s+\xi < A) = P(2nd) = \int_0^A (1-e^{-\lambda(A-s)}) \frac{\lambda e^{-\lambda s}}{(1-e^{-\lambda A})} ds = \frac{1-(\lambda A e^{-\lambda A} + e^{-\lambda A})}{1-e^{-\lambda A}}. \quad (6)$$

We now have all of the information necessary to determine the probability of generating an anti-message with a timestamp within the aggressive window. Let $(Anti)$ be the event that an LP generates an anti-message with a timestamp within the aggressive window. The probability of this event occurring is the probability that an LP processes the "complete_service" message aggressively, receives an arrival message with a timestamp less than the timestamp of the "complete_service" message, and generates a message with a timestamp within the aggressive window. Noting the independence of these events this probability is

$$P(Anti) \approx P(C)\,P(Ar=1)\,P(2nd)\,.5 + P(C)\,P(Ar=2)\,P(2nd)\,.75 \quad (7)$$

The first term in Equation (7) is the probability of processing the "complete_service" message aggressively. The second term is the probability of receiving one arrival message. The third term is the probability of generating a message within the aggressive window. The fourth term is the probability that the arrival message has a timestamp less than the timestamp of the "complete_service" message. This probability is .5 since both messages have the same probability distribution. The other terms are similarly defined. Note that the probability of receiving more than two arrival messages is negligible and is not included in the equation. Since there are terms left out of the equation (because they have such a low probability of occurring) we note that Equation (7) is an approximation of the probability of generating an anti-message.

4.2. Second Generation Anti-Message

We compute the probability of generating a second generation anti-message in a similar manner. There are four conditions necessary for a second generation anti-message to be produced. First, the LP must receive an anti-message. Second, the LP must process the "complete_service" message aggressively. Third, the timestamp of the anti-message must be less than the timestamp of the "complete_service" message. Fourth, the "complete_service" message must have generated a message with a timestamp within the aggressive window. We have calculated the probability of all of these conditions except for the probability that the timestamp of the anti-message is less than the timestamp of the "complete_service" message. We now derive this probability.

An anti-message that falls within the aggressive window will have a timestamp that is gamma distributed, conditioned on being within the aggressive window. This is because the timestamp of the "complete_service" message is exponentially distributed within the aggressive window, and the timestamp of the message it generates is an exponential increment from the timestamp of the "complete_service" message. The probability that the timestamp of an anti-message is less than the timestamp of the "complete_service" message is therefore the probability that a gamma distributed random variable is less than an exponentially distributed random variable, given that both random variables are within the aggressive window. Let (G) be the event that the timestamp of an anti-message is less than the timestamp of the "complete_service" message. We give the probability of (G) below without elaboration.

$$P(Anti < complete\_service \mid both\ within\ window) = P(G) = \frac{.25 - e^{-\lambda A} + .75e^{-2\lambda A} + .5\lambda A e^{-2\lambda A}}{(1 - e^{-\lambda A} - \lambda A e^{-\lambda A})(1 - e^{-\lambda A})} \tag{8}$$

We now have all of the information necessary to calculate the probability of producing a second generation anti-message. Let $Anti^2$ be the event that an LP produces a second generation anti-message. Then the probability of this event is

$$P(Anti^2) = P(Anti)\,P(C)\,P(2nd)\,P(G). \tag{9}$$

Assuming the aggressive window size is set to its maximum value of $1/\lambda$, the probability of generating an anti-message is 0.046. Under these same conditions the probability of producing a second generation anti-message is 0.003. Given the very low probability of generating a second generation anti-

message it can be seen that the probability of cascading rollbacks developing is negligible. In the analysis that follows we ignore the effects of second or higher order generation anti-messages because the probability of one being produced is negligible. We do however account for the effects of first generation anti-messages.

5. Expected Number of Messages Reprocessed

As noted above a rollback occurs when a message with timestamp t has been processed aggressively and at some point in the future the LP receives a message with timestamp s such that $s < t$. In this case the LP performs a rollback to logical time s and begins processing from this point forward. All messages with timestamps greater than s are considered invalid and must be reprocessed. It is also possible that a message may have to be reprocessed more than one time. In this section we derive the expected number of messages that an LP must reprocess due to rollbacks.

