Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

NPSI ADAPTIVE SYNCHRONIZATION ALGORITHMS FOR PDES

Sudhir Srinivasan
Paul F. Reynolds, Jr.

Department of Computer Science
University of Virginia
Charlottesville, VA 22903, U.S.A.

ABSTRACT

Adaptive approaches to synchronization in parallel discrete event simulations hold significant potential for performance improvement. We contend that an adaptive approach based on low cost near-perfect system state information is the most likely to yield a consistently efficient synchronization algorithm. We suggest a framework by which NPSI (near-perfect state information) adaptive protocols could be designed and describe the first such protocol - Elastic Time Algorithm. We present performance results which show that NPSI protocols are very promising. In particular, they have the capacity to outperform Time Warp consistently in both time *and* space.

1 INTRODUCTION

The synchronization problem remains the central challenge in PDES (Fujimoto 1990, Fujimoto and Nicol 1992). We present initial results of research into a novel class of protocols which adapt dynamically to changes in the simulation using low-cost near-perfect system state information. We refer to these protocols as *NPSI (Near-Perfect State Information) adaptive protocols.* Given the inherently dynamic nature of simulations (Nicol and Reynolds 1990), we believe NPSI adaptive protocols offer the best hope of finding a consistently efficient, general protocol.

We assume familiarity with the common approach to PDES (Fujimoto 1990), namely the partitioning of a simulation into logical processes (LP's). Each LP is itself a sequential discrete event simulator which can schedule events at other LP's using timestamped messages. LP's must execute events without violating causality constraints (effectively). Typically, this is the responsibility of a *protocol.* The nine design variables in Reynolds (1988) define the design space for protocols. An *aggressive* protocol is one which executes events without the guarantee of freedom from causality errors. A protocol has *risk* if it propagates messages based on aggressive or inaccurate computation. Based on these two variables, a *conservative* protocol is non-aggressive and without risk, while an *optimistic* protocol is aggressive and with risk. These two categories form the end points of a spectrum of protocols with limited optimism.

Adaptiveness for PDES protocols is defined in Reynolds (1988) as the capability of a protocol to modify the bindings of one or more of its design variables during the simulation. Several protocols have been proposed that limit optimism, but most of them are not adaptive by this definition. In order to make dynamic decisions based on system state, processes must be provided with near-perfect state information at low cost. Gathering such information using typical communication networks is infeasible due to the high cost of such communication. This has been the major obstacle in the study of NPSI adaptive protocols.

We assume an asynchronous dynamic feedback system which provides each LP with a near-perfect snapshot of the system state (in a reduced form) at very low cost. This is done for three reasons: (i) we believe NPSI adaptive protocols have significant potential, (ii) a feasible implementation for the feedback system is a high-speed reduction network, and (iii) an implementation exists for such a network (Reynolds, Pancerella, and Srinivasan 1992). As we shall see, our first NPSI adaptive protocol shows significant improvements over pure Time Warp (Jefferson 1985), in both time and space.

2 PREVIOUS WORK

We categorize protocols that limit optimism based on the criterion for limiting optimism:

i) *Window based*: Only those events within a (common or independent) window are executed aggressively. Similarly only those messages within a (possibly different) window are sent out. This ensures that all of the LP's remain close to each other in logical time. Uncontrolled echoing and cascading rollbacks cannot occur. Examples are described in Sokol, Briscoe, and Wieland (1988), Lubachevsky, Weiss, and Shwartz (1989), Reiher and Jefferson (1989), McAffer (1990), Turner and Xu (1992), Dickens (1993), and Steinman (1993).
ii) *Space based*: The boundaries for limiting optimism are spatial rather than temporal. In general, the

processors are divided into clusters which use Time Warp internally. Interaction among clusters is without risk. Examples are described in Gimarc (1989) and Rajaei, Ayani, and Thorelli (1993). An interesting special case is when each cluster contains exactly one LP, resulting in a risk-free system (Dickens and Reynolds 1990, Mehl 1991, Steinman 1991 and Bellenot 1993).

