Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

A UNIFIED FRAMEWORK FOR CONSERVATIVE AND OPTIMISTIC DISTRIBUTED SIMULATION

V. Jha
R. L. Bagrodia

October 1993
CSD-930036

A Unified Framework For Conservative And Optimistic Distributed Simulation [1]

Vikas Jha
Computer Science Department
University of California at Los Angeles
Los Angeles, CA 90024
Email: jha@cs.ucla.edu
Phone: (310) 825-4885

Rajive L. Bagrodia[2]
Computer Science Department
University of California at Los Angeles
Los Angeles, CA 90024
Email: rajive@cs.ucla.edu
Phone: (310) 825-0956

Abstract

A great deal of research in the area of distributed discrete event simulation has focussed on evaluating the performance of variants of conservative and optimistic methods on different types of applications. Application characteristics like lookahead, communication patterns etc. have been found to affect the suitability of a specific protocol to simulate a given model. For many systems, it may be the case that different subsystems possess contradictory characteristics such that whereas some subsystems may be simulated efficiently using a conservative protocol, others may be more amenable to optimistic methods. Furthermore, the suitability of a protocol for a given subsystem may change dynamically. We propose a parallel simulation protocol that allows different parts of a system to be simulated using different protocols, allowing these protocols to be switched dynamically. A proof of correctness is presented, along with some preliminary performance measurements.

[1]This research was partially supported by NSF, ONR, Hughes Aircraft Co., and MICRO.

[2]Author to whom correspondence should be addressed

1 Introduction

A great deal of research has been done to evaluate the performance of variants of conservative and optimistic methods on different types of distributed discrete-event simulation applications ([Fujimoto 90] reviews many of these studies). Some consensus has emerged on gross application characteristics like lookahead, communication patterns among the primary model components, and checkpointing granularity that determine the suitability of using a specific protocol to simulate a given model. For many systems, it may be the case that different subsystems possess contradictory characteristics; whereas some subsystems may be simulated efficiently using a conservative protocol, others may be more amenable to optimistic methods. Furthermore, the suitability of a protocol for a given subsystem may change dynamically. For instance consider a gate level model of a circuit. If the signal activity in a given portion of the circuit is reasonably uniform and regular, conservative null-message methods may be suitable. However, if the signal activity becomes sporadic, an optimistic protocol that reduces blocking may be more effective. Performance reasons aside, in some applications, it may not be possible to execute some processes optimistically because of the sheer size of the state (many copies of which have to be maintained in optimistic schemes), whereas the rest of the system may have characteristics suited to optimistic execution. The need to develop efficient protocols for the execution of dynamic systems has led to considerable interest in designing simulation protocols which unify conservative and optimistic methods, thereby combining the advantages of both.

This paper describes a simulation protocol called CON-OPT which attempts to bridge the gap between optimistic and conservative simulation methods. CON-OPT allows different parts of a simulation model to be executed using either conservative or optimistic protocols. The execution mode for a submodel may also be changed (from conservative to optimistic, or vice-versa) dynamically.

Optimistic schemes like Time Warp [Jefferson 85] have two major components -

the Local Control Mechanism(LCM), which ensures that events are eventually executed in correct order, and the Global Control Mechanism (GCM) which is concerned with global issues such as I/O handling, termination detection, memory management etc. The key observation is that a similar logical division can be made for conservative algorithms. For each simulation object (or LP) in the model, we define the Earliest Input Time (EIT) as the lower bound on the timestamp of any future messages that may be delivered to that LP. The GCM is used to compute the EIT for each LP and the LCM allows only messages with timestamps less than EIT to be processed.

The choice of a GCM can be shown to be orthogonal to that of an LCM. This means that an LP may autonomously choose to be optimistic or conservative by selecting the appropriate Local Control Mechanism, whereas the model uses a single Global Control Mechanism. Examples of candidate GCMs include the Global Virtual Time mechanism used by optimistic methods or the null message based scheme used by conservative methods. This permits an optimistic LP to exploit optimizations like lookahead and null messages that are typically used only with conservative methods; similarly a conservative LP may benefit from an arbitrarily dynamic communication topology, and a relaxation of the FIFO channel requirement typically needed for conservative methods.

The paper also presents a formal proof of correctness for the CON-OPT protocol. As many of the existing simulation protocols like Time Warp [Jefferson 85], CMB algorithm [Chandy 81], and the conditional event algorithm [Chandy 89a] are special cases of this protocol, this generates a correctness proof for all the protocols under a consistent set of assumptions.

Rest of the paper is organized as follows. Section 2 describes the distributed simulation model and the assumptions made. Section 3 describes the simulation protocol. We give a proof of correctness for the algorithm in section 4. Section 5 describes how processes can change their mode of operation at the runtime. Relationship of CON-OPT to the existing simulation protocols and some new algorithms that can

be derived as special cases of CON-OPT are discussed in section 6. Section 8 discusses assumptions made regarding the communication topology and lookahead in CON-OPT. Section 7 presents some performance results. The related work is briefly discussed in section 9. Section 10 gives the conclusions and directions for the ongoing research.

2 Model

A physical system is a finite set of physical processes (PP) and a finite set of events; each event represents some interaction among the processes. In the *simulation model*, each PP is represented by a *logical process*(LP). An event is represented by a message communication among the corresponding LPs. Each message in the model has a logical timestamp that corresponds to the (relative) physical time at which the corresponding event occurred in the physical system. For efficiency reasons, an LP may simulate an arbitrary number of LPs for an arbitrary interval of simulation time (i.e. an arbitrary region of the space-time [Chandy 89b]). For simplicity, we will assume that each LP simulates exactly one LP for the entire simulation interval.

Execution of a simulation model on a parallel architecture requires the allocation of the LPs in the model among the available physical processors, and the definition of a synchronization protocol to ensure that messages in the distributed model are (eventually) processed by the LPs in the global order of their timestamps. This paper addresses only the latter problem. Before describing the protocol, we list a set of assumptions that form the basis for our design. Similar assumptions have been imposed by most existing protocols.

