Minimizing the Null Message Exchange in Conservative Distributed Simulation

Syed S. Rizvi, K. M. Elleithy
Computer Science and Engineering Department
University of Bridgeport
Bridgeport, CT 06605
{srizvi, elleithy}@bridgeport.edu

Aasia Riasat
Department of Computer Science
Old Dominion University
Norfolk, VA 23529
ariast@cs.odu.edu

Abstract— **The performance of a conservative time management algorithm in a distributed simulation system degrade s significantly if a large number of null messages are exchanged across the logical processes in order to avoid deadlock. This situation gets more severe when the exchange of null messages is increased due to the poor selection of key parameters such as lookahead values. However, with a mathematical model that can approximate the optimal values of parameters that are directly involved in the performance of a time management algorithm, we can limit the exchange of null messages. The reduction in the exchange of null messages greatly improves the performance of the time management algorithm by both minimizing the transmission overhead and maintaining a consistent parallelization. This paper presents a generic mathematical model that can be effectively used to evaluate the performance of a conservative distributed simulation system that uses null messages to avoid deadlock. Since the proposed mathematical model is generic, the performance of any conservative synchronization algorithm can be approximated. In addition, we develop a performance model that demonstrates that how a conservative distributed simulation system performs with the null message algorithm (NMA). The simulation results show that the performance of a conservative distributed system degrades if the NMA generates an excessive number of null messages due to the improper selection of parameters. In addition, the proposed mathematical model presents the critical role of lookahead which may increase or decrease the amount of null messages across the logical processes. Furthermore, the proposed mathematical model is not limited to NMA. It can also be used with any conservative synchronization algorithm to approximate the optimal values of parameters.**

Keywords— conservative distributed simulation, Lookahead, logical processes null messages, null message algorithm.

I. INTRODUCTION

This paper presents a mathematical model for a conservative distributed simulation system that uses null messages to avoid deadlock. The term distributed refers to distributing the execution of a single run of a simulation program across multiple processors [1]. By distributing the execution of a computation across N processors, one can finish the computation up to N times faster than if it were executed on a single processor. Therefore, the main reason behind the use of distributed simulation is to reduce the overall simulation execution time.

One of the main problems associated with distributed simulation is the synchronization of distributed execution. If not properly handled, synchronization problems may degrade the performance of a distributed simulation environment [2]. Time management algorithms are, therefore, required to ensure that the execution of the distributed simulation is properly synchronized. Two main classes of time management algorithms are *conservative* and *optimistic*. This paper focuses on the performance issues related to the conservative null message algorithm (NMA) that uses null messages to avoid deadlock and provide synchronization among the logical processes (LPs). The selection of values for several critical parameters such as lookahead, null message ratio (NMR), and frequency of transmission plays an important role in the generation of null messages. If these values are not properly chosen by a simulation designer, the result will be an excessive number of null messages across each LP. This situation gets more severe when the NMA needs to run to perform a detailed logistics simulation in a distributed environment to simulate a huge amount of data as specified in "in press" [9]. This paper provides a quantitative criterion to limit an excessive number of null messages exchanged by predicting the optimal values of the critical parameters. The reduction in the null message exchange minimizes the transmission overhead and hence improves the overall system performance. In addition, we show that the performance of a conservative distributed simulation system degrades if the NMA generates an excessive number of null messages.

The rest of the paper is organized as follows. In section II, we provide an overview of the conservative protocols, focusing on the null message protocol (NMP) and its related problems. In section III, we derive the proposed mathematical

model that approximates the optimal values of the key parameters. Section IV provides a comprehensive discussion on various optimizations that we have incorporated in our proposed mathematical model. In addition, section IV gives a brief discussion on the numerical and simulation results. Finally, we conclude in section V.

