Int j simul model 12 (2013) 2, 69-81
ISSN 1726-4529 Original scientific paper

A LOGICAL PROCESS SIMULATION MODEL FOR CONSERVATIVE DISTRIBUTED SIMULATION SYSTEMS

Rizvi, S. S.
Department of Information Sciences and Technology
Pennsylvania State University, USA
E-Mail: srizvi@psu.edu

Abstract
This paper presents a new logical process (LP) simulation model for distributed simulation systems where Null Message Algorithm (NMA) is used as an underlying time management algorithm (TMA) to provide synchronization among LPs. To extend the proposed simulation model for *n* number of LPs, this paper provides a detailed overview of the internal architecture of each LP and its coordination with the other LPs through sub-system components and models such as communication interface and simulation executive. The proposed architecture of LP simulation model describes the proper sequence of coordination that need to be done among LPs though different subsystem components and models to achieve synchronization. To execute the proposed LP simulation model for different set of parameters, a queuing network model is used. Experiments will be performed to verify the accuracy of the proposed simulation model using the pre-derived mathematical equations. Our numerical and simulation results can be used to observe the exchange of null messages and overhead indices.
(Received in April 2012, accepted in January 2013. This paper was with the author 3 months for 3 revisions.)

Key Words: Discrete-Event Simulation, Distributed Systems, Large-Scale Networks, Queuing Model, Parallel Computing

1. INTRODUCTION

This paper presents a new logical process (LP) simulation model for a conservative distributed simulation that uses Null Message Algorithm (NMA) as an underlying synchronization mechanism to avoid deadlock or recover from it if it occurs. The proposed simulation model is based on the mathematical equations derived in [1] for quantifying the null message transmission. The term distributed refers to distributing the execution of a single run of a simulation program across multiple processors [2]. Therefore, the main motivation for using the distributed simulation is to reduce the overall simulation execution time and to improve the event processing rate. However, one of the requirements of distributed simulation is the need of proper synchronization of multiple event-messages that may be geographically distributed and concurrently executed. If not properly handled, synchronization problems may degrade the performance of a distributed simulation environment [3]. Time management algorithms (TMAs) are therefore required to ensure that the execution of the distributed simulation is properly synchronized. Two main classes of TMAs are conservative and optimistic. This paper focuses on the performance issues related to the conservative distributed simulation for relatively large-scale networks.

In conservative distributed simulation, event-messages are required to execute by LPs such that local causality constraint requirement must not be violated [4]. To do that, each LP must know which event-messages are safe to process. Since all logical processes (LPs) do not have a consistent view of the state of the entire system, LPs must exchange timing information to synchronize with each other. If LPs do not properly synchronize, the distributed simulation system may go into a deadlock situation. A NMA is used specifically for deadlock avoidance originally developed by Chandy, Misra, and Bryant [5]. If simulation goes into a deadlock, an excessive numbers of null messages may need to be transmitted between LPs to resume the normal processing of events. This transmission overhead may increase significantly if other important parameters such as Lookahead and frequency of transmission are not chosen appropriately by simulation designers. This situation gets more severe when conservative simulation is used to perform a detailed logistics simulation in a distributed environment to simulate a huge amount of data [6].

The choice of what optimal values a simulation designer should use for variety of parameters becomes more difficult when we have relatively a large-scale network containing millions of end hosts and routers. This clearly demands a persistent simulation model with some well driven underlying mathematical equations that can be used by simulation designers to predict the behaviour of conservative simulation for large-scale networks. Although, previous studies have evaluated the performance of conservative synchronization algorithms for transmission overhead (e.g., see [1]), most of these studies have conducted using commercial network simulators without using a simulation model. It is, therefore, hard to generalize these research results for *n* number of LPs and could not be used to realize the modelling and simulation of a large-scale network.

In this paper, we present a new LP simulation model for distributed simulation systems where NMA is used as an underlying TMA to provide synchronization among LPs by exchanging null messages. The proposed simulation model is based on the mathematical equations derived in [1] for quantifying the synchronization messages. Our proposed LP simulation model provides a detailed view of sub system components and models such as communication interface, simulation engine, and application to describe the proper order in which coordination need to be done between participating LPs to safely execute event-messages with a minimum transmission overhead. To support the simulation model, mathematical equations will be derived to generalize the hypothesis for *n* number of LPs. To execute the proposed LP simulation model for different set of parameters, a queuing network model is used. Our numerical and simulation results can be used to observe the exchange of null messages and overhead indices. These results can then be further used to predict the behaviour and performance of different simulation models under different set of parameters with varying values for parallel and distributed systems.

The rest of the paper is organized as follows: Section 2 provides an overview of the conservative protocols, focusing on the Null Message Protocol (NMP) and its related problems. Section 3 presents the proposed LP simulation model with a comprehensive discussion on various components and their implementations. Numerical and simulation results of the proposed system are presented in Section 4. Finally, we conclude the paper in Section 5.

2. EVENT SYNCHRONIZATION THROUGH CONSERVATIVE TMA

The need for message exchange, interoperability, reduced latency, and parallel execution are motivating principles behind dividing a single large simulation into number of events and distributing them into distant nodes in a distrusted network [7]. Each of the above mentioned principles clearly require synchronization to be the least common denominator. In the contents of distributed discrete-event simulation (DES), synchronization refers to the coordination of events that are running simultaneously on different distant nodes. Simultaneous execution of events increases the synchronization requirements when the events that must synchronize can run concurrently on different machines in a relatively large network. TMA have now become the lingua franca to provide synchronization between discrete-events in distributed networks.