Assumption 1 *All messages with the same timestamp are processed simultaneously by an LP (and the corresponding PP).*

Assumption 2 *Each LP is a total function of its inputs.*

Assumption 3 *For a given initial state and inputs (with the same timestamp) , the LP will produce the same output(s) and reach the same final state as the corresponding PP(***Fidelity***).*

Assumption 4 *Given a finite set of inputs with timestamp t, an LP generates a finite set of outputs with timestamps $t + \epsilon$ or greater, where ϵ is a positive constant.*

Assumption 5 *Communication channels on the multiprocessor are reliable and FIFO.*

Assumption 6 *Sufficient memory is available at each processor to store all incoming messages (i.e. we ignore the problem of flow control).*

Henceforth we use **x**, **y**, **z** to refer to PPs, and **p**, **q**, **r** to refer to LPs. The preceding identifiers may also be used to subscript variables used in the description of the protocol; thus v_p is used to refer to the variable v for LP p.

Definitions

We introduce a few terms that occur frequently in the protocol description:

input-set refers to the set of inputs with a given timestamp; the input-set at p with timestamp t is represented by $I_p(t)$. A sequence of input-sets is referred to as an **input-sequence**; $IS_p(t)$ refers to the subsequence of input-sets with timestamps *less* than t received by p.

output-set refers to the set of outputs (possibly with different timestamps) that are a function of a given input-set; $O_p(t)$ refers to the set of outputs produced by process p as a result of processing $I_p(t)$. An **output-sequence** is defined similarly to an input-sequence; $OS_p(t)$ refers to the output-sequence that is a function of $IS_p(t)$.

Lookahead for a process p, at simulation time time t, is said to be la, if p can accurately predict all the outputs it will generate in the interval $[t, t + la]$. By assumption 4, la is at least ϵ.

Logical Process

We abstract an LP by the set of data structures used by the LCM or GCM to implement the corresponding simulation protocol. The following data structures are defined:

Input Queue contains inputs received by the LP that have not been processed. $\mathbf{T_p}$ refers to the earliest timestamp in the *input queue*; if the queue is empty, $T_p = \infty$.

History contains a history of the LP over some interval of simulation time; the duration of the interval depends on the LCM of the LP. We use $h_p(t_1, t_2)$ to refer to the history of LP p in the interval $[t_1, t_2]$. For a given t, $h_(t)$ is a 5-tuple – $< t, I(t), St, O(t), la >$, where St represents the state of the LP after processing input-set $I(t)$, and the other elements are as described above. The history is sorted by the timestamp of each entry. The entry with the earliest timestamp is denoted by $S[0]$, whereas the entry with the largest timestamp, also known as the *current* state, is denoted by $S[n]$. As will be shown later, for a conservative LP, the history need only contain a single entry, the one with the largest timestamp.

3 Simulation Protocol

The simulation protocol can be logically divided into two orthogonal parts - Local Control Mechanism(LCM) and Global Control Mechanism(GCM). The LCM is responsible for executing events in the correct timestamp order, and GCM for global issues like progress, I/O handling, termination detection, and memory management. Following definitions are used in describing the Local and Global Control Mechanisms:

Earliest Input Time(EIT) At any execution instant, the EIT for a given p is referred to as eit_p and is defined to be the timestamp of the earliest input message

that the LP may receive in the future. The history of each LP contains at most one entry with timestamp less than its EIT. i refers to the minimum EIT in the system.

Earliest Output Time (EOT) At any execution instant, the EOT for a given p is referred to as eot_p and is defined to be the timestamp of the earliest output that the LP may send in the future. eot_p is equal to $min(T_p, eit_p) + S[0].la$.

Earliest Conditional Output Time(ECOT) At any execution instant, the ECOT for a given p is referred to as $ecot_p$ and is defined to be the timestamp of the earliest output that the LP may generate in the future, *assuming no more inputs are received by the LP until then* . $ecot_p$ is equal to $T_p + S[i].la$, where i is the largest integer such that $S[i].t < T_p$. Note in absence of any lookahead, ECOT would the equivalent of LVT in Time Warp.

3.1 Local Control Mechanism

LCM ensures the correctness of simulation either by *blocking* to ensure that only correct messages are processed(Conservative) or through *rollbacks* upon discovering causality errors(Optimistic). Note that LPs process an antimessage exactly like the corresponding positive message. However, if a message and its antimessage, both are present in an input(output)-set, they are assumed to cancel each other. If there are more than one *eligible* (as defined below) LPs on a processor, any of the conservative LPs can be chosen to be executed. If there are no eligible conservative LPs, the optimistic (eligible) LP with the smallest T_p is chosen.

3.1.1 Optimistic Processes

An optimistic LP is eligible to be executed whenever it's *input queue* is non empty. Let p be the LP selected to execute, and let $I(t)$ be the first input-set in its *input*

queue.

- If $t > S[n].t$: Execute $< St', O, la >:= p(S[n].St, I)$. Send out the messages in O. Append the entry $< t, I, St', O, la >$ to the *history*,
- If $t \leq S[n].t$: Find the largest i, such that $t > S[i].t$. Rollback upto $S[i]$, and recompute using the input-sets from the *history*, after adding I to the input-set with timestamp t.

Lazy cancellation and lazy recomputation optimizations [Jefferson 85] can be used in case of rollbacks.

3.1.2 Conservative Processes

A conservative LP is eligible to be executed if its $T_p < eit_p$ Let p be the selected LP, and let $I(t)$ be the first input-set in its *input queue.* (it follows from lemma 3 that $t > S[0].t$).

- Execute $< St', O, la >:= p(S[0].St, I)$. Send out the messages in O. Append the entry $< t, I, St', O, la >$ to the history. The previous entry would be deleted, since, only one entry with $t < EIT$ needs to be kept. Note, therefore, that for a conservative LP, there is only one entry in the *history* at any given time. As mentioned before, I and O need not be saved for the entry $S[0]$.

