DISTRIBUTED SYNTHETIC NATURAL ENVIRONMENT REPRESENTATION FOR PARALLEL DISCRETE EVENT SIMULATIONS

Thomas Stanzione
Forrest Chamberlain
TASC
Reading, MA

Dr. Alan Evans
SAIC
Burlington, MA

ABSTRACT

Distributed heterogeneous simulations pose many problems for representations of a temporally and spatially consistent synthetic natural environment (SNE), especially if that environment is dynamic. The generation and distribution of the SNE in the run time formats of the simulation applications is one hurdle that must be overcome. In a dynamic environment, this data must be updated in an efficient, consistent manor as the exercise is taking place. Parallel Discrete Event Simulations (PDES), such as JSIMS, pose additional problems for a dynamic SNE. For instance, optimistic approaches may require that changes made to the simulation environment be “rolled back” if a simulation entity needs to revert to a time previous to the change. Changes to the SNE may require large amounts of data and processing time, so this requirement has serious run time implications. Also, each simulation entity may require a different view of the environment, depending on its location and notion of simulation time.

Under the DARPA Advanced Simulation Technology Thrust program, the Framework of Reusable Objects for the Synthetic Natural Environment (FROST) project is working to identify and address these issues, in order to provide a dynamic SNE representation that could be used in a distributed PDES simulation, such as JSIMS. The FROST approach is to develop a ground truth environment management system that uses PDES events for database updates, along with a commercial object oriented database management system. This paper will describe the issues associated with a distributed SNE in a PDES environment, how our approach addresses these issues, and our design efforts to date.

ABOUT THE AUTHORS

THOMAS STANZIONE is the manager of the Computer Generated Forces section at TASC. He is the Program Manager for the FROST project, and was the Program Manager for the ModSAF terrain database representation improvements project for the DARPA STOW program. His interests include data representations for terrain reasoning and terrain database generation for simulation applications. Mr. Stanzione holds BS and MS degrees in Photographic Science from the Rochester Institute of Technology.

FORREST CHAMBERLAIN is a Senior Member of the Technical Staff in the Computer Generated Forces section at TASC. Forrest has been involved in Computer Generated Forces work since joining TASC in 1994. He is the TASC program manager for the Advanced Synthetic Command Forces (ASCF) project, and is a key technical contributor to the FROST project. His interests include behavioral representations and synthetic environments. Mr. Chamberlain holds an MS in electrical and computer engineering from Carnegie Mellon University.

DR. ALAN EVANS is the manager of the Burlington office of SAIC and is the technical lead on the FROST project. He has worked in Advanced Distributed Simulation for over five years with primary research interests in simulation systems architecture, object modeling, distributed object technology and synthetic environments. Dr. Evans has a Ph.D. in Mathematics from Michigan State University and an MS in Computer Science from New York University.

DISTRIBUTED SYNTHETIC NATURAL ENVIRONMENT REPRESENTATION FOR PARALLEL DISCRETE EVENT SIMULATIONS

Thomas Stanzione
Forrest Chamberlain
TASC
Reading, MA

Dr. Alan Evans
SAIC
Burlington, MA

INTRODUCTION

Distributed heterogeneous simulations pose many problems for representations of a temporally and spatially consistent synthetic natural environment (SNE), especially if that environment is dynamic. The generation and distribution of the SNE in the run time formats of the simulation applications is one hurdle that must be overcome. Parallel Discrete Event Simulations (PDES), such as JSIMS, pose additional problems for a dynamic SNE. For instance, optimistic approaches may require that changes made to the simulation environment be "rolled back" if a simulation entity needs to revert to a time previous to the change. Changes to the SNE may require large amounts of data and processing time, so this requirement has serious run time implications. Also, each simulation entity may require a different view of the environment, depending on its location and notion of simulation time.

DISTRIBUTED SNE ISSUES

All end users and role players in an enterprise, and thus all applications in a system, should have SNE data, derived from a common source, if they need it. This could be achieved via pre-distribution, by a central server, or by a truly distributed solution. Experience has shown that (1) defining a common notion of global environmental ground truth is notoriously difficult, even in a fairly homogeneous context, and that (2) consumers of SNE data make local optimizations to global environmental ground truth. That is, consumers of SNE data have application-specific needs for data content and coverage, especially in heterogeneous systems.

