Space Uncertain Simulation Events: some Concepts and an Application to Optimistic Synchronization

Francesco Quaglia and Roberto Beraldi
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy

Abstract

The notion of temporal uncertainty, expressed as the possibility for an event to occur in an interval of simulation time, rather than at a specific instant, has been recently proposed and exploited in order to enhance the performance of parallel/distributed simulation systems. In this paper we propose the concept of "spatial uncertainty" expressed as the possibility for a simulation event to occur in one of a set of points within the simulated system space. How to handle/exploit space uncertain events in an optimistic simulation system is discussed. Also, experimental results for optimistic simulations of a Personal Communication System (PCS) modeled with space uncertain events are reported. These results show the ability of spatial uncertainty to increase the performance of the parallel simulation system by providing a more flexible approach to synchronization.

1 Introduction

Increasing the performance of a parallel/distributed simulation system, whether it be based on conservative or optimistic synchronization, is still a core issue to address. To this purpose, the notion of temporal uncertainty of simulation events [6] has been recently proposed and exploited, especially in the context of federated simulation systems. Basically, a time uncertain simulation event is an event for which we do not define a precise timestamp for its occurrence. Instead, we admit an interval of simulation time within which it eventually occurs. The advantage of temporal uncertainty derives from the fact that the execution of events at any Logical Process (LP) is ordered on the basis simulation time intervals, rather than precise timestamps, with any two events considered as concurrent in case the corresponding intervals overlap and they are not causally dependent on each other. This increases the flexibility of synchronization since concurrent events lead to the admission of multiple correct event execution orders at an LP. In the context of optimistic synchronization, this allows reducing the amount of rollback in the parallel execution [2], in the case of conservative synchronization it allows overcoming problems due to poor lookahead in the simulation model [6, 11].

Actually, temporal uncertainty has its justification in the fact that any simulation is only an approximation of the simulated system. Therefore there are some aspects that might not be directly captured by the simulation model; we call this phenomenon as *discrepancy*. Assigning an uncertainty to the occurrence time of a simulation event is a way to express the existence of the discrepancy. As an example, consider a wireless communication system simulation, where the service area is partitioned into cells. Typically, the simulation model assigns to each cell a defined geometric shape in a way that there is no overlapping between cells (e.g. cells are modeled as non-overlapping hexagons). On the other hand, real wireless systems have cells that overlap, thus giving rise to portions of the coverage area that are served by multiple base stations. This implies that, in the real system, the hand-off procedure of a mobile that switches between cells can occur when the mobile is in any point of the overlapped portion of the coverage area. This peculiarity, not explicitly captured by the simulation model, is some way expressed by assigning a temporal uncertainty to the occurrence time of the hand-off event in the simulation model.

In this paper we present the concept of *spatial uncertainty* that expresses the possibility for a simulation event to occur in a spatial region of the simulated system, rather than in a precise point. As for the case of temporal uncertainty, spatial uncertainty is a mean to express the previously mentioned discrepancy phenomenon, which might appear in specific simulation applications. To make clear the latter assertion, consider again the example above regarding the wireless system simulation, where cells are modeled as non-overlapping hexagons. Consider a mobile within a cell, say cell A, that moves outside this cell, thus giving rise to a hand-off event in the simulation. Also, suppose, as shown in Figure 1.a, that an exact point p is defined for the hand-off just around a corner of cell A. In this case the simulation model only admits the possibility for the mobile to enter cell B. However, in the real system we might have overlapping between cells B and C around point p therefore either cell might accept the mobile and serve it. This discrepancy can be captured, and expressed, by assigning to the hand-off event a spatial region, say the segment L in Figure 1.b, thus re-introducing in the simulation model the possibility that either cell B or cell C accepts the mobile as it could occur in the real system.

Similarly to temporal uncertainty, spatial uncertainty can provide a mean to improve the run-time behavior of parallel/distributed simulation systems. Specifically, the possibility for an event to occur in any point within a spatial region means that multiple LPs in the parallel/distributed simulation system, i.e. all those LPs modeling a portion of that

Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS'04)
1087-4097/04 $20.00 © 2004 **IEEE**

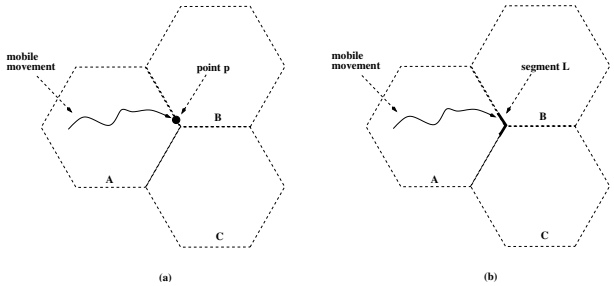

Figure 1. Hand-off in a Point p (a) or on a Segment Region L (b).

spatial region, might take care of the execution of that event. Therefore, the decision on which LP should really execute that event can be taken at run-time in a way to improve specific performance metrics for the parallel/distributed simulation system. This aspect, together with mechanisms for handling space uncertain events, will be explored in this paper for the case of optimistic simulation. Then we also report the results of an experimental study on optimistic simulations of a Personal Communication System (PCS) based on spatial uncertainty, thus providing a quantification of the performance benefits that can be achieved just through the exploitation of such a kind of uncertainty.