In general, synchronization protocols can be categorized into two different families: conservative and optimistic. Conservative synchronization is performed using the NMA and its variant techniques. Some recent advancements and interesting results in the optimistic algorithms can be found in [7-10]. Conservative protocols fundamentally maintain causality in event execution by strictly disallowing the processing of events out of time-stamp order [4]. Some recent research on conservative algorithms in DES can be found in [11-14]. An effort to combine conservative and optimistic synchronization algorithms on a common layered architecture framework is proposed in [15].

Examples of conservative mechanisms include Chandy, Misra and Byrant's NMP [5], and Peacock, Manning, and Wong [16] avoided deadlock through null messages. Some intelligent approaches to null message generation include generation on demand [17], and generation after a time-out [18]. Some earlier research on DES has focused on variants of NMP, with the objective of reducing the high null message overhead. For instance, Bain and Scott [19] attempt to simplify the communication topology to resolve the problem of transmitting redundant null messages due to low Lookahead cycles. Cota and Sargent [20] focused on the skew in simulation time between different LPs by exploiting knowledge about the LPs and the topology of the interconnections.

3. A LOGICAL PROCESS SIMULATION MODEL

In parallel discrete-event simulation (PDES), the simulated system is partitioned into a set of subsystems that are simulated by a set of processes that communicate by sending and receiving time-stamped messages [2]. We assume that the simulation system consists of *n* number of LPs which are devised to execute events that may occur at discrete points in time. In addition, we also assume that a total of *n* number of event-messages (E_{MSG}) are generated by *n* number of LPs. The total event-messages are further divided into two subsets represented as follows:

$$\langle E_L, E_R \rangle \subseteq E_{MSG} \qquad (1)$$

where E_L and E_R represent the local and remote event-messages respectively, generated by LPs.

Multiple simulation applications (*SAs*) communicate with each other to execute both local and remote events-messages concurrently. In general, each *SA* typically executes events asynchronously in a non-decreasing time stamp order by strictly following the local causality constraint requirement. Specifically, a group of events that are processed by an LP represents a sub-system model (*SSM*) of the total simulation with respect to the simulation time (T_s). The complete architecture of simulation is shown in Fig. 1. A communication system (*CS*) enables LPs to exchange data with each other during the processing of both local and remote event-messages. In addition, LPs are synchronized via the *CS*. The synchronization within each *SA* is achieved via the communication interface (*CI)*. Each *CI* can exchange event's time stamp information with the other *CI* through the *CS*.

Within each simulation application (say *SAi*), we have one communication interface (*CIi*) that operates on top of the corresponding simulation executive (*SEi*). The *CI* acts as a global controller within the *SA* to prevent the violation of the local causality constraint requirement. This is typically done by each *CI* within the *SA* by simply blocking an *SE* to processes unsafe event-messages. To determine which events are saved to execute by an LP, each *CI* needs to periodically exchange messages with the other *CIs* via the *CS*. Thus, the overall synchronization is achieved with the cooperation of all participating *CIs*.

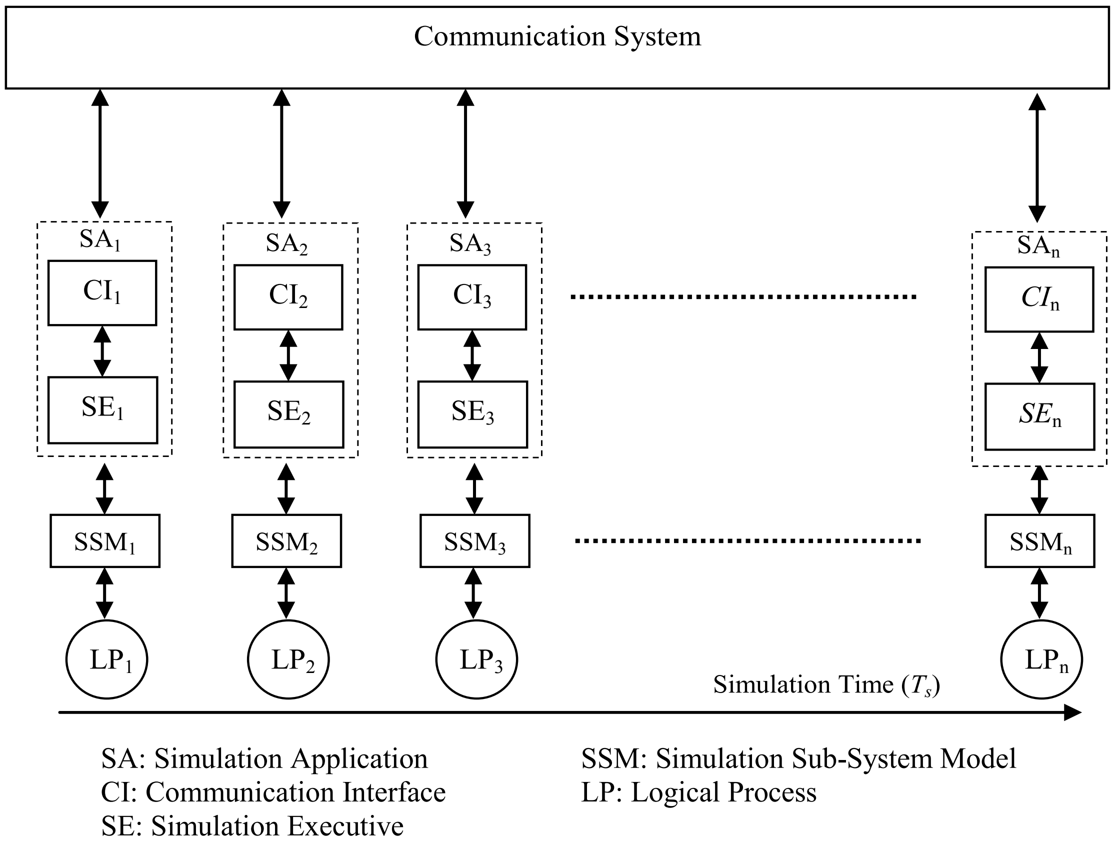

Figure 1: Architecture of LP simulation model consisting of simulation application, communication interface, and simulation executive.