II. Related Work

Event synchronization is an essential part of parallel simulation. In general, synchronization protocols can be categorized into two different families: conservative and optimistic. Conservative protocols fundamentally maintain causality in event execution by strictly disallowing the processing of events out of timestamp order. The main problems faced in conservative algorithms are overcoming deadlock and guaranteeing the steady progress of simulation time.

Examples of conservative mechanisms include Chandy, Misra and Byrant's NMP [6], and Peacock, Manning, and Wong [11] avoided deadlock through null messages. The primary problem associated with null messages is that if their timestamps are chosen inappropriately, the simulation becomes choked with null messages and performance suffers. Some intelligent approaches to null message generation include generation on demand [8], and generation after a time-out [5]. Some earlier research on discrete event simulation has focused on variants of NMP, with the objective of reducing the high null message overhead. For instance, Bain and Scott [4] attempt to simplify the communication topology to resolve the problem of transmitting redundant null messages due to low lookahead cycles. Other recent developments [10] have focused on incorporating knowledge about the LP into the synchronization algorithms. Cota and Sargent [7] focused on the skew in simulation time between different LPs by exploiting knowledge about the LPs and the topology of the interconnections.

Although earlier work has aimed to optimize the performance of the NMA by proposing the variants of the NMP [3, 4, 8, 10], it has not addressed reducing the exchange of null messages that is caused by improper selection of the parameters. This paper provides a mathematical model that approximates the optimal values of parameters in order to minimize the null message exchange across the LPs, while still maintaining a consistent parallelization.

The principal problem is that the NMA uses only the current simulation time of each LP and the lookahead value to predict the minimum time stamp of messages it can generate in the future. These messages with the minimum time stamp are then used to avoid deadlock. As a result, if one of the important parameters such as the lookahead value is chosen poorly, the performance will degrade significantly due to an excessive number of null messages. However, the prediction of minimum time stamps of messages can be improved by understanding the relationship between the time stamp and the lookahead value. The proposed mathematical model helps designers to choose appropriate values for lookahead to intelligently generate the null messages.

III. Mathematical Model

A conservative distributed simulation environment involves synchronization overhead which is added due to the distributed nature of simulation. With NMA, this overhead is mainly associated with the transmission of null messages. Therefore, when comparing the performance of a conservative distributed simulation environment using NMA with the performance of sequential execution, the message overhead can make a significant performance difference between the two approaches. Before developing the mathematical model, it is worth mentioning some of our key assumptions.

A. Key Assumptions

- For NMA, we assume that the value of lookahead may change during the execution of a lookahead period. This assumption makes it easier to analyze the variation in null message overhead with respect to different values of lookahead.
- We assume that each LP is initialized with a constant event arrival or job intensity rate (i.e., a uniform distribution of event-messages). This assumption will be used to analyze the relationship of event arrival rate with the lookahead values.
- For the frequency of message transmission, we assume that all messages are equally distributed among the LPs. Unless otherwise stated, we use the term all messages to refer to both null and event messages.
- Finally, we assume that a fixed size message is transmitted between LPs.

B. Definition of System Parameters

All model variables, along with their definition, are listed in Table I. Based on NMA, we assume that each LP maintains two clock times, one for each of its input and output neighbors. One is the minimum receiving time (MRT) for the input neighbor LP and the second is the minimum sending time (MST) for the output neighbor LP. The MRT contains the minimum simulation time the LP can receive an event from an input neighbor LP, where as the MST contains the minimum simulation time the LP might send a message to its output neighbor LP. These times play an important part in computing the timestamp for a null message. The performance (P) of a conservative distributed simulation environment mainly depends on the amount of computation required for processing an event per second. In addition, the event arrival rate (ρ) represents the number of events that occur per second (in practice, events occur per simulation second). Unlike performance, the parameter ρ mainly

depends on the model. Lookahead (L) is measured in seconds. As mentioned earlier, the value of L changes over the execution of lookahead period. Frequency of transmission (F_T) is the frequency of sending a message from one LP to another. T_{Null} represents the timestamp of a null message sent from one LP to another. T_{Null} is the sum of the current simulation time and the lookahead value. In other words, one may consider T_{Null} as an equivalent of MST for an LP (i.e., the value of T_{Null} is always updated by the sender LP to its current MST). This relationship can be expressed as: $T_{Null} = MRT + L$.