The remainder of this paper is structured as follows. In Section 2 we present a simple, pragmatical model for space uncertain events and provide indications on how to map the model on parallel simulation methodology. Section 3 presents a discussion on how to exploit spatial uncertainty in optimistic synchronization, and describes additional mechanisms required by an optimistic simulator to handle the uncertainty. The results for the experiments with the PCS application are reported in Section 4.

2 Spatial Uncertainty in Parallel Simulation

In this paper we are interested in spatial uncertainty as a mean to improve the run-time behavior of a parallel/distributed simulation system. In particular, we focus on the reduction of the wall-clock time on a whole run for an optimistically synchronized parallel simulator. For this reason we provide in this section the description of an event model with spatial uncertainty that well fits the requirements of parallel/distributed simulation methodology. However, we argue that the notion of spatial uncertainty might be also exploited in other ways that we reserve for future work. As an example, a space uncertain event might give rise to multiple different outcomes depending on the decision we take on the final position for the occurrence of the event. Therefore, we might associate with each space uncertain event a decision point that enables cloning of the simulation [8]. This might allow fast exploration of multiple execution paths due to sharing of portions of the computation on different paths, independently of whether the simulator executes sequentially or concurrently.

The remainder of this section is structured as follows. We first introduce the event model expressing spatial uncertainty as an extension of a classical event model for parallel simulation, then we provide hints on the correctness of the execution of a simulation model with space uncertain events.

2.1 A Model for Space Uncertain Events

A discrete event simulation model is classically viewed as a set $S = \{s_1, s_2, \ldots, s_n\}$ of state variables, also referred to as *state of the simulation model*, and a sequence of state changes, also referred to as events, occurring at discrete points (timestamps) in simulation time.

In parallel discrete event simulation, the state S is partitioned into disjoint subsets of state variables S_j (with $j \in [1, m]$) such that $S = \bigcup_{\forall j \in [1,m]} S_j$. The peculiarity of such a partitioning is that each event, namely each state transition, involves a single subset S_j. Therefore, the state transition function associated with a given event takes as input values only the variables within S_j and updates only those same variables. In practice, this is implemented by assigning the management of each subset S_j of state variables to a specific LP, namely LP_j, which is entirely responsible for the execution of the state transition function associated with the event. We can model such a classical approach by associating with each event e the tuple $\tau(e) = [timestamp, content, position]$, where $timestamp$ is the simulation time for the event occurrence, $content$ is its content (which determines the state transition function to be invoked), and $position$ is a value in the interval $[1, m]$. The latter value represents the index of the subset of state variables involved in the state transition associated with the event, therefore the state variables involved in the transition due to the event e are only those in $S_{\tau(e).position}$ ([1]).

The notion of spatial uncertainty we propose is based on an extension of the previous classical perspective, and deals with the partitioning of the simulation model state S into the previously mentioned disjoint subsets S_j. Specifically, we associate with the event e a tuple $\tau(e) = [timestamp, content, \Pi]$, where the first two elements have the same meaning as above, while Π is a non-empty set of values, all different from each other, belonging to the interval $[1, m]$. Π specifies multiple possibilities as respect to the subset of state variables that are involved in the state transition associated with the event e. Therefore, it expresses uncertainty on where, within the state of the simulation model, the event e will have an effect. In practice this means that we have multiple LPs that can take care of the execution of the event e, which expresses the fact that the state transition associated with the event e can involve one of a set of distinct parts of the system under investigation.

A special instance occurs when $|\tau(e).\Pi| = 1$. In this case we obtain that the event e has no degree of uncertainty since, as respect to the partitioning of the state S, we exactly know which is the subset of state variables where the state transition function associated with $\tau(e).content$ needs to be applied. This also shows how the classical approach with no spatial uncertainty can be seen as a special case of a simulation with space uncertain events, namely the case in which the set Π has cardinality one for all the events.

[1] We use the notation $\tau(e).x$ to indicate the value of element x within the tuple associated with the event e.

2.2 Correctness Criteria with Spatial Uncertainty

When dealing with parallel simulation, we need to address the problem of correctness for the model execution, i.e. the synchronization problem. This is because concurrency is admitted for the execution of events acting on distinct subsets of state variables S_j (i.e. distinct LPs) and the execution of an event acting on a given subset can schedule events destined for other subsets, therefore we have dependencies among state transitions occurring on different subsets of state variables.

For the case of no spatial uncertainty, we say that the simulation model execution is correct if, for each pair of events e and e' such that $\tau(e).position = \tau(e').position$ and $\tau(e).timestamp < \tau(e').timestamp$, then the state transition function associated with $\tau(e).content$ is executed before the one associated with $\tau(e').content$. This is reflected by the classical perspective according to which the execution of events at each LP in non-decreasing timestamp order is a sufficient condition for the correctness of the simulation model execution, i.e. the correctness of simulation results. We want to exploit this same perspective for the case of simulation models with spatial uncertainty. In other words, we want to define a property of the model execution such that timestamp ordering is preserved at each LP. A natural way to approach such an issue is to define a property based on the positions we admit for each event and on how the admitted positions combine for distinct events within the model execution itself.