iii) *Penalty based*: It is assumed that the recent past is a good predictor of the near future. Based on their recent behavior, some LP's are penalized (and consequently block) while others are favored (and consequently continue). Examples are described in Reiher and Jefferson (1989) and Ball and Hoyt (1990). In Madisetti (1993), the penalty is based on the difference between an LP's logical clock and *estimates* of the logical clocks of other LP's.

iv) *Knowledge based:* Rollback information is used to restrict the propagation of potentially incorrect computation. Examples are described in Madisetti, Walrand, and Messerschmitt (1988) and Prakash and Subramanian (1991).

v) *Probabilistic*: A special process decides to synchronize all LP's periodically, by sending synchronization messages (Madisetti, Hardaker, and Fujimoto 1992).

These protocols perform better than Time Warp under specific tests. However, they will have limited performance in general due to one or more of the following: (i) the criterion for limiting optimism (window size, cluster size, penalty thresholds, probabilities, etc.) is predetermined (ii) the decision to limit optimism is based solely on local history (iii) the LP's are loosely synchronous (i.e. not asynchronous).

Recently, three protocols have been proposed which we categorize as *state based* protocols. They differ from those above in two significant ways: they are adaptive in that the LP's continually adjust their optimism, and optimism is limited based on non-local state information. These protocols are similar to NPSI protocols. The first two (Hamnes and Tripathi 1994, Ferscha and Tripathi 1994) are similar in that they both utilize channel information to decide when LP's should wait and for how long. The NPSI approach has an advantage because LP's can receive information from all of their predecessors whereas channel-based protocols provide information only about immediate predecessors. The third protocol (Das and Fujimoto 1994) adaptively limits the memory consumption of LP's and, consequently, limits their optimism. NPSI adaptive protocols limit the optimism of LP's and consequently limit memory consumption. The NPSI approach has the potential for eliminating the need for costly memory management schemes completely.

3 EFFECT OF LIMITING OPTIMISM

Aggressive protocols incur three direct costs: state saving, rollback and memory management. Limiting aggressiveness and risk introduces a fourth cost: *lost opportunity*, characterized by the potential loss in performance when an LP stops executing events or sending messages even though it is safe to do so. To obtain good performance, protocols must attempt to minimize:

state saving cost + rollback cost +
memory management cost + lost opportunity cost

While limiting optimism tends to decrease the first three of these, it also tends to increase the fourth; see Figure 1. This trade-off must be balanced properly in order to obtain the best possible performance. To do so, protocols must distinguish incorrect computations from correct ones and limit the propagation of the former while allowing the latter to progress.

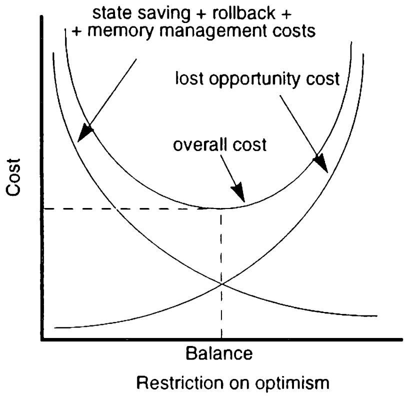

Figure 1: Trade-off Introduced by Limiting Optimism

PDES's are very dynamic in nature, i.e. locality of events changes as the simulation progresses. Typically, this is due to an information flow which is translated into a causal chain of events among LP's. Since the propagation of such chains is based on probabilistic decisions and input parameters of the simulation, it is impossible to determine the flows a priori (except in special cases). This suggests that in order to balance the trade-off above, a protocol must also be dynamic, adapting its behavior in response to observed changes in the system.

4 NEAR-PERFECT STATE INFORMATION

Two key requirements for a protocol to be consistently efficient are that it is dynamic and that it uses feedback from the simulation to adapt. Ideally, these requirements are met by providing LP's with perfect state information. However, this is impossible to achieve due to various latencies in computing distributed snapshots. Therefore, we consider an approximation of perfect state information.