In order to measure the performance, it is imperative to consider one parameter that can compute simulation time advancement. As mentioned earlier, the performance is determined by the processing of a number of events per second whereas the event arrival rate is characterized by the number of events that occur per second. Taking these facts into account, the simulation time advancement can be defined as a ratio of performance to event arrival rate. This can be expressed mathematically as:

$$Simulation\ Time\ Advancement = STA = P/\rho \quad (1)$$

MRT represents the earliest time an LP can receive an event from its input neighbor. MRT is analogous to the clock associated with each incoming link of an LP. The value of MRT is updated through a null message coming from other LPs on the output link of a receiving LP. MST, on the other hand, represents the minimum time of an LP that may send a message to its output neighbor LP. A sender LP sends null messages to other LPs to avoid a deadlock situation. The timestamp for these null messages is determined by the current MST of that LP.

Each LP maintains a simulation time clock that indicates the timestamp of the most recent event processed by the LP. Any event scheduled by an LP must have a timestamp at least as large as the LP's simulation time clock when the event was scheduled [1]. This requirement is also referred as the local causality constraint. To strictly follow this requirement, a large number of null messages can be transmitted by LPs before the non null-messages can be processed. This large message overhead may degrade the performance of a conservative distributed simulation. It is, therefore, worth computing the ratio of null messages to the total messages transmitted among LPs. The null message ratio can be simply defined as the ratio of total number of null messages to total messages where total messages include both null and event messages. Mathematically, it can be expressed as follows:

TABLE I
System Parameter Definition

Parameter	Definition
P	Computation required for processing an event per second
ρ	Event arrival rate (events per second)
MRT	Minimum receiving time
MST	Minimum sending time
L	Lookahead
STA	Simulation time advancement
F_T	Frequency of transmission
T_{Null}	Timestamp of a null message
T_S	Current simulation of a LP
T_{Total}	Total simulation time in seconds

$$Null\ Message\ Ratio(NMR) = \frac{Total\ Number\ of\ Null\ Messages}{Total\ Messages} \quad (2)$$

IV. Optimization Of Critical Parameters Via The Proposed Mathematical Model

This section provides an analysis of the proposed mathematical model for a conservative distributed simulation environment. The numerical analysis provides several examples of parameters-optimization which are based on the mathematical equations and properties discussed above.

A. Impact of Null Messages on the Distributed Simulation Environment performance

Null messages are used to avoid deadlock in distributed simulation environment. As mentioned earlier, the computation of a null message involves the current simulation time of an LP and a lookahead value. The NMA performs well as a deadlock avoidance mechanism and gives good performance as long as the message overhead is not sufficiently high. The message overhead depends on the frequency of null message transmissions. Ignoring the fact that the transmission of null messages becomes essential when deadlock approaches in a distributed simulation environment, the value of lookahead also plays a critical role in increasing or decreasing the amount of null messages across the LPs. In other words, the value of lookahead is a design choice which should be appropriately chosen with respect to other system parameters.