Our analysis is simplified by the very limited amount of aggressive processing we are considering. This significantly restricts the number of events that need to be considered in our analysis. For example, the probability of receiving three arrival messages is negligible. The probability of receiving two arrival messages and an anti-message is also negligible. Thus the only events with significant probability of occurring are the event that an LP receives one arrival message, two arrival messages or one arrival message and an anti-message. Other events could be considered but the probability of their occurring is so small that the expected values we compute would not be affected in any significant way. Since the probability of these events occurring is non-zero however we note that our equations for the expected number of messages reprocessed are an approximation.

Our analysis is also simplified by assuming a particular real time ordering of events. That is, we assume all aggressive messages are processed before any arrival messages are received. This may not be strictly true, but the effect of this assumption is to potentially over-estimate the damage caused by an arrival message. Similarly, we assume any reprocessing of messages caused by the receipt of the first arrival message is complete by the time a second arrival message or an anti-message is received. Again this is a worst case assumption.

We begin by considering the probability of reprocessing one aggressive message given that one arrival message is received. In order to reprocess one aggressive message the LP could process exactly one aggressive message and receive one arrival message with a timestamp less than the timestamp of the aggressive message, or the LP could process two aggressive messages and receive one arrival message with a timestamp less than one, but not both, aggressive messages and so on. As discussed above, only a very few such combinations have a significant probability of occurring. Let $(RP = K)$ be the event that an LP must reprocess K messages. The most significant terms of the probability of invalidating (and thus having to reprocess) one aggressive message given one arrival message are

$$P(RP=1 \mid Ar=1) \approx P(A=1,Ar=1)\ .5 + P(A=2,Ar=1)\ .5 + P(A=3,Ar=1)\ .375 + P(A=4,Ar=1)\ .25. \tag{10}$$

The first term in Equation (10) is the probability of processing one aggressive message and receiving one arrival message. The next term (.5) represents the probability that the timestamp of the arrival message is less than the timestamp of the aggressive message. This probability is .5 because (with high probability) both messages have the same timestamp distribution. The other terms are similarly derived. Note that the probability of processing more than four aggressive messages is negligible and is not considered.

Next we give the most significant terms for the probability of reprocessing two aggressive messages given that one arrival message is received. In order for this to occur an LP must receive an arrival message with a timestamp less than the timestamps of exactly two aggressive messages.

$$P(RP=2 \mid Ar=1) \approx P(A=2,Ar=1)\ .25 + P(A=3,Ar=1)\ .5^3 3 + P(A=4,Ar=1)\ .5^4 6 \cdots \tag{11}$$

The first term in Equation (11) is the probability of processing two aggressive messages and receiving one arrival message. The second term is the probability that the arrival message has a timestamp less than the timestamps of both aggressive messages. The other terms are similarly derived.

Below we give the most significant terms for the probability of invalidating three and four aggressive messages given one arrival message. The terms have a meaning similar to the equations discussed above.

$$P(RP=3 \mid Ar=1) \approx P(A=3,Ar=1)\ .5^3 + P(A=4,Ar=1)\ .5^4\ 4... \tag{12}$$

$$P(RP=4 \mid Ar=1) \approx P(A=4,Ar=1)\ .5^4 \cdots \tag{13}$$

Now consider the probability of having to reprocess one aggressive message given that an LP receives two arrival messages. The second arrival message may invalidate the first arrival message as well as any aggressive messages processed. One way to invalidate exactly one message given two arrival messages is to process zero aggressive messages and for the second arrival message to invalidate the first. Alternatively, the LP could process one aggressive message and the second arrival message could invalidate either the aggressive message or the first arrival message, but not both. Again there are an infinite number of ways this can occur but only a few have any significant associated probability. We enumerate the most significant terms below.

$$P(RP=1 \mid AR=2) \approx P(A=0,Ar=2)\ .5 + P(A=1,Ar=2)\ .5 + P(A=2,Ar=2)\ .375 + \tag{14}$$
$$P(A=3,Ar=2)\ .25 + P(A=4,Ar=2)\ .15625...$$

