2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications

Accelerating Distributed Discrete Event Simulation through Exchange of Conditional Look-Ahead

Desheng Fu, Matthias Becker, Helena Szczerbicka
University Hannover
FG Simulation
Welfengarten 1
30167 Hannover, Germany
{fu, xmb, hsz}@sim.uni-hannover.de

***Abstract*—Distributed discrete event simulation is a very important method today to analyze the behavior of large models. We investigate the practical implementation of distributed discrete event simulation with conservative synchronization and its acceleration through dynamic estimation of process-to-process look-ahead. Since the dynamic look-ahead changes with time, we have to face the situation, that the look-ahead between some logical processes is decreased temporarily. The shortened look-ahead has a very negative influence to the performance of the simulation and it is hard to avoid. However, this effect can be reduced by introducing some extra mechanisms in the simulation.**

In this paper, we present a mechanism to optimize the simulation for the situation that the look-ahead between some processes is very short. This mechanism is based on exchange of conditional look-ahead and broadcast of invalidation announcement. Our evaluation shows reduction of the execution time of a majority of distributed simulations, especially when the estimated look-ahead is stochastically too conservative.

I. Introduction

Discrete Event Simulation (DES) is a very important method today to analyze the behavior of large models. The distributed execution of DES, known as *Distributed Discrete Event Simulation (DDES)*, has many advantages compared with the sequential simulation [1]. Especially, it is expected that the distributed execution of a DES system can reduce the execution time as many other distributed algorithms do. However, depending on the models to simulate, this cannot be achieved if the *Logical Processes (LPs)* are tightly coupled and the *Look-Ahead (LA)* between these LPs is very short [1].

We investigate the practical implementation of DDES with conservative synchronization and its acceleration by dynamic estimation of process-to-process LA [2]. Since the process-to-process LAs of many models change rapidly with time, we have to face the situation, that some of the LAs are shortened temporarily. The temporarily shortened LAs significantly reduce the performance of DDES. In general, we cannot avoid such situations, but the simulator can be optimized to reduce that damage.

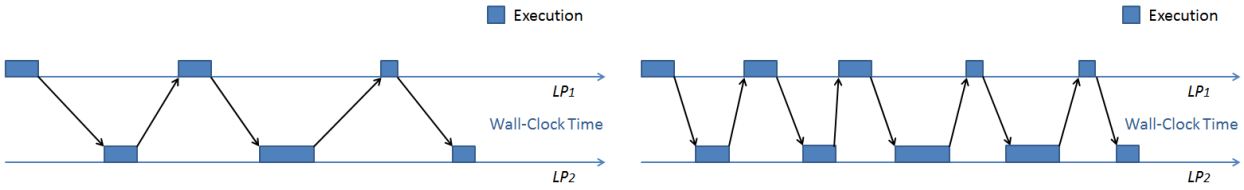
In this paper, we present an approach to extend the LAs by exchange of *Conditional Look-Ahead (CLA)* and broadcast of *Invalidation Announcement (IA)*. In addition, we discuss some typical cases where the approach can be applied. As our evaluation shows, our approach can increase the LA between some certain LPs in many cases. It is optimized to the situation that the LA is short, but the possibility, that some external events should be exchanged right after the time interval defined by the LA, is very small. In other words, the mechanism is optimized to the situation that the estimated look-ahead could be stochastically too conservative (multi-agent system, stochastic Petri-net, cell biology [3] etc.).

The remainder of this paper is organized as follows: the related work on the estimation of LA is introduced in section II. In section III our approach is presented. Section IV is an evaluation of our approach based on a case study followed by a conclusion of this paper in Section V.