3.2 Global Control Mechanism

The objective of the Global Control Mechanism(GCM) is to provide as good(i.e. as high) an estimate of *Earliest Input Time(EIT)* as possible for every LP. Note that we don't necessarily require one global estimate for every LP. Computing good estimates for EIT is crucial for the **progress** of conservative processes, since, they process only the inputs that have timestamps less than EIT. On the other hand, the optimistic processes need good estimates of EIT to efficiently manage their memory space(by

collecting the fossils from *history*, *input* and *output queues*) and for committing I/O, but, not to guarantee local progress.

3.2.1 GVT based GCM

One simple global control mechanism is simply the *GVT* method used in Time Warp and other optimistic methods. In a global snapshot of the system, GVT is defined as the minimum of all the ECOTs and the timestamps of the messages in transit. [Bellenot 90] describes some *GVT* algorithms. Note that in our case, *ECOT* also takes into account the lookaheads specified by the LPs, hence, the *GVT* estimates computed might be better than the ones computed otherwise. The computed GVT forms an estimate of EIT for all the LPs in the system. It is assumed that any process can send messages to any other process in the system. If, however, the *predecessor set*, which is the transitive closure of the source set(the set of processes that the entity can receive messages from), is known for an LP, its EIT is equal to the minimum of ECOTs over only the predecessor set and the messages in transit from these processes. GVT with predecessor set optimization may compute better estimates for EIT for each entity but will allow inflexible communication topologies(discussed in detail in section 8.1).

3.2.2 Null message based GCM

Another possible global control mechanism is based on null messages, traditionally used in conservative methods. Every process sends the value of its *Earliest Output Time(EOT)* to all its destination processes using null messages. The *Earliest Input Time(EIT)* of a process is defined to be the *minimum of the highest values of EOT received from each of its predecessors.* It is initialized to ϵ for every LP. The frequency of sending null messages can be varied as long as the new value of EOT is eventually communicated. This scheme requires every process to know all its predecessors and successors(the whole system in the default case). Also, the estimates of EIT computed by each process may be different. This method is able to take advantage

of the knowledge of the communication topology in calculating EIT, and doesn't require any global communications. However, it doesn't allow an arbitrarily dynamic communication topology.

3.2.3 Combination

The third possibility is to execute both the *GVT* algorithm and the null message scheme together. They can be executed independently , since, they don't interfere with each other. The *EIT*, for any process, would be the maximum of the value computed by each method. Although, this would incur a larger overhead than either of the methods, it would give better estimates too.

4 Proof of Correctness

In order to simplify the proof, we assume that the communication topology is static, and that there is only one LP per processor. Let p, q, and r refer to the LPs corresponding to the PPs x, y, and z, respectively, for the rest of this section.

Lemma 1 *Processing of any input-set with timestamp t results in production of a finite number of outputs, all with timestamp greater than t.*

Proof:
If the input-set results in forward computation, it schedules a finite number of new messages all with timestamps greater than t(assumption 4). If it causes a rollback, a finite number of output-sets, all having outputs with timestamps $> t$, are cancelled. Recomputation uses a finite number of input-sets with timestamps $\geq t$, hence, sends a finite number of outputs, all having timestamps $> t$.

Lemma 2 (Extended Fidelity) *Let $IS_p(t)$ and $IS_x(t)$ be the input-sequences processed by the LP p and the PP x, respectively(starting in the same initial state). Let $OS_p(t)$ and $OS_x(t)$ be the corresponding output-sequences produced by them. If $IS_p(t) = IS_x(t)$, then, $OS_p(t) = OS_x(t)$, and p and x reach the same final state.*

$t - S[0].la \Rightarrow t' < eot_p - S[0].la \Rightarrow t' < min(eit_p, T_p)$. Therefore, p must have received an input with timestamp $t' < eit_p$ after sending eot_p. This contradicts the assumption that (m, t) is the earliest violator message in the system.

Case 2(GVT based GCM): The last GVT computation before the arrival of m must have returned a value eit_q which implies that there must have been a global snapshot S *before* the arrival of m whose GVT was $\geq eit_q$. In S, $T_p + la \geq eit_q, \forall p$, and all the messages in transit bear timestamps $\geq eit_q$(by the definition of GVT). Therefore, by the definition of lookahead, the system can't produce or deliver a message with timestamp less than eit_q. Hence, contradiction.

Let eit refer to the minimum EIT in the system, at any point during execution.

Lemma 4 (Safety) *Let $IS_p(eit)$ be the input-sequence* **received**, *with timestamp less than eit, by an LP p, at any point in execution. Let $IS_x(eit)$ be the input-sequence, with timestamp less than eit, processed by the PP x. Then, $IS_p(eit) = IS_x(eit)$, for any LP p and its corresponding PP x.*

Proof:

Assume the contrary. Let $t < eit$ be the *earliest* timestamp at which an $IS_p(eit)$ differs from $IS_x(i)$, for any LP p and its corresponding PP x. Therefore, $I_p(t) \neq I_x(t)$.

Since, t is the earliest time where the logical and physical sequences differ, $IS_q(t) = IS_y(t), \forall q$, and corresponding y.

Let $OS_y(t)$ be the output sequence with timestamp less than t for a PP y in the physical system. Consider an execution instant where all $IS_q(t)$ have been processed in the logical system. Let $OS'_q(t)$ be the output sequence with timestamp less than t produced at this point by an LP q. By lemma 2, therefore, $OS_y(t) = OS'_q(t), \forall y$, and corresponding q.

$I_x(t)$ is equal to the set of messages with timestamp t in $OS_y(t), \forall y$.

Since, $t < eit$, by lemma 3, all the messages in $OS'_q(t)$ have been delivered(assuming reliable communication), for all q. Therefore, $I_p(t)$ is equal to the set of messages with timestamp t in $OS'_q(t), \forall q$.