We assume the existence of a dynamic *feedback system* that operates asynchronously with respect to the

LP's and provides them with low-cost, near-perfect information. One solution is to use a high-speed reduction network. A global reduction network is one in which binary, associative operations (minimum, summation, etc.) are used to reduce state information so that, say, the minimum simulation clock value can be computed across all LP's. Our experience with the design, construction and testing of a global reduction network (Reynolds, Pancerella, and Srinivasan 1992) suggests that a network can operate at very high speeds (less than 20 nanoseconds per stage in the tree). We have used this network for the performance analysis described later.

5 NPSI ADAPTIVE PROTOCOLS

NPSI adaptive protocols are optimistic protocols in which the aggressiveness and risk are controlled dynamically using near-perfect state information. There are two phases in the design of NPSI adaptive protocols:

- identifying the information on which the decision to limit optimism is to be based;
- designing the mechanism that translates this information into control over an LP's optimism;

There are many choices for each of these. To facilitate independent study of each, we uncouple them by introducing a quantity called *error potential* (EP_i). The value of EP_i is used to control LP_i's optimism. The framework we propose is shown in Figure 2. The NPSI adaptive protocol keeps each EP_i up-to-date as the simulation progresses, by evaluating M_1 at high frequency using state information it receives from the feedback system. Similarly, M_2 reflects new values of EP_i in the event execution and communication rates dynamically. The goal of our research is to design mappings M_1 and M_2 such that their combination forms an adaptive protocol that performs well consistently.

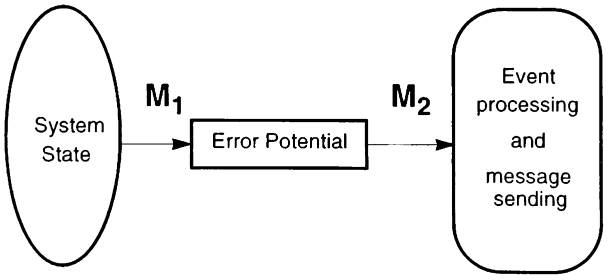

Figure 2: General Framework for NPSI Protocols

To achieve optimal performance, LP's must identify computations that will be rendered incorrect in the future and limit their propagation. This requires the ability to predict the future which is difficult at best. In the framework described above, error potential is a way of labeling computation as *potentially incorrect*. EP_i indicates the likelihood of LP_i's computation becoming incorrect in the near future: the higher the value of EP_i, the higher this likelihood. The key to consistently good performance is to devise an M_1 which will predict the nature of the LP's computation (i.e. whether that computation will be rolled back or not) accurately most of the time. An inaccurate M_1 can produce a low value of EP when the computation is erroneous, resulting in higher rollback costs, or a high value of EP when the computation is correct, resulting in higher lost opportunity cost.

M_2 must be such that higher values of EP_i result in less aggressiveness and risk at LP_i. A simple scheme is to establish a *threshold* such that if the value of EP exceeds this threshold, event execution and communications are suspended until EP falls below the threshold again. A more sophisticated scheme would reduce event execution and communication rates gradually as EP increases. This deceleration can be achieved by inserting delays at appropriate points.

6 ELASTIC TIME ALGORITHM

We describe the *elastic time algorithm* (ETA), the first NPSI adaptive protocol. ETA has been implemented; preliminary performance analysis results are presented here. The protocol can be specified completely by describing the two mappings M_1 and M_2.

6.1 M_1: Computing EP_i

M_1 is the following function:

$$M_1 : EP_i = \text{logical clock}_i - GVT$$

where GVT (global virtual time — Jefferson 1985) is defined as the minimum of the logical clocks of all LP's and the timestamps of any messages in transit. Thus, ETA is based on near-perfect values of GVT being available to the LP's. See Srinivasan and Reynolds (1993).