Each simulation executive (*SEi*) operates in an event-driven model on a corresponding simulation sub-system model (*SSMi*) to execute events. On the other hand, each LP (say *LPi*) occupies a corresponding sub-system model (say *SSMi*) to generate local events and schedule remote events for the neighbouring LPs. If local events are generated by *LPi* denoted as E_{Li}, then their execution must belong to *SEi* which operates on *LPi* within a single *SA*. This relationship can be expressed as:

$$E_{Li} \in \left(SE_i, LP_i\right) \text{ where } E_{Li} \in E_L \tag{2}$$

Similarly, remote events generated and scheduled by LPi (say E_{Ri}) must belong to one or more neighbouring LPs. This can be written as:

$$E_{Ri} \in \left(SE_j, LP_j\right) \tag{3}$$

where $j \neq i, j = 1 \ldots n$ and $E_{Ri} \in E_R$.

As mentioned in our system model, we assume that the simulation system consists of *n* number of LP$_S$ where each LP maintains one input FIFO queue per neighbouring LP that stores the corresponding incoming event-messages. Similarly, each LP also maintains one output queue per neighbouring LP to transmit the remote event-messages to the other LPs via *CI*. Both input and output FIFO queues referred as IQ_i and OQ_i, respectively for a neighbouring LP_i. Our system model is based on the proposed internal architecture as shown in Fig. 2.

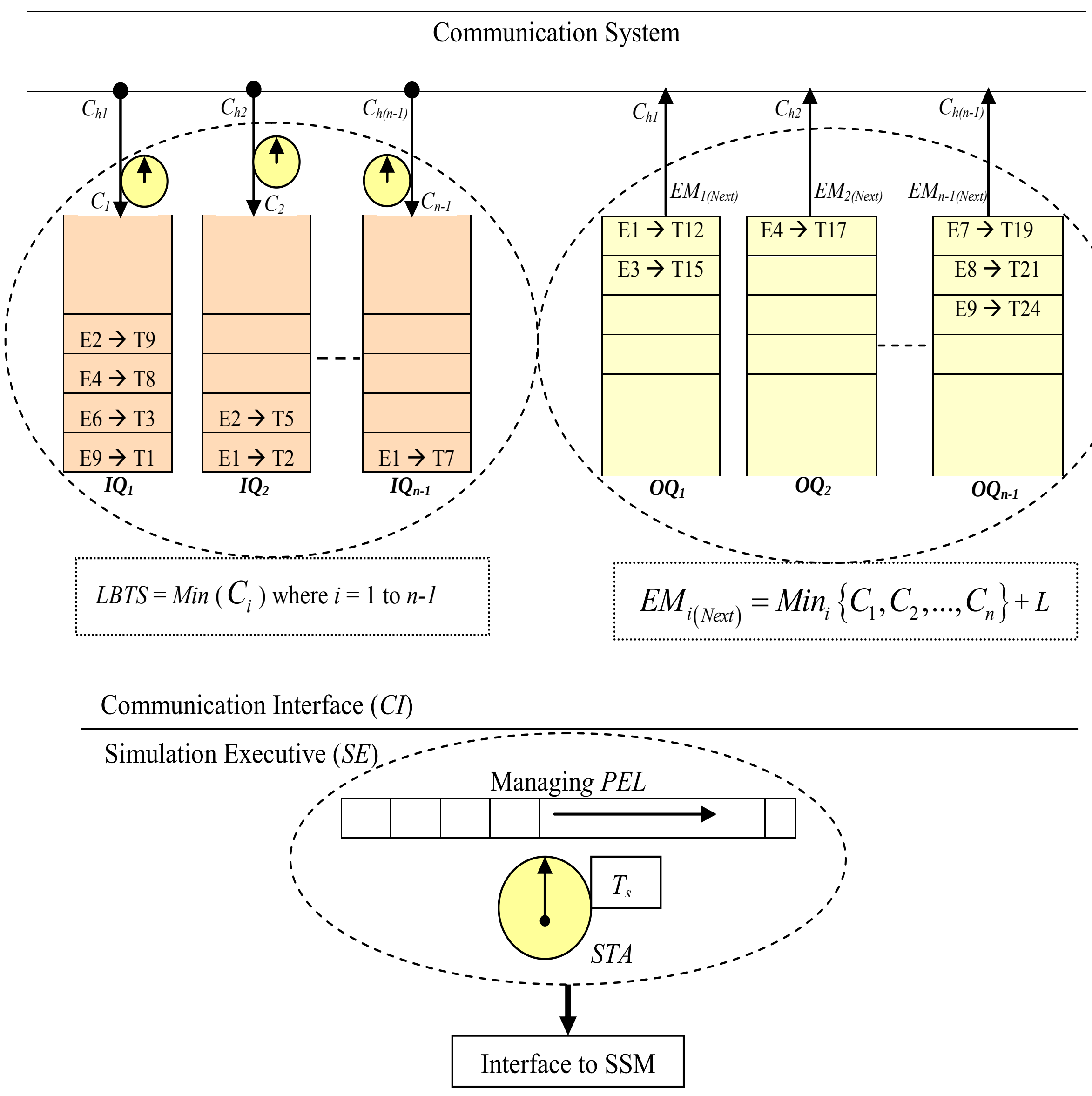

Figure 2: System model based on the proposed internal architecture of an LP.

We assume that a total of n number of channels CH_{Total} exist within each LP that include both incoming and outgoing channels. This relationship can be expressed as follows:

$$\langle IC_h, OC_h \rangle \subseteq CH_{Total} \tag{4}$$

where IC_h and OC_h in (4) represent the incoming and outgoing channels respectively, for each LP in the simulation system as shown in Fig. 2.

