Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

EFFICIENT DATA CONSISTENCY IN HLA/DIS++

Sudhir Srinivasan

Mystech Associates, Inc.
5205 Leesburg Pike, Suite 1200
Falls Church, VA 22041, U.S.A.

ABSTRACT

Simulation exercises based on the DIS paradigm are faced with a severe scalability limitation due to the use of time-out updates or "heartbeat" messages for maintaining consistency despite message loss. Since they carry information that is redundant and useless in most cases, these messages represent a waste of network bandwidth and computational resources. This paper presents an elegant solution that eliminates heartbeats without compromising consistency. Rather than using additional schemes and software components as others have done, the solution proposed here consists of careful use of reliable and unreliable communications based on a key insight into the communication properties of DIS-based simulations. Not only does our solution eliminate heartbeats, it also minimizes additional resource consumption by using reliable communications minimally. Finally, due to its simplicity, it can be easily incorporated into HLA-compliant simulations.

1 INTRODUCTION

A distributed simulation consists of a set of simulators distributed geographically and interconnected via a system of networks (local and wide-area), simulating a common application, typically a battle. The Distributed Interactive Simulation (DIS) paradigm and protocols define a common way for simulators to exchange information so as to maintain a consistent view of the battlefield at each simulator. In this paper, we focus on one aspect of these protocols, namely, the *time-out update*, also known in the literature as the *heartbeat*. This data consistency feature of the DIS paradigm requires each simulator to periodically send an update on the state of each *entity* it simulates, even if the state of the entity has not changed since the last update. The DoD High Level Architecture (HLA) (McGarry, Weatherly and Wilson, 1995), which is the chosen architecture for the next generation of DIS, referred to as DIS++, provides a *minimum rate* communication service to be used by DIS-based simulations to implement heartbeats. Clearly, heartbeats impose a severe scalability limitation. With the advent of large exercises such as STOW (Calvin et.al., 1995) which are expected to simulate on the order of hundreds of thousands of entities, the scalability problem due to the heartbeat has become critical to the point of making such exercises infeasible. Consequently, there is a severe need for a scalable solution.

While reliable communication can eliminate the need for heartbeats, it introduces problems of its own (excessive bandwidth consumption and potentially higher communication latency). We describe a new solution to the problem that combines unreliable and reliable communications in a simple and elegant way to yield an effective, scalable solution. The main features of our solution are:

- it eliminates the need for heartbeats;
- it uses reliable communication only when required, thus minimizing the number of acknowledgment messages and retransmissions;
- it is simple to implement — specifically, the HLA provides the required primitives, so that our solution can be incorporated trivially by HLA-compliant simulations.

Due to the power and simplicity of our solution, we strongly believe it can solve the scalability problem for large-scale distributed simulation exercises.

2 THE PROBLEM

DIS (Dahmann and Wood, 1995) refers to two things: a *paradigm* for interoperability among distributed real-time simulators, and an IEEE standard set of *protocols and data structures* that implement this paradigm. With the advent of the HLA as directed by the DoD Modeling and Simulation Master Plan (DMSO, 1995), it is expected that the DIS community will adopt the HLA as the architecture for DIS++, the next generation of DIS.

Irrespective of the implementation (protocols vs. HLA), the basic DIS paradigm is expected to be an important technology in future distributed simulation exercises. Essentially, a distributed simulation based on the DIS paradigm consists of a set of geographically distributed simulators tightly synchronized to a global wall-clock time. The simulators maintain a consistent view of the battlefield by exchanging ground truth such as state updates (e.g. position and velocity) and interactions (firing of a weapon) and modifying their local databases accordingly. To reduce the number of state updates, DIS uses a mechanism called *remote entity approximation* (or *dead reckoning*, DR) wherein simulated entities use a common algorithm to estimate the trajectory of other entities based on the most recently received updates. With this scheme, state updates are generated only when the actual trajectory of an entity differs from the trajectory estimated by the DR algorithm by some threshold amount. In the rest of this paper, "DIS" refers to the DIS paradigm rather than the protocols, unless specified otherwise.

DIS does not assume a reliable communication medium underlying the simulators. In other words, it is possible for messages (in particular, state updates) to be lost. To compensate for lost updates, DIS includes a *time-out update* mechanism, commonly known as the *heartbeat*. Under this mechanism, an entity must generate an update if a given amount of time has elapsed since the last one, even though its state or appearance have not changed. The periodicity of these heartbeats is adjusted (typically, 5 seconds) so that the error due to lost updates is within acceptable bounds. Thus, the primary use of the heartbeat mechanism is to maintain data consistency in the face of lost messages. Heartbeats have also been used to detect the removal of an entity from an exercise — an entity that has not sent a heartbeat for a certain number of time-outs is considered deleted.