The rationale behind this M_1 is that if an LP is far ahead of others, it is likely to be rolled back soon and should be slowed down. One can imagine an LP and its predecessors as pins moving along a logical time line with an elastic band around them. The farther an LP moves away from the rest, the slower its progress due to the restraining pull of the elastic band. When the LP farthest behind moves forward, the restraint on the LP farthest ahead is reduced so that it may quicken its pace again. As load locality changes among LP's, this scheme adapts by restraining those LP's far ahead in logical time.

6.2 M_2: Controlling Optimism

Given a value of EP_i computed by M_1, we use the following function to scale it to a delay value, δ_i:

$$\delta_i = s \cdot \frac{EP_i}{MaxEP_i}$$

where $MaxEP_i$ is the maximum value of EP_i observed thus far and s is a scaling factor (we defer discussion on s to Section 7). The event processing loop of a Time Warp LP is modified as shown in Figure 3 to incorporate adaptive delaying. Some interesting features of this M_2 must be noted:

i) The blocked state of the LP is not "opaque" in that while in this blocked state, the LP observes its input channels for messages that may cause it to rollback. If such a message arrives, the waiting is aborted and rollback is initiated. Also, the LP may perform useful work in the blocked state such as converting messages that it receives (that do *not* cause a rollback) into future events.
ii) EP_i and δ_i are updated in each iteration of the loop in the blocked state.
iii) The waiting scheme is not memoryless. The wait timer is started only once at the beginning of the wait period. As LP_i goes through successive iterations of the wait-loop, its wait time increases whereas δ_i decreases (because its logical clock is constant and GVT is monotonically increasing). Thus, LP_i interprets each new value of δ_i as an estimate of the amount of time it should have waited since the start of the waiting period. When this value becomes smaller than the time it has actually waited, LP_i exits the wait-loop.
iv) This M_2 provides direct control over an LP's aggressiveness only. The LP's risk is controlled only to the extent that while in a blocked state, the LP does not send out messages. Clearly, it is possible (and perhaps desirable) to have a separate mechanism to limit risk.

```
While there are events to be processed
   Update logical clock and process event
   Start wait timer
   do
       Receive messages and exit from loop if
           there is a message that will cause
           rollback or if the message has
           timestamp = GVT
       Update GVT, $EP_i$ and $\delta$
       Read wait timer
   while wait timer value < $\delta$
   Rollback if necessary
   Process messages
   Save state
   Collect fossils
endwhile
```

Figure 3: Event Processing Loop with Adaptive Waiting

Putting ETA in perspective with previous protocols, we observe that it is a state based protocol that is similar to window based protocols with two significant differences:

- It is completely asynchronous — there are no barrier synchronizations to negotiate windows.
- Each LP's logical time window may be considered infinite but the event execution rate drops rapidly as the LP moves away from the base of the window.

6.3 Performance Analysis

We present the results of performance tests on ETA. We describe the testing environment, followed by test cases, metrics used and results.

Hardware: The hardware consists of a cluster of four Sparc 2 workstations connected by Ethernet and a *parallel reduction network* (PRN) that we have designed and built (Reynolds, Pancerella, and Srinivasan 1992). Each workstation communicates with its own auxiliary processor (AP) through dual-ported memory (DPRAM). The four AP's are connected to the PRN which computes and disseminates globally reduced values at very high speeds.

Software: The two primary software components are the Time Warp LP's executing on the workstations and the AP's respectively. The LP's use aggressive cancellation and do not support event pre-emption; they include the M_1 and M_2 for ETA. The GVT computation algorithm for the AP's is described in Srinivasan and Reynolds (1993).