II. Related Work

Investigations about LA estimation can be divided into two groups. The LA estimation for conservative synchronization algorithms (e.g. [4] [5]) must be also *conservative* in general to prevent causality errors. Such investigations are often called *look-ahead exploitation*. The main purpose of conservative LA estimation is to provide a longer LA for the conservative synchronization algorithms by estimating the local state. For example, Nicol [6] introduced the *implicit look-ahead* in FCFS (First Come First Served) stochastic queuing network, which was extended by Lazowska [7] to RR (Round Robin) as well as some other queuing networks. The basic idea of Nicol and Lazowska is to pre-simulate the predictable random behaviors in the future. In stochastic queuing networks, they pre-sample the service times of processes into a "future list", the LA is then the time length to the end of next service defined by the future list. Another notable work was done by Meyer el. al. [8]. They investigated conservative algorithms for wireless network simulation and reported the positive

1550-6525/14 $31.00 © 2014 IEEE
DOI 10.1109/DS-RT.2014.30



(a) Time-Creep with Long Transmission Delay

(b) Time-Creep with Short Transmission Delay

Fig. 1: Time-Creep with Different Transmission Delays of LA

effect of the LA estimation in wireless networks. Liu and Nicol [9] also investigated the LA estimation in wireless network with simulator SWAN. They considered the IEEE 802.11 CSMA/CA DCF protocol on MAC layer and estimated the LA from the CSMA/CA behavior of a station. Through estimation of next time to transmit and the next potential push time on the MAC layer, the LA is increased and the execution time is significantly decreased, especially in the network with a low traffic load.

The other group of investigations estimated the LA for optimistic time-warp algorithms (e.g. [10]). Compared to the estimation for conservative algorithms, *over-estimation* is allowed. Such estimation failures will be repaired later with the optimistic time-warp algorithm as described earlier. The main purpose of these LA estimation is to provide a longer LA to reduce the overhead of the algorithm. For example, Maritini et al. [11] introduced the *tolerant synchronization* by accepting event errors during the synchronization. The interval where the event errors might occur is predicted and time-warp algorithm is applied to the time interval to resolve the occurred errors. Ferscha et al. [12] [13] estimated the LA in Petri nets aggressively to reduce the cost of the applied optimistic algorithm, this can also be considered as a conservative solution to the causality error without hard synchronization and an optimistic algorithm is applied to resolve the occurred causality errors. This idea is extended by Kunz et al. [14]. They announced their investigation as an improvement of the optimistic algorithm. Park et al. [15] introduced the *relaxed synchronization* to queuing networks. In his investigation, the LA will also be estimated aggressively to reduce the execution time.

In our approach, we also allow over-estimation. However, we do not apply any time-warp algorithm. The over-estimation in our approach must be detected before any causality error occurs. In other words, the estimated conditional LA in our approach is a way to exchange information between LPs. The truly applied LA will be estimated later conservatively with help of the received information.

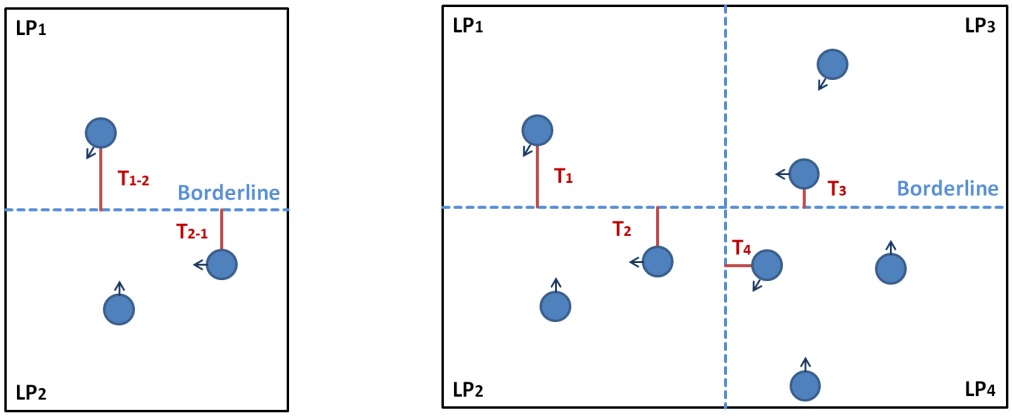
III. Our Approach

In this section, we present our approach to synchronize LPs with CLA and IA.