Before proceeding, let us introduce an additional, simple notation: we denote as $location(e)$ the value belonging to $\tau(e).\Pi$ that identifies where, within the simulation model state S, the state transition function associated with $\tau(e).content$ really acts. In other words, $S_{location(e)}$ identifies the subset of state variables really involved in the execution of that transition function while the model execution proceeds. Hence, $LP_{location(e)}$ is the LP that really takes care of the execution of this event.

We can now envisage the following property, we call ACceptability ($\mathcal{AC}$), for the model execution:

Property 2.1 - ACceptability ($\mathcal{AC}$).
Model execution is acceptable if, for each pair of events e and e' such that $(\tau(e).\Pi \cap \tau(e').\Pi \neq \emptyset) \wedge (\tau(e).timestamp < \tau(e').timestamp)$, at least one of the following two conditions holds: (1) the state transition function associated with $\tau(e).content$ is executed before the state transition function associated with $\tau(e').content$; (2) $location(e) \neq location(e')$.

In other words, property $\mathcal{AC}$ asserts that, given any pair of events e and e' having the potential to work on a same subset S_x of state variables within the state of the simulation model, we accept to process them with no ordering constraint among each other in case we are able to select distinct locations for them while the model execution proceeds. Trivially, in any acceptable model execution, for each subset of state variables S_x, events eventually located on this subset are all executed in non-decreasing timestamp order, which translates in that each LP_x is guaranteed to execute its events in non-decreasing timestamp order. Therefore the $\mathcal{AC}$ property can be seen as a criterion that expresses correctness of the model execution in the same classical terms that characterize simulations with events having no spatial uncertainty. At the same time, any parallel/distributed simulation system built to ensure the $\mathcal{AC}$ property could take advantage from higher flexibility while synchronizing the LPs. Specifically, the flexibility might come out by proper specification of the function $location(e)$ determining which LP really takes care of the execution of the event e. In the next section we discuss a scheme, layered on top of classical optimistic synchronization, to build simulation systems providing acceptability in the model executions as expressed by Property 2.1.

3 Optimistic Synchronization with Space Uncertain Events

We focus on the classical approach to parallel/distributed simulation, where LPs exchange messages to notify each other about newly scheduled events. In this context we are interested in the relation between the optimistic approach to synchronization, as designed for events with no spatial uncertainty [9], and the notion of acceptable model execution in the sense of Property 2.1. More precisely, we aim at extending basic mechanisms underlying the optimistic approach to synchronization to embed within the simulation system the ability to achieve, in an efficient way, an acceptable model execution. The first step towards this direction is to properly define an algorithm specifying the function $location()$, namely an algorithm that establishes how to determine final positions for the events. To this purpose, we introduce an algorithm called Rollback If No Additional Chance (RINAC). It specifies the function $location()$ in the attempt to keep low the occurrence of rollback. The last part of this section is devoted to the management of cancellation of events when considering that their location is established through RINAC. As we shall discuss, this issue will be tackled through a proper Multicast Cancellation scheme (MC).

3.1 Rollback If No Additional Chance

When we need to define an algorithm specifying the function $location()$ in a way to provide an optimistic simulation system supporting the $\mathcal{AC}$ property, several different approaches might be devised. As a first example, upon the scheduling of the event e by a given LP, say LP_x, it might be possible to collect information, e.g. via polling/reply messages, on the current simulation clock value for all the LPs in the set $\phi = \{LP_j | j \in \tau(e).\Pi\}$. Then LP_x might address the event e to the LP in the set ϕ, say LP_y, having simulation clock lower than $\tau(e).timestamp$, if any. In such a case, unless the simulation clock of LP_y oversteps $\tau(e).timestamp$ due to processing of other events before the notification message for e finally arrives at LP_y, the event e can be processed by LP_y in timestamp order, thus not causing violation of the $\mathcal{AC}$ property ([2]). This avoids the need for rollback procedures to recover acceptability in the model execution. Another approach would be to let all the LPs in the set ϕ to execute an explicit coordination algorithm, e.g. via rounds of messages, to resolve the location

[2] This is because, there is no event e' different from e, with $\tau(e').timestamp > \tau(e).timestamp$, such that $location(e') = location(e)$ and the state transition function associated with $\tau(e').content$ is executed before the one associated with $\tau(e).content$.

Behavior of LP_i upon the receipt of a notification message $m[e, visited]$:

1. **if** ((local simulation clock $\leq \tau(e).timestamp$) $\vee$ $(visited = \tau(e).\Pi - \{i\})$)
2. accept the event e for processing;
3. **else**
4. $visited = visited \cup \{i\}$;
5. select $x \in \tau(e).\Pi - visited$;
6. forward $m[e, visited]$ to LP_x;

Figure 2. The RINAC Algorithm.

for the event e, again with the aim to address the event e itself to an LP that can process it in timestamp order. On the other hand, the overhead due either to polling/reply messages sent/collected by LP_x or to the explicit coordination algorithm among the LPs in ϕ might be excessive, thus giving rise to a simulation system unable to really exploit spatial uncertainty for performance purposes.