Test cases: We employ a parameterized synthetic workload generator to create our test cases with the important parameters being event execution time, average timestamp increment, state saving cost, communication topology and distributions, number of local events per message, and state saving and fossil collection frequencies. Time consuming actions such as event executions and state saving are simulated by busy loops; all other mechanisms such as rollback, restoring state, sending antimessages and fossil collection and data structures such as saved state list and antimessage list are implemented in detail. A synthetic workload generator is used instead of actual applications because it allows us to mimic those applications without the excessive time and effort required to implement each of them. Our workload generator is very similar to the PHOLD model (Das and Fujimoto 1994).

We tested ETA on several workloads and observed that it outperformed Time Warp on all of them. We present results for the following:

a) **Workload 1** consists of four LP's with the T_1 communication topology shown in Figure 4 (a torus). The number on an arc is the probability that a generated message is sent along that arc. The workload is self-initiating (each event schedules the next local event — Nicol 1991) and the probability of an LP sending a message after an event is 0.2.

Messages may cause rollbacks, but do not schedule events.

b) **Workload 2** also uses topology T_1 but is message initiating (messages cause events to be scheduled). Each message schedules a job event. Execution of a job event creates an output event, which generates a message. Every LP has 25 events initially. Thus, this workload resembles a closed queueing network with density 25.

c) **Workload 3** is an implementation of the *echoing* example described in Lubachevsky, Weiss, and Shwartz (1989). LP_0 and LP_1 execute in self-initiating mode, sending messages to LP_2 after each event. In addition, they exchange a single message that schedules a message-initiating event. This message causes rollbacks of increasing amplitudes at LP_0 and LP_1.

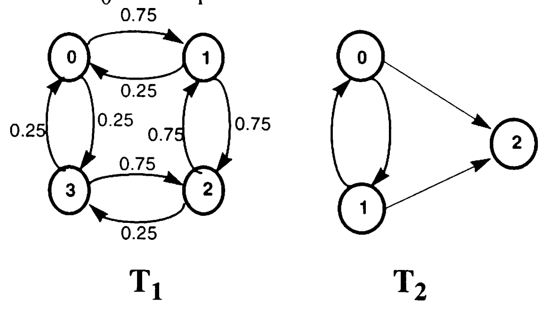

Figure 4: Communication Topologies for Test Workloads

For all three workloads, the mean event execution time is 100 µs, the mean state saving time is 25 µs and state saving and fossil collection are performed after each event.

Metrics: The most important metric is completion time. Yet another metric is rollback time, which is the time an LP spends rolling back (including state restoration and sending of antimessages). Since limiting of optimism is expected to reduce rollback time, ideally to zero, this metric is a good indicator of how close the actual performance is to the goal.

An important aspect of ETA is the scaling factor, s in M_2. s translates the value of EP_i from logical time to a delay in real time. The range of EP_i is dependent on the logical time increments and the rate at which LP's execute events, send messages, etc. Since these factors differ considerably across applications, the value of s that maximizes performance will be different for each application. Thus, s is a good choice for the independent variable in the performance tests. The problem of determining s dynamically is discussed in §6.3.3

6.3.1 Results

Figures 5, 6 and 7 show the variation of completion time and rollback time with the scaling factor s for the three workloads respectively. When s=0, δ=0 and ETA is essentially identical to Time Warp. As s increases, the aggressiveness and risk of the LP's decrease. The completion time in Figure 5 has the expected parabolic shape based on the trade-off shown in Figure 1. The absence of such a parabolic shape in Figures 6 and 7 is due to the nature of the workloads. There is very little concurrency in workload 2 owing to the small event execution times, the high latency of Ethernet and high connectivity of topology T_1. A simple critical path analysis of workload 3 shows that it is also inherently sequential. Generally, the waiting period at each LP increases with s, approaching sequential execution. However, a threshold is reached such that further increase in s does not increase waiting due to the following: consider LP_i waiting to execute an event. When the last event with timestamp less than LP_i's logical clock has been executed and all messages with timestamps less than LP_i's logical clock have been received, GVT will equal LP_i's logical clock causing EP_i (and δ_i) to drop to zero and LP_i to come out of waiting in the next iteration. This analysis demonstrates that ETA has the capability to approach sequential execution but not become arbitrarily slower than it during phases in a simulation where there is so little concurrency that parallel execution is detrimental.