A. Time-Creep Problem & Look-Ahead

One of the most important challenges in DDES with conservative synchronization is the well-known time-creep problem. It happens when some of the LPs are tightly coupled and the LA between these LPs is very short (or is hard to estimate). To reduce the execution time of the simulation, we should apply a more precise estimation of the LA. If the estimated LA is still too short, sequential simulation should be considered instead of distributed simulation. When we estimate the process-to-process LAs dynamically, the distributed simulation should be preferred when each process-to-process LA is large enough most of the time. However, we have to face the situation, that some of these LAs are shortened temporarily or a small amount LAs between certain LPs are very short.

When the LA is very short, the time advance of the system is very slow due to the extra cost. First of all, the LPs have to exchange many messages (e.g. null-message) which are costly. In addition, the LPs have to synchronize with each other at a high frequency. Most notably, the frequency of the synchronization often depends on the shortest LA in the whole model. The synchronization, especially the global synchronization, reduces the parallelism, since the fast LPs must be blocked to wait for synchronization of the slow LPs. Please note that in practice, the LPs are blocked not only for synchronization. Depending on the applied synchronization algorithm, the LPs might need to be blocked for the LA estimation, which will be discussed later. Furthermore, the cost of execution of synchronization after the LPs are blocked depends strongly on the transmission delay between the LPs including the time to process received messages. In the idealistic situation without any transmission delay, the performance is almost the same as with sequential simulation. The more time the transmission takes, the slower the time advances. An example is shown in Figure 1, where the synchronization always takes place when an LP is blocked (such as the null-message algorithm [4] [5]). In practice, the transmission delay between the LPs



(a) Multi-Agent System with Two LPs

(b) Multi-Agent System with Four LPs

Fig. 2: Multi-Agent Systems

is quite different. If two LPs are located at the same computer with shared memory, the transmission delay is very short compared with two LPs located at two different computers connected with LAN. Obviously, the transmission delay could further increase if the two computers are connected by the Internet.

In this paper we present a new synchronization mechanism with several advantages. First of all, there is no global synchronization in general. The mechanism only blocks LPs when it is really necessary. Secondly, it is able to increase the LA, especially the shortest LAs in the model and the LAs between LPs with long transmission delay. Thus the frequency of the costly synchronization is reduced. However, the mechanism has its own cost. Thus it may not work well for some models, which will be discussed later.

B. Solution with Conditional Look-Ahead

Our approach to increase the LA is achieved by exchange of the Conditional Look-Ahead (CLA). CLA is an extension to the original LA, which is only valid when some certain conditions are fulfilled [9]. During the advance of the time, the conditions and CLA can be invalidated any time after the original LA expires. To keep the CLA conservative and to avoid causality error, a mechanism must be applied, so that the related LP receives an Invalidation Announcement (IA) before the CLA is invalidated. In addition, it should be guaranteed that the new LA after applying the IA is non-negative, so that no causality error occurs. The new LA can be sent with IA. Alternatively, an IA may contain a time stamp. The IA will be "processed" at this time to inform that the CLA is no more valid, and the new LA after the invalidation is zero.

The reason, why the mechanism could be helpful to increase the LAs, is quite simple. The dynamic LA estimation during the simulation benefits from the knowledge of the current state of the model and thus could provide a longer LA than the static estimation in most cases. However, we should limit the cost of the dynamic estimation so that the total execution time is reduced. In the practical implementation, one of the two basic ideas is considered in general. First of all, each LP could estimate the LA only based on its local state (e. g. [4] [5]). With this idea, the cost of the estimation is minimized and there is no unnecessary blocking, but the estimated LA could be too conservative. In addition, each LP or a central controller could block all LPs to take a static global observation for the LA estimation (e. g. [1] [16]). With this idea, the best LA is provided but it enforces to block all LPs as a global synchronization each time the estimation is done. As mentioned earlier, it reduces the parallelism of the simulation and thus is costly. Our mechanism combining the basic ideas of the two solutions could provide a better performance. On one side, each LP exchanges information (e.g. CLA, IA) with neighboring LPs to provide a long LA. On the other side, no global synchronization is necessary. Obviously, this mechanism has its own cost, but it is still considerable when the possibility that the CLA is invalidated before it expires is very small.