Now consider the number of ways the second arrival message can invalidate two messages. The LP can process the first arrival message and one aggressive message and have the second arrival invalidate both messages. Alternatively, the LP can process two aggressive messages and the first arrival message and have the second arrival message invalidate two out of the three of these messages. The significant terms of this probability are given below.

$$P(RP=2 \mid Ar=2) \approx P(A=1,Ar=2)\ .25 + P(A=2,Ar=2)\ .5^3\ 3 + \tag{15}$$
$$P(A=3,Ar=2)\ .5^4\ 6 + P(A=4,Ar=2)\ .5^5\ 10...$$

Below we give the significant terms for the probability of invalidating three messages given two arrival messages. We note that the probability of invalidating four messages given three arrival messages is negligible.

$$P(RB=3 \mid Ar=2) = P(A=2,Ar=2)\ .5^3 + P(A=3,Ar=2)\ .5^4\ 6 + P(A=4,Ar=2)\ .5^5\ 10... \tag{16}$$

Now consider the number of messages that must be reprocessed due to the receipt of an anti-message. Recall that the probability of an LP generating an anti-message (P(Anti)) is given in Equaiton (7). Another way to view the probability of an LP generating an anti-message (falling within the aggressive window) is that P(Anti) percentage of the messages generated are cancelled. Thus if an LP receives

an arrival message there is a P(Anti) probability that this message will be cancelled.

Note that an LP must receive an arrival message in order to receive an anti-message. Further note that the probability of an LP having more than one arrival message cancelled is negligible and is not considered. As discussed above, the probability that an LP receives two arrival messages and an anti-message is also negligible and is not considered.

Recall that the timestamp of an anti-message that falls within the aggressive window is gamma distributed conditioned on being within the aggressive window. Thus the probability that an anti-message will have a timestamp less than the timestamp of an aggressive message is the probability that a conditional gamma distributed random variable is less than a conditional exponentially distributed random variable. This probability is given in Equation (11) above. Recall that we use the notation P(G) to represent this probability.

Calculating the number of ways an anti-message can invalidate one aggressive message is similar to other derivations given above. The significant terms for the probability of an anti-message invalidating one aggressive message are given below.

$$P(RP=1 \mid Ar=1, P(Anti)) \approx P(A=1, Ar=1, P(Anti))\, P(G) \tag{18}$$
$$+\ P(A=2, Ar=1, P(Anti))\, P(G)\,(1-P(G))$$
$$+\ P(A=3, Ar=1, P(Anti))\, P(G)\,(1-P(G))^2\, 2$$
$$+\ P(A=4, Ar=1, P(Anti))\, P(G)\,(1-P(G))^3\, 4\ldots$$

Below we give the significant terms for the probability of an anti-message invalidating two aggressive messages. The probability of an anti-message invalidating three or more aggressive messages is negligible.

$$P(RP=2 \mid Ar=1 P(Anti)) \approx P(A=2, Ar=1, P(Anti))\, P(G)^2 \tag{19}$$
$$+\ P(A=3, Ar=1, P(Anti))\, P(G)^2\,(1-P(G))\, 3$$
$$+\ P(A=4, Ar=1, P(Anti))\, P(G)^2\,(1-P(G))^2\, 6\ldots$$

The expected number of messages reprocessed is the expected number reprocessed due to the receipt of one arrival message plus the expected number reprocessed due to the receipt of a second arrival

message plus the expected number reprocessed due to the receipt of an anti-message. Again note that other cases could be considered but would not add any significant amount to the expected value.

$$E[RP] = P(RP=1 \mid Ar=1) + 2P(RP=2 \mid Ar=1) + 3P(RP=3 \mid Ar=1) + 4P(RP=4 \mid Ar=1) \quad (20)$$
$$+ P(RP=1 \mid Ar=2) + 2P(RP=2 \mid Ar=2) + 3P(RP=3 \mid Ar=2)$$
$$+ P(RP=1 \mid Ar=1, P(Anti)) + 2P(RP=2 \mid Ar=2, P(Anti))...$$

In Figure 2 we plot the expected number of messages that must be reprocessed as a function of the aggressive window size. In Figure 3 we plot the expected number of messages processed aggressively as a function of the aggressive window size. This is shown for comparison purposes.