For instance, consider the following simulation example that demonstrates the impact of lookahead on the overall performance of a system. Let a single LP process an event in 0.1 seconds and the rate at which events arrive be 0.25 events per second (i.e., events arrive for processing once every 4 seconds). In addition we compute event arrival rate by dividing the total number of event message to the simulation time. Mathematically, this can be expressed as:

$$\rho = Total\ numebr\ of\ event\ messages/T_{Total} \quad (3)$$

Using (1), one can easily approximate the STA. The value of STA can tell us how many null messages an LP needs to transmit to break a deadlock situation. For the above system parameters, the result would be P/ρ = 40. Thus this implies that 40 null messages are required to advance the simulation time to the next event. However, if we assume that the lookahead value is 10 times greater than the processing time value (i.e., $L = 0.1 \times 10 = 1\text{sec}$), then only approximately 4 null messages must be transmitted to avoid deadlock. In other words, a lookahead of one second yields an increase in *MRT* of one simulation second per step as shown in Fig. 1. Similarly, a lookahead value, which approaches the processing time, may significantly degrade the overall performance of a conservative distributed simulation environment. This degradation in performance is evident in Fig. 1. It can be concluded from the simulation results shown in Fig. 1 that a large number of null messages must be transmitted in order to advance the simulation time of each LP if the value of lookahead is quite small compared to the mean simulation time. Note that the purpose of this example is to demonstrate the behavior of null message algorithm for different values of lookahead.

B. *Characteristics of Event Arrival Rate and Lookahead*

Observing the simulation results of Fig. 1, one can compute an ideal value of lookahead that minimizes the null message overhead while at the same time maintains an acceptable performance for a conservative distributed simulation environment. It can be seen that the number of null messages approaches 1 as the value of lookahead approaches the inverse of the event arrival rate. Thus, this leads us to the following hypothesis that the ideal value of lookahead should be at least equal to or greater than the inverse of the event arrival rate. Mathematically, this relationship can be expressed as follows:

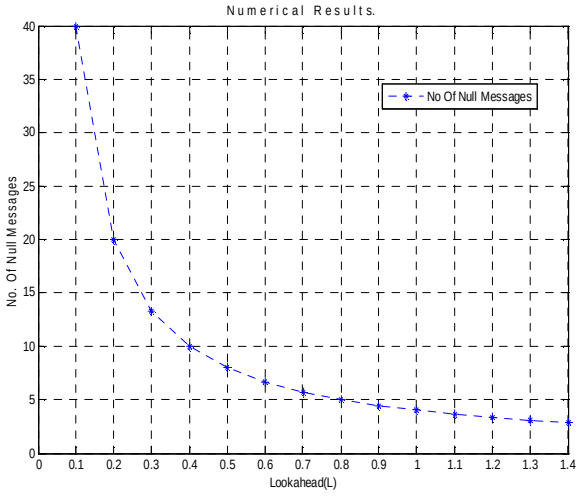

Fig.1. L versus number of null messages

$$Lookahead\,(L) \geq inverse\ of\ \rho \Rightarrow L\rho \geq 1 \quad \text{Property (1)}$$

For instance, if we assume that L is equal to 4 seconds and the event arrival rate is 0.25 events per second, then the result will be the transmission of only one null message and, thus improved performance.

C. *Null Message Ratio*

Another important relationship to be analyzed is the ratio of total number of null messages to the total messages per LP. Consider the following simulation example which shows the variations in null message overhead with respect to event arrival rate, processing time, and the lookahead values. Let the processing rate of a single LP be 50 event messages per second (i.e., *P*=50 event messages per second = 0.02 second per event) and let the event arrival rate be 10 events per second computed using (3) (i.e., $\rho = 0.1$ seconds between each event).

Using the lookahead value from the previous example (i.e., initially it is 10 times the processing time required by a single event), then the ratio of null messages to total messages can be computed using (2) as follows: When *L* = 10, *P* = 10 X 0.02 = 0.2 seconds, the number of null messages that need to transmitted is 50. We can interpret this numerical result as a lower bound for null message overhead as shown in Fig. 2. It should be noted that the value of L in this example is much less than the inverse of event arrival rate and this can be considered as one of the main reasons for the large number of null messages (a 50% null message ratio) and a lower bound of message overhead.