We are interested in this paper in the definition of an algorithm specifying the function $location()$, which can be implemented with minimal overhead. More precisely, we neither want to collect information about current simulation clocks of the LPs, nor want to execute an explicit coordination algorithm. Instead, our proposal is based on a forwarding mechanism according to which an LP that receives a notification message for an event e accepts the event in case its simulation clock is lower than the event timestamp. Otherwise it rejects the event, and the notification message is forwarded to another LP among the possible destinations for that event. This is done in the hope that the next destination LP is able to maintain acceptability in the model execution without the need for rollback procedures. To avoid cycles of forwarding, that might lead to a situation in which no LP eventually accepts the event (thus the event itself is not eventually executed), forwarding can be done only among the set of LPs not yet visited by the notification message. If all the possible destination LPs have been visited, the LP lastly visited in the forwarding scheme cannot reject the event, even in case the event timestamp is lower than the current simulation clock of that LP. Therefore, we need to execute rollback on that LP to recover acceptability in the model execution since there is no more chance to visit another LP in the forwarding scheme. Hence the name RINAC (Rollback-If-No-Additional-Chance) for the algorithm we propose to specify the function $location()$.

The pseudo-code for RINAC is shown in Figure 2. Actually, to support RINAC we assume each notification message m for a given event also carries a set of values, called $visited$, initially empty, that contains the identifiers of all the LPs that have already been visited during the forwarding scheme. By line 1 of the algorithm, LP_i accepts to process the event in case no rollback needs to be executed or it is the last LP visited through the forwarding scheme. Otherwise it updates the $visited$ set to include its identifier, namely the value i (see line 4), selects a not yet visited LP among the possible destinations for the event (see line 5) and then forwards the notification message to that LP.

We argue that the cost of managing RINAC is definitely lower than the cost of collecting information on current simulation clock values, or executing explicit coordination. On the other hand, as respect to those approaches, RINAC might increase the time required to explore all the possible destination LPs on time for the event to be addressed to any of them and processed in timestamp order. This is because possible destinations are explored serially through the forwarding of the event notification message. Such an additional delay in the exploration of the possible destinations might cause non-minimal rollback length since we are actually delaying acceptance of the event, with possible execution of rollback, until we become aware that there is no chance to process the event in timestamp order at any LP among the possible destinations. Anyway this phenomenon is expected to have a limited impact each time the set $\tau(e).\Pi$ has relatively low cardinality. Also, non-minimal rollback length might be anyway acceptable if we are able to keep low the frequency of rollback occurrence by carefully locating the events among the LPs, which is exactly the aim of RINAC. Overall, RINAC represents a tradeoff between the cost of management of spatial uncertainty and advantages from its exploitation.

We also note that RINAC might be particularly effective in case the simulation system makes use of the so called kernel approach to implement core functionalities. Specifically, when using this approach, the LPs are typically executed as application level threads, and each processor runs an instance of a dispatcher thread (DT) that controls and schedules a set of LPs for event execution according to a given algorithm, e.g. Smallest-Timestamp-First [10]. DT has control on the event queues of those LPs, on their current simulation clock values, and so on. In this case, upon the arrival of an event notification message at DT, the RINAC algorithm is used to verify whether the condition in line 1 holds for the destination LP. If the condition does not hold and forwarding of the notification message is required towards another LP belonging to the set controlled by that same DT, then the message does not really need to be re-sent since it is already buffered within a memory area accessible by DT itself. This reduces the real number of message send operations performed to explore the set of possible destinations for a given event.

There is a final observation we would like to bring to the reader's attention. As already pointed out in Section 2.2, the function $location(e)$ identifies the LP that finally executes the event e. This means that the value of $location(e)$ might theoretically be considered as not definitely determined until an LP really executes the event e. In practice, RINAC determines the value of $location(e)$ by definitely establishing which LP accepts to eventually process the event e, with no possibility to switch to another LP even in case the event e remains unprocessed for a while. In other words, with no loss of correctness, we are studying the case in which the final position for the event is established prior to the event is really executed. It is established just upon the acceptance of the event, for future processing, at a given LP. We reserve for future work investigations on the possibility to keep the establishment of the location provisional until the event execution really occurs, which might further increase flexibility in the synchronization scheme.

Behavior of LP_x when cancelling the event e:
1. $\forall i \in \tau(e).\Pi$
2. send $m^-[e]$ to LP_i;

Figure 3. Multicast Cancellation.

3.2 Multicast Cancellation

When the location of events is established through RINAC, we need to consider an adequate scheme for cancelling a scheduled event e due to rollback. Specifically, the main problem in the cancellation issue is that, when LP_x needs to cancel an event e it has previously scheduled, it might have no knowledge on the identity of the LP that finally accepted the event e for processing. Therefore it has no knowledge on where to send the antimessage for the event e. As for the algorithm specifying the function $location()$, we do not want LP_x to collect information about the final location for the even e prior to sending the antimessage. This is due to both the overhead for information collection and the fact that this approach might excessively delay the send operation of the antimessage itself. Therefore, we have devised a cancellation scheme called Multicast Cancellation (MC), according to which LP_x multicasts the antimessage for the event e to all the LPs defined as potential locations for the event e.

