A New Approach to Avoid Deadlock in Parallel Discrete Event Simulation (PDES) System

Hemen Patel, Syed S. Rizvi, and Khaled M. Elleithy
Department of Computer Science, University of Bridgeport, Bridgeport 06604, USA
{hemenp, srizvi, elleithy}@bridgeport.edu
Tel: (203) 576-4703
Fax: (203) 576-4766

Keywords: Conservative distributed simulation, Null messages, logical process with null message, look ahead.

Abstract
A conservative distributed simulation requires all logical processes (LPs) to follow the causality constraint requirement. This implies that all event-messages are processed in strictly timestamp order. Apart from the timestamp of each event generated by LPs, synchronization between all LPs is the second most important requirements. Finally, there must not be a deadlock in the distributed environment. A deadlock may occur when there is no events present in the queue of LP. In such case, to avoid deadlock, Chandy-Misra-Bryant presented an algorithm called *Null Message Algorithm* (NMA) [3]. These null messages are passed as an event-message to other LPs and it stored in one of queues of LPs. This null message indicates that till the time stamp of that null message, all other events in the queue which have lesser time stamp than null message's time stamp are safe to process. It means that there won't be any arrival of any events from that logical process until current simulation time is equal to the time stamp of the null message. With the time stamp of the null message, a Lookahead value is added to the time stamp of that null message. This Lookahead value can be measure on certain kind of parameters such as delay to transmit a message, propagation delay, etc. therefore, calculating value of Lookahead is the most important part as Lookahead value affects the performance of the conservative distributed event simulation. Proper value of Lookahead can reduce the number of null messages which decreases the traffic of the network. In this paper, we demonstrate some calculation on the Lookahead which shows the performance of the distributed event simulation.

1. INTRODUCTION

The term conservative refers to the causality constraint. In such kind of distributed simulation, events which are processed already can't be rolled back. It shows that all events presented in the queue of LP must be in sorted time stamp order or an arrived event's time stamp must be greater than the biggest time stamp event presented in the logical process' queue. All event messages violating causality constraint are rejected by the receiving logical process. In the simulation, each logical process has logically different queues for the rest of logical process present in the distributed environment. All event messages received by the receiving logical process are stored in that logical queue which is meant for the sending logical process. If one of those queue becomes empty then simulation stops and it is said that deadlock occurred in the network.

To avoid deadlock, we transmit null messages at certain simulation time or at certain occurrences of events. Hence, now it is required to avoid deadlock, we have to send null message indicating the safe time to process events. Chandy-Misra-Bryant presented an algorithm called *Null Message Algorithm* (NMA). [3] This algorithm sends either null message or event message at each simulation time completion. In NMA, value of the Lookahead is added to the time stamp of the null message. This value of Lookahead is critical as it depends on the transmission time and propagation time of the distributed environment. We have to choose value for the Lookahead carefully in order to avoid deadlock and reduce the number of the null messages. In NMA, value for the Lookahead is chosen inappropriate. This is one of the main problems present in NMA. Therefore, numbers of null messages are increased to more. In this paper, we present a model that will avoid deadlock and also calculate appropriate value for the Lookahead.

2. RELATED WORK

There are two approaches in the distributed environment to process events. One is the conservative approach and the second one is optimistic approach. Both approaches have their own advantages and disadvantages. An optimistic approach has roll back mechanism which is not present in the conservative approach. Likewise, conservative approach has causality constraint which does not exist in the optimistic approach. An approach is different but goal of both approaches are same to simulate processes in a distributed environment [4].

Many algorithms are there in conservative algorithm for the distributed environment simulation. Chandy-Misra-Bryant presents an algorithm based on null message which itself is called null message algorithm.[3] One drawback of that algorithm is that the timestamp of the null message is chosen randomly due to which there are null messages in the network when deadlock is occurred in the network. Bain and Scott try to simplify network topology to re solve problem of null

messages overhead. [1] All these works are done to optimize the performance of the conservative distributed event simulations.

3. PROPOSED MODEL

When conservative distributed algorithm is there, there are few parameters where everyone has to take care of them. Those parameters are like synchronization, deadlock, propagation delay, transmission delay, latency, etc. Evaluating the performance of a distributed simulation algorithm, we have to consider all those parameters as well as overhead of null messages. We have made some assumption in order to make model clearly explain and defined analytically. Those assumptions are described in the next paragraph of this part. I have defined certain terms which are mentioned in the table 1 of this paper.