The significant reduction in completion time in Figure 7 demonstrates that ETA can avoid unstable situations such as echoing. The instability is manifest in the large variations in both curves in Figure 7 for small values of s. It is important to note that in all three graphs, the rollback time is close to zero when completion time is minimized. This suggests the reduction achieved by ETA is close to the maximum possible.

6.3.2 Memory Considerations

In the general case, memory consumption appears to be a serious problem with Time Warp. Excessive memory consumption is usually due to so-called *runaway processes* - LP's that execute events faster than other LP's so that (i) they have a large number of processed events as yet uncommitted, and (ii) they schedule a large number of unprocessed events at other LP's. Several memory management schemes have been proposed (Lin 1992) to reclaim memory from future events (since this memory cannot be reclaimed by fossil collection). We expect that NPSI adaptive protocols will eliminate the need for these schemes for two reasons. First, any approach that limits the aggressiveness of LP's inherently reduces memory requirements by not permitting runaway processes to move too far ahead. Second, an adaptive, memory-based flow control scheme (Das and Fujimoto 1994) can be integrated naturally with NPSI protocols by including information about the memory availability of successor LP's (perhaps immediate successors only) in the mapping

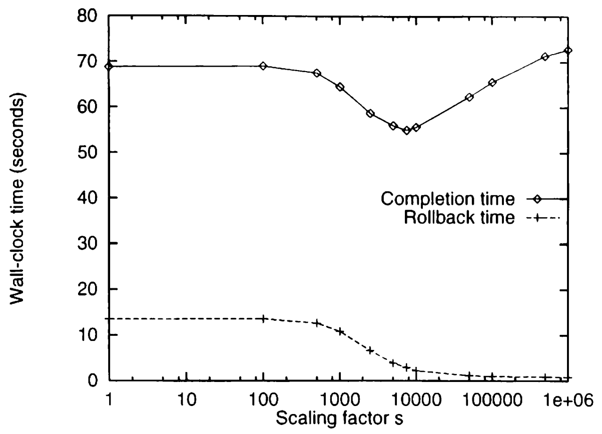

Figure 5: Performance for Workload 1

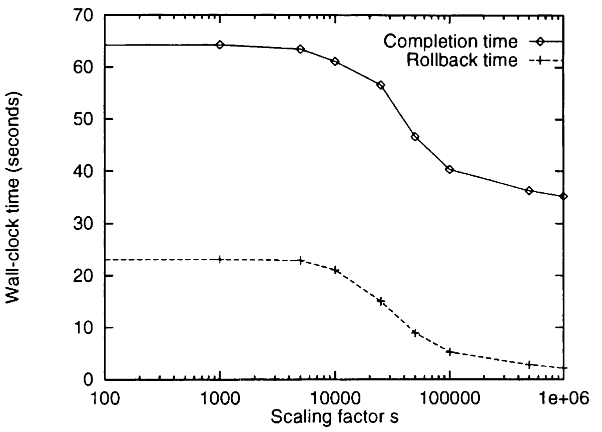

Figure 6: Performance for Workload 2

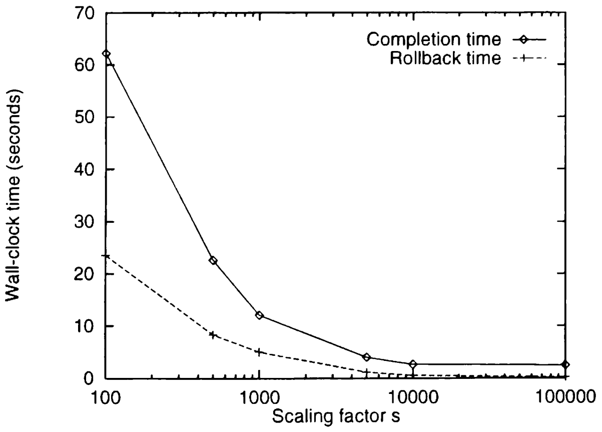

Figure 7: Performance for Workload 3

M_l. In this way, an LP could slow down when any of its successors is at risk of running out of memory.