Therefore, $I_p(t) = I_x(t)$ (contradiction).

Lemma 5 *Eventually, all the messages in the system with timestamp less than eit will be processed.*

Proof:

All the messages ever to be generated in the system with timestamp $< eit$ have already been delivered(lemma 3). Since, optimistic LPs process available inputs (in increasing timestamp order) without blocking, and conservative LPs process inputs with timestamp less than their EIT (in increasing timestamp order), all the messages with timestamp $< eit$ will eventually be processed.

Lemma 6 (Liveness) *Eventually, the minimum EIT of the system will be* $\geq eit + \epsilon$.

Proof:

By lemma 5, all the messages with timestamps less than eit will eventually be processed. In such a state, there are only a finite number of messages in transit with timestamps in the interval $[eit, eit + \epsilon)$ (by lemma 1 and the fact that the system has processed only a finite number of input-sets so far).

Null message based GCM: The value of EOT for any LP p is $\geq eit + \epsilon$ (since $min(eit_p, T_p) \geq eit$). Eventually, a null message carrying this value will be delivered to all the LPs that p can send a message to. Therefore, eventually all the LPs would receive an EOT $\geq eit + \epsilon$ on each one of their input channels, thus, making their EITs $\geq eit + \epsilon$.

GVT based GCM: Eventually, all the messages in the interval $[eit, eit + \epsilon)$ which are in transit will be delivered. Any new messages generated will have timestamp $\geq eit + \epsilon$(lemma 1).

Eventually, the earliest unprocessed message at an optimistic LP will have a timestamp $\geq eit + \epsilon$. At this point, the value of their ECOT would be atleast $eit + \epsilon$. Since, the conservative LPs are required to know the value of ϵ, and their $T_p \geq eit$, the value of their ECOT would $\geq eit + \epsilon$ too. Therefore, the value of GVT is $\geq eit + \epsilon$. This

value of GVT will eventually be returned by the GVT computing algorithm and will be assigned to the EIT of every LP.

5 Dynamically changing modes of Operation

At some point in execution, some of the processes may want to change the mode of operation from conservative to optimistic or vice-versa, because of changed lookahead, network traffic or some other reason.

5.1 Conservative to Optimistic

This doesn't require any special transitory phase. The process can instantaneously change its mode and start following the rules of optimistic execution.

5.2 Optimistic to Conservative

There are two options possible:

5.2.1 Option 1

Rollback the process to state $S[0]$. Mark all the inputs with timestamp greater than $S[0].t$ as unprocessed. Discard all the *history* entries except $S[0]$. Change the mode to conservative and start executing. This option is simpler and instantaneous(doesn't require any transitory stage), but, causes some amount of recomputation.

5.2.2 Option 2

The process has to be in a transitory phase called *OtoC* as long as n is greater than 0 i.e. there are more than one element in the *history* . The local control mechanism of this state is given below. Assuming, the global control mechanism keeps advancing the *EIT*, it can be shown that eventually, there will be only one state left in the queue, namely, $S[0]$, at which point the process can switch to the conservative mode

of operation.

Local Control Mechanism for OtoC

An LP is eligible to be executed whenever it has an unprocessed input with timestamp $< S[n].t$. In case there are many such LPs, the one with earliest unprocessed message is selected.

Let p be the selected LP, and let $I(t)$ be the first input-set in its *input queue*.

- Find the largest i, such that $t > S[i].t$. Rollback upto $S[i]$, and recompute using the input-sets saved in the *history*.

Lazy cancellation and lazy recomputation optimizations can be used.

6 Relationship to Other Simulation Protocols

The distributed simulation scheme outlined above encompasses many of the conservative and optimistic schemes that we know of. If all the processes in the system use the conservative LCM, then using null message based GCM corresponds to the **Chandy-Misra-Bryant** algorithm. If, on the other hand, GVT based GCM is used, the resulting protocol is same as the **Conditional Event** algorithm [Chandy 89a]. Similarly, if all the processes use the optimistic LCM, along with GVT based GCM, the result is the traditional **Time Warp**[Jefferson 85] scheme.

Using a combination GCM with conservative processes gives rise to a **new conservative algorithm** whose implementation is discussed in [Jha 93]. Using the null message based GCM with optimistic processes gives rise to a **new optimistic** method. Using null messages to compute EITs in optimistic algorithms removes the need for global computations of GVT(or can be used in conjunction with GVT, as in combination GCM) and may give better estimates of EIT since it uses lookahead and the topology information in calculating it.

Using the predecessor set optimization for the GVT based GCM(hence applicable to conditional event algorithm and Time Warp) corresponds to calculating Target Specific Information as described in [Pancerella 93].

For the conditional event algorithm(conservative LPs, with GVT based GCM), we defined ECOT to be equal to the earliest unprocessed message plus the lookahead. This just gives a lower bound on the timestamp of the next conditional output The exact value of the next conditional output timestamp may be obtained by actually executing(tentatively, subject to rollback) the next input(in cases where it wouldn't be executed otherwise, since the processes are conservative). Using this estimate for the ECOT may result in better estimates for the GVT(at the expense of possible local rollback/state saving overheads). This corresponds to the **Breathing Time Bucket** algorithm [Steinman 92](with a synchronous GVT computing algorithm).

In section 2, we assumed for simplicity that each SP corresponds to one LP for the entire simulation time interval. If, however, we divided the entire simulation interval into n intervals and assigned all the LPs for one of the n intervals to one SP, and used the optimistic LCM for all the SPs, the resulting algorithm would correspond to a **Time Parallel** simulation [Lin 91].

7 Performance

In this section, we present some preliminary results on the performance of the CON-OPT protocol for a small synthetic benchmark. The benchmark was designed to isolate the type of configurations that are expected to perform better with a mixed protocol than with a pure optimistic or a pure conservative one. Rather than implement the protocol directly on a parallel architecture, we chose to first evaluate it in a simulated environment where the protocol could be studied for a wide range of overhead costs. The protocol was simulated using the Maisie[Bagrodia 92a] simulation language.