Fig. 1 represents an internal architecture of logical processes. For the sake of simplicity, we assume that the simulation system consists of 3 LPs that can directly communicate with each other. However, in practice, the LP may reach to an arbitrary value of N. An LP can not only schedule an event for the neighboring LPs (remote events generation) but can also schedule an event for itself (local event generation). Since LPs are connected with each other using a mesh topology, each LP must have n-1 number of logical queues. In Fig.1 the total numbers of LPs are 3, it implies that each LP contains 3-1 that is 2 queues in it. When LP1 schedule event for LP2, it sends a message to LP1 with the time stamp. That message is received by LP2 and it is saved in the logical queue for the LP1 which is present in the LP2. Likewise, scheduling of events takes place and simulation cycles finishes. Each message must have causality constraint, it also applies here. We assumed that Lookahead value is calculated based on transmission time, and the propagation time. Therefore, the value of the Lookahead is directly propositional to the value of transmission time and value of propagation time. Therefore the equation for the

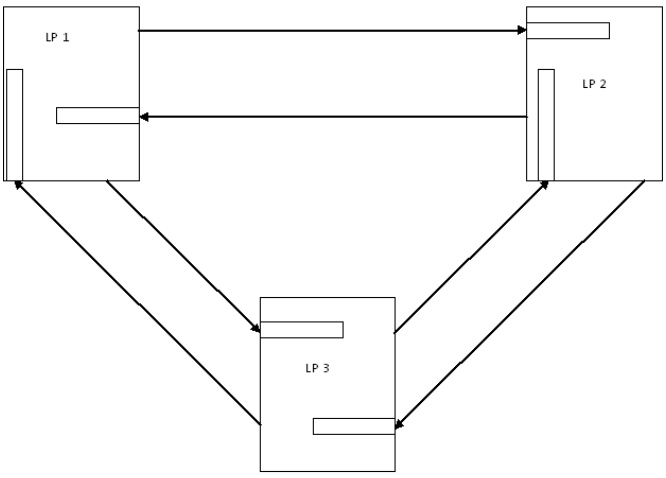

Fig1. Logical Processes architecture

TABLE I
SYSTEM PARAMETERS AND DEFINITIONS

L	Look Ahead
$T_{transmission}$	Transmission time : Time taken to generate (send) message
$T_{propagation}$	Propagation time : Time taken to travel from source LP to destination LP by a message
$Total_{left\ duration}$	Total time of duration of each event present in the logical queue
T_{event}	Simulation time that an event take to execute
T_{null}	Time stamp of Null Message

Lookahead is as below:

$$L = T_{transmission} + T_{propagation} \quad (1)$$

Transmission time and propagation time is dependent on the distributed network. Therefore, fix value for them can't be taken generally for the all distributed network. Furthermore, We calculate value of L for the each two neighboring LPs which are used for calculating timestamp of the null messages. Also We have assumed that depending upon topology of the network, value of the Transmission Time and Propagation Time is calculated dynamically. Therefore, there will be no furthermore derivation for the Lookahead value.

One more assumption is that, every event message present in the queue has predefined simulation duration in the unit of simulation time. Hence, we can able to know that how much simulation time that all events present in the queue takes.

Therefore, we can calculate that time of duration by the following equation:

$$Total_{left\ duration} = \sum T_{event} \quad (2)$$

This $Total_{left\ duration}$ is calculated for each logical queue present in the LP. At every simulation time; value for $Total_{left\ duration}$ is updated. When it is found that $Total_{left\ duration}$ is twice of the look ahead value, at that simulation time, a signal is sent to the neighboring LP that queue is going to empty. After receiving signal from empty queue one's LP, receiving LP simply sends a null message to the empty queue one's LP. By the simulation time that null message reaches to the empty queue one's LP, all remaining events which was before at the time of sending signal, has been processed by empty queue one's LP. And null message will be processed at the next simulation time.

Hence, we can reduce wait time or ideal time for the LPs. Let's make it more clearly with an example. Suppose LP1 has a logical queue for LP2 in itself. When value of $Total_{left\ duration}$ for the queue LP2 in LP1 is equal or less than value of Look Ahead L, at that simulation time a signal is sent from LP1 to LP2 indicating that a queue is going to become empty. At the receiving end, LP2 sends null message with time stamp $T_{null} + L$.