A scenario is shown in Figure 2a as example, where a multi-agent system is simulated distributedly. In this example, the whole area is divided by a borderline, and each part is simulated with one LP (LP_1 and LP_2). There are several agents. They can move in the area to solve some certain problems with built-in intelligence, which is hard to be considered for the estimation of LA in most cases. In general, such a DDES system is simulated with time-warp. Without any information about the agents in the other part of the area, no positive LA can be considered for the possibility, that one agent crosses over the borderline (hand-in) at the next time point

and leaves back (hand-out) immediately. Therefore, distributed conservative synchronization without central intelligence cannot be applied. However, since the cost of time-warp depends on the cost to duplicate the agents including the state of its intelligence, which can be very costly, we may still want to simulate the model with conservative synchronization. In these situations, the conservative synchronization with central intelligence could be considered, for example *synchronized execution algorithms* [1] [16] with a specified synchronization algorithm (e.g. [17]). With this algorithm, we take a static observation to the whole area. Since there is currently no agent which is right on the borderline, the LPs have to exchange no information in the short future, and the bi-directional LA could be calculated as the minimal time needed for an agent to move to the borderline regarding the position as well as the maximal speed of each agent (T_{2-1} in the example). There are many different algorithms to synchronize the LPs with the estimated LA, the discussion of these algorithms is out of scope of this paper.

Without global observation, this idea could still be achieved through exchange of extra information between LPs. This could be formally described with CLA and IA as shown in Figure 3a. The condition of the CLA sent from LP_1 to LP_2 (CLA_{1-2}) is that LP_2 will make no change to the local state of LP_1 (no external event or IA from LP_2 will be processed by LP_1). The length of CLA_{1-2} depends on the model. It is at least same as LA and it is larger than LA in many cases (T_{1-2} vs. 0 in the example). Similarly, the CLA sent from LP_2 to LP_1 (CLA_{2-1}) is determined. Furthermore, the condition of the CLA sent from LP_1 to LP_2 breaks if LP_2 does send an external event to LP_1 with time stamp T. In that case an IA is sent by LP_2 to itself to inform the invalidation of CLA_{1-2} at T. Since T is anyway after the current time of LP_2, no causality error occurs.

C. Fully-Connected Topology and Line Topology

The approach discussed in the previous subsection could also be applied to the model with more LPs, where all LPs are directly connected (*fully-connected topology*). A scenario of multi-agent system with four LPs is shown in Figure 2b and the general model is shown in Figure 3b. When all LPs could communicate with each other directly and there is LA between each pair of LP, each LP could send an identical CLA to all other LPs with the condition that its state will not be modified by any other LP. When the first external event is sent from one LP to another, an IA with the same time stamp will be broadcasted so that all CLAs from the LP, where the external event should be processed, are invalidated. Obviously, the global CLA sent from this LP is shorter than any process-to-process CLA from this LP. Thus no matter to which LP the external event is sent, the time stamp of the external event is larger than the global CLA. The IA is thus in the future of any other LPs and there is no causality error caused by the external event and the IA. There is a large cost to reconstruct the CLAs after the IA is broadcasted. Thus the solution with CLA should only be applied when the probability that

1. Initialize
for each neighbor LP_r **do**
  Let $localID_r = 0$.
  Let $remoteID_r = 0$.
end for

2. Estimate & Send CLA
Let CLA_{min} be the minimal time of state change propagation from one interface to another.
for each neighbor LP_r **do**
  Estimate the CLA_r to LP_r with the condition that the local state will not be changed by any other LP.
  if $CLA_r < CLA_{min}$ **then**
    Let $CLA_{min} = CLA_r$.
  end if
end for
for each neighbor LP_r **do**
  Send CLA_{min} to LP_r with ID $localID_r$.
end for