We use ***virtual time*** to refer to the timestamps of the application being simulated by CON-OPT, and *simulation time* to refer to the timestamps of the CON-OPT simulator. The execution time of actual messages, null messages, communication delays and other overheads in the actual parallel simulator are modeled by the *hold* and other Maisie constructs. Therefore, the simulation time it takes for the Maisie simulator to simulate CON-OPT for HORIZON amount of virtual time is an estimate of the real time it would take for CON-OPT, implemented on a parallel architecture (with number of processors equal to the number of LPs), to simulate the application for HORIZON amount of virtual time.

The benchmark is shown in Figure 1. The network consists of 5 processes. Processes C and D are source processes, E is a sink process and A and B are server processes that service incoming jobs in FIFO order. C generates *jobs* with a fixed inter-arrival time given by *serv_time*; for every job message generated by C, with probability P, it also generates and sends a *straggler* message with timestamp $t - 5 * serv\_time$. If A is an optimistic process, receipt of a straggler messages will cause A to rollback, and the frequency of rollbacks can be controlled by varying P. D sends a single message with timestamp 0; this message is serviced by B and with probability 1, is sent back to B. Communication topology is (potentially) a completely connected network between A and B. However, channels between A and B are never used to send any job messages (although they are used in the implementation of the GCM). The (virtual) service times for A and B are equal to *serv_time* and *10*serv_time*, respectively. Both of them take 1 unit of simulation time to execute any message.

We run the benchmark using both GVT based and null message based GCMs. Only overhead is rollbacks versus blocking time. We assume that the GCM computation messages (GVT related messages or null messages) incur no processing or transmission costs. Also, optimistic LPs don't incur any state saving overheads. Therefore, the only overhead that is incurred by an optimistic LP is due to recomputation (following a rollback) of certain messages. A conservative LP only suffers from