The pseudo-code for MC is shown in Figure 3, where the notation m^- is used to specify we are sending an antimessage, i.e. a negative message. With MC we expect to promptly catch and annihilate the event e that needs to be cancelled, this is because all the possible destinations are notified about the need for cancellation at once. Therefore we do favor the potential to annihilate the event e prior it is really executed. On the other hand, with this scheme we need to send $|\tau(e).\Pi|$ antimessages for each event e to be cancelled. However, we argue that the communication overhead due to multicasting can be kept low in any case the cardinality of $\tau(e).\Pi$ is relatively low. Also, the number of real send operations might be reduced in case proper solutions for multicasting can be included within the simulation system. This might be the case of the kernel based approach mentioned in the previous section, since sending a set of antimessages to LPs controlled by the same dispatcher thread (DT) could actually be implemented by sending a single antimessage to that DT, indicating the identities of all the receivers. Also, if the communication layer provides facilities for efficiently multicasting information, then MC could actually be implemented through a multicast send operation with adequate specification of the multicast group. We note that MC simply specifies where to send the antimessages in case we deal with spatial uncertainty resolved through RINAC. However it does not indicate when the send operation should occur. This means that MC can be coupled with either lazy or aggressive policies for the definition of the time instant for the send operation of antimessages [7].

Actually when dealing with MC, there is the possibility that the antimessage $m^-[e]$ arrives at the LP, say LP_y, that finally accepts the event e before the arrival of the corresponding positive message $m[e, visited]$. This is because the message $m[e, visited]$ can be delivered to LP_y through the forwarding scheme after the delivery of $m^-[e]$. However, this is not a problem since classical optimistic synchronization admits the possibility for an antimessage to arrive before the corresponding message (e.g. for example due to non-FIFO message delivery between LPs). In this case the antimessage is simply kept within a queue of pending antimessages. It will eventually annihilate the corresponding message upon its arrival.

The real problem with MC is that the event e is accepted by a single LP, while all the LPs that are potential destinations for the event e receive the antimessage $m^-[e]$. Therefore we obtain annihilation between m and m^- on a single LP, while on all the other LPs the antimessage $m^-[e]$ remains pending. This creates problems in case the event e, after being cancelled, is re-scheduled. In this case there is the possibility that a new instance of the message $m[e, visited]$ arrives at an LP that accepts the event, but that holds a copy of a pending antimessage $m^-[e]$. In this case, the new instance of the event e is cancelled via an antimessage destined for cancellation of the previous instance. Therefore we obtain that the event disappears from the system, which obviously means incorrect execution of the simulation model. To avoid this problem we can piggyback on each notification message for an event e a serial number, established by the LP that scheduled the event e, which uniquely identifies that message, i.e. that instance of the event, in the system . At the same time, all the antimessages multicasted for catching a given message should carry that same serial number, so that annihilation occurs only in case the serial numbers on m and m^- coincide at the LP that accepted the event. With this solution we avoid cancelling an event via an antimessage destined for cancellation of a previous instance of that same event, thus no event disappears in the system.

We note that an antimessage $m^-[e]$ remaining pending at an LP due to the fact that the event carried by the corresponding message $m[e, visited]$ is not accepted/delivered by/to that LP, can be eventually discarded just while executing fossil collection procedures. Specifically, upon computation of Global Virtual Time (GVT), $m^-[e]$ can be discarded in case the GVT value is larger than $\tau(e).timestamp$. Therefore the storage usage due to pending antimessages that will not eventually match with the corresponding positive message can be run-time controlled via classical fossil collection mechanisms.

4 An Experimental Study

In this section we report the results of an experimental study carried out using a Personal Communication System (PCS) simulation application as a test-bed. In this study, we would like to verify whether spatial uncertainty, exploited through RINAC, is able to really provide a more flexible approach to synchronization, with respect to classical optimistic synchronization, thus giving rise to better run-time behavior of the simulation system. On the other hand, we also would like to observe whether the simulation model with space uncertain events provides performance indices for the PCS system (e.g. call-block frequency) that exhibit no significant error with respect to the values that can be obtained in case of no uncertainty in the model when consider-

ing the same amount of events simulated by the simulation system. In other words, we would like to observe the impact of spatial uncertainty on the accuracy of the simulation results that are produced. The remainder of this section is organized as follows. We first provide details on the testing environment we have used. Then we present the simulation model and the performance metrics we have selected. Finally, we report the obtained results.

4.1 Testing Environment

The experiments were all performed on a myrinet cluster of 4 Pentium III 866 MHz (256 Mbytes RAM). All the PCs of the cluster run LINUX (kernel version 2.2.15) and are equipped with M3M-PCI64C myrinet cards employing LANai 9 technology. CCL v3.0 [12] has been used to support message passing. Message exchange among LPs hosted by the same machine does not involve operations of the CCL layer. We make use of the kernel approach to implement the core functionalities of the simulation system, therefore each LP is implemented as an application-level thread and there is an instance of an optimistic simulation engine, namely the dispatcher thread (DT), on each machine. DT manages the local event list (resulting as the logical collection of the event lists of the local LPs) and schedules LPs for event execution according to the standard Smallest-Timestamp-First algorithm [10]. Dynamic memory allocation/release based on standard `malloc()/free()` calls is adopted for the entries of the event lists. Checkpointing is performed for any LP before the execution of each new event by exploiting functionalities provided by CCL. GVT calculation and fossil collection are executed periodically. Finally, antimessages are sent on an aggressive basis. Also, in case of MC, one antimessage for each potential receiver LP of the event to be cancelled is sent, therefore no specific optimization has been implemented while supporting this type of cancellation.