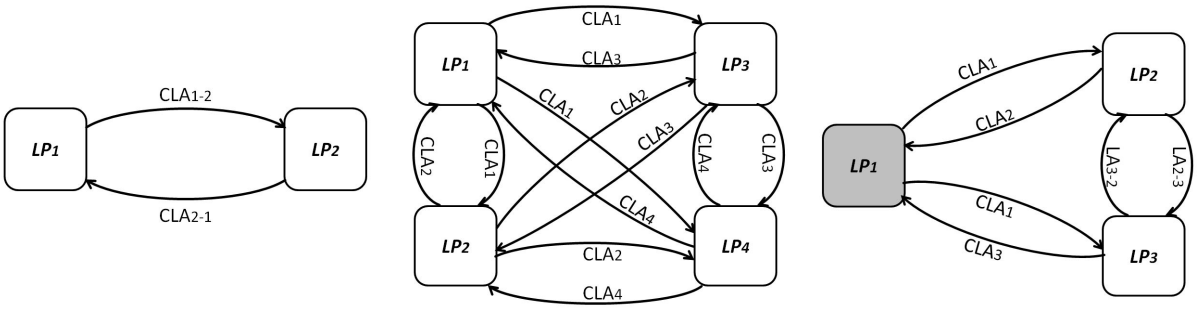
3. Apply Received CLA
Input: the received CLA CLA_r from LP_r with ID id.
if $id \geq remoteID_r$ **then**
  Apply CLA_r as the current LA from LP_r.
end if

4. Schedule External Event & Broadcast IA
Input: the event E to LP_r with time stamp T.
Send E to LP_r.
for each LP_t in any fully-connected topology with the local LP and LP_r **do**
  Send IA to LP_t with time stamp T and parameter LP_r.
end for
Do **2.** to estimate and send the new CLA.

5. Process Received External Event
Input: the event E from LP_r.
Process E.
for each LP_t in any fully-connected topology with the local LP and LP_r **do**
  $localID_t$++.
end for
Do **2.** to estimate and send the new CLA.

6. Process Received IA
Input: the IA from LP_r with parameter LP_t.
Set the current LA from LP_r to 0.
$remoteID_r$++.

Fig. 4: Pseudocode of the algorithm



(a) Model with Two LPs (b) Model with More LPs (Fully Connected) (c) Model with More LPs (Two Groups)

Fig. 3: Different Models to Apply CLA

CLA is invalidated before its regular expiration is very small. The mechanism increases the short LAs, but it might reduce the long LAs in the system. In this aspect, it balances the LAs in the system.

Similarly, this approach could be applied to the models where all LPs are connected as a chain (*line topology*) as long as the state change of one LP will not cause the immediate state change of a neighboring LP. The approach discussed in the previous section can be applied to each *interface* between a pair of LPs. During the calculation of CLA for one interface, it is assumed that the state of the LP could be changed any time by a neighboring LP from another interface. Thus the CLA is limited by the minimal time needed that state change "propagated" from one interface to another. In the example of multi-agent simulation, it is the minimal time needed for an agent to cross the area simulated by this LP.

Furthermore, the approach of CLA works well with a combination of fully-connected topology and chain topology. In that case, an interface is between a LP and a group of LPs organized as a fully-connected topology. An example will be discussed later as case study. Compared with the synchronized execution based on conservative LA, more messages will be exchanged between these LPs to exchange CLAs and IAs. But we do not need any static observation to the whole system and the LPs will not be blocked due to the observation in general. Our approach is thus considerable, especially when the estimated LA is stochastically too conservative. A practical implementation of the mechanism with slight simplification is shown in Figure 4, where each LP estimate a global CLA for all neighboring LPs instead of a CLA per interface. Since it is unknown whether the external event will be processed at the specified LP before sent IAs are processed at some other LPs, serial IDs are managed to compare the number of processed external events and IAs between each pair of LPs.

D. Grouping the Logical Processes

In the simulation of large models, there are often many LPs. To apply the mechanism to these simulations, the LPs could be divided into groups according to the transmission delay, so that the transmission delay between LPs in the same group is short and the transmission delay between LPs in different groups is long. Within each group, the local synchronized execution is applied and the group acts like a single LP. The solution with CLA is applied as discussed in the previous subsections for the synchronization between groups. Since the transmission delay within the groups is short, the cost of such local synchronized execution is much less than the global synchronized execution. However, the local synchronization must be done more often to calculate CLAs needed for the synchronization between groups. Thus the combination with synchronized execution and CLA is only considerable when the LA is stochastically too conservative as mentioned earlier.