Dynamic Environment

In a dynamic environment, this data must be updated in an efficient, consistent manner as the exercise is taking place. One of the major advances in DoD simulation technology over the past five years has been the advent of dynamic SNE data. SNE objects and their attributes can change during the course of an exercise, and new objects can be created. The range of variability is enormous, from highly localized changes to the geometry of terrain or the articulation of a door or window in a building, to the fluctuation with time of gridded weather data on a global scale. A static, pre-distributed Synthetic Natural Environment is no longer feasible.

Some immediate architectural observations are possible. First, the mere existence of dynamic objects in the SNE makes a distributed solution necessary. A central server for dynamic SNE data would be the easiest to implement and maintain from a developer's point of view, but introduces performance issues that must be addressed. Furthermore, a central server can introduce a single point of failure. One major issue in designing a dynamic SNE mechanism has to do with how changes are distributed. While distributing them at the finest level of detail is the most obvious way to guarantee correlation, doing so can be very inefficient. A possible optimization strategy is to use a generic or parametric description of changes but having applications implement these changes in their own ways leads to consistency problems: How does one ensure that all updates are equivalent?

Consistency

While some progress has been made in maintaining and distributing dynamic SNE data, the simulation community has yet to come to terms with an even more difficult set of challenges summed up in the word consistency. Different applications may have different runtime representations of the SNE. Some of these differences may be historical, having arisen along parallel development paths. Some differences in runtime representation are more basic, being driven by variable performance requirements of target applications and varying system capabilities. However, the most fundamental differences in SNE representations are due to different application requirements for content, based on differing requirements for functionality. One obvious fundamental area of mismatch is the need for both textured imagery and vector data, as required by visualization and planning systems, respectively.

A more pervasive difference is in the need for representations of an object or collection of objects at different levels of resolution. Aggregate combat models may understand a forest or wooded area as relevant to maneuver planning for a battalion, whereas a tactical team training simulator may very well need to model individual trees. The terrain skin itself may need to be represented at levels of resolution ranging from a gridding at hundreds of meters between sample points down to highly complex polygonal models of sub-meter resolution. For every type of object in the SNE, it is possible to find applications and users wishing to represent objects of that given type at different levels of resolution. For system designers and implementors, the need for consistency across different levels of resolution and across different types of representation presents a severe challenge. A server/client approach would make it easier to maintain consistency but harder to retain performance.

Considerable progress has been made by SEDRIS (Synthetic Environment Data Representation Interchange Specification) [STRICOM 1997] in bringing a common object model and common software interface into the pipeline between SNE data producers and consumers. Of course, the SEDRIS object model is highly flexible, as it must be to support the data requirements of a wide range of consumers. Thus, while SEDRIS has been able to provide data from many producers to many hungry consumers through a common interface, it has not solved the problem of consistency of data among the various representations enabled by its flexible model.

Data Distribution

The mechanisms used for distribution of SNE data may be an issue, since data distribution mechanisms used for ground truth data characterizing entities, whether platforms or aggregate units, may not be adequate, due to the following fundamental differences of these data types:

- Entity data sparsely populates the gaming area, whereas SNE data covers the gaming area

- Many types of SNE objects are “large” in spatial extent and irregularly shaped – The fact that an entity has a well-defined location and a relatively compact minimum bounding box makes managing such data and queries about them much easier. In the case of large and irregularly shaped areal features such as vegetation or bodies of water, or essentially unbounded linear features such as transportation networks, support for even simple queries is difficult. Performing a “fence locate” query which returns all features in a given spatial window leads to dynamic indexing schemes.

- The quantity of SNE data is orders of magnitude greater – Detailed platform level terrain databases,

such as the STOW SW Asia playbox, approach a gigabyte in size. Coverage at a coarser level of resolution of larger areas, as planned for JSIMS builds, will probably be on the same order of magnitude. Dynamic SNE objects, METOC data in particular, will probably be implemented at JSIMS IOC as gridded fields, requiring significant resources for storage and reassembly.