4.2 Simulation Model

In a PCS system, base stations provide communication services to mobile units. In our simulation model the service area is partitioned into cells, each modeled by a distinct LP. A cell represents a receiver/transmitter having some fixed number of channels allocated to it. The model is call-initiated [5] since it only simulates the behavior of a mobile unit during conversation, i.e. the movement of a mobile unit is not tracked unless the unit itself is in conversation. Therefore, the model is organized around two entities, namely cells and calls. Call requests arrive to each cell according to an exponential distribution [3, 4, 5] with inter-arrival time t_{int} seconds. All the calls initiated within a given cell are originated by the LP associated with that cell, therefore no external call generator is used. The state vector of any LP records statistics, information about busy channels and, for each channel, information about features of the mobile unit involved in the ongoing call (e.g. scheduled call termination time, call initiation time, class of the mobile unit etc.), if any. As a result, the size of the state vector depends on the amount of channels associated with the cell. We have selected a model with 100 channels per cell, which gives rise to LP state vector size of about 4 Kbytes.

There are three types of events, namely hand-off, due to mobile unit cell switch, call termination and call arrival. A call termination simply involves the release of the associated channel, whose identifier is maintained into the event compound structure, and statistics update. A call arrival checks if there is at least an available channel. In the negative instance, the incoming call is simply counted as a block, otherwise an available channel is allocated for the call. When a hand-off occurs between adjacent cells, the hand-off event at the cell left by the mobile involves the release of the channel and statistics update. Similarly, the hand-off event at the destination cell checks for channel availability, and allocates an available channel, if any, for the call. If there is no available channel, then the call is simply cut off (dropped). In our model there are two distinct classes of mobile units. Both of them are characterized by residence time within a cell which follows an exponential distribution [3], with mean 3 minutes (fast movement units) and 30 minutes (slow movement units), respectively. The average holding time for each call associated with both fast and slow movement units is 2 minutes. When a call arrives at a cell, the type (slow or fast) of the mobile unit associated with the incoming call is selected from a uniform distribution, therefore any call is equally likely to be destined to a fast or a slow movement mobile unit.

Cells are modeled as hexagons, therefore all the cells, except bordering cells of the coverage area, have six neighbors. In the experimental study we have varied the model size (i.e. the amount of cells, namely the amount of LPs) between 16 and 256, with even distribution of the LPs on the 4 machines of the cluster. Variation of the model size allows us to observe the simulation system behavior while varying the degree of parallelism in the model execution which, in its turn, typically gives rise to different rollback patterns. We have set the expected call inter-arrival time per cell t_{int} to 1.4 seconds, thus obtaining channel utilization factor of about 85%.

In this simulation model, a single type of events can be scheduled among distinct LPs, namely the hand-off events. Specifically, when the mobile involved in conversation switches out from a cell, the corresponding LP schedules a hand-off event for the destination cell. We use a classical random walk model for mobility [1], which means each mobile can switch out from a cell in any point along the border of the cell. However, on top of random walk we build an uncertainty segment that establishes a spatial region for the hand-off. More precisely, we establish through random walk the median point of a segment L that represents the whole spatial region where the hand-off can occur. We have varied the length of the segment L between 0% and 95% of the length of a single side of the hexagonal cell. The case of 0% means that the length of L is null, therefore we obtain an exact point for the hand-off, just defined according to classical random walk; in other words, we have a simulation model with no spatial uncertainty since the destination cell (i.e. the destination LP) for the hand-off is precisely defined. Instead, for non-null values of the length of L we have a model in which there might be uncertainty on the destination cell for the hand-off. Specifically, there might be two possible destinations. Obviously, the larger L, the higher the likelihood that the hand-off is uncertain in space

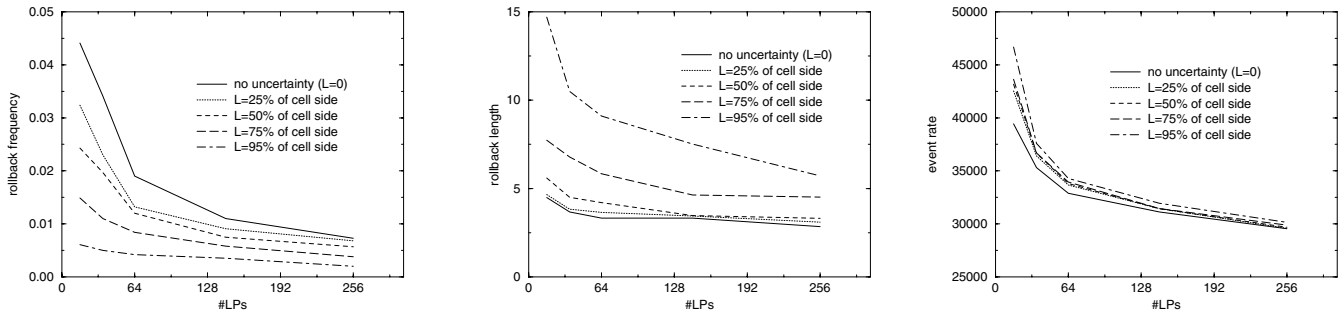

Figure 4. Performance of the Simulation System.