Based on the above expression, each LP (say LP_i) is connected to other LP (say LP_j) via a communication channel ($C_{h(i,j)}$) where $C_{h(i,j)} \in IC_H$. An input clock is associated with each end of the communication channel that holds the time stamp of the last event-message received on each link. For instance, a clock maintained by LP_i for the neighbouring LP_j is referred as C_i that holds a copy of a time stamp of event-message at the head of a FIFO queue where initially the value of C_i is set to zero. The maintenance of input clock C_i for each communication channel C_h between LP_i and LP_j can be expressed as: $C_i \in C_{h(i,j)}$. In addition, the value of C_i will be used to determine the lowest event among all the FIFO queues maintained by LP_i for all the neighbouring LPs such as:

$$EM_{i(Next)} = Min_i \{C_1, C_2, ..., C_n\} \qquad (5)$$

where $EM_{i(Next)}$ is the time horizon up until which LBTS is allowed IQ_i to progress by simulating internal or external events, since no events can arrive with a time stamp smaller than $EM_{i(Next)}$.

Once $EM_{i(Next)}$ is determined, the lowest event will be extracted from the respective IQ_i of LP_i and sent to *SEi* for the execution. Similarly, each LP (say LP_i) maintains one OQ_i per output communication channel for *n*-1 number of neighbouring LPs that holds the remote event-messages scheduled by LP_i. Each communication interface (for instance CI_i of LP_i) extracts the remote event-message from one of the OQ_i with respect to the minimum sending time and sends to one of the neighbouring LPs.

4. PERFORMANCE ANALYSIS

Experiments have been performed to verify the proposed simulation model. Our numerical and simulation results can be used to observe null message transmission and overhead indices. These results can then be used to predict behaviour and performance for different simulation models under different set of parameters with varying values. For the sake of performance analysis, we simulate 5 different cases. The whole system is modelled in C++.

For our study, we used the queuing network model described in [21]. Numerical analysis is performed based on the set of equations derived in [1] for null message quantification. The event-message traffic can be controlled by controlling the number of messages generated by each of the LPs (λ). The LPs trigger themselves at random intervals of time to generate new event-messages that include both local and remote events where we consider the random interval as a design parameter. An event-message can traverse multiple hops to reach the destination LP. However, if an event-message exceeds the maximum number of hops (H_{MAX}), it will be discarded by one of the receiving LPs. This feature is useful to avoid looping problems of event-messages within the network.

Two types of propagation delays need to be specified namely, LAN delay and Internet delay. LAN delay represents the delay between the LP and a router whereas the Internet delay represents the delay between two routers within the subnet. In addition to delays, we also specify the service time (S_T) and the buffer length (B_L) for each router. For the ease of implementation, a static routing algorithm is considered which will be evaluated once the network topology is known. The complete network topology along with different parameters is presented in Fig. 3. The same network topology (see Fig. 3) can be replicated to extend the model for *n* number of subnets where each subnet consists of several intermediate routers, buffers, and a group of LPs. For the sake of performance evaluation, we use TCP traffic which is originated by the end hosts (i.e., a set of LPs in each group) with respect to the message generation rate. Specifically, each LP can generate a message of 100 K bytes from each server. The complete details of parameters are specified in each of the cases that we evaluate.

4.1 Case I: Multiple output lines per LP

Fig. 4 shows the null message transmission with the following simulation parameters: simulation time is 500 s, *L* is uniformly distributed per output line (*O*). The number of output line may vary from 0 to 8 for all cases as shown in Fig. 4. The numerical results for Case I are presented in Table I. Both numerical and simulation results present a comparison of null message transmission per LP versus multiple output lines.