In other words, property (1) shows that the product of L and ρ should be greater than or equal to 1 in order to achieve better performance. Since for the above example, $STA = P/\rho = 0.2$ seconds per step (i.e., the value of MRT increases by 0.2 second in each transmission of a null message), the product of L and ρ is about 2, which conforms the characteristic of property (1). If the value of L linearly decreases during the execution of a lookahead period, the resultant performance will be degraded due to the increase in null message traffic as shown in both Table II and Fig. 2. The numerical results of Table II imply that in order to achieve good performance, the parameter L should not only satisfy property (1) but also remain stable (ideally growing with respect to simulation time).

D. *Processing Rate and Null Message Overhead*

In order to understand the relationship between processing rate and message overhead, consider the following example where we reduce the processing rate in the previous example by 50% (i.e., now a single LP can process 25 events per second). Furthermore, we use the same event arrival rate from the previous example using (3) (10 events per second). Given these changes, the new computation of null messages yields a

reduction in null message overhead by 50% as shown in Fig. 3. This is because of the increase in the lookahead value that increases the MRT by 0.4 seconds per null message transmission instead of 0.2. Fig. 3 illustrates that the product of lookahead and the event arrival rate has significantly increased due to the reduction in the processing power of an LP. Thus, the increase in $L\rho$ ensures a better performance for a conservative distributed simulation environment.

E. Effects of Multiple LPs on the Performance

This section presents a brief discussion on the use of multiple LPs and its corresponding effect on the null message overhead as well as on the overall system performance. Consider an example where four LPs are interacting together to perform tasks. If each LP processes 25 event messages, then four LP should process 25/4 messages (recall one of our assumptions about uniform event message distribution) where each of them has an equal computing power (i.e., one event processing in every 0.04 seconds). This implies that an average of 6.25 events per second will be processed by each LP. In addition, as we have already seen in the previous example that a singe LP processes one event message in 0.04 seconds, four LPs approximately accomplish the same job in 0.01 seconds. If we use the event arrival rate of 10 events per second, then the resultant STA will be approximately 0.1 seconds and consequently the required null message transmission will tend toward 100 messages. This numerical result demonstrates that the null message overhead grows as the number of LPs grows in the system. Mathematically this relationship can be expressed as:

$$(Null\ Message\ Overhead) \propto (Number\ of\ Neighbor\ LPs)$$

Property (2)

Where ' $\propto$ ' represents the sign of proportionality.

In this example, although the number of null messages is increased significantly, the required execution time for the same number of events is also reduced 4 times. This numerical result is achieved since we distribute the execution of events across four LPs that complete the required processing up to four times faster than if it were executed on a single LP.

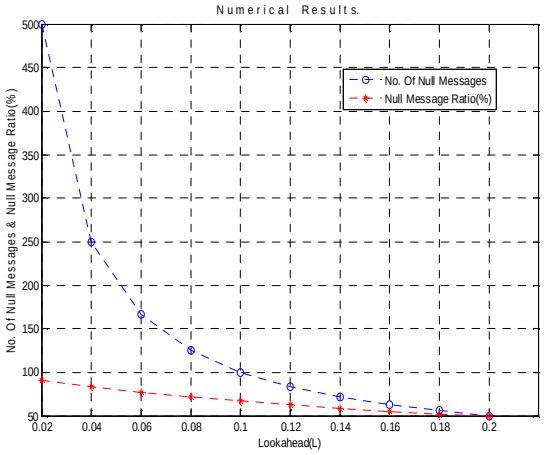

Fig.2. Frequency of transmission versus performance.