Heartbeats impose a severe scalability limitation on DIS exercises. This scheme worked well for the original SIMNET protocols because SIMNET exercises were on a small scale (few tens of entities). DARPA's STOW program expects to conduct exercises involving on the order of 100,000 entities by the year 2000 (Calvin et.al., 1995). A large number of these entities are expected to be in a steady state most of the time (i.e. their state and appearance change very infrequently). Unfortunately, DIS requires these entities to generate updates at every time-out. Clearly, this will place excessive demands on the network (for transporting messages) and processors (for generating and consuming the messages). This (mostly redundant) traffic is projected to constitute a significant portion of the total traffic in large-scale exercises. The problem will only worsen as DIS++ incorporates new kinds of entities such as dynamic terrain and environmental effects (smoke and weather). Consequently, there is a significant need for a scalable solution that eliminates or minimizes the use of heartbeats.

3 RELATED WORK

To date, only two efforts have addressed the problem: the STOW real-time information transfer and network (RITN) project (Van Hook, Calvin and Smith, 1995) and the DIS lite & query protocol (Taylor, 1995).

The STOW approach is based on a key characterization of the state of DIS entities into *active* and *quiescent* states. A quiescent entity is one that generates only heartbeats (we defer a formal definition to Section 4.2 since our solution is based on these definitions as well). The main idea behind STOW's *consistency protocol* is to reduce the number of heartbeat messages generated by combining the updates for *all* quiescent entities at a simulator into a single message at each timestep. The protocol requires simulations to effectively maintain a shared *consistency list* that is used to track the set of quiescent entities and their versions. Using the information in the consistency lists, entities receiving a heartbeat update can determine the quiescent entities for which they do not have current information and request this information using a *negative acknowledgment* (NACK) scheme. All entities that do not have current information send a NACK message indicating the missing pieces of information to the emitter of the updates which responds with the missing information. Since a large number of NACK's could be generated in a short period of time (NACK's follow a bursty pattern), the protocol incorporates a *NACK suppression* scheme using back-off in which a sender of a NACK waits a random amount of time before sending its NACK. If the NACK of another sender is received during this back-off period, the sending of the NACK is aborted.

Clearly, the STOW approach is fairly complex (the consistency list is a shared data structure with a single writer and multiple readers, the implementation of which is non-trivial; so also the NACK suppression scheme). Recognizing scalability limitations of the consistency protocol, STOW has proposed *consistency agents* to off-load some of the consistency tasks from the simulations. A key factor to good performance is quiescence detection. Simulations must be able to accurately determine when entities have become quiescent or active. Quiescence is usually detected when an entity has been generating heartbeats for some period of time. This threshold period must be chosen very carefully since it introduces a performance trade-off: if quiescence is detected too often, the overhead of the consistency protocol will degrade performance; on the other hand, if quiescence is not detected often enough, the benefits of the protocol are not

observed. To simplify this problem, the STOW program adopted a weaker definition of quiescence where entities are considered quiescent only when they are stationary (trading off some performance in the process). It is important to note that the consistency protocol requires heartbeats to be *broadcast* to all simulations (whereas state updates would be multicast to only those simulations that require them). As such, the scalability of the STOW scheme improves upon that of DIS only in that it is limited by the number of simulations rather than the number of entities. While this represents a significant improvement, we believe it will prove insufficient as exercises that involve large numbers of simulations are planned.

The DIS-Lite & Query Protocol (DLQP) is based on an analysis of the contents of the *entity state protocol data unit* (ESPDU), the most often used unit of data exchange in DIS. DLQP proposes alternate "lightweight" PDU's that would be used instead of the ESPDU's, to reduce the bandwidth consumed. Further, it defines "minimal" versions of these new PDU's that would be used for the heartbeats, again, to reduce bandwidth consumption. Finally, it proposes increasing time-out intervals to reduce the number of heartbeats. A query protocol is used to fill in gaps of information, much like the NACK scheme of the consistency protocol. DLQP uses a suppression scheme for queries similar to that used for NACK's in the consistency protocol. Since DLQP is based on the structure and contents of the ESPDU, it is not clear how it will be extended to DIS++ which will use the HLA interface specification for data exchange, rather than the ESPDU. A more detailed comparison of DLQP with the STOW consistency protocol is given in (Smith and Van Hook, 1996).