- Ownership of SNE objects is not well-defined – Distributed systems enforce consistency by granting update privileges or locks on objects or their attributes. In the case of battlefield entities, there is typically a single owner of this privilege, in fact a single thread or process is normally responsible for state changes to a given entity. However, in the case of SNE objects, many if not all federates or processes may be eligible to change the state of an object. This poses challenges for the software infrastructure, particularly in light of the large spatial extent of SNE objects.

PARALLEL DISCRETE EVENT SIMULATION ISSUES

The JSIMS program plans to use a Parallel Discrete Event Simulation (PDES) implementation for its simulation engine. The Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES) [Steinman 1991] framework will be used for this engine. Simple event-based simulations model systems of interest by sequentially executing and evaluating a time-ordered sequence of discrete events, each representing a change in the system. As events occur, they often create new (future) events, which must be scheduled according to when they will happen.

The challenge in PDES is to partition this stream of events into multiple streams which can be executed in parallel, while achieving the same results as in the serial execution. To achieve this, each parallel process is typically assigned responsibility for one or more distinct elements of the system, and thus processes all events relevant to those elements. To ensure correctness, it is sufficient to ensure that each process executes all of its events in time order [Overeinder 1991]. However, since each event can spawn additional events, guaranteeing ordered execution is non-trivial. A number of techniques have been developed to address this “time management” issue, which can be divided into two broad categories: conservative, and optimistic. Conservative strategies use restrictions on the behavior of events and simulations to guarantee that no event is ever executed out of order. Optimistic strategies, on the other hand, execute events speculatively and “roll back” the results of any events found to have been executed out of order.

Simulation systems that use an optimistic time management scheme, such as is planned for JSIMS, pose interesting problems in simulating a dynamic SNE. In particular, the fact that SNE data is large, dense, and difficult to localize make it especially challenging to model in an optimistic manner. This leads to three related difficulties. First, most roll back schemes involve some form of state copying before each event is processed; the size of the SNE data can make this impractical. Differential state saving schemes have been developed, but can be difficult to implement for SNE data due to the non-linear and sometimes data-intensive nature of SNE changes. Second, SNE data is difficult to transmit on demand due to its size. PDES applications frequently use interest management techniques to limit information flow into each simulation process; thus applications do not receive state updates for objects they are not currently interested in. As a simulation’s zone of interest changes, it must acquire the current state of newly interesting objects over the network. Unfortunately, infrastructure software tuned to the needs of entity

simulation is often incapable of supporting the volume of data generated by the transmission of this state data for the SNE. Finally, even with interest management, each time-managed object is typically interested in a large volume of SNE data at any given time. Applications aiming to achieve fine-grained parallelism can simulate hundreds or even thousands of separately time-managed objects on a given CPU. For such applications, it is impractical to maintain a separate SNE store for each simulated object, yet each object must see a version of the SNE consistent with that objects unique simulation state.

FROST DESIGN

The DARPA Framework of Reusable Objects for the Synthetic Natural Environment (FROST) project, which is part of the Advanced Simulation Technology Thrust program, is investigating a distributed ground truth environment database and management system approach for use in the Joint Simulation (JSIMS) enterprise of constructive simulations. FROST is a joint project with TASC as the prime contractor and SAIC as subcontractor. The FROST team is investigating issues associated with a distributed ground truth environment approach for providing all SNE data to heterogeneous simulation systems, including platform and unit level simulations, as well as C4I systems. The FROST design is based on an object-oriented framework solution. Classes and methods are defined to implement a ground truth environment management system. The FROST team is investigating the feasibility of using existing elements of the JSIMS infrastructure and/or a commercial database management system to implement much of this functionality. We have investigated both relational and object-oriented commercial database management systems, including ObjectStore, Objectivity, and Oracle8. We are working with the JSIMS SNE and infrastructure developers to determine how this approach fits into the JSIMS architecture, especially within the JSIMS Object Services (JOS), Mission Space Objects (MSO), and Parallel Discrete Event Simulation time management architecture. The FROST team is developing a prototype of this approach, which should be available in September of 1999.

Ground Truth Environment Management System

A distributed ground truth environment database and management system, as shown in Figure 1, could be used to help solve many of the problems associated with consistent, dynamic SNE representations. In a system such as this, a distributed ground truth environment database is maintained, which contains all of the source data needed to generate the run time, application specific environment representations. The ground truth environment management system provides the data distribution and real time compilation mechanisms to maintain consistent representations among simulation applications. Application specific database content, area extent, and level of resolution are managed using interest management techniques.