Alternatively, we could consider the synchronization based on conservative LA within each group and synchronization based on CLA between groups. A scenario with three LPs is shown in Figure 3c. As illustrated there, the three LPs are divided into two groups. LP_1 belongs to the first group, LP_2 and LP_3 belong to the second. We consider that each group is simulated with an individual computer with sufficient number of processors. Since the transmission delay between LP_1 and LP_2 as well as between LP_1 and LP_3 are very long, we will extend them with CLA. First of all, LP_1 sends CLA to LP_2 and LP_3 (CLA_1) with the condition, that its own state will be modified neither by LP_2 nor by LP_3. If LP_2 sends an external event to LP_1 with time stamp T for example, it will send an IA to itself to inform that the CLA_1 is no more valid after T. As discussed earlier, there is no causality error. In addition, it will send the same IA to LP_3. This IA will be arrived in time without causality error if the LA from LP_2 to LP_3 (LA_{2-3}) is shorter than the sum of LA from LP_2 to LP_1 (LA_{2-1}) and LA from LP_1 to LP_3 (LA_{1-3}). Similarly, it must be guaranteed that the LA from LP_3 to LP_2 (LA_{3-2}) is shorter than the sum of LA from LP_3 to LP_1 (LA_{3-1}) and LA from LP_1 to LP_2 (LA_{1-2}). Otherwise, the mechanism

might cause causality errors and it cannot be applied. In practice, we only consider LA_{2-1} and LA_{3-1} and ignore the LA_{1-3} and LA_{1-2} due to the extra cost of communication. Most important, we will limit LA_{2-3} and LA_{3-2} so that it is always shorter than LA_{2-1} and LA_{3-1}. Furthermore, LP_2 sends CLA to LP_1 with the condition, that LP_1 will not modify the state of LP_2 and LP_3 with external events. If LP_1 sends an external event to LP_2, an IA will be sent to itself as discussed above.

This alternative solution can be seen as a trade-off between the external LAs and internal LAs. On one side, any LA sent from LP_i to any LP in the same group is limited by

$$min\{LA_{i-k}|LP_k \text{ belongs to the other group}\}.$$

CLA could be applied to extend these LAs anyway, but the limit must be kept. On the other side, the LA / CLA sent from LP_i to any LP in the other group could be extended with condition. If the condition really helps to extend the LA depends strongly on the model and the partitioning. Generally it is hard to determine due to the unknown state of LPs in the same group. But in some special situations it could help to break some immediate communication loops.

IV. Evaluation

As case study, we consider the simulation of a multi-agent system. The whole area is divided as 9 sub-areas (3×3) as shown in Figure 5a and each sub-area will be simulated with one LP. There are n agents in the whole area, which are placed randomly at the start with a random direction. They move straight forward with a constant speed. When an agent reaches one border of the whole area, it will be rebounded. However, we only consider the distance between the agent and borderlines when calculating LA, since the simulator itself holds no extra knowledge about the intelligence in the agent. In addition, the simulator could anyway ascertain that the direction and speed will not be changed in a very short future. Furthermore, each agent saves information about the agent's "meetings" (distance is short enough) as well the time of last meetings for each of them.

The model is implemented with our simulator named *Universal Simulation Engine (USE)* [1]. The program runs on a single computer equipped with an Intel i7 CPU. We consider the "simulation of simulation" to study the case. In other words, we simulate a distributed simulator of the given model. Besides the model itself, we also simulate core behaviors of the distributed simulator, including the transmission delay, cost to lock and unlock a LP, the execution power of each LP, wall-clock time of the virtual simulator, etc. The result represents the simulation result with 9 individual computers each with a Intel i7 CPU.

[1]The simulator is available at http://www.useproject.com/.