4 TOWARDS A NEW SOLUTION

At first thought, it seems the solution to the heartbeat problem lies in reliable communications: if all messages were sent reliably, there would be no need for heartbeats. Unfortunately, reliable communication has its own costs. The main drawback of reliable communication is that reliability is achieved using acknowledgment messages and retransmission of lost messages which increases the bandwidth consumed, compared to unreliable communication. This introduces an interesting trade-off that is an open problem worthy of investigation: on the one hand, the reliable communications for *all* updates increases the bandwidth consumption; on the other hand, the use of heartbeats with unreliable communications also increases bandwidth consumption — which is better? Ironically, while reliable communication protocols are designed to alleviate the loss of messages, it is precisely that situation that causes the most problems for these protocols. It is easy to reach a state where the loss of a message causes a sender's retransmission window to fill up and thus halt further sending of new messages (thereby increasing the latency of communication for those messages). This second problem is severe enough due to the real-time requirements of distributed simulations to make reliable communication *for all messages* infeasible (at least at the time of writing — as we move towards more reliable communication media, reliable communications may prove to be the solution).

4.1 Requirements

From the above discussion, some combination of reliable and unreliable communication appears necessary, to solve the scalability problem of heartbeats. Ideally, the solution must utilize unreliable communications most of the time and use reliable communications only when required. The goal is to develop a solution that eliminates (or minimizes) the use of heartbeats while minimizing the use of acknowledgments and retransmissions.

Ideally, a scheme that eliminates heartbeats would be preferable. In practice, the DIS community has used heartbeats for a second purpose: detecting the loss of entities. An entity from which some specified number of contiguous heartbeats have not been received is deleted to prevent other entities from unrealistically engaging or dead-reckoning it. If heartbeats are eliminated, we must address this issue of "dropped entities" — i.e. how to detect entities that are either no longer operational or that cannot communicate with the rest of the simulation.

In keeping with the philosophy of the STOW RITN project (Van Hook, Calvin and Smith, 1995), we argue that the primary focus of efforts in this area should be on the data consistency aspects, for the following reasons:

- The issues of data consistency and detecting dropped entities were originally coupled because a single mechanism (heartbeats) could address both. *There is no other reason to link these two issues.* Since the heartbeat mechanism is proving inadequate to solve the data consistency issue, we argue that the two issues should be decoupled and independent solutions developed for each.
- The decoupling of the two issues allows the use of more efficient schemes (such as query-reply schemes using unicasts rather than multi- or broadcasts) to detect dropped entities
- Such outages are temporary in nature and during such outages, interactions are meaningless by definition (Smith and Van Hook, 1996). Schemes developed to detect such situations must focus on quick detection rather than recovery since no protocol can compensate for the semantic gap created by the sudden loss of entities.

4.2 Solution Strategy

Under the DIS paradigm, state updates must be emitted under three conditions:

- dead-reckoning threshold is exceeded
- appearance has changed
- time-out period has elapsed

An entity that emits updates under the first two conditions is said to be in an *active* state while an entity that emits updates only under the last condition is said to be in a *quiescent* state. Note, stationary entities (whose appearance is not changing) are quiescent by definition, but quiescent entities are not necessarily stationary — a moving entity that does not deviate significantly from its dead-reckoned path is also quiescent. Typically, entities transition between these states in the course of a simulation: an active entity could become quiescent; a quiescent entity could become active; a quiescent entity could move to another quiescent state (for example, by changing its appearance only).

To understand the solution we propose, we must first understand the nature of communications among DIS simulations. A very important requirement is low latency of communication. DIS simulations operate in real-time (typically with humans in the loop). To be realistic, the communication latency must be low enough (~ 100 ms) so that the occurrence of an event at one simulator and the notification of the same at another are effectively instantaneous. An important consequence of this real-time requirement is that it limits the benefits of reliable communications, as follows: the loss of a message implies the information in the message will not reach its destination at the time it would have reached if the message had not been lost. In a sense, we can say the damage is already done when a message is lost. Any remedial action (such as a retransmission) can only reduce the effect of this damage, but cannot eliminate it. Another factor that limits the benefit of reliable communication is the property that a new update makes any older update obsolete immediately. Thus, a retransmission by a reliability mechanism pays off only if it happens closer in time to the original lost message than to the next update. Recalling the definition of active and quiescent states of entities, this implies that reliable communications will not provide significant benefit when an entity is in an active state. Since the goal is to eliminate the heartbeats in the quiescent states, we have the following insight which is key to the solution proposed here:

Observation: Reliable communications will have the most benefit if used while transitioning to a quiescent state.

Based on this observation, we describe next the solution we propose to solve the scalability problem due to heartbeats.

