Multicasting in DIS:
A Unified Solution

Sudhir Srinivasan
Bronis R. de Supinski

Computer Science Report No. CS-95-17
March 21, 1995

Multicasting in DIS: A Unified Solution

Sudhir Srinivasan
ss7a@uvacs.cs.Virginia.EDU
http://uvacs.cs.Virginia.EDU/~ss7a/

Bronis R. de Supinski
brd5y@uvacs.cs.Virginia.EDU
http://uvacs.cs.virginia.edu/~brd5y/

Department of Computer Science
University of Virginia,
Charlottesville, VA 22903
(804) 982-2294

Abstract

Multicasting, as an alternative to broadcasting, has great potential to improve DIS scalability by reducing the demands on network bandwidth and computational resources. We take an eclectic approach to incorporating multicasting into DIS, blending new ideas based on insights discussed here with the best schemes proposed previously. To our knowledge, no previous work has provided such a unification. The simplicity and completeness of our approach make it more promising than previous ones.

In general (and in DIS in particular), the multicast problem should be considered as consisting of two parts: i) definition and use of multicast groups, and ii) efficient implementation of these groups. We describe a method for defining and using multicast groups, present its implementation requirements and show these to be realizable. Further, we provide insights into the configuration of DIS networks in order to obtain the most benefit from multicasting. Finally, we note that management of static objects is inherently related to any multicast solution. The integration of static object management with our multicast solution is made possible by its static nature and use of a regular grid.

Appeared in: *ELECSIM 1995*, Electronic conference held on the Internet from April 10 - June 18, 1995

1. Introduction

The broadcasting by each simulation entity of state changes poses the most appreciable and immediate limit to the scalability of the DIS architecture. As the size of the simulated environment increases, it becomes likely that these state changes are irrelevant to many of the receiving simulation entities. This leads to the use of relevancy filtering, whereby simulation entities simply discard irrelevant messages. While this reduces the computation load on the entities, as the number of entities increases, relevancy filtering can consume unacceptable levels of computation power. Multicast schemes are frequently proposed to reduce both network bandwidth requirements and the demands of relevancy filtering. We propose a comprehensive scheme which simplifies relevancy filtering, reduces network bandwidth requirements and integrates static object management.

DIS is a distributed system of entities interacting with each other to simulate some complex real-world activity, such as a battle. In order to simulate a common scenario coherently, entities must be aware of the states of other relevant entities, where relevancy may be defined by proximity in the virtual environment (VE) (also known as simulated environment, virtual world, conceptual space, etc.). Thus, one of the critical aspects of DIS is the timely exchange of state information among entities. In SIMNET (an ancestor of DIS), this exchange is achieved by having each entity broadcast its state periodically to all other entities. While this scheme works for the small-scale systems for which SIMNET was intended, it will not scale to the systems being considered for DIS which are expected to have millions of entities. For a more detailed analysis of this scalability problem of DIS, see [MZP95].

The reason broadcasts were used in earlier systems is that the VE's were small enough that each entity needed to be aware of most other entities. Large scale DIS exercises are expected to simulate much larger VE's than before. As the size of the VE grows, the communication requirements for entities become relatively localized. If state changes are broadcast to all entities, each entity must perform relevancy filtering on the messages it receives and discard those that are irrelevant to its simulation. Thus a broadcast scheme generates a lot of unnecessary work which can be avoided if the state exchanges are done in a more controlled manner. This locality of communication suggests the use of multicasting instead of broadcasting. By propagating each message only to those entities that require it, we can minimize relevancy filtering as well as reduce network bandwidth.

In the following section, we place the multicast problem in the context of DIS. We then propose a multicast scheme based on a static, regular partitioning of the VE which is used to determine static, possibly irregular, regions of the VE to use as multicast groups. While static schemes are falling out of favor, we believe after thorough analysis that they are still the most effective way to reduce DIS message overhead. Next, we discuss the relationship between static object management and the multicast problem and demonstrate that an efficient static object management mechanism is consistent with our multicast strategy. Finally, we note the importance of the physical design of the network and discuss an important class of simulation entities, broad receivers, for this design.

2. The Multicast Problem

The multicast problem in DIS is two sided in that many entities require the state changes of an individual simulation entity and an individual entity requires the state changes of several other entities. That is, communication in DIS is both one-to-many and many-to-one. The immediate effect of this requirement is that entities multicast to the group of entities interested in their state changes, while listening to any multicast groups whose state changes are relevant. Thus a multicast scheme must provide an efficient mechanism for determining both the group to which the entity will send and the groups to which it listens.

A multicast solution cannot be proposed in isolation. The network architecture must reflect the communication requirements (both static and dynamic) of the simulation entities. If it does not, no multicast scheme can be effective.

Additionally, a successful multicast solution must accommodate the management of static objects. The state of static objects may change as a result of the actions of simulation entities, and thus state changes of relevant entities must reach these objects. Further, state changes of these objects affect the actions of simulation entities, and thus must be sent to interested entities. Clearly, this facet of static object management is simply a specific instance of the multicast problem.

The multicast problem must therefore be viewed as consisting of two parts:

- the definition of multicast groups (addresses) and the way in which the application uses them
- the implementation of these multicast groups in the network architecture

These two are largely orthogonal, although some design choices in one may have implications in the other.

We focus primarily on the first part of the multicast problem. In the context of DIS, this requires: linking multicast groups with the VE; establishing schemes employed by the entities to use (i.e send to and listen to) these multicast groups; and integrating static object management with the definition and use of multicast groups

Efficient implementation of multicasting is still being researched in the networking community. Some simple implementations exist but these are not efficient. We study the particular requirements placed on the implementation by a large-scale distributed system such as DIS and suggest a simple implementation that meets these requirements.

3. Multicast Groups

Multicast groups (MCG's) may be organized based on different criteria. Several excellent criteria are suggested in [MZP95, DIS93], for example, physical proximity, functional relationships, fidelity requirements, temporal behavior. In most of these cases, the design of the MCG's is fairly straightforward. However, groups based on spatial proximity among entities in the VE (called the *spatial class* in [MZP95]) present some interesting challenges. Intuitively, spatial MCG's seem to be the most obvious way of exploiting multicasting in DIS. If entities are close to each other in the VE, they are likely to perceive each other, requiring the corresponding simulators to communicate. Correspondingly, if entities are far apart in the VE, their simulators may not need to communicate. By defining MCG's based on spatial proximity, we may thus minimize or even eliminate unnecessary message traffic and relevancy filtering.

However, it is very important to note that *proximity in the VE does NOT imply proximity in the real world*, i.e., two entities engaged in combat with each other may be simulated by computers located in different continents. All message traffic between these two entities *must* be routed to the network segments that contain the simulators in both continents. In the worst case, every message could potentially go to every network segment in the DIS network (because there is at least one entity in each segment that listens to any particular MCG). In this case, multicasting will degrade to the equivalent of broadcasting and could in fact become worse than broadcasting depending on how the MCG's are used and the how the multicast is implemented. This observation suggests that care must be taken in configuring the network. We provide some insights later.

3.1 A unified solution

The scheme we propose for the first part of the multicast problem, the definition and use of MCG's, can be best understood in four steps:

- static partitioning of the VE
- static allocation of MCG's
- determination of the MCG's to which each entity "sends"
- determination of the MCG's to which each entity "listens"

The key word above is "static". The scheme is static because an allocation of MCG's to points in the VE is determined prior to an exercise and remains fixed throughout the exercise. This is as opposed to dynamically changing MCG's where, for instance, a "large enough" group of entities constitutes a MCG which moves with the entities

as they move through the VE. Such a dynamic approach has several problems:

- when is a group “large enough” to warrant a MCG?
- when is a group “too large” so that it should be split into smaller ones?
- when are two or more MCG’s close enough in the VE that their members should begin to receive each others’ messages?
- do entities change MCG’s and if so, when?
- how to incorporate static objects?

Most of these problems can be solved by using some form of distributed arbitration which is expensive and therefore may not be feasible in a real-time environment such as DIS. Thus, while such a dynamic scheme has some intuitive appeal, it appears to have more drawbacks than benefits.

3.1.1 Static partitioning of the VE

The first step is to partition the VE into a *regular* grid structure. This grid discretizes the VE in a regular way which makes it easy to compute memberships in MCG’s. The grid must be of a simple, regular shape such as square, rectangle or hexagon. The granularity of the grid is influenced by several factors and must be selected carefully. Since the same grid will be used to partition the static objects, the grid cells must be small enough so that they each contain a manageable number of static objects. On the other hand, the finer the grid, the more often an entity has to recompute its grid position. If the grid cells are too large, MCG’s will tend to cover entities that do not need to communicate with each other.

One factor which makes this trade-off difficult is the widely varied density of static objects common in large scale VE’s. Hierarchical grids have been suggested as a solution to this varied density. With hierarchical grids, dense areas can be divided into finer resolution grid cells, thus reducing the number of static objects in each grid cell. As we will demonstrate in section 4, explicit hierarchical grids are not necessary for static object management. They are beneficial for the definition of MCG’s only if the perception capabilities of simulation entities changes with location.

Regardless of the grid partitioning scheme, the grid granularity must be based on a variety of factors. These include the terrain, the nature of the engagement, the strategy chosen for allocating MCG’s and the variance of the density of static objects. We believe experience will play a significant role in choosing a good grid size.

3.1.2 Static allocation of MCG’s

Having discretized the VE with a regular grid, the MCG’s may be overlaid on the grid in two ways:

1. *Regular* - the MCG’s are formed by superimposing a second regular grid over the grid that partitions the VE such that each cell in the former encompasses one or more cells in the latter (Figure 1). In its simplest form, the two grids are identical so that each cell in the grid corresponds to a single MCG, as suggested by [JoM92] (Local Effects category) and [MZP95] (example hexagonal MCG’s). However, there are competing factors for this simple scheme: on the one hand we wish to make the MCG’s, and hence the grid cells, large so as to reduce the number of MCG’s; on the other hand, we wish to keep the grid cells small so that the number of static objects per cell is manageable by a single computer. The more general mapping shown in Figure 1 allows a good balance in this trade-off. The main advantage of this scheme is that a simple mathematical formula may be used to determine the MCG’s relevant to an entity given its position in the VE.
2. *Irregular* - the MCG’s can be arbitrarily shaped groups of adjacent grid cells. Figure 2 shows an example. This scheme allows us to take advantage of terrain information in defining MCG’s. For instance if we know an impassable mountain separates two groups of entities (thus limiting their interaction), the two sides of the mountain can be assigned separate MCG’s. Similarly, if it is known that a large area of the VE is going to have very little activity, it can be assigned a single MCG as opposed to the many MCG’s that would be used in a regular allocation. The main advantage of this scheme is that by selecting the

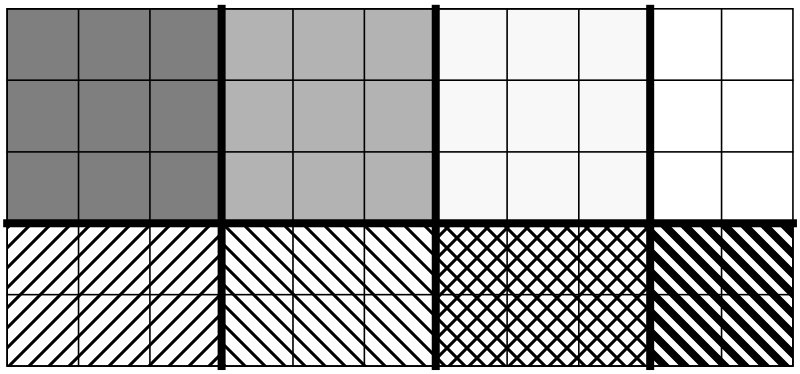


Figure 1 - Regular shaped multicast groups based on a regular grid

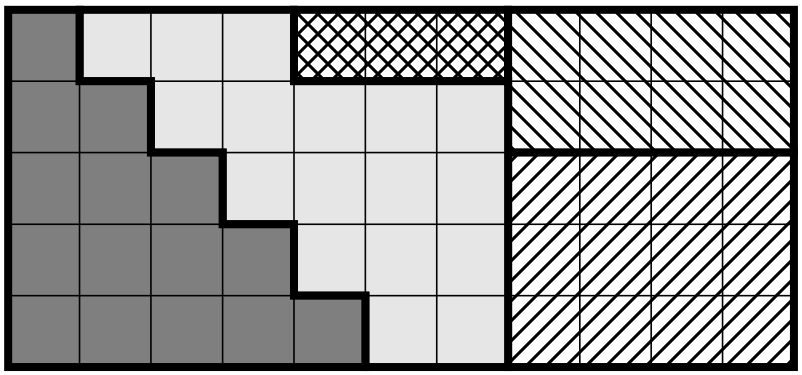
shapes and sizes of the MCG's carefully, we can dramatically reduce the number of multicast addresses being used as well as unnecessary communications.

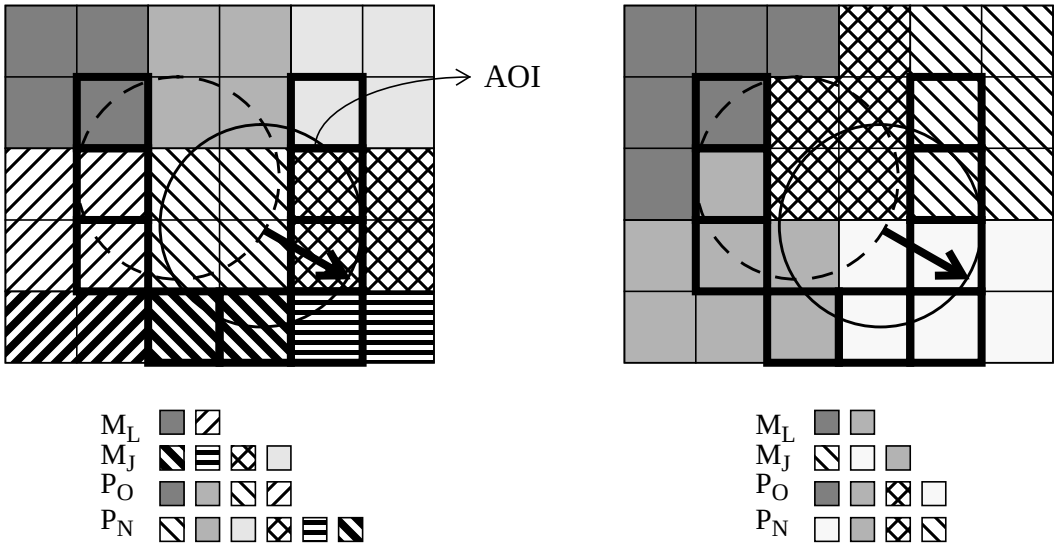
We wish to emphasize the separation between partitioning the VE using a grid and overlaying the MCG's on this grid. This separation allows us to define irregularly shaped MCG's while alleviating the biggest problem of irregularly shaped grids, the determination of boundary crossings. In our scheme, a simple mathematical formula can be used to determine the grid cell corresponding to a given position. Determining the MCG corresponding to a grid cell can be as simple as a table look-up.

Regular MCG's have two main disadvantages. First, it is possible to have too many MCG's. As an example, consider the situation where the VE spans continents and the grids are defined on the order of kilometers (typical when tanks are engaged in battle). It is clear that this is wasteful because MCG's are being allocated to areas of the VE, such as the ocean, where little or no activity will take place. Secondly, since this scheme does not explicitly allow us to take advantage of terrain information, it may result in unnecessary communication.

Irregular MCG's have some significant drawbacks as well. As we shall see later, entities change the set of MCG's they are listening to incrementally as they move. As pointed out in [MZP95], computing the change is trivial with regular MCG's such as hexes (because the change is also regular). However, with irregular MCG's, this task may require relatively more computation

Figure 2 - Irregularly shaped multicast groups based on a regular grid





(a) Regular MCG's (a) Irregular MCG's

Figure 3 - Incrementally tracking the AOI

in spite of the regular grids. Secondly, the success of this scheme depends on a careful analysis of the terrain database and the nature of the exercise. In reality, it may not be possible to predict exactly where the action will take place. In other words, incorrect analyses can lead to poor allocation of MCG's and consequent performance penalties. Finally, as mentioned in [ShB92], changes in terrain databases may change the allocation of MCG's requiring a re-analysis.

The primary factors that must be used to decide between regular and irregular MCG's appear to be: the scale of the VE, availability of multicast addresses, nature of the terrain and the cost of doing incremental changes in MCG membership using irregularly shaped MCG's.

3.1.3 Sending to MCG's

Each entity follows two rules to determine which MCG to send its messages to. First, it sends its periodic state updates to the MCG that corresponds to its current location in the VE. The second rule is for when an entity affects another entity in a different grid cell than its own. In this case, the entity causing the effect must send the message to the MCG corresponding to the grid cell in which the target entity resides. Generalizing this, when an entity causes changes in entities that reside in grid cells other than its own, it must send the messages to the MCG's corresponding to each of those grid cells (duplicates can be eliminated). Note, in the case of self-guiding munitions, the munition is instantiated as a separate entity and as it moves through different cells, it must send updates to the MCG through which it is currently moving.

3.1.4 Listening to MCG's

The sending scheme outlined above can be thought of as a "distributed mailbox" system. Each MCG acts like a mailbox into which certain entities drop messages. It is the responsibility of other entities to "watch" any mailboxes that are relevant to them and receive those messages. This scheme fits well with the reality that it is the sensors that perceive other entities passively. The *perception envelope* or *Area Of Interest (AOI)* of a sensor determines the entities it can observe. Correspondingly, the AOI of an entity determines the MCG's to which it listens. In the VE, the AOI is typically approximated as a circle centered at the entity. The MCG's that are covered (even partially) by this circle are the ones to which this entity must

listen. We call this set of MCG's the *working set* of an entity.

Since the AOI moves with the entity but the MCG's are static, a simulator must track its working set. A logical approach is to maintain the working set incrementally (i.e. only compute the change as the entity moves). Figure 3 shows how the use of a regular partitioning grid makes it easy to track the working set in our scheme even with irregular MCG's. For this discussion, we define the *periphery set* of the AOI of an entity as the set of all grid cells that are in the AOI and are adjacent to one or more grid cells that are not in the AOI. To determine the change in its AOI, each entity computes four sets of MCG's: M_L is the set of MCG's corresponding to the grid cells that leave the entity's AOI, M_J is the set of MCG's corresponding to the grid cells that join the entity's AOI, P_O is the periphery set corresponding to the entity's old AOI and P_N is the periphery set of the entity's new AOI. Then M_L - P_N is the set of MCG's that the entity need no longer listen to and M_J - P_O is the set of MCG's it must begin listening to.

Steinman and Wieland [StW94] point out an important potential source of error due to the fact that an entity computes its current grid location only periodically. If an entity crosses into a new grid cell between successive grid location computations, there is a time period during which it is in error regarding its current grid location. In this period, it will continue to send its state updates to the MCG corresponding to its old grid location when in fact it should be sending them to the MCG of the new location. There may be sensors which can sense this entity in its new location in reality but cannot do so in the simulation because the old MCG is beyond their AOI. A similar problem may occur when an entity crosses into and out of a grid cell in between successive grid location computations. The authors [StW94] present a satisfactory solution using "fuzzy grids" which is incorporated into our scheme very easily. Their solution is to inflate the AOI of sensors by fixed amounts to compensate for the fact that other entities as well as the sensors themselves are moving. In our scheme, the inflated AOI should be used to determine the set of MCG's (figure 3) that an entity must listen to.

4. Static objects

In order to be realistic, simulations must incorporate static objects in the terrain such as buildings, rock formations, trees or bomb craters. Clearly, there can be a very large number of such objects in a small area of terrain. As the simulation proceeds, these static objects may be affected (buildings may be destroyed, craters may form, etc.). These changes must be made known to participating entities because the changes may affect the course of the rest of the simulation. Since these objects are usually multi-state [Mil92] and can be affected by multiple entities at the same time in the simulation, the main problem is to provide all entities with a consistent view of the state of these objects. There are basically two ways of solving this problem: distributed consensus or centralized arbitration. The former has obvious time costs which make it infeasible for DIS. Miller [Mil92] proposes the use of special Object Management Processors (OMP's) whose sole responsibility is to maintain the states of static objects. These processors listen to the DIS network for messages that could cause changes in the states of any static objects for which they are responsible. Upon receiving such messages, the OMP's process them and communicate state changes to other entities. Another responsibility of the OMP's is to inform newly arrived entities of the states of all static objects.

Although the static object problem is not directly related to the multicast problem, its solution must be consistent with the multicast scheme we have outlined. The OMP's, like other entities, must send and listen to appropriate MCG's in order to perform their duties. To fit into our multicast scheme, the OMP's can be treated as entities whose AOI's span only those grid cells in which static objects that they manage are located. Thus, the OMP's send messages to and receive messages from the same set of MCG's. When an entity moves to a new position and begins sensing a new grid cell as a result, it sends a message to the MCG corresponding to that cell asking for information about all static objects in that cell. The

OMP managing the objects for that cell must then respond by sending the data reliably to the requesting entity.

The allocation of static objects in the terrain to OMP's can impact message traffic. For instance, since the management of static objects is not computationally intensive, it is possible that each OMP manages a large number of objects. Therefore, the OMP's may span many MCG's and must receive messages from all of them, increasing the traffic on that segment of the network. This has implications on network configuration which we describe later. As noted, dense areas of static objects are a significant factor in determining the size of regularly shaped MCG's. With irregularly shaped MCG's, the size of each MCG can reflect the local density of static objects. Thus, OMP's are likely to span fewer MCG's on average with irregularly shaped MCG's compared with regular ones.

As yet, we have assumed that a single OMP manages all static objects of a grid cell. This situation is highly desirable as it minimizes the cost of the reliable transfer of the state of static objects. However, when the number of static objects in a grid cell is very large, it may be impractical for a single OMP to manage all of the objects. A popular solution to this problem is to use hierarchical grids.

Our solution is to have multiple OMP's for grid cells in which the density of static objects is significantly greater than the average density of the overall VE. Our multicast scheme guarantees that all OMP's will receive any messages relevant to their static objects, regardless of the number of OMP's managing the objects of a grid cell. This solution for the variable density of static objects incurs almost no additional cost for grid cells which do not require multiple OMP's. Determination of the assignment of static objects to OMP's does not require the participation of other simulation entities. For any grid cell, the process of assigning objects to OMP's can be restricted to those OMP's which will manage its static objects. This feature allows the number of OMP's for a grid cell to be determined dynamically, which simplifies the task of managing dynamically created static objects, such as a downed aircraft or incapacitated tank. Our solution to the problem of the widely varied density of static objects in large scale VE's effectively implements transparent hierarchical grids for the purpose of static object management.

5. Implementation issues

The challenge in implementing multicasting is to efficiently route a message sent to a particular MCG only to those nodes or network segments containing those nodes that are listening to that MCG. Efficiency in this case is two-fold: low-latency for message transmissions and quick response to changes in MCG membership (i.e., as nodes in the system change their working set of MCG's, the implementation should adapt to these changes fast enough so that no messages are lost). To our knowledge, no implementation exists currently that satisfies both of these goals. In most distributed computations, the requirement of quick dynamic response to MCG-membership changes may not be very stringent. However, it is particularly important in DIS since loss of messages can detract from the fidelity of the simulation.

5.1 Current technology

One implementation of multicasts is in the MBONE [Eri94] which operates over the Internet. Agents located around the network communicate with each other periodically, exchanging information about their locations and MCG sets. Thus, each agent has sufficient information locally to route multicasts to appropriate neighbors. Any change in the topology or MCG set must be communicated to all agents by message exchanges. The time cost of such an operation makes this scheme infeasible for DIS.

5.2 A simple implementation

We suggest a simple multi-tiered implementation that cannot be applied to a large-scale, general network such as the Internet but may certainly be considered for a DIS-specific network such as DSI. This multi-tiered implementation is consistent with the two-tiered implementation of the DIS Strawman document [Bea92]. We assume geographically distributed sites are interconnected

by some system of WAN's. This forms the top level of the network hierarchy. At each site, we may have some hierarchical organization of local networks. Each local network connects to its higher level network through a gateway. This gateway maintains a list of MCG's to which the nodes in the lower level networks are listening. It is assumed that changes in these lists can be propagated fairly quickly up the tree within a site. The scheme is very simple: all messages at the top level (WAN's) are broadcast to all WAN's (and therefore all sites). At each level in the hierarchy below the WAN's, the gateways perform filtering by transmitting incoming messages to only those networks at the lower level that have registered a need to listen to the MCG in which that message was sent. This system will obviously not reduce WAN traffic but has the potential to reduce LAN traffic and relevancy filtering significantly, provided the networks are configured with care.

5.3 Impact of network configuration

An entity determines that the state changes of other entities are relevant based on logical proximity (i.e. in the VE) and not physical proximity (i.e. in the network configuration). Logical proximity will change over the course of an exercise and thus entities cannot be physically located to reflect logical proximity throughout an exercise. Further, such placement may not be desirable for a variety of reasons. To improve upon broadcasting, the combined interests of simulation entities on a network segment must be less than the entire VE. This implies that for any multicast scheme to be effective, the physical design of the network must reflect the communication realities of the simulation entities.

One approach may be to configure the networks so as to minimize the number of MCG's of interest to the nodes in any segment. Simple as it sounds, this may be very hard to achieve for two reasons: first, it may not be possible to know a priori which MCG's are of interest to a particular node and second, even if this set is known, it will most likely change as the simulation proceeds. These problems are compounded by common events such as node crashes (requiring re-configuration of local networks while the exercise proceeds) and dynamic relocation of simulators to other machines in the site.

However, there are special cases where we may take advantage of our knowledge of the entities. We define "broad-receivers" as simulation entities, such as long-range sensors, whose AOI includes all, or nearly all, of the VE, thus spanning many MCG's. For these entities, relevancy filtering offers little or no advantage. Further, the presence of even one of these broad receivers on a network segment essentially reduces it to a broadcast segment. If one is present on each segment, then a broadcast network results, regardless of the multicast solution. In this specific case, the solution is simple: locate the broad-receivers of a site on a separate segment. Other similar examples are OMP's that span a large number of MCG's and fast movers such as aircraft. The AOI of a fast mover can be artificially inflated to reduce the rate at which it changes its working set of MCG's, making it similar to a broad-receiver.

6. Summary

Multicasting is a way of reducing the demands on network bandwidth and computational power in DIS. The multicast problem consists of two parts: structure and use of multicast groups and their implementation. In this paper we have focused primarily on a solution to the first part and discussed DIS-specific multicast implementation issues. The scheme we have outlined for the definition and use of space-based multicast groups is a unification of several previous proposals with some novel contributions of our own. Finally, we have pointed out that the solution to the problem of managing static terrain objects must be taken into account when considering multicast in DIS. We have integrated one such solution into our multicast scheme.

Although our scheme builds on previous work, the unified perspective we have provided appears to be a first. In a sense, this unification goes beyond the sum of its parts by capturing only the best of previous schemes and rejecting the rest. For example, our scheme is completely distributed. We have eliminated the need for centralizing agents such as Area of Interest Managers [MZP95],

EOMAN's [StW94] and group leaders [MZP95]. This reduces not only the complexity of the system but also communication latencies (the scheme in [StW94] requires three messages). Further, there is no physical act of registering in and out of multicast groups (also checking in/out or join/ leave) on the part of the entities. Each entity simply informs its local network interface of the set of groups from which it is interested in receiving messages each time this set changes. The implementation must ensure that these changes are effected quickly. The rationale is that any implementation of multicasts must necessarily deal with changes in multicast group membership (and rightly so). Consequently, there seems to be no necessity to solve the same problem at the application level.

Acknowledgments

We are grateful to Prof. Paul Reynolds for motivating this paper, for proof-reading it and for discussions and suggestions. The unification concept arose from discussions with the participants of Prof. Reynolds' seminar on Distributed Simulation. We thank Adam Ferrari, Anand Natrajan, Anh Nguyen, Andrea Salas and Chenxi Wang for those discussions and their feedback. We also thank Bert Dempsey for advice on the current state of multicasting in current network technology.

References

[Bea92] Beaver, R., etal., "Strawman Distributed Interactive Simulation Architecture Description Document Volume I," Loral Systems Company, March 1992.

[DIS93] DIS Steering Committee, "The DIS Vision," Institute for Simulation and Training, October 1993.

[Eri94] Eriksson, H., "MBONE: The Multicast Backbone," CACM, Vol. 37, No. 8, August, 1994, pp. 54-60

[JoM92] Johnson M. and Myers, S., "Allocation of Multicast Message Addresses for Distributed Interactive Simulation," 6th Workshop on Standards for the Interoperability of Defense Simulations, Vol. II, pp. 109-114.

[MZP95] Macedonia, M.R., Zyda, M.J, Pratt, D.R. and Barham, P.T., "