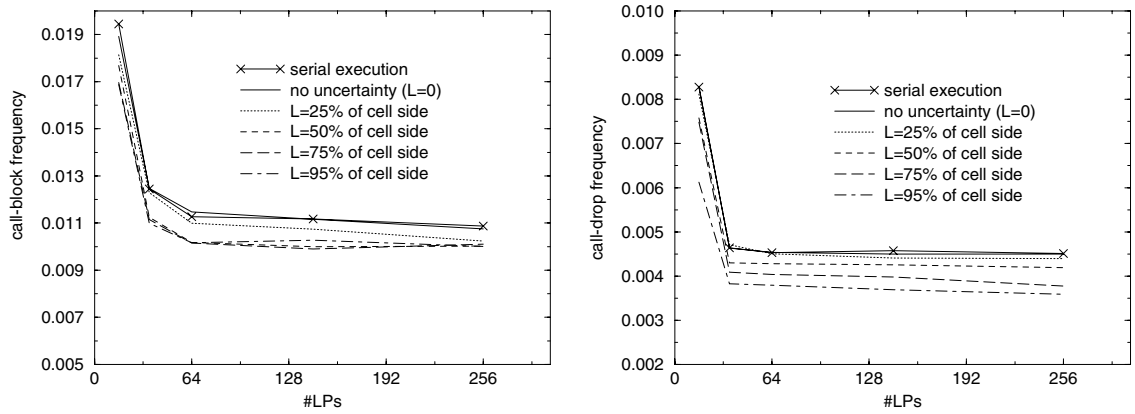

Figure 5. Statistics Related to the PCS.

due to the fact that two different cells might accept the mobile. Therefore, increasing L actually means studying the system behavior while the amount of uncertain events increases. Actually, the upper value for L we consider, i.e. 95% of the length of the side of the hexagonal cell, originates a situation in which a hand-off event has two possible destinations with probability 0.95. In other words, for that value of L, the large majority of hand-off events are space uncertain. As a final preliminary observation, each time the hand-off has two possible destinations due to the fact that the segment L covers a portion of the border that is adjacent to two different cells, then the first LP visited by the forwarding scheme in RINAC is the one associated with the cell touched by the median point of the segment L.

4.3 Performance Metrics

As already pointed out, when dealing with spatial uncertainty we are interested in considering two different effects. One is the possible improvement in the run-time behavior of the simulation system (in our case through RINAC), the other one is the effect of the uncertainty on proper statistics related to the simulated system, namely the PCS. We consider both these effects.

As respect to performance metrics for the simulation system, we report measures related to the event rate, classically evaluated as the amount of committed simulation events per second, which is an indicator of the speed of the model execution. We also report values related to the rollback frequency (classically evaluated as the ratio between the number of rollbacks and the number of executed simulation events) and the rollback length (classically evaluated as the average number or rolled back events at each rollback occurrence), which are indicators of the amount of rollback experienced during the execution. Specifically, the amount of rollback is typically expressed as the product pr between the rollback frequency and rollback length, and this product measures the probability for whichever executed event to be eventually rolled back. The quantity $1 - pr$ is a metric referred to as the efficiency of the simulation model execution. It indicates the probability to carry out productive simulation work, i.e. to execute events that are not eventually rolled back. We report data related to rollback frequency and rollback length just because we want to observe whether possible variations in the execution speed due to spatial uncertainty derive from variations in the rollback pattern (e.g. a reduction in the amount of rollback). As respect to proper statistics of the PCS, we report observed values for both the frequency of call-block and the frequency of call-drop, which are two classical parameters characterizing any PCS. Each reported value is computed as the average over 5 runs, all done with different seeds for the pseudo-random generation. At least 2×10^6 simulation events have been committed in each run.

4.4 Results

The obtained results related to the performance of the simulation system are reported in Figure 4. By the plots we obtain that spatial uncertainty exploited through RINAC is able to strongly reduce the frequency of rollback occurrence as compared to the case of no spatial uncertainty, i.e. the case of $L = 0$ ([3]). Also, a greater reduction of the rollback frequency is observed in case of larger values for L. This is an expected behavior when considering that larger values for L mean higher likelihood for the hand-off event to be addressable to two distinct LPs. Therefore, increasing L actually means increasing the percentage of hand-off events that can be forwarded to the second chance LP through RINAC in case they cannot be accepted by the first chance LP without the need for rollback. As an extreme, when $L = 95\%$ of the length of the cell side, we have a reduction of the frequency of rollback on the order of up to 86%. On the other hand, we note that the reduction of the rollback frequency is non-negligible (it is between 26% and 44%) even for lower values of L, namely values between 25% and 50% of the length of the cell side.