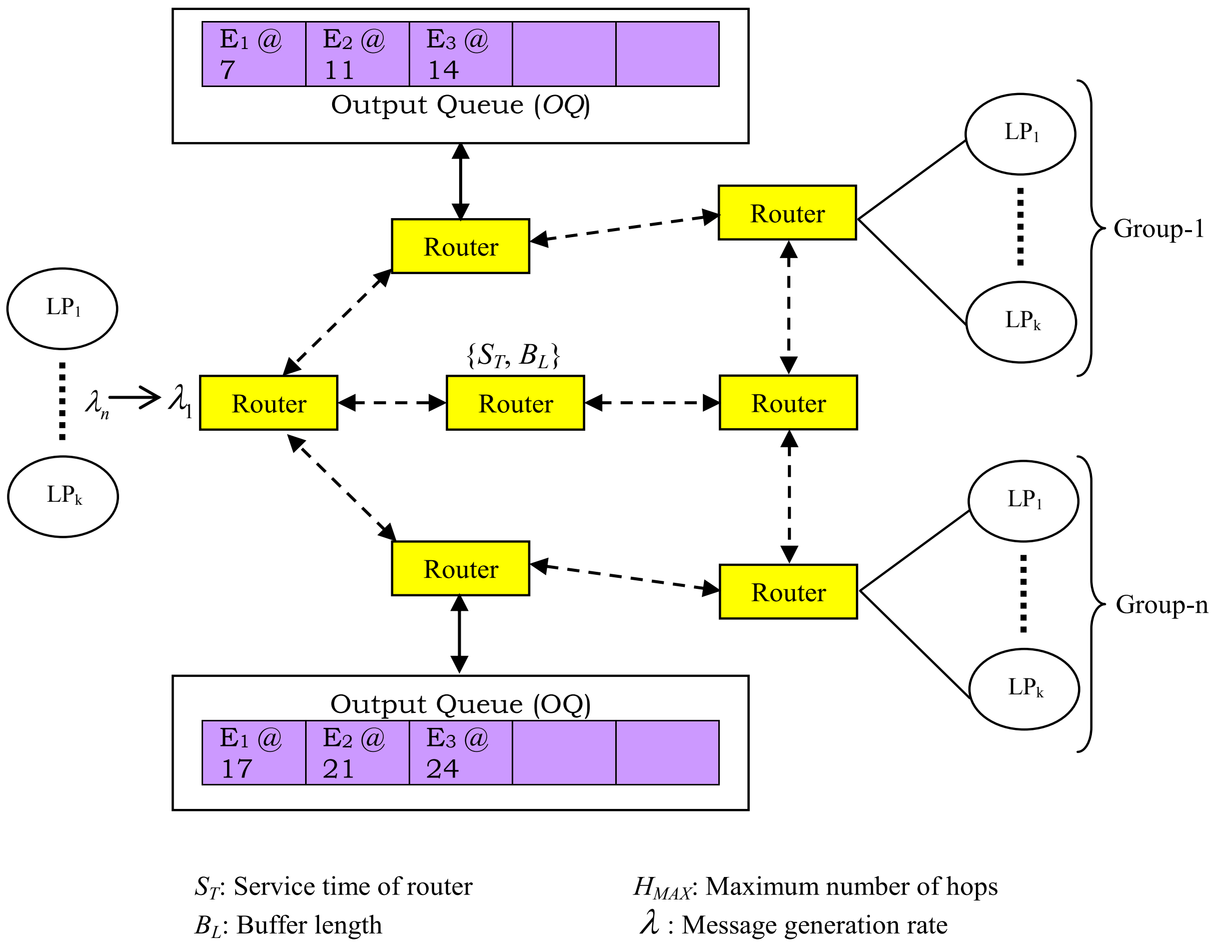

Figure 3: Network topology with varying parameters. LPs are divided into *n* Groups. Output queues carry event-messages with time stamps.

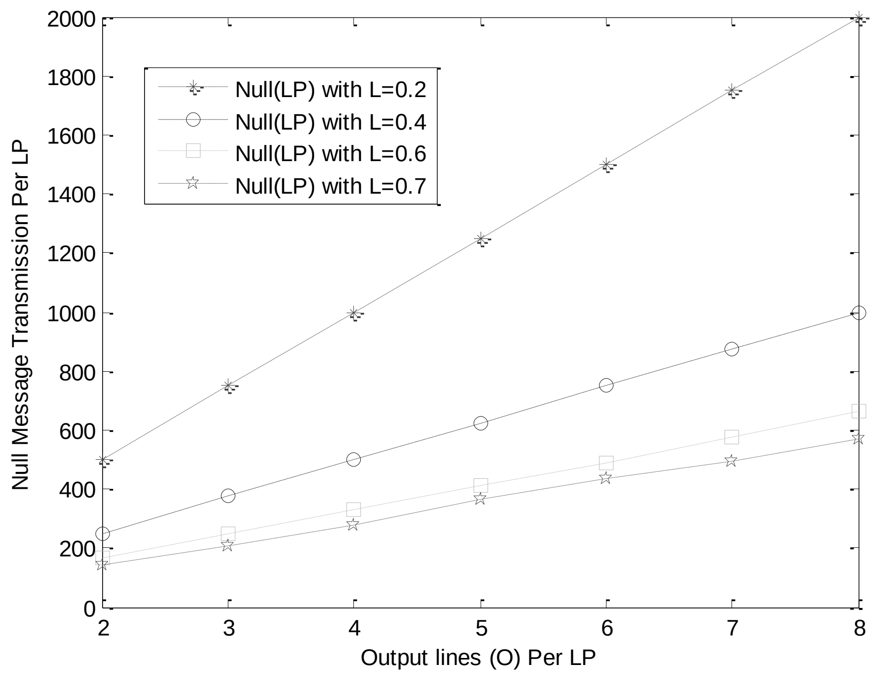

Figure 4: Multiple Output lines per LP versus Null message transmission per LP.

Table I: Multiple Output lines versus Null messages.

Output lines *(O)* per LP	**Null messages per LP when *L* = 0.2**	**Null messages per LP when *L* = 0.4**	**Null messages per LP when *L* = 0.6**	**Null messages per LP when *L* = 0.7**
2	500	250	166.66	142.87
4	1000	500	333.33	277.75
6	1500	750	487.6	433.28
7	1750	875	577.7	496.73
8	2000	1000	661.8	571.62

4.2 Case II: Multiple LPs with multiple output lines per LP

In Case II, we assume that we have multiple LPs with *O* output lines (fixed per LP). Let the output lines per LP is 4 with the simulation time (*Ts*) of 500 s. Fig. 5 shows the null message transmission with the uniformly distribution of Lookahead values per output lines (*O*) where each output line is assumed to be fixed for each LP (i.e., *O* = 4). The numbers of LPs are varied from 1 to 10 as shown in Fig. 5. The numerical results for Case II are presented in Table II.