A common environmental interface provides a consistent interface to all applications, thus constraining the design of the application specific environment representations and reducing developer investment in interface design.

Description of Components

The major components of this approach are the Ground Truth Environment Management System (GTEMS), which is itself composed of the Ground Truth Environment Database (GTED) and Application Specific Environment Databases (ASEDs), and the

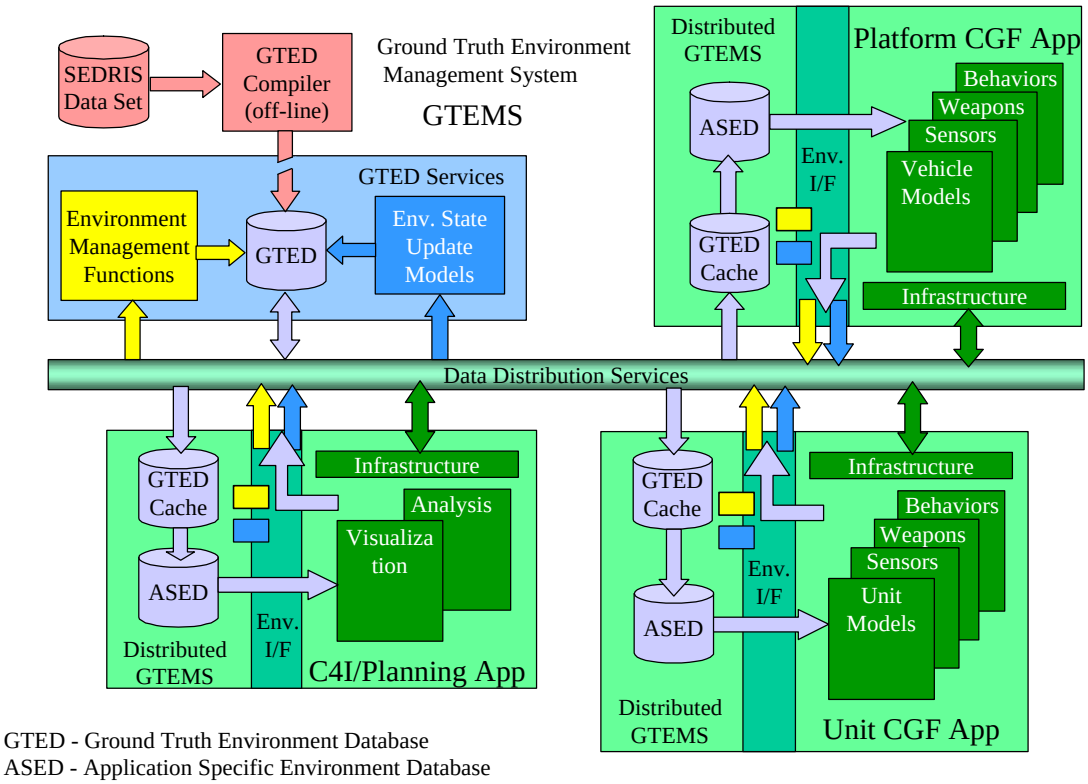

Figure 1: Distributed Ground Truth Environment Database and Management System

Environmental Interface. This section will describe these components in greater detail.

The GTEMS can be viewed as a distributed application that manages the ground truth and application specific environment databases. The GTED, which is the heart of the GTEMS, contains all of the environment data needed by any of the supported applications, including terrain, ocean, and atmospheric data. The GTEMS supports and manages the other processes, such as multiple resolution environmental state update models and management routines for initialization and dynamic updates. Future simulation systems will require that SNE data be changeable at run time. This makes it impossible to guarantee that data will always be available locally for each query, and indicates that simple pre-distribution of all SNE data is not enough to ensure availability. Furthermore, the requirement to support run time modification necessitates the existence of a mechanism for propagating changes from the originating application to all other interested applications.