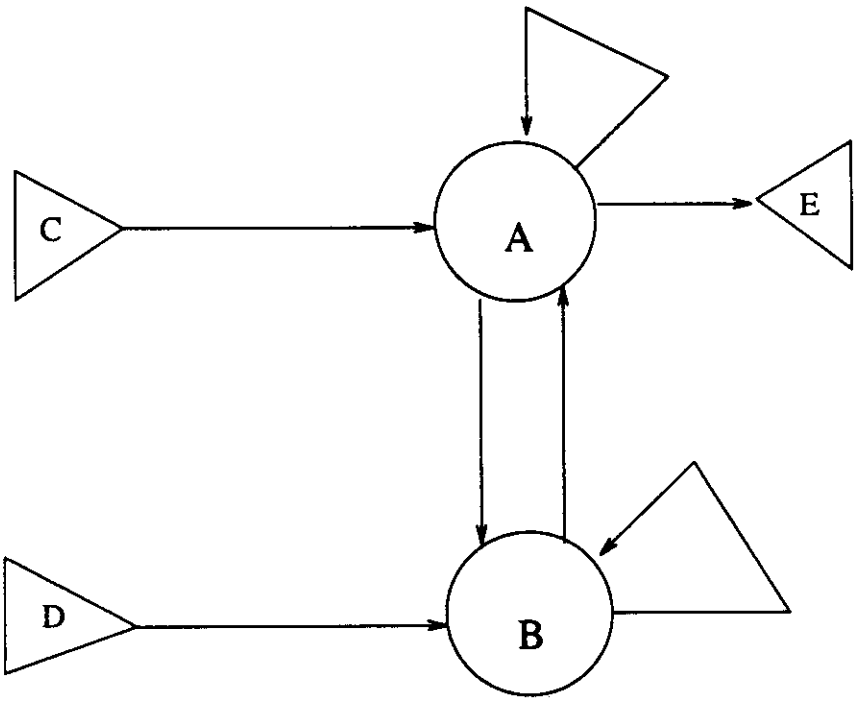

Figure 1: Network 1 simulated on CON-OPT

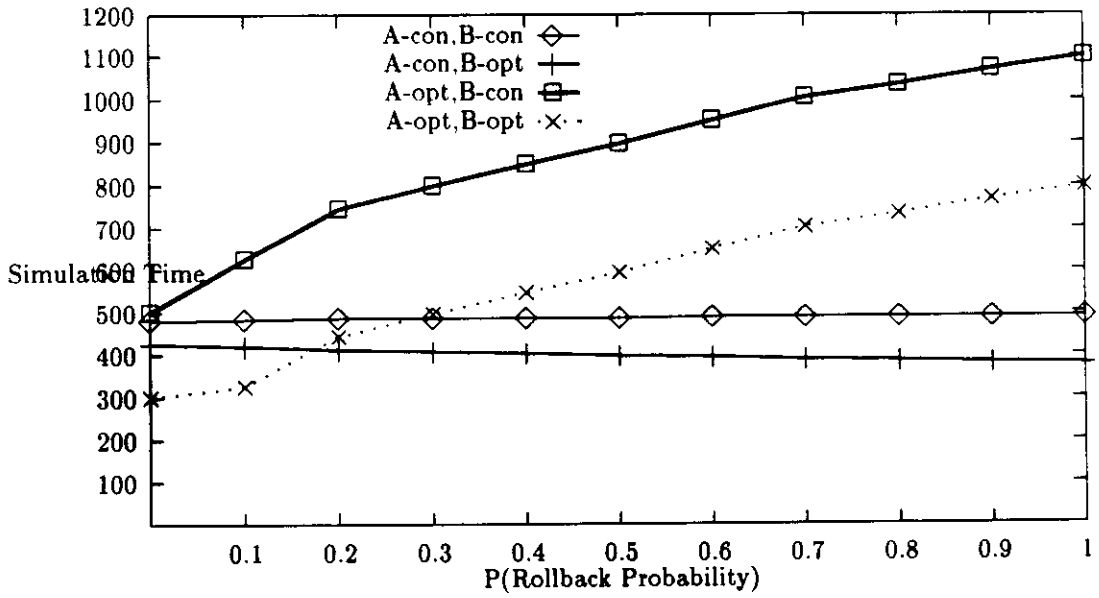

Figure 2: Network 1 using GVT based GCM: Simulation time versus P

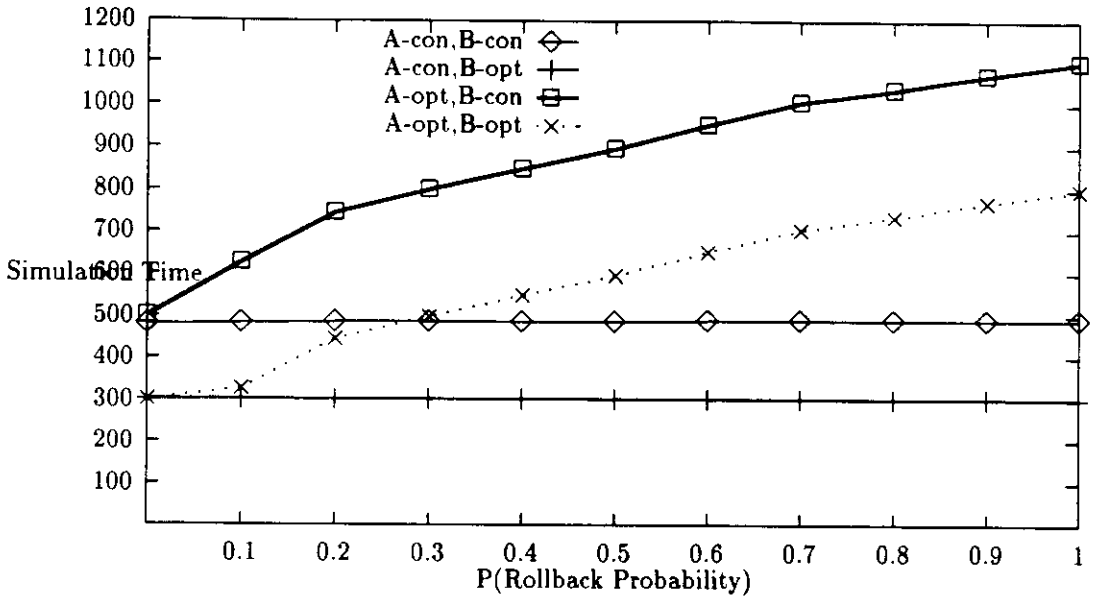

Figure 3: Network 1 using null message based GCM: Simulation time versus P

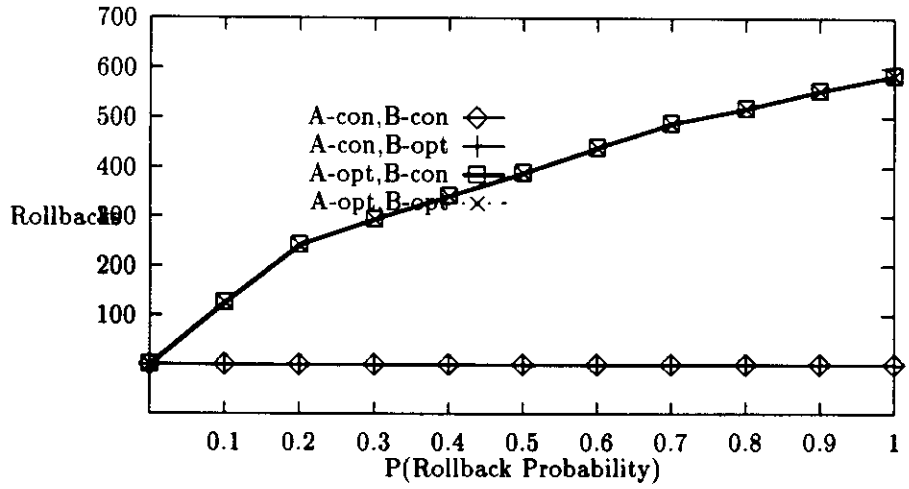

Figure 4: Network 1 using GVT based GCM: Rollbacks versus P

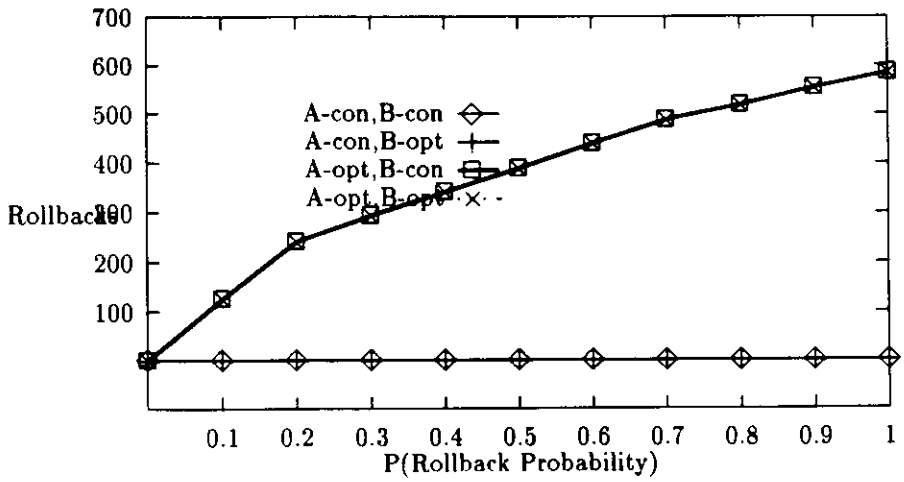

Figure 5: Network 1 using null message based GCM: Rollbacks versus P

overhead due to blocking of messages. Four different combinations are possible for the execution modes of the servers, and the performance of the protocol was determined for each mode. Each experiment was executed for a duration that was sufficient for each server to process 200 job messages.