To complete this whole process, it takes almost twice of L simulation time. It implies that after 2L simulation time; null message reaches to the LP1. And at the same time, LP1 has no events left in queue. Therefore, LP1 process that null message. Hence, wait time for LP1 is reduced to nearly 0 simulation time. All these steps are shown in the Figure 2, 3, 4 and 5. It clear shows that idle time for LP1 is zero.

Algorithm

```
While (simulation is not over)
{
        If(Total left duration <= L)
        {
                Send Signal to LP;
        }
                        .
                        .
        Process events which are in Queue;
                        .
                        .
}
```

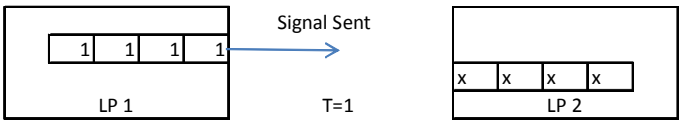

Fig. 8. Scenario at T=1; L=4

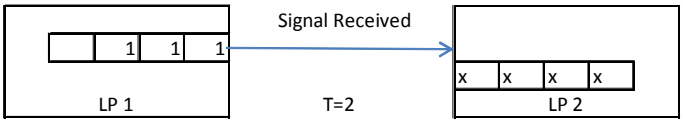

Fig. 8. Scenario at T=2; L=4

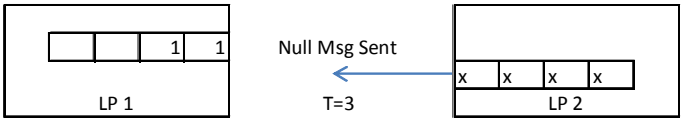

Fig. 8. Scenario T=3; L=4

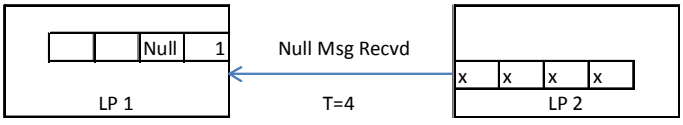

Fig. 8. Scenario t=4; L=4

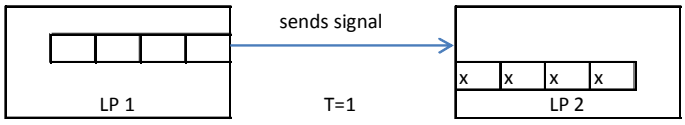

Fig. 4. Original Algorithm Scenario T=1; L=4

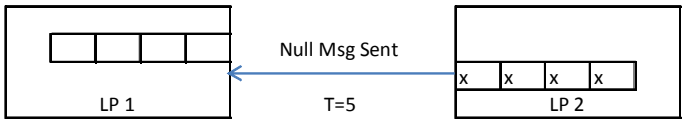

Fig. 4. Original Algorithm Scenario T=5; L=4

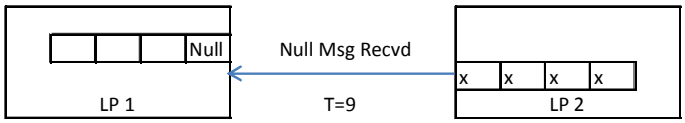

Fig. 4. Original Algorithm Scenario T=9; L=4

Given pseudo code is modified version of the original null message algorithm. While comparing original modified null message algorithm with original algorithm, we can see the idle time for LP in the original algorithm is more than idle time for LP in the modified algorithm. Figure 6, 7, 8 show original algorithm idle time. In the case of original algorithm LP's idle time is 9 for the given example. From this example, it is clear that original algorithm has more idle time for LPs than my proposed model which is modified original algorithm.

4. PERFORMANCE EVALUATION

For the sake of performance evaluation and experimental verifications, we present two cases that incorporate a verity of different parameters such as Lookahead, frequency of transitions, and non uniform distribution of Lookahead among the LPs.

4.1. A Single LP goes dormant

In such case, if dormant LP can send Null Messages to other LPs then my proposed model works fine. As dormant LP may receive event scheduled by other LPs, it will become non-dormant. And if dormant LP can't send Null Messages to other LPs then my proposed model can't work. As dormant LP can't send Null Message and deadlock is still remain in the distributed environment.