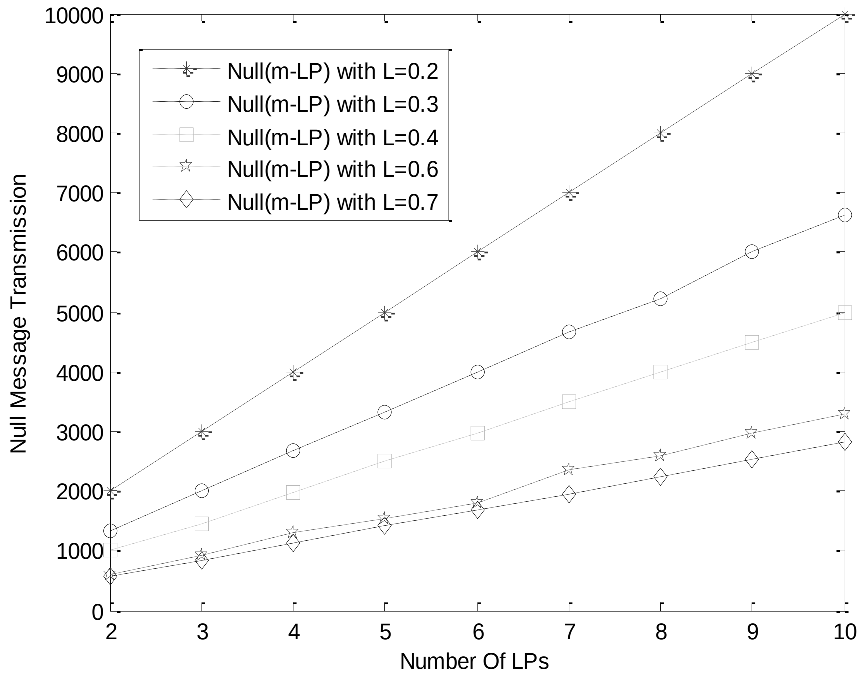

Figure 5: Multiple LPs with fixed output lines per LP versus Null message transmission.

Table II: Multiple LPs versus Null messages.

Number of LPs	**Null messages for *m*-LP when *L* = 0.2**	**Null messages for *m*-LP when *L* = 0.3**	**Null messages for *m*-LP when *L* = 0.4**	**Null messages for *m*-LP when *L* = 0.6**	**Null messages for *m*-LP when *L* = 0.7**
2	2000	1333.33	999.76	605.45	565.55
4	4000	2666.64	1979.44	1302.34	1139.95
6	6000	3998.65	2976.44	1800.45	1685.47
8	8000	5231.53	3992.33	2598.36	2234.68
10	10000	6632.64	4976.45	3303.89	2825.48

4.3 Case III: 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. 6 shows the null message transmission with the following simulation parameters: simulation time is 500 s, *L* is non-uniformly distributed per output lines (*O*). The numbers of output lines are varied from 1 to 10 as shown in Fig. 6. 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. The numerical results for Case III are presented in Table III.

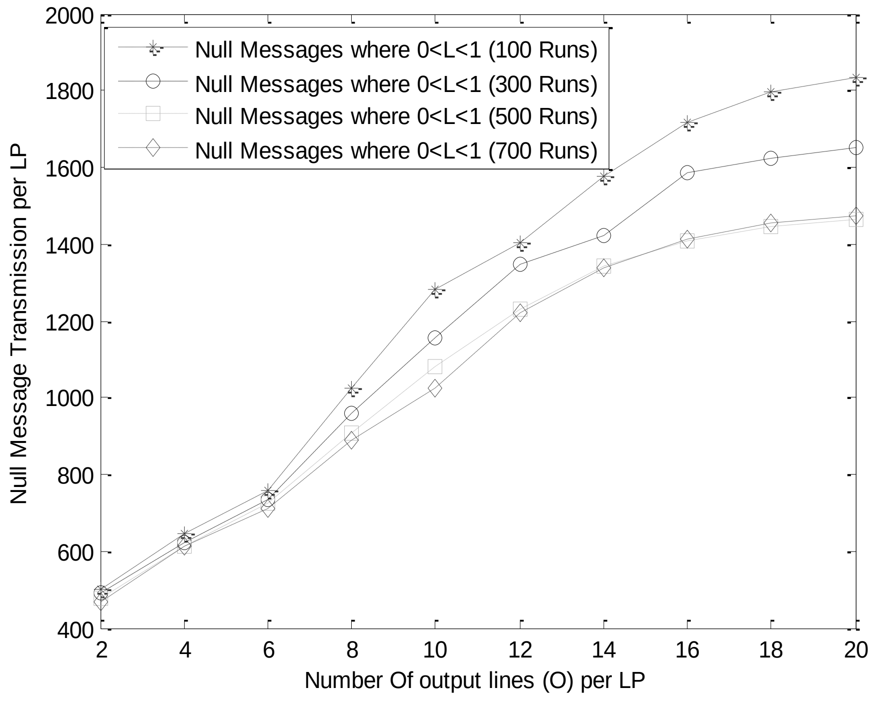

Figure 6: Multiple Output lines per LP with non-uniform distribution of lookahead versus Null message transmission.

Table III: Multiple output lines per LP.

Number of output lines per LP	**Null messages per LP where 0 < *L* < 1 (100 runs)**	**Null messages per LP where 0 < *L* < 1 (300 runs)**	**Null messages per LP where 0 < *L* < 1 (500 runs)**	**Null messages per LP where 0 < *L* < 1 (700 runs)**
2	500.93	489.45	475.45	468.34
6	756.23	733.35	725.34	709.34
8	1024.65	957.27	909.65	887.56
10	1280.43	1155.65	1080.34	1023.34
14	1577.34	1420.53	1342.33	1338.45
16	1717.33	1586.34	1410.37	1414.34
18	1797.45	1625.64	1445.45	1453.56
20	1835.34	1653.34	1464.76	1473.34

4.4 Case IV: Multiple LPs with multiple fixed output lines where each output line can have 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. 7 shows the null message transmission with the following simulation parameters: simulation time is 500 s, *L* is non-uniformly distributed per output lines (*O*). The numbers of LPs are varied from 1 to 20 as shown in Fig. 7. Also, it should be noted that the value of *m* and *O* are both varying quantities 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 behaviour of Lookahead. This can also be considered as irregular networks due to the non-uniform distribution. The numerical results for Case IV are presented in Table IV.