The Ground Truth Environment Database (GTED) is a dynamic repository of data used to create consistent SNEs for use by the simulation applications. The GTED interface to the ASEDs provides a spatial organization which is implemented efficiently for use by the ASED compilers, utilizing the spatial indexing organization mechanism within the SEDRIS data model as the main organization construct. Other organization mechanisms, such as level of detail and state information, could be used as secondary organization constructs. Since the GTED will be large and need to be persistent, portions of it will probably be stored on disk. In order to provide data to distributed applications, latencies from disk

access and the network will have to be accounted for. Steps can be taken to ameliorate the latency issue. In particular, the GTEMS system can maintain a local cache of relevant GTED data in the process space of each application. The GTEMS manages these caches to ensure consistency and timeliness. In exercises in which occasional long latencies are unacceptable, the local caches can be made very large, effectively eliminating network latency for static data, at the cost of increasing application memory requirements. In order to help satisfy composability requirements, the GTED would be designed to allow sub-sampling both by spatial extent and by SNE content, through the use of an off-line pre-exercise process and by the GTEMS at exercise initialization. This capability has design implications such as how the data are spatially organized and the localization of indices within the data structures.

The Application Specific Environment Databases (ASEDs) are the run time versions of the SNE that the supported applications will use. The ASED object model could be a subset of the GTED object model, tailored to the run time SNE requirements of the specific applications. The ASEDs may also contain data derived from GTED data, if the derivation of that data is more efficient than storing it in the GTED representation.

A question that may be asked about this approach is why ASEDs are needed at all, i.e. why not use the GTED as the run time format for all applications. The simple answer to this is that different application types require very different run time SNE data representations. A good example of this are the differences between computer generated forces and computer image generator run time formats, where the former use the SNE mostly for planning and the latter use it for displaying textured polygons. The algorithms these applications use are closely coupled to the run time data formats, for efficiency reasons. Since future CGF systems will need to federate with other simulation systems, but still require correlated and consistent environmental representations, the GTED representation will have to be different from any of the ASED representations in order to support all of the ASED formats.

The Environmental Interface (EI) component can be divided into four main classes of functions: environmental state queries, environmental effects, ASED management, and environmental state updates. The environmental state queries would provide direct access to the geometry, features, topology, and METOC data in the ASEDs. These functions could be implemented as methods which can query data in the ASED. Examples of functions in this class are elevation lookup and temperature at a location. The environmental effects functions are algorithmic methods that typically access multiple environmental state variables. Examples of functions in this class are line of sight, transmission loss, and mobility analysis. The ASED management functions provide an interface to the GTEMS data management routines. These functions provide support for local data caching, interest management, and search space management. The environmental state update interface functions allow the modification of SNE data. Some of these functions modify the GTED data directly, while others will launch non-trivial processes that affect multiple environmental state variables. Examples of functions in this class are smoke, hydrology, and cavitation models.

Design Issues

One of the major design issues with this approach is how the ASEDs are populated and distributed. Just-in-time compilation from the GTED to the ASED formats, with interest management schemes used to limit the amount of data that needs to be locally cached within each application, seems like the most flexible approach. Environmental databases are becoming very large and complex, and the memory demands on simulation applications are growing enormously to hold these data.

Managing only the areas of the environment that are being used by the application will decrease the amount of memory required, and make dynamic updates more manageable because only the areas being used will have to be updated. Another benefit to locally caching only the data that is being used is that the amount of data compaction does not have to be as great as it is in some of the run time formats currently in use. This makes the compilation process more manageable, and makes just-in-time compilation possible.

Performance is a key requirement for all components of most simulations. For the SNE, performance can be modeled as a function of two parameters: local query response time and, for distributed SNE implementations, network transmission time. Experience has shown that the most effective way to optimize local query response time is to tailor the data organization both to the data content and to the nature of the queries of greatest importance. For diverse applications, this can only be achieved by using application-specific data formats for responding to local queries. To optimize network transmission time, the ideal solution is to store all data locally. Failing that, it is critical that the data required for most queries be available locally at the time that the queries are made. This is often achieved by pre-distributing much or all of the relevant data to interested applications at exercise start-up. However, data sets may be too large to make pre-distribution practical, especially if that data needs to stay be updated based on changes to the environment. Dynamic interest management can be used to reduce the amount of data that needs to be pre-distributed.