The model to simulate is considered as a combination of four fully-connected topologies and several line topologies between the stars. For example the fully-connected topology FC_1 contains LP_1, LP_2, LP_4, and LP_5. If LP_1 sends an external event to LP_4, it will broadcast an IA to LP_2, LP_4 ("included" in the external event), and LP_5. Furthermore, LP_4 belongs to two different fully-connected topologies (FC_1 and FC_3). In other words, it has two interfaces. The state changes of LP_4 due to the LP_1 will not cause immediate state change of LP_7 for example due to the minimal time needed for an agent to cross the sub-area simulated by LP_4. In many models of multi-agent system including this one, the minimal time needed that the state change propagated from one interface to another interface of a LP is longer than any CLA provided by the LP and it will be thus ignored.

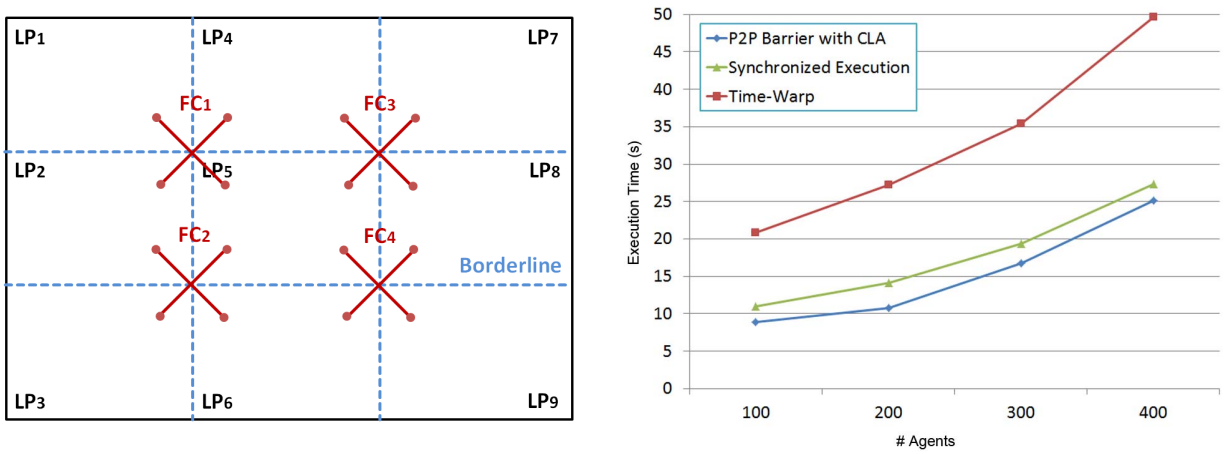
The CLA is utilized by the *Process-To-Process Barrier (P2P Barrier)* synchronization of USE. A barrier represents a specified bounded time that the process should never reach. During the simulation, each process sets a barrier to every process including itself based on the CLA. These barriers will be altered (moved back) before the CLA expires. If there is no event exchange between two processes at all or the causality is maintained by extra logical constraints, a barrier pointing to infinity should be considered.

As comparison, we consider the standard time-warp algorithm as well the global synchronized execution based on butterfly-barrier synchronization [17]. The result of the simulation is shown in Figure 5b. It is obvious that the simulation takes more time if there are more agents to simulate. The execution time for time-warp is very long due to the extra cost to save the state of each agent including the state of its AI. This algorithm could be further optimized but it is out of scope of our discussion. The result with global synchronized simulation and the result based on CLA are comparable. Our approach takes about 10% - 24% shorter time compared with global synchronized execution depends on the number of agents in the system.

V. Conclusion

We presented a new approach to accelerate Distributed Discrete Event Simulation (DDES) by exchange of Conditional Look-Ahead (CLA) and broadcast of Invalidation Announcement (IA). Our approach could increase the LA between some certain LPs in many cases. It is optimized to the situation, that the estimated Look-Ahead (LA) is stochastically to conservative, and the probability that an external event really occurred right after the expiration of the LA is very small.