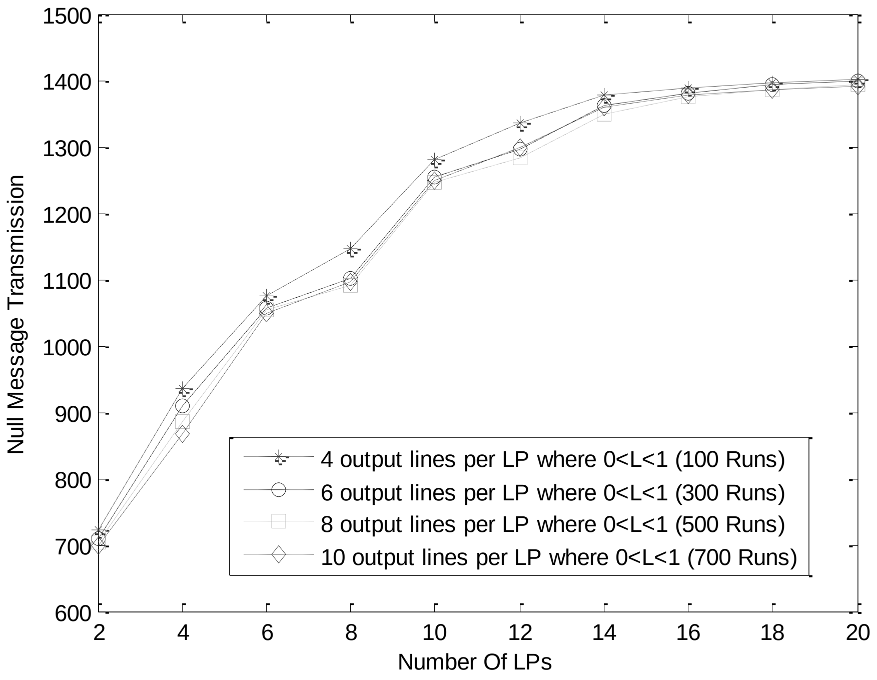

Figure 7: Multiple LPs and multiple fixed output lines with non-uniform distribution of lookahead value versus Null message transmission.

Table IV: Non-uniform lookahead distribution per Output line.

Number of LPs	**Null messages per LP where 0 < *L* < 1 (100 runs)**	**Null messages per LP where 0 < *L* < 1 (300 runs)**	**Null messages per LP where 0 < *L* < 1 (500 runs)**	**Null messages per LP where 0 < *L* < 1 (700 runs)**
2	723.54	710.45	704.34	697.54
6	1074.82	1055.52	1053.33	1049.34
10	1280.34	1254.60	12.45.32	1248.43
14	1378.56	1360.53	1348.29	1357.92
18	1395.75	1393.73	1386.75	1384.48

4.5 Case V: Multiple LPs with multiple fixed output lines with the frequency of transmission

For this simulation, we assume that we have multiple LPs that can have fixed number of output lines where each line of an LP send null messages with a certain frequency. This frequency of transmission can be considered as a portion or percentage of the Lookahead value discussed above. Fig. 8 shows the null message transmission with the following simulation parameters: simulation time is 500 s, frequency of transmission is assumed to be fixed for each output line (*O*). The numbers of LPs are varied from 1 to 20 as shown in Fig. 8.

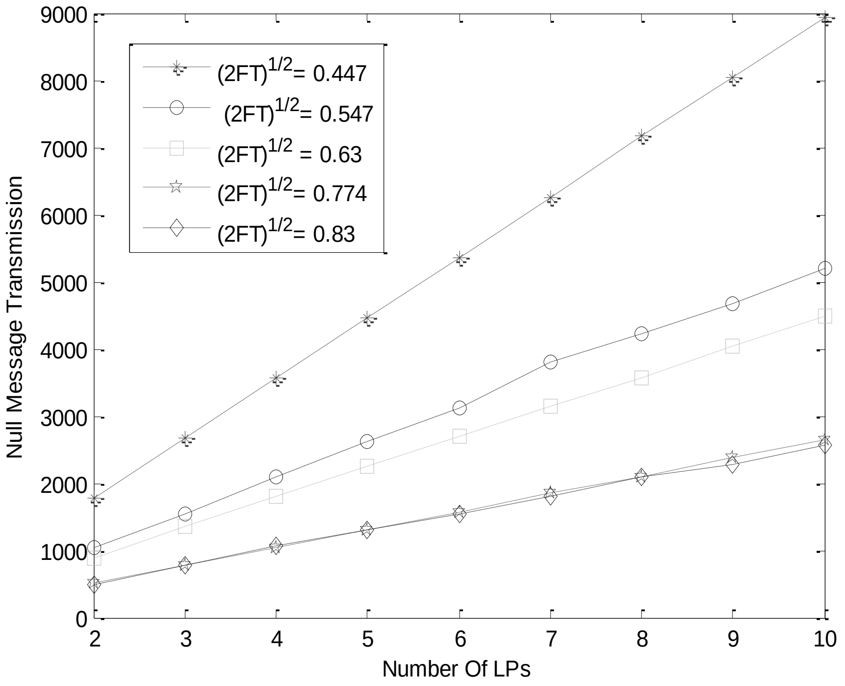

Figure 8: Multiple LPs with multiple fixed output lines with the frequency of transmission versus Null message transmission.

Also, it should be noted that the value *O* is assumed 4 for this particular scenario. The numerical results for Case V are presented in Table V. The use of certain frequency of transmission can stop the unnecessary generation of null messages by causing a comparatively larger time period for null message transmission. In harmony with our expectation, the number of null messages decreases when compared with the non-uniform distribution of Lookahead.

Table V: Multiple LPs with the frequency of transmission.