As can be seen, the expected number of aggressive messages processed is 1.5 for the maximum aggressive window size considered (100% of the mean service time). For this same aggressive window size the expected number of messages reprocessed is approximately .22. Therefore we expect to have to reprocess approximately one out of every six messages processed aggressively.

6. Expected Number of States Saved

We now calculate the expected number of states that must be saved in order to allow limited aggressive processing. First, state must be saved for each of the aggressive messages processed. The distribution for the number of aggressive messages processed is given in Equation (2).

Figure 2.

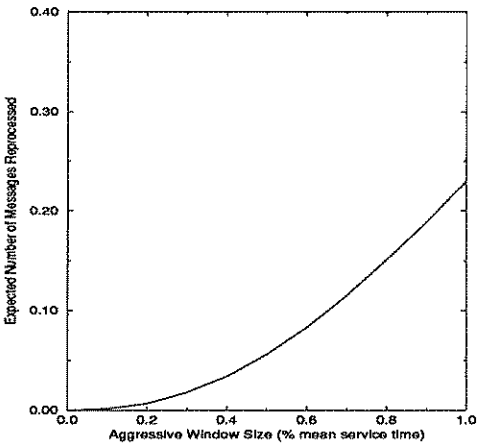
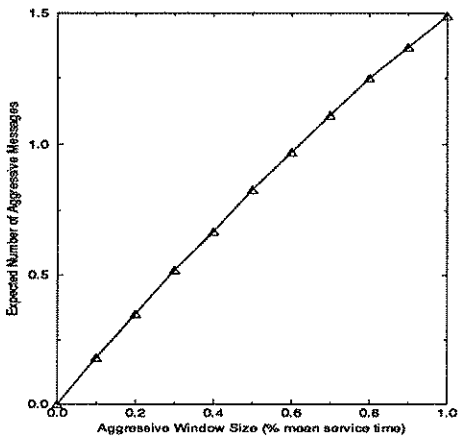


Figure 3.



$$E[A] = P(A=1) + 2P(A=2) + 3P(A=3) + 4P(A=4) \tag{21}$$

State must also be saved before each message is reprocessed. This is because a message may be reprocessed more than once. The expected number of messages reprocessed is given in Equation (20). Finally, state must be saved before processing any arrival messages since these messages can also be invalidated. The expected number of states saved due to arrival messages is the expected number of arrival messages. Noting that the probability of receiving more than two arrival messages is negligible we give the expected number of arrival messages below.

$$E[Ar] = P(Ar=1) + 2P(Ar=2) \tag{22}$$

The expected number of states saved is the sum of Equation (20), Equation (21) and Equation (22).

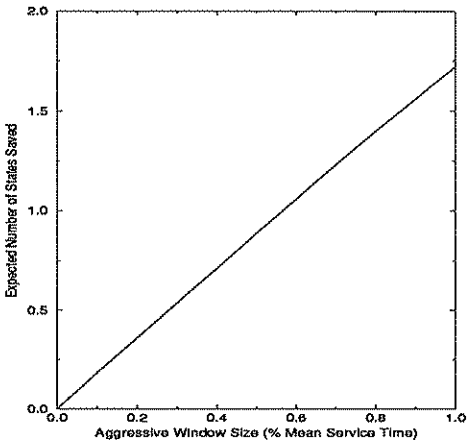
$$E[SS] = E[A] + E[RP] + E[Ar] \tag{23}$$

In Figure 4 we plot the expected number of states that must be saved as a function of the aggressive window size.

7. Expected Improvement

As discussed above, we measure the expected improvement as the number of messages processed between global synchronization points in the non-aggressive version of the algorithm compared to the number of messages processed between global synchronization points in the aggressive version. The number of messages processed in the non-aggressive version is the number of messages processed in the Lookahead window. Equation (1) gives the probability of processing K messages in the Lookahead

Figure 4.



window. The number of messages processed in the aggressive version is the total number of messages processed aggressively minus the cost of this aggressive processing. As discussed above, we account for the number of messages reprocessed by subtracting this number from the number of messages processed aggressively. We account for state saving costs by subtracting the number of states saved times the costs of saving state from the number of messages processed aggressively. As discussed above, we model state saving costs relative to the cost of processing a message. We give the expected improvement due to aggressive processing, including the costs of aggressive processing, below.