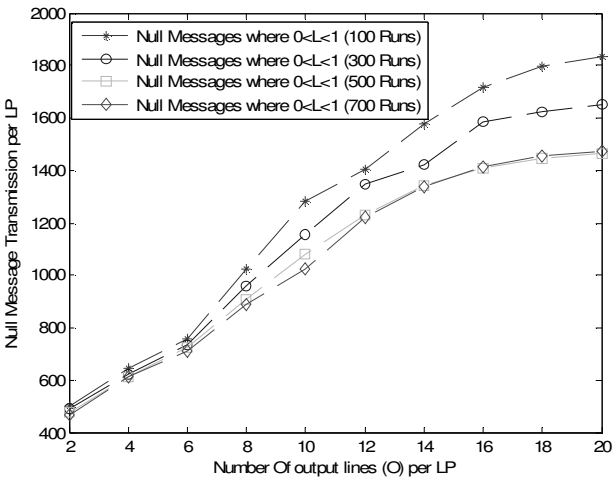

Fig.9. Multiple Output Lines per LP with Non-Uniform Distribution of Lookahead versus null message transmission

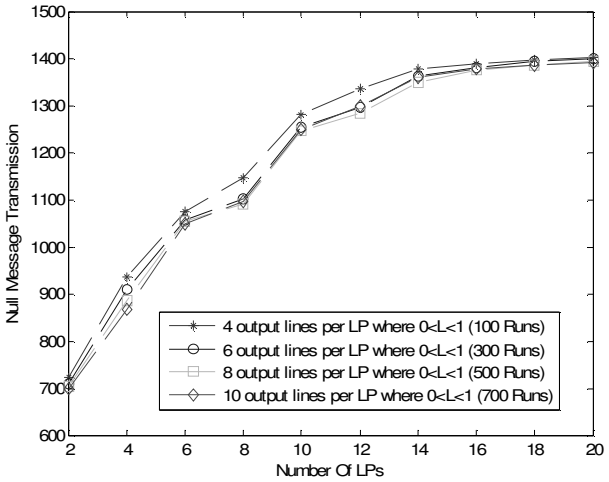

Fig.10. Multiple LPs and Multiple Fixed Output Lines with non-uniform distribution of Lookahead value versus null message transmission

4.2. Multiple Output Lines per LP with Non-Uniform Distribution of Lookahead

For this simulation, we assume that we have single LP that has *O* number of output lines where each output line of an LP can have different value of Lookahead (*L*). Fig.9 shows the null message transmission with the following simulation parameters: simulation time = 500 sec, *L* is non-uniformly distributed per output lines (***O***) of an LP. The numbers of output lines are varied from 1 to 10. Also, it should be noted that the value of Lookahead is chosen randomly within the range of 0 to 1 and assigned to each output line at run time. This random selection may control the generation of unnecessary null messages as long as the value is chosen appropriately.

4.3. Multiple LPs with Multiple Fixed Output Lines with Different Lookahead Value

For this simulation, we assume that we have multiple LPs that can have fixed number of output lines where each line of an LP can have different value of Lookahead (*L*). Fig.10 shows the null message transmission with the following simulation parameters: simulation time = 500 sec, *L* is non-uniformly distributed per output lines (***O***). The numbers of LPs are varied from 1 to 20 as show in Fig.10. Also, it should be noted that the value of ***m*** and ***O*** are both varying quantity for this particular scenario. This random selection may control the generation of unnecessary null messages as long as the values are chosen appropriately. In harmony with our expectation, the number of null messages increases due to an increase in number of LPs. However, this increase in null messages is limited and controlled due to random behavior of Lookahead. This can also be considered as irregular networks due to the non uniform distribution.

5. CONCLUSION

From my modified algorithm, we can reduce idle time for the LPs. But number of messages or signals transferred is increased. In other words, line utilization is increased. Idle time for the LPs is most likely equal to zero but there may be effect of distributed environment to the idle time. Though idle time won't be the more than Lookahead value of the LP for the proposed model, which is again less than the original algorithm because for the original algorithm idle time is almost 2 times of the Lookahead value.

REFERENCES

[1] 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.

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

[3] 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.

[4] Syed S. Rizvi, Khaled. M. Elleithy and Aasia Riasat, "Optimizing the Performance of NMA for Conservative Synchronization in Parallel and Discrete Event Simulation"