Number of LPs	**Null messages for *m*-LP $(2F_T)^{1/2}$= 0.447**	**Null messages for *m*-LP $(2F_T)^{1/2}$= 0.547**	**Null messages for *m*-LP $(2F_T)^{1/2}$ = 0.63**	**Null messages for *m*-LP $(2F_T)^{1/2}$= 0.77**	**Null messages for *m*-LP $(2F_T)^{1/2}$= 0.83**
2	1788.9	1033.59	894.42	516.79	497.56
4	3577.8	2079.54	1793.34	1032.34	1076.54
6	5366.4	3123.76	2684.45	1576.87	1546.43
8	7174.3	4235.76	3567.54	2087.78	2086.43
10	8942.3	5187.77	4485.87	2654.87	2567.65

5. CONCLUSION

In this paper, we presented an LP simulation model for a distributed computer simulation. For the proposed simulation model, we used both queuing network model with the varying parameter's network topology and mathematical equations derived in our earlier work. In addition, for extending the proposed simulation model for *n* number of LPs, this paper provided a detailed overview of the internal architecture of each LP and its coordination with the other LPs through sub-system components and models such as communication interface and simulation executive. The transmission of local and remote event-messages through incoming and outgoing channels is considered in the proposed LP simulation architecture as well as supported by the mathematical expressions. In addition, a detailed overview of a sub-system model is presented with the supporting mathematical equations (e.g., maximum number of hops, buffer length, service time etc.) to solve the looping problem of event-messages. 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. It is left to further studies to experimentally verify the implementation of the proposed simulation model for the optimistic TMA simulation.

REFERENCES

[1] Rizvi, S.; Riasat, A. (2008). Reducing null message traffic in large parallel and distributed systems, *Proceedings of the 13th IEEE Symposium on Computers and Communications*, 1115-1121

[2] Fujimoto, R. (2003). Parallel simulation: distributed simulation system, *Proceedings of the 35th Conference on Winter Simulation*, 124-134

[3] Teo, Y.; Ng, Y.; Onggo, B. (2002). Conservative simulation using distributed shared memory, *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, 3-10

[4] Park, A.; Fujimoto, R.; Perumalla, K. (2004). Conservative synchronization of large-scale network simulations, *Proceedings of the 18th Workshop on Parallel and Distributed Simulation,* 153-161

[5] Chandy, K. M.; Misra, J. (1979). Distributed simulation: a case study in design and verification of distributed programs, *IEEE Transactions on Software Engineering,* Vol. SE-5, No. 5, 440-452, doi:10.1109/TSE.1979.230182

[6] Belfore, L. A.; Mazumdar, S.; Rizvi, S. S.; Garcia, J.; Bitts, D.; Blancett, C.; Paredes, E.; Moulton, D.; Quinones, W.; Jones, K.; Browning, J. (2006). Integrating the joint operation feasibility tool with JFAST, *Proceedings of the Fall 2006 Simulation Interoperability Workshop*

[7] Rizvi, S. S.; Riasat, A.; Elleithy, K. M. (2010). An efficient optimistic time management algorithm for discrete-event simulation system, *International Journal of Simulation Modelling,* Vol. 9, No. 3, 117-130, doi:10.2507/IJSIMM09(3)1.146

[8] Liu, Q.; Wainer, G. (2008). Lightweight Time Warp – A novel protocol for parallel optimistic simulation of large-scale DEVS and cell-DEVS models, *Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, 131-138, doi:10.1109/DS-RT.2008.15

[9] Wang, J.; Tropper, C. (2007). Optimizing time warp simulation with reinforcement learning techniques, *Proceedings of the 39th Conference on Winter Simulation*, 577-584

[10] Leye, S.; Uhrmacher, A. M.; Priami, C. (2008). A bounded-optimistic, parallel beta-binders simulator, *Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications,* 139-148, doi:10.1109/DS-RT.2008.32

[11] Patel, H.; Rizvi, S. S.; Almazaydeh, L.; Riasat, A. (2010). Performance model for a conservative distributed simulation environment using null messages to avoid deadlock, *International Journal of Computer Science & Information Technology*, Vol. 2, No. 2, 1-8, doi:10.5121/ijcsit.2010.2201

[12] Rizvi, S.; Elleithy, K.; Riasat, A. (2006). Minimizing the null message exchange in conservative distributed simulation, *International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering*, 443-448

[13] Peschlow, P.; Martini, P. (2007). Efficient analysis of simultaneous events in distributed simulation, *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, 244-251

[14] Madl, G.; Dutt, N.; Abdelwahed, S. (2007). Performance estimation of distributed real-time embedded systems by discrete event simulations, *Proceedings of the 7th ACM & IEEE International Conference on Embedded Software*, 183-192

[15] Rizvi, S.; Elleithy, K. (2011). A generic optimized time management algorithms (OTMA) framework for simulating large-scale overlay networks, *Proceedings of the 14th Communications and Networking Simulation Symposium (CNS'11) at the 2011 Spring Simulation Multiconference*, 27-34

[16] Peacock, J.; Wong, J.; Manning, E. (1979). A distributed approach to queuing network simulation, *Proceedings of the 1979 Winter Simulation Conference*, 399-406

[17] Peacock, J.; Wong, J.; Manning, E. (1980). Synchronization of distributed simulation using broadcast algorithms, *Computer Networks*, Vol. 4, 3-10

[18] Davis, N. J.; Mannix, D. L.; Shaw, W. H.; Hartrum, T. C. (1990). Distributed discrete-event simulation using null message algorithms on hypercube architectures, *Journal of Parallel and Distributed Computing*, Vol. 8, No. 4, 349-357, doi:10.1016/0743-7315(90)90133-A

[19] Bain, W.; Scott, D. (1998). An algorithm for time synchronization in distributed discrete event simulation, *Proceedings of the SCS Multiconference on Distributed Simulation,* Vol.19, No. 3, 30-33

[20] Cota, B.; Sargent, R. (1989). An algorithm for parallel discrete event simulation using common memory, *Proceedings of the 22nd Annual Simulation Symposium*, 23-31

[21] Thondugulam, N. (1998). *Unsynchronized parallel discrete event simulation* (Thesis submitted to the Division of research and advanced studies), University of Cincinnati