There are several critical characteristics of this approach to note. Firstly, applications would only access the SNE data through the ASEDs Environmental Interface. The Environmental Interface, in turn, would process those requests through interaction with the ASED; i.e. they would not access the GTED directly. This helps maintain encapsulation, and ensures that each application is served by an efficient representation, tailored to its needs. Secondly, all changes would be made through the update models in the GTEMS to ensure the consistency of the GTED; allowing changes directly to the various ASED representations would make ensuring consistency nearly impossible due to varying levels of detail. Thirdly, the SEDRIS data model should be used as the starting point to define the content of the GTED, since the SEDRIS model adequately represents the full range of SNE data. Fourthly, the interface to the SNE data should have the same look for all applications. All queries for the same information should use the same methods for access, but different algorithms may be run in those methods depending on the application. And finally, metadata should be associated with the ASEDs which would be used by the GTEMS to populate them, including information about SNE content and level of resolution.

PDES Infrastructure

As described previously, there are a number of technical challenges when implementing a distributed SNE within an optimistic PDES simulation. In this section, we will describe the infrastructure designed by the FROST team to address these challenges. This infrastructure was designed with two basic assumptions in mind. First, that each application will simulate hundreds or thousands of individually time-managed “entities,” each requiring a time-managed view of a dynamic SNE. And second, that changes to SNE data within any given entity’s immediate area of interest will generally be infrequent.

With these assumptions in mind, the FROST team came to the following conclusion: SNE data cannot be replicated and separately time-managed for each entity, but rather must be treated as a shared resource for all entities simulated on a given host. The difficulty, then, is in ensuring that each entity sees, or behaves as if it sees, a view of the SNE consistent with the events it has processed to date (and no others).

There are a number of ways to achieve this. The most obvious would be to run the entire simulation using conservative time-management, resulting in there being only one "correct" view of the SNE at any given moment. However, this solution was rejected as incompatible with JSIMS requirements. Another solution would be to maintain a conservatively managed shared base-state for the SNE, and store and locally apply change events for each entity as it accesses SNE data. This seems to be a reasonable solution, but it would be computationally expensive because it involves performing the same operations repeatedly for entities interested in the same data. The alternative that the FROST team settled upon is to manage knowledge

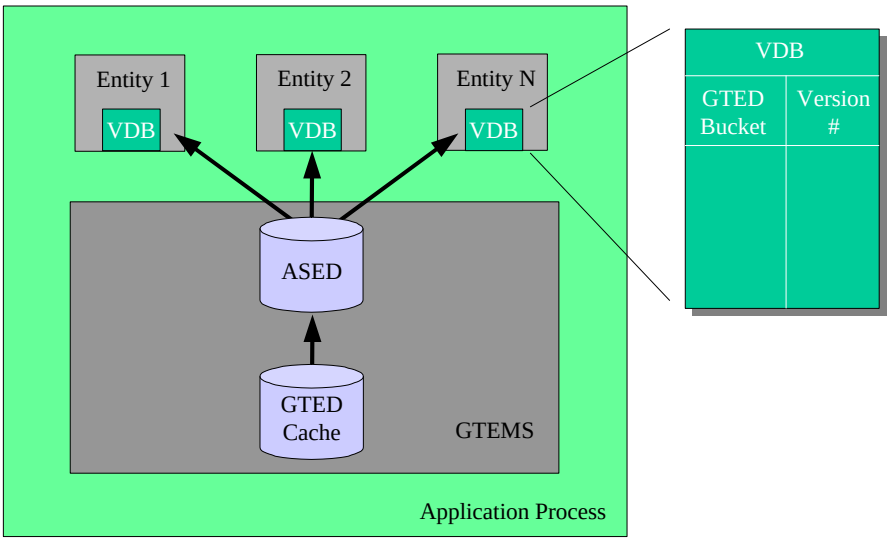

Figure 2: FROST PDES Infrastructure uses Version Databases (VDB) for each simulation entity

about changes to the SNE optimistically with the entities, but manage the SNE data itself conservatively.