ETA does not include information about successors' memory usage in its M_l. Despite this, we observed significant savings in memory requirements due to limited aggressiveness. We measured the average and maximum size of the saved-state list (in terms of the number of entries in the state list) as an indicator of the LP's processed but uncommitted memory requirements. The maximum size is important because the workstation must have sufficient memory to store that much state even if it is a rare occurrence.

Figures 8, 9 and 10 illustrate the substantial savings in memory consumption (despite the fact that only state-saving space is being considered here). In the two stable workloads (Figures 8 and 9), the savings in maximum state list size is noteworthy. In the echoing workload (Figure 10), it is interesting to observe that the average state list size is very high. This is because for small values of s, the state lists grow unboundedly due to instability.

6.3.3 Further Issues

Cascading of rollbacks tends to occur when the event grain is comparable with the cost of a single rollback, which is proportional to the cost of sending antimessages. Thus, the communication cost of the architecture determines the granularity of events at which Time Warp performance degrades due to high rollback costs. Provided the NPSI assumption is satisfied at this granularity, we expect ETA to produce reductions in rollback (and completion) time similar to those presented here on any architecture.

Memory consumption becomes a problem in Time Warp when LP's simulate at different speeds. Since this occurs primarily due to load imbalance (dissimilar timestamp increments and event execution times) rather than any architectural feature, the memory performance of ETA is expected to be architecture independent.

One of the conclusions drawn from the results above is that the scaling factor (s) must be chosen properly to ensure good performance. A scheme to automatically tune the value of s is being developed. In Srinivasan and Reynolds (1994), we have proposed two metrics to aid this task. Extensive testing of ETA on larger systems using simulations is underway.

7 SUMMARY

We have introduced a new class of synchronization protocols called NPSI (near-perfect state information) adaptive protocols. These differ from previous approaches to adaptiveness in that they base their adaptive decisions on near-perfect information about the state of relevant parts of the entire system. In Ramamritham, Stankovic, and Zhao (1989), it has been shown that a load sharing policy that assumes perfect state information at zero cost offers the best solution. Correspondingly, we believe that NPSI adaptive protocols will provide a general, efficient solution to the synchronization problem of PDES.

A framework has been suggested for the design of NPSI adaptive protocols. Based on this framework, the