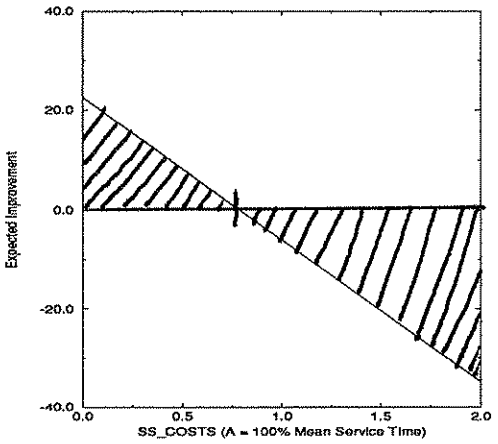
$$E[I] = \frac{(E[A]+E[Ar]+E[LA]-E[RP]-E[SS]scosts)}{E[LA]} \tag{24}$$

In Equation (24) *scosts* is the relative cost of saving state. In Figure 5 we plot the expected improvement in the number of messages processed between global synchronization points as a function of the costs of saving state. Note that in Figure 5 the aggressive window size is set to its maximum value of 100% of the mean service time distribution.

8. Conclusion

As can be seen in Figure 5 adding aggressiveness is beneficial when the cost of saving state is approximately half of the costs of processing a message. The benefit can be very substantial when the costs of saving state are insignificant as would be the case with hardware support (Reynolds *et al.* 1992, Buzzel *et al.* 1990). The maximum benefit is approximately 2100% when state saving costs are zero.

Figure 5.



When state saving costs are twice the costs of processing a message however there would be a penalty of approximately 3300% for the aggressive processing.

Note that our model addresses the question of expected improvement in terms of the expected number of messages processed between global synchronization points. Given this measure of improvement it can be shown that the expected improvement does not degrade *as the number of LPs approaches infinity*. Thus the very important scalability properties of the non-aggressive version of the algorithm are maintained. Our model does not however prove that the modified algorithm will necessarily complete in less time than the non-aggressive version. This is because our model does not consider real time behavior. Conditions can be contrived under which the non-aggressive version would complete in less time even though the aggressive version processed significantly more messages between global synchronization points. We think however that it is reasonable to conclude that when significantly more messages are processed between global synchronization points, and this includes the costs of aggressive processing, that the aggressive version will generally complete in less time. Current research is focusing on including real time behavior in our model.

REFERENCES

Akylidiz, I.F., L. Chen, S.R. Das, R.M. Fujimoto and R.F. Serfozo 1992. "Performance Analysis of "Time Warp" With Limited Memory". *1992 ACM Sigmetrics Conference.*

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.

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.

Dickens, P.M. and P.F. Reynolds 1991. "A Performance Model for Parallel Discrete Event Simulation". *1991 Winter Simulation Conference.* December, 1991, 618-626.

Dickens, P.M. and P.F. Reynolds, Jr. 1992a. "Performance Results for an Aggressive Global Windowing Algorithm in an Open System". In Preparation.

Felderman, Robert and Leonard Kleinrock 1991. "Bounds and Approximations for Self-Initiating Distributed Simulation Without Lookahead." Submitted (September 1991): *ACM Transactions on Modeling and Computer Simulation.*

Fujimoto, R. 1990. "Parallel Discrete Event Simulation". *Communications of the ACM.* October, 1990, pgs. 30-53.

Gupta, Anurag, I.F. Akylidiz and R.M. Fujimoto 1991. "Performance Analysis of Time Warp With Multiple Homogeneous Processors". *IEEE Transactions on Software Engineering.* Volume 17 (No. 10):1013-1027, Oct. 1991.

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

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 1989. "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.

Nicol, David M. 1991. "The Cost of Conservative Synchronization in Parallel Discrete Event Simulation". To appear. *Journal of the ACM.*

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.

Reiher, P., R. Fujimoto, S. Bellenot, D. Jefferson. "Cancellation Strategies in Optimistic Execution Systems. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation.*, January, 1990.

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.

Reynolds, P.F., C. M. Pancerella and S. Srinivasan. "Making Parallel Simulations Go Fast". To appear. *1992 Winter Simulation Conference.*