5 THE SOLUTION

The rationale behind our scheme (developed in Section 4) is summarized in Figure 1. In an active state, the entity generates new state updates often enough so that the time-out does not occur. Reliable communication does not provide significant benefit in this state because new updates (being generated frequently enough) subsume older updates. In other words, the time to retransmit the lost message (including timing out waiting for an acknowledgment) will typically be close to or greater than the time to the next new state update, significantly reducing or eliminating the benefit of the retransmission. In the quiescent state, we wish to eliminate the s completely. Since the heartbeats are generated primarily to ensure data consistency in the event some simulators do not receive some of these updates, our solution is to use reliable communication while transitioning to a quiescent state, thus eliminating the need for heartbeats in the quiescent state.

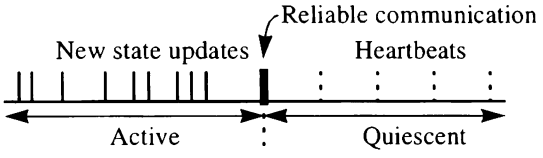

Figure 1: Rationale

To implement our solution, a simulation follows the state diagram of Figure 2. The upper entry in each state describes the state while the lower entry (in italics) describes the communication paradigm used in that state. In an Active state, the entity sends updates using unreliable communications (as is done in DIS now). The occurrence of a time-out is taken as an indication that the entity is transitioning to a quiescent state and the entity enters the Transition state where it sends an update using reliable communication. In the Transition state, one of two things can happen: if the entity generates a new update, then it is put back into the Active state and the update is sent using unreliable communication as before; if a second time-out is generated, the entity enters the Quiescent state where updates are suppressed. The entity comes out of the Quiescent state when it generates its next new state update, at which time it moves to the Active state. Note, it is possible to reduce the Transition and Quiescent states into a single state that sends only one message with reliable communication and suppresses all subsequent messages. With this optimization, a single time-out is needed to transition the entity out of the Active state — thereafter, the timer can be disabled until the entity re-enters the Active state. It is very important to note that the two classes of messages (reliable and unreliable) should follow different logical paths in the

communications software of a simulation. This is required so that hold-ups in the reliable communication channel (such as a full retransmission buffer) do not interfere with the unreliable communications.

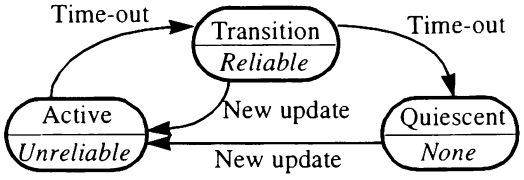

Figure 2: Implementation

The simple scheme we have proposed above has several powerful advantages:

- It eliminates heartbeat messages completely — the only update is in the transition state and this update is required to preserve the effect of the heartbeats (data consistency).
- It uses reliable communication only where it is most likely to be beneficial.
- It uses reliable communication at a minimum, resulting in a minimum number of projected acknowledgments and retransmissions (and thus, extra bandwidth consumed).
- Reliable communications do not interfere with unreliable communications, so that in an active state, the entity's updates can be delivered in a timely manner.
- It does *not* require quiescence detection; in a sense, the quiescence detection is built into the scheme (note also, we are using the stronger definition of quiescence here, where a quiescent entity need not be stationary).
- Any retransmissions required due to lost messages are handled by the communications software rather than the simulation, which will lead to better performance since the retransmissions can presumably be performed more efficiently at the communications layer.
- Since updates are usually sent to several receivers, the use of reliable communications has the benefit that retransmissions are sent only to those receivers that did not receive the original message. With the heartbeats of DIS, effectively, the retransmission is sent to *every* receiver of the original update.
- It can benefit single-entity simulators as well as those that simulate a number of entities, such as synthetic forces.
- It is relatively easy to implement.

The power of the solution may be enhanced by the following optimization. Recall, the solution utilizes the property that newer updates subsume older ones. Consider an entity that has just moved into the Transition state (Figure 2). Soon after, it generates a new state update, moving it back into the Active state. In such a case, it would be beneficial if the simulation had the capability to abort the reliable transmission of the message it sent in the Transition state (if such a transmission is in progress). While this capability may not exist with off-the-shelf communication software such as TCP, it is one that can be built in easily. This optimization could be useful in the case where an entity toggles back and forth between the Active and Transition states.

Theoretically, the scalability of this solution is limited only by the number of entities in the Transition state simultaneously (i.e. the number of entities using reliable communications at any time). In practice, this number is expected to be very low. Optimizations such as controlling time-out intervals and aborting unnecessary retransmissions (described above) can further improve scalability.