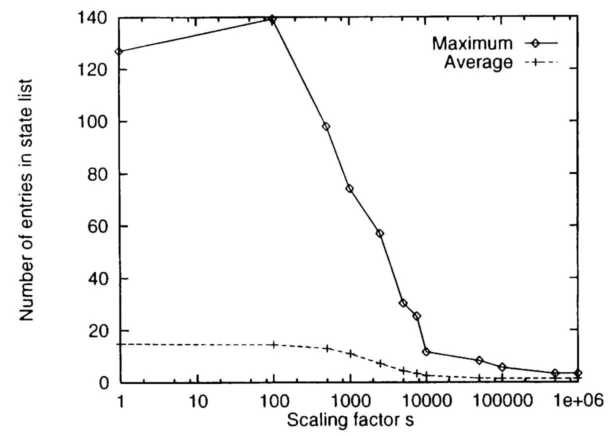

Figure 8: Memory Consumption for Workload 1

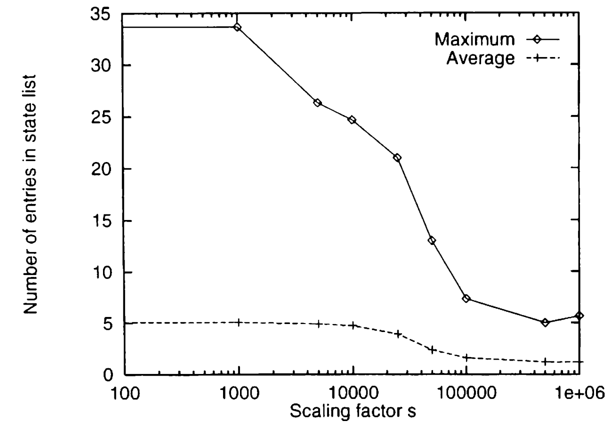

Figure 9: Memory Consumption for Workload 2

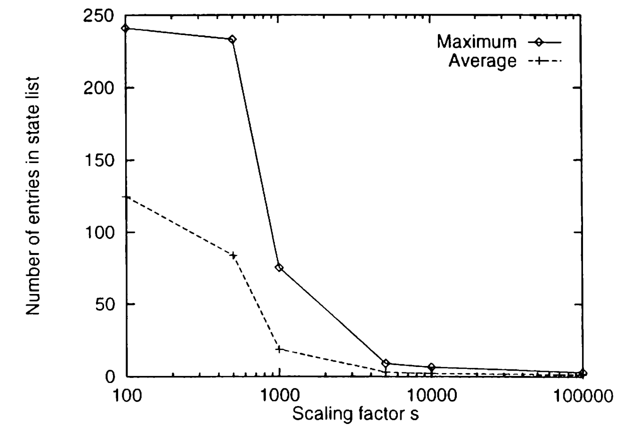

Figure 10: Memory Consumption for Workload 3

Elastic Time Algorithm has been designed and implemented. For this implementation, near-perfect state information is computed and disseminated through a high-speed reduction network. The protocol has been tested with several workloads, the results from three of which have been presented here. From these results, it is evident that NPSI adaptive protocols can outperform pure Time Warp in both time and space. Some issues must be resolved before any conclusive statements can be made about the relative performance and usability of NPSI adaptive protocols. These include: designing a scheme for the LP's to tune any parameters of the protocol automatically; testing on larger systems; designing more NPSI protocols and comparison with other adaptive protocols. Ongoing research into some of these issues has been described.

ACKNOWLEDGMENTS

We are grateful to Bronis de Supinski for proof- reading the paper and for his insightful suggestions. We thank the members of the Smart Interconnection Networks group at the University of Virginia for their suggestions. This work was supported by Mystech Associates, Inc. (Academic Affiliates Program).

REFERENCES

Ball, D. and S. Hoyt. 1990. The adaptive Time-Warp concurrency control algorithm. *Proceedings of the SCS Multiconference on Distributed Simulation*, 174-177.

Bellenot, S. 1993. Performance of a riskfree Time Warp operating system. *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, 155-158.

Das, S. and R.M. Fujimoto. 1994. An adaptive memory management protocol for Time Warp parallel simulation, *SIGMETRICS '94*, 201-210.

Dickens, P.M. 1993. Analysis of an aggressive global windowing algorithm. Ph.D. thesis, Computer Science Department, University of Virginia, Charlottesville, Virginia.

Dickens, P.M. and P.F. Reynolds, Jr. 1990. SRADS with local rollback. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, 161-164.

Ferrari, D. and S. Zhou. 1987. An empirical investigation of load indices for load balancing applications. Report number CSD-87-353, Computer Science Division, University of California at Berkeley.

Ferscha, A. and S.K. Tripathi. 1994. Parallel and distributed simulation of discrete event systems. Report number CS-TR-3336, Computer Science Department, University of Maryland at College Park.

Fujimoto, R.M. 1990. Parallel discrete event simulation. *CACM*, Vol. 33, No. 10, 30-53.

Fujimoto, R.M. and D.M. Nicol. 1992. State of the art in parallel simulation. *Proceedings of the 1992 Winter Simulation Conference*, 246-254.

Gimarc, R.L. 1989. Distributed simulation using hierarchical rollback. *Proceedings of the 1989 Winter Simulation Conference*, 621-629.