Figure 2, and Figure 3 plot the simulation.time needed to simulate the network as a function of the probability P, for each of the 4 execution modes for the two GCMs. (the source and sink LPs behave the same irrespective of whether they are conservative or optimistic). It shows that for P equal to 0.0, a pure optimistic configuration is amongst the best (also better than the pure conservative configuration), whereas for higher values of P, the mixed configurations - A:CON, B:OPT is the best. This is to be expected since if A is executed optimistically, it would incur too many rollbacks. However, if both A and B are made conservative, all B's events would execute after A's since they have higher timestamps. Having A as conservative, and B optimistic allows them to execute their events independently (and in parallel). Figure 4 and Figure 5 plot the amount of rollback (total number of states rolled back during the entire execution) versus P. They confirm that the main contributing factor to the

relative execution times is the amount of rollback. Note that these experiments ignored overhead for computation of GVT as well as that due to state-saving.

8 Discussion

In this section, we discuss the assumptions that we make regarding the communication topology and the process lookaheads.

8.1 Communication Topology

An ability to support dynamically changing communication topology is often cited as a major advantage of optimistic schemes over the conservative ones. We believe, however, that whether a dynamic communication topology can be supported in a simulation protocol or not depends on the Global Control Mechanism used. Besides, there is a trade off between the performance gained by exploiting the knowledge of communication topology and the flexibility of a completely dynamic configuration.

A null message based GCM can't allow arbitrarily dynamic configuration. The main problem is illustrated in figure 6. A channel is to be created from process b to process c at time t. But, by the time this information reaches process c, the value of its EIT might already have been advanced beyond time t, causing a possible causality violation.

However, [Jha 93] describes how a restricted form of dynamic topology can be supported even in this case. Any channels created between the parent and the child process at the time the child process is spawned don't suffer from a possible causality violation described above. In order to create any other channel, say between b and c at time t, both b and c should receive channel creation messages with timestamp less than or equal to t on *existing* channels. In many applications this requirement is easy to satisfy. Once again, this restriction would apply to any protocol, optimistic or conservative, using the null message based GCM.

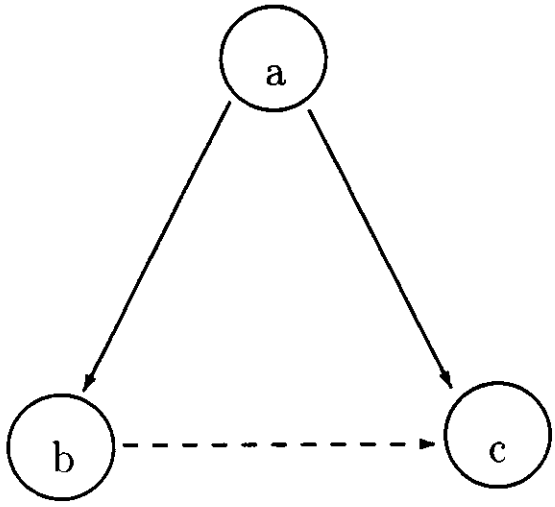

Figure 6: Creating channels dynamically

GVT based GCM assumes a completely connected network, and hence can support an arbitrarily dynamic configuration. Hence any simulation protocols, whether optimistic e.g. Time Warp, or conservative e.g. Conditional Event Algorithm or Breathing Time Bucket, that use a GVT based GCM, can support a dynamic configuration. Besides, they don't require the LPs to *know* the exact communication topology, since they assume a completely connected network.

Note, however, that using the predecessor set optimization, as described in section 3, for GVT calculation requires processes to know their predecessor sets. This would, therefore, restrict the dynamic creation of channels as described before.

8.2 Lookahead Assumptions

The basic assumption we make regarding lookahead is the existence of the positive constant ϵ (assumption 4) for every LP. It can be shown that existence of such a predetermined constant is also required for the progress of even pure optimistic methods like Time Warp. In CON-OPT, under the GVT based GCM, the conservative LPs need to *know* the value of the ϵ, whereas the optimistic LPs don't need to *know* the

value of ϵ, for the system to make progress. Under the null message based GCM, however, at least one LP (optimistic or conservative) in each cycle has to *know* the value of ϵ.

Under a different assumption, namely, that messages with same timestamp at a given LP can be processed in an arbitrary order (as long as they obey the causal order), it can be shown that pure conservative conditional event algorithm doesn't require even the existence of ϵ to make progress (with the obvious condition that systems with infinite physical messages take infinite amount of time to simulate), whereas Time Warp still does.

If the system has good lookahead properties (and the LPs know the value of lookahead), both GVT based GCM and the null message based GCM are able to utilize this knowledge to give better estimates of EIT. However, the null message based GCM does a better job of exploiting this knowledge since it combines it with the knowledge of communication topology. Hence, if the communication topology is sparse, message communication is regular, and lookaheads are good, it is preferable to use null message based GCM. However, if the topology is dense, there are small number of messages in the system or the communication is irregular, and the lookaheads are bad, a GVT based GCM would perform better. In case of these characteristics changing during the execution, either the GCM can be dynamically switched, or a combination GCM (which uses both null messages and GVT, and computes the maximum of the EIT values obtained from both) can be used.

9 Related Work

The efforts in combining the optimistic and conservative protocols have thus far concentrated on either adding limited amount of optimism to conservative protocols, or limiting the optimism of the optimistic protocols. The methods in the former class [Dickens 91], [Steinman 92] allow processes to compute messages optimistically, but,

potentially erroneous messages are not sent to other processes. The methods in the latter class [Sokol 88], [Reiher 89] attempt to reduce rollbacks by preventing certain LPs from progressing too far into the future. A third approach consists of forming clusters of LPs, using an optimistic protocol within each cluster, and a conservative protocol for inter-cluster synchronization [Rajaei 93]. The most promising approach, in our opinion, is one which allows different parts(LPs) of the system to choose, dynamically, to be either optimistic or conservative. Composite ELSA [Arvind 92] is one such protocol. However, it requires each message to contain extra information (to specify whether it is certain or guessed) whose processing may constitute extra overhead. Additionally, it requires the communication topology to be statically determined.