The design that the FROST team developed is shown in Figure 2. The approach is as follows. The SNE data is partitioned into "buckets",each of which has a current version number. SNE data is maintained conservatively, along with its version information, in a GTED cache and one or more ASEDs in each application process (multiple ASEDs are allowed on a host only to allow the simulation of entities requiring different ASED formats within a single application). At the same time, each entity maintains an optimistically managed "version database" (VDB) of what it believes, based on the events it has processed, to be the current version number of each bucket in which it is interested. When an application requires SNE data, it first checks to make sure that the version numbers for each relevant bucket data, it first checks to make sure that the version numbers for each relevant bucket match those stored in its local table. If any of the currently available data has an old version number, then the application stalls until the desired data is available. Because the SNE data is managed conservatively, it can never be the case that data with a newer version number than that desired by the application is available. If an application processes several events and then receive a message predating those events and specifying a change

in version for relevant SNE data, then it must roll back to the time of the change event, and ensure that it acquires the specified data.

Consider the following illustrative example: Entity E1 is performing maneuvers in a region covered by SNE bucket B1. E1 receives an event notifying it that at time T1, B1's version number will change from 3 to 4. So long as E1 has not yet processed any events with timestamps later than T1, it simply updates its version table for B1 and proceeds. If at some time later than T1 it requires access to the SNE, it will check the shared SNE repository to make sure that version 4 is available. If it is, then E1 will continue simulating. If not, it will wait until the new data is present, and then proceed. To understand why the data might not yet be available, remember that the GTED and ASED repositories are managed conservatively, and so will not process the change event until there is no possibility that it will be rolled back.

The impact of this approach is that changes to the SNE tend to result in conservative execution of entities interested in the affected regions. In the worst case, frequent and widespread changes could cause the entire simulation to run conservatively over an entire execution run. However, in the more frequent case of less pervasive SNE changes, most of the simulation will run optimistically most of the time, without suffering the massive data or computation replication costs associated with providing a separate SNE representation for each simulated entity. The FROST program plans to prototype this approach, and will present experimental results in the future.

Figure 2 shows the architecture on the GTED client side only, and focuses on elements relevant to time management. Not shown are GTED servers, which are separate applications on the network responsible for maintaining GTED "ground truth". Also note that data enters the components shown in Figure 2 through two paths: simulation events update each entity's VDB, and cache synchronization methods populate each application's GTED cache as ground truth evolves.

One could argue that the time management scheme described above is inefficient, and that it is imperative that a means be found to manage SNE data optimistically to avoid stalling applications while waiting for global simulation time to advance after changes to the SNE. We believe that that could be achieved by storing changes to the (conservative) ASED local to each entity. However, such a scheme would have two primary drawbacks. First, it would involve massive replication of potentially large datasets. For example, consider a series of change messages describing a periodic update to weather data across the entire SNE playbox. Large portions of this data would need to be replicated and integrated into local ASED lookaside buffers for each entity; this volume of data could easily overwhelm an otherwise well-designed system. Second, implementing such a scheme would add yet another level of data representation that must be kept consistent with the others. Rather than being able to look to the ASED for "the" answer to any SNE query, the Environmental Interface would now need to keep track of and integrate data from both the conservative ASED and the entity-local lookaside store. While this could clearly be done, it would add considerable complexity to the system. Thus, we currently plan to pursue the Version Database approach previously described. Nonetheless, should experimentation show that our current design has unacceptable performance characteristics, we may revisit the decision not to pursue optimistic storage of ASED changes as described here.

CONCLUSION

This paper describes an approach for managing distributed synthetic environment data in a parallel discrete event simulation environment. The FROST team is

prototyping this approach using an object oriented database management system for the persistent SNE database and the SPEEDES framework for PDES. Results of this prototyping effort will be documented as part of the FROST project.

REFERENCES

Overeinder, B., Hertzberger, B., Sloot, P., (1991) “Parallel Discrete Event Simulation”, University of Amsterdam, www.cpsc.ucalgory.ca, April 1991.

Steinman, J. (1991) "SPEEDES: Synchronous Parallel Environment for Emulation and Discrete-Event Simulation", In proceedings of the SCS Western Multiconference on Advances in Parallel and Distributed Simulation (PADS91). Vol. 23, No. 1, Pages 95-103.

.STRICOM (1997) “SEDRIS Overview”, STRICOM Document from www.sedris.org, November 1997.