We presented two typical topologies where CLA could be applied to accelerate the simulation. First of all, if all LPs could communicate with each other directly (fully-connected topology), each LP could send an identical CLA to all other LPs with the condition, that its local state will not be modified by any other LP.



(a) Model

(b) Result

Fig. 5: Case Study

The IA should be broadcasted after any external event is sent within the whole system. In addition, the approach can be applied to a chain of LPs (line topology) as long as the state change of one LP will not cause the immediate state change of a neighboring LP. Most important, the approach also works for a combination of the both topologies. We also discussed the possibility to apply the solution with CLA to more complex models through grouping of LPs. By the case study of multi-agent simulation, the execution time of the simulation is reduced by up to 24% compared with a globally synchronized execution.

REFERENCES

[1] R. M. Fujimoto, *Parallel and Distribution Simulation Systems*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1999.

[2] D. Fu, M. Becker, and H. Szczerbicka, "On the potential of semi-conservative look-ahead estimation in approximative distributed discrete event simulation," in *Proceedings of the 2013 Summer Computer Simulation Conference*, ser. SCSC '13. Vista, CA: Society for Modeling & Simulation International, 2013, pp. 28:1–28:8.

[3] S. Leye, A. Uhrmacher, and C. Priami, "A bounded-optimistic, parallel beta-binders simulator," in *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*, Oct 2008, pp. 139–148.

[4] R. E. Bryant, "Simulation of packet communication architecture computer systems," Cambridge, MA, USA, Tech. Rep., 1977.

[5] K. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *Software Engineering, IEEE Transactions on*, vol. SE-5, no. 5, pp. 440 – 452, sept. 1979.

[6] D. M. Nicol, "Parallel discrete-event simulation of fcfs stochastic queueing networks," *SIGPLAN Not.*, vol. 23, no. 9, pp. 124–137, Jan. 1988.

[7] Y. Lin and E. Lazowska, "Exploiting lookahead in parallel simulation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 4, pp. 457–469, 1990.

[8] R. A. Meyer and R. L. Bagrodia, "Improving lookahead in parallel wireless network simulation," in *Proceedings of the 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 262–.

[9] J. Liu and D. M. Nicol, "Lookahead revisited in wireless network simulations," in *Proceedings of the sixteenth workshop on Parallel and distributed simulation*, ser. PADS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 79–88.

[10] D. R. Jefferson, "Virtual time," *ACM Trans. Program. Lang. Syst.*, vol. 7, no. 3, pp. 404–425, Jul. 1985.

[11] P. Martini, M. Rümekasten, and J. Tölle, "Tolerant synchronization for distributed simulations of interconnected computer networks," *SIGSIM Simul. Dig.*, vol. 27, no. 1, pp. 138–141, Jun. 1997.

[12] A. Ferscha and G. Chiola, "Self-adaptive logical processes: the probabilistic distributed simulation protocol," in *Simulation Symposium, 1994., 27th Annual*, apr 1994, pp. 78 –88.

[13] A. Ferscha, "Probabilistic adaptive direct optimism control in time warp," in *Parallel and Distributed Simulation, 1995. (PADS'95), Proceedings., Ninth Workshop on (Cat. No.95TB8096)*, jun 1995, pp. 120 –129.

[14] G. Kunz, M. Stoffers, J. Gross, and K. Wehrle, "Know thy simulation model: analyzing event interactions for probabilistic synchronization in parallel simulations," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '12. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 119–128.

[15] H. Park and P. Fishwick, "A fast hybrid time-synchronous/event approach to parallel discrete event simulation of queuing networks," in *Simulation Conference, 2008. WSC 2008. Winter*, dec. 2008, pp. 795 –803.

[16] S. Jafer and G. Wainer, "Global lookahead management (glm) protocol for conservative devs simulation," in *Distributed Simulation and Real Time Applications (DS-RT), 2010 IEEE/ACM 14th International Symposium on*, Oct 2010, pp. 141–148.

[17] D. M. Nicol, "Noncommittal barrier synchronization," *Parallel Comput.*, vol. 21, no. 4, pp. 529–549, Apr. 1995.