TABLE II
L Versus Null Messages and NMR (%)

Lookahead (L)	Null Messages	NMR (%)
0.020	500.000	90.900
0.040	250.000	83.330
0.060	166.660	76.920
0.080	125.000	71.420
0.120	83.330	62.400
0.160	62.500	55.000
0.180	55.550	52.000
2.00	50.000	50.000

F. Frequency of Transmission and the Computational Power of an LP

Another important relationship that we should analyze in our analysis is the variation in the computational power of an LP with respect to the frequency of transmission of null messages. If we increase the message transmission between two LPs, the result will be reduced computing power for each LP (i.e., the number of event-messages processed per second per LP will be reduced). This is due to the fact that an increase in the message transmission between LPs forces the LPs to spend more time dealing with these messages instead of processing the real event-messages. Thus, this leads us to the following mathematical hypothesis:

$$frequency\ of\ transmission \propto {}^{1}\!/_{computing\ power} \Rightarrow F_T \propto 1/P$$

Property (3)

Recalling (1), if we substitute the value of P, property (3) becomes,

$$F_T \propto 1/P \Rightarrow F_T \propto \frac{1}{STA}\rho \Leftrightarrow \frac{\rho STA}{P} \propto F_T$$ Property (4)

Or equivalently, property (4) can be written for performance such as:

$$P \propto \frac{\rho STA}{F_T}$$ Property (5)

If we assume that we have an average value for L (note that the value of L is considered to be poor if it is very small compared to STA), then it can be approximated as STA (i.e., $L \cong STA$ for an average case). Property (5) can now be written as:

$$P \propto \rho L/F_T$$ Property (6)

For instance, if we consider a large value of lookahead, for example, 10 seconds, and let the event arrival rate be 1000 events per second, then the number of events processed per

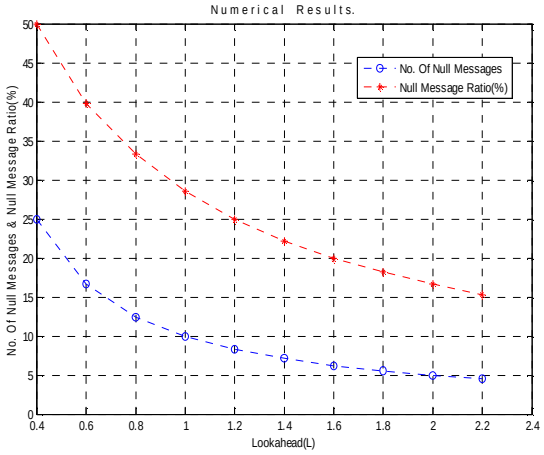

Fig.3. L versus null messages and NMR (%)

seconds for a range of F_T can be computed using property (6), as shown in Fig. 4.

G. *System Behavior with a Dormant LP*

Distributed simulation that uses the null message algorithm assumes that the simulation environment consists of a collection of LPs that communicate with each other by sending and receiving time stamped messages. Each LP in distributed simulation environment maintains local state information and a list of time stamped events that have been scheduled for the LP. This list of scheduled events contains both internal and external events. The internal and external scheduled events are handled by separate queues. In addition, the LP never blocks on the internal queue containing messages it schedules for itself. However, if any of the external queues that have the smallest clock (i.e., *MRT*) are empty, the LP blocks. Thus, this implies that the system behavior that has a dormant LP is only vulnerable to external events. In other words, the system remains stable and works smoothly if a single LP stops generating internal events as shown by the characteristics of the derived properties. However, the overall performance of the system may degrade slightly due to the passive state of an LP for internal events generation. On the other hand, in the presence of deadlock, the termination of external event generation by an LP can put the whole system in a non-continuous null message transmission cycle. Consequently, the whole system remains in the deadlock situation. This is because a finite cycle of null message transmission is required to avoid a deadlock situation. If this cycle does not go through, all the LPs, the deadlock situation will not be resolved. Finally, we believe that a single dormant LP does not have any severe effects on the performance if a system is working without a deadlock. But once a deadlock is reached, the dormant LP causes the cycle of null messages to stop.