Introducing conservative LPs in an optimistic system can be compared to the throttling mechanisms described in [Reiher 89]. In Window based throttling, all the LPs in the system are allowed to process events only with in a specific time window. The problem with this approach is that it penalizes the LPs doing correct work also, along with those doing incorrect work. Also, the window size is sensitive to the application and the actual timestamps of the messages. The penalty based throttling tries to alleviate this problem by penalizing specific LPs that have done incorrect work (measured by the number of negative messages sent) recently. However, the problem in this case is that the only choice available is either to schedule a penalized process (in which case it continues to do incorrect work, and also incurs the overheads of state saving etc.) or to not schedule it all (in which case it does no work at all). CON-OPT provides the alternative of allowing it do guaranteed correct work (by making it conservative). Other advantages of CON-OPT include the possibility of using GCMs other than the GVT based GCM used in the throttling mechanisms.

10 Conclusion and Ongoing Research

We have described a distributed simulation scheme in which various logical processes executing in either optimistic or conservative modes can interact with each other correctly. The processes are also allowed to change their mode of execution on the fly.

Our protocol allows the user to combine the features so far thought to be characteristic of optimistic methods like dynamic configuration with conservative processes. Similarly, it allows optimistic processes to take advantage of features normally associated with conservative LPs e.g. exploiting the knowledge of lookahead and communication topology. Any of these features can be used in conjunction with a mix of conservative and optimistic LPs too.

Presence of a conservative process amongst optimistic ones doesn't restrict the progress of the optimistic ones in a big way. The optimistic processes won't receive the (possibly incorrect) messages from the conservative process that they would have received had the process been optimistic. However, they can still receive and process possibly incorrect messages from the optimistic processes and the correct messages from conservative processes, without blocking, which they would not have been able to if they had been conservative. The other thing that might happen is that the fossil collection in the optimistic processes might be delayed because the presence of a conservative process might lower the value of EIT computed.

Currently, CON-OPT allows only one GCM for the whole system (different LPs could be using different LCMs). We are working on extending it to allow different GCMs for different clusters of LPs, also allowing them to switch the GCM. We are in the process of implementing CON-OPT on a parallel architecture. Based on the insights gained from running different types of benchmarks on this implementation, we would formulate and implement policies to transparently switch LCM and GCM of individual LPs, and cluster of LPs.

References

[Arvind 92] D. K. Arvind, and C. R. Smart, Hierarchical Parallel Discrete Event Simulation in Composite ELSA. In *Proceedings of 6th Workshop on Parallel and Distributed Simulation*, pages 205-210.

[Bagrodia 92a] R. L. Bagrodia, and W. Liao, A Language for Iterative Design of Efficient Simulations. Technical Report UCLA-CSD-920044, Computer Science Department, UCLA, Los Angeles, October 1992.

[Bellenot 90] S. Bellenot, Global Virtual Time Algorithms. In *Proceedings of the Multiconference on Distributed Simulation.* January 1990, pages 122-127.

[Chandy 81] K. M. Chandy, and J. Misra, Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Communications of The ACM*, Vol. 24, No. 11, Pages 194-205, November 1981.

[Chandy 89a] K. M. Chandy, and R. Sherman, The Conditional Event Approach to Distributed Simulation. In *Proceedings of the SCS Simulation Multiconference on Distributed Simulation*, pages 93-99, March 1989.

[Chandy 89b] K. M. Chandy, and R. Sherman, Space-Time and Simulation. In *Proceedings of the SCS Simulation Multiconference on Distributed Simulation*, pages 53-57, March 1989.

[Dickens 91] P. Dickens, and P. Reynolds, A Performance Model for Parallel Simulation. In *Proceedings of the 1991 Winter Simulation Conference*, pages 618-626.

[Fujimoto 90] R. Fujimoto, Parallel Discrete Event Simulation. *Communications of The ACM*, Vol. 33, No. 10, pages 30-53, October 1990.

[Jefferson 85] D. Jefferson, Virtual Time. *ACM TOPLAS*, Vol. 7, No. 3, pages 404-425, July 1985.

[Jha 93] V. Jha, and R. Bagrodia, Transparent Implementation of Conservative Algorithms in Parallel Simulation Languages. To appear in *Proceedings of the 1993 Winter Simulation Conference.*

[Lin 91] Y. B. Lin, and E. D. Lazowska, A Time-Division Algorithm for Parallel Simulation. *ACM TOMACS*, Vol. 1, No. 1, pages 73-83, January 1991.

[Pancerella 93] C. M. Pancerella, and P. F. Reynolds, Disseminating Critical Target-Specific Synchronization Information in Parallel Discrete Event Simulations. In *Proc. of 7th Workshop on Parallel and Distributed Simulation*, pages 52-59, May 1993.

[Rajaei 93] H. Rajaei, R. Ayani, and L. Thorelli, The Local Time Warp Approach to Parallel Simulation. In *Proc. of 7th Workshop on Parallel and Distributed Simulation*, pages 119-126, May 1993.

[Reiher 89] P. Reiher, and D. Jefferson, Limitation of Optimism in the Time Warp Operating System. In *Proceedings of 1989 Winter Simulation Conference*, pages 765-769.

[Sokol 88] L. M. Sokol, D. P. Briscoe, and A. P. Wieland, MTW: A Strategy for Scheduling Discrete Simulation events for Concurrent Execution. In *Proceedings of SCS Conf. on Distributed Simulation*, pages 34-42.

[Steinman 92] J. Steinman, SPEEDES: A Unified Approach to Parallel Simulation. In *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*, pages 75-83.