6 HIGH LEVEL ARCHITECTURE/DIS++

One of the advantages of the solution we have proposed here is its ease of implementation. This is especially true for simulations that are HLA-compliant. The HLA is a part of a common technical framework being developed by the Architecture Management Group (AMG) of the Defense Modeling and Simulation Office (DMSO) to promote interoperability and reuse among simulations. It consists of a set of compliance rules, an object model template (OMT) to describe the object models of simulations (and federations of simulations) and an interface specification describing the interfaces using which the simulations interact. The services described in the interface specification are provided by a *run-time infrastructure* (RTI). Among other things, the RTI provides communication services to the simulations. Currently (Architecture Management Group, 1996), the RTI is expected to provide both reliable and unreliable (called *best-effort*) communication. As such, the RTI provides the primitives needed to implement our solution. HLA-compliant simulations can simply choose the appropriate RTI service depending on the state they are currently in.

The RTI being developed currently by the Defense Modeling and Simulation Office is a prototype designed primarily for demonstrating the HLA concept in practice. As such, some aspects of its design are guided by schedule rather than by performance. Therefore, while our solution can be implemented using the RTI prototype, performance may not be satisfactory. We expect the performance to improve with future RTI implementations.

7 CONCLUSION

The use of time-out updates (or heartbeats) in the DIS paradigm poses a scalability limitation for distributed simulation exercises. This problem has become acute with the advent of large-scale simulation exercises such as STOW and alternate schemes are needed. We have proposed an elegant yet powerful solution to this problem. Our solution consists simply of judicious use of reliable and unreliable communications based on a key insight into the communication requirements of DIS simulations and does not require any additional software/hardware schemes. It completely eliminates the use of heartbeats. Further, it uses reliable communications only when required and in a way such that they need not interfere with the normal, unreliable communications used for state udpates. Consequently, it minimizes the use of acknowledgments and retransmissions. Overall, we believe it will significantly improve the scalability of distributed simulations. Finally, our solution is easy to implement, especially in HLA-compliant simulations.

REFERENCES

Architecture Management Group. 1996. Interface Specification for the HLA, Version 0.5. Working document available from URL: http://www.dmso.mil/docslib/hla/IF_Spec_v0.45.doc, Defence Modeling and Simulation Office, Alexandria, Virginia.

Calvin, J.O., J. Seeger, G.D. Troxel, and D.J. Van Hook. 1995. STOW Realtime Information Transfer and Networking system architecture. In *Proceedings of the 12th Workshop on Standards for the Interoperability of Distributed Simulations*, 343-353, Institute for Simulation & Training, Orlando, Florida.

Dahmann, J.S., and D.C. Wood. 1995. Editors, Special Issue on Distributed Interactive Simulation. *Proceedings of the IEEE*, Vol. 83, No. 8.

DMSO. 1995. DoD Modeling and Simulation Master Plan, Defense Modeling and Simulation Office, Alexandria, Virginia, October 1995.

McGarry, S., R. Weatherly, and A. Wilson. 1995. The DoD High Level Architecture (HLA) Run-time Infrastructure (RTI) and its relationship to distributed simulation. In *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, 595-606, Institute for Simulation & Training, Orlando, Florida.

Smith, J.E. and D.J. Van Hook. 1996. Comparison of consistency protocol vs. DIS-Lite. In *Proceedings of the 14th Workshop on Standards for the Interoperability of Distributed Simulations*, 875-884, Institute for Simulation & Training, Orlando, Florida.

Taylor, D. 1995. DIS-Lite and Query protocol. In *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, 697-703, Institute for Simulation & Training, Orlando, Florida.

Van Hook, D.J., J.O. Calvin, and J.E. Smith. 1995. Data consistency mechanisms to support distributed simulation. In *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, 797-806, Institute for Simulation & Training, Orlando, Florida.

AUTHOR BIOGRAPHY

SUDHIR SRINIVASAN is a research scientist at Mystech Associates, Inc., conducting research in parallel and distributed modeling and simulation. He received his Ph.D. in Computer Science from the University of Virginia in 1995 and Bachelor of Engineering also in Computer Science from the Bangalore University, India, in 1990. From August 1995 through February 1996, he was also a research associate at the University of Virginia, working on identifying fundamental issues in linking models at different levels of resolution. His main research interest is in parallel and distributed simulation, especially the High Level Architecture. He is supervising the on-going development of a performance analysis tool for the HLA. His other interests are in parallel and distributed computing and networking. He has published several papers in conferences and journals and is a member of the ACM and the IEEE Computer Society.