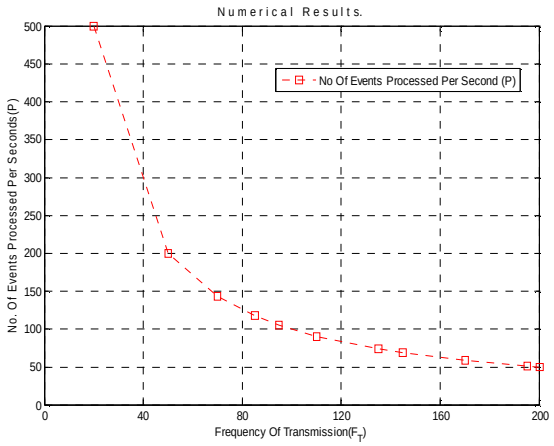

Fig.4. Frequency of transmission versus performance.

V. Conclusion

We have proposed a mathematical model to predict the optimum values of critical parameters that have great impact on the performance of NMA. The derived properties of the proposed mathematical model account for the cases when the NMA would send too many null messages. The proposed mathematical model provides a quick and practical way for simulation designers to predict whether a simulation model has potential to perform well under NMA in a given simulation environment by giving the approximate optimal values of the critical parameters. We have experimentally verified that if critical parameters, specifically the lookahead value, are chosen intelligently, we can limit the transmission of null messages among the LPs and consequently improve the performance of NMA in a distributed simulation environment. It is left to further studies to experimentally verify the implementation of the proposed mathematical model on other conservative synchronization algorithms.

References

[1] R. M. Fujimoto, "Distributed Simulation system," *preceding of the 2003 winter simulation conference*. College of Computing, Georgia Institute of Technology, Atlanta.

[2] Y.M. Teo, Y.K. Ng and B.S.S. Onggo, "Conservative Simulation using Distributed Shared Memory," *Proceedings of the 16th Workshop on Parallel and Distributed Simulation (PADS-02), IEEE Computer Society*, 2002.

[3] B. R. Preiss, W. M. Loucks, J. D. MacIntyre, J. A. Field, "Null Message Cancellation in Conservative Distributed Simulation," *Distributed Simulation 91 Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, 1991.

[4] W. L. Bain, and D. S. Scott, "An Algorithm for Time Synchronization in Distributed Discrete Event Simulation", *Proceedings of the SCS Multiconference on Distributed Simulation*, 19, 3 (February), pp. 30-33, 1988.

[5] N. J. Davis, D. L. Mannix, W. H. Shaw, and Hartrum, T. C., ''Distributed Discrete-Event Simulation using Null Message Algorithms on Hypercube Architectures,'' *Journal of Parallel and Distributed Computing*, Vol. 8, No. 4, pp. 349-357, April 1990.

[6] K. M. Chandy and J. Misra, "Distributed Simulation: A case study in design and verification of distributed programs", *IEEE Transactions on Software Engineering*, SE-5:5, pp. 440-452, 1979.

[7] B. A. Cota and R. G. Sargent, ''An Algorithm for Parallel Discrete Event Simulation using Common Memory,'' *Proc. 22nd Ann. Simulation Symp.*, pp. 23-31, March 1989.

[8] J. K. Peacock, J. W. Wong, and E. Manning, ''Synchronization of Distributed Simulation using Broadcast Algorithms,'' *Computer Networks*, Vol. 4, pp. 3-10, 1980.

[9] L. A. Belfore, S. Mazumdar, and S. S. Rizvi et al., "Integrating the joint operation feasibility tool with JFAST," *Proceedings of the Fall 2006 Simulation Interoperability Workshop*, Orlando Fl, September 10-15 2006.

[10] D. M. Nicol and P. F. Reynolds, ‘‘Problem Oriented Protocol Design,’’ *Proc. 1984 Winter Simulation Conf.*, pp. 471-474, Nov. 1984.
[11] J. K. Peacock, J. W. Wong, and E. Manning, ‘‘A Distributed Approach to Queuing Network Simulation,’’ *Proc. 1979 Winter Simulation Conf.*, pp. 39 9-406, Dec. 1979.