As already discussed in Section 3.1, RINAC might delay the execution of rollback procedures in case no LP along the forwarding chain is encountered, which has simulation clock lower than the event timestamp. This delay, in its turn, might increase the length of rollback. However, this phenomenon does not seem to appear in the results. Specifically, the execution with spatial uncertainty exploited through RINAC does not exhibit significant increase of the rollback length for values of L up to 50% of the cell side. It exhibits increase of the rollback length only in case of very large values of L, namely 75%-95% of the length of the cell side. By this phenomenon we argue that the increase in the rollback length for this particular application is not primarily related to delay in the execution of rollback procedures possibly caused by the forwarding scheme underlying RINAC. Instead, it should be mainly due to the fact that when L is largely increased we get a very strong reduction of the rollback frequency, which means the simulation execution is becoming more "asynchronous". Therefore simulation clocks at LPs on different processors might slightly diverge, which, as expected, tends to negatively impact the rollback length. However, the increase in the rollback length is less relevant as respect to the reduction we obtain in the rollback frequency. Therefore, even for very large values of L, the amount of rollback in the execution gets lower than the one achieved with no spatial uncertainty. Direct effect of the combination of results for the rollback frequency and the rollback length is an increase in the execution speed, namely in the event rate, when spatial uncertainty is exploited. This increase is more evident for the case of larger values of L (due to the much larger reduction of the rollback frequency as compared to lower values), and when considering higher degree of parallelism in the model execution, namely a lower amount of LPs. Specifically, when the number of LPs is 16, we get an increase in the execution speed between 10% and 20%. This gain tends to decrease while increasing the number of LPs. However, we would like to note that the reduction of the gain in the execution speed when the number of LPs is increased (while keeping the same size of the underlying computing system) is not a flaw of RINAC. Instead, it is due to the fact that reducing the degree of parallelism leads to an execution with very low amount of rollback even in case of no uncertainty and classical optimistic synchronization, therefore the improvements in the run-time behavior achievable through spatial uncertainty and RINAC are necessarily limited.

Figure 5 reports statistics related to the PCS system. Actually, we have also reported the call-block and call-drop frequencies estimated through serial executions of this same simulation model. By the results we get that statistics obtained with values of L up to 50% of the length of the cell side are very close to those achieved with serial execution or with parallel execution with no spatial uncertainty (i.e. the case of parallel execution with $L = 0$). Specifically, call-block and call-drop frequencies obtained with L set up to 50% of the length of the cell side differ by no more than 10% from the values obtained with serial execution or parallel execution with no uncertainty. On the other hand, for larger values of L, those parameters show values that remain within the 20% of those achieved with serial execution or parallel execution with no uncertainty.

[3]Recall that $L = 0$ means we have an exact point for the hand-off of the mobile. Therefore the destination cell, that is the destination LP, for the hand-off event is precisely defined.

5 Summary

In this paper we have presented the notion of spatial uncertainty of simulation events, with an application to optimistic parallel/distributed simulation. As we have shown, spatial uncertainty allows more flexible synchronization, with consequent improvement in the run-time behavior of the simulation system. We have also shown that the numerical results obtained for a PCS simulation application with space uncertain events are relatively close to those obtained with no uncertainty when considering the same amount of committed simulation events.

References

[1] I. F. Akyildiz, Y. B. Lin, W. R. Lai, and R. J. Chen. A new random walk model for PCS networks. *IEEE Journal on Selected Areas in Communications*, 18(7):1254–1260, 2000.

[2] R. Beraldi and L. Nigro. A Time Warp based on temporal uncertainty. *Trans. of the Society for Modeling And Simulation*, 18(2):60–72, 2001.

[3] A. Boukerche, S. K. Das, A. Fabbri, and O. Yildz. Exploiting model independence for parallel PCS network simulation. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, pages 166–173. IEEE Computer Society, May 1999.

[4] C. D. Carothers, D. Bauer, and S. Pearce. ROSS: a high performance modular Time Warp system. In *Proceedings of the 14th Workshop on Parallel and Distributed Simulation*, pages 53–60. IEEE Computer Society, May 2000.

[5] C. D. Carothers, R. M. Fujimoto, and Y. B. Lin. A case study in simulating PCS networks using Time Warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*, pages 87–94. IEEE Computer Society, June 1995.

[6] R. M. Fujimoto. Exploiting temporal uncertainty in parallel and distributed simulation. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, pages 46–53. IEEE Computer Society, May 1999.

[7] A. Gafni. Space management and cancellation mechanisms for Time Warp. *Tech. Rep. TR-85-341, University of Southern California, Los Angeles (Ca,USA)*, 1985.

[8] M. Hybinette and R. M. Fujimoto. Cloning parallel simulations. *ACM Trans. on Modeling and Computer Simulation*, 11(4):307–407, Oct. 2001.

[9] D. R. Jefferson. Virtual time. *ACM Trans. on Programming Languages and System*, 7(3):404–425, July 1985.

[10] Y. B. Lin and E. D. Lazowska. Processor scheduling for Time Warp parallel simulation. In *Advances in Parallel and Distributed Simulation*, pages 11–14, 1991.

[11] M. L. Loper and R. M. Fujimoto. Pre-sampling as an approach for exploiting temporal uncertainty. In *Proceedings of the 14th Workshop on Parallel and Distributed Simulation*, pages 157–164. IEEE Computer Society, May 2000.

[12] F. Quaglia and A. Santoro. CCL v3.0: Multiprogrammed semi-asynchronous checkpoints. In *Proceedings of the 17th Workshop on Parallel and Distributed Simulation*, pages 21–30. IEEE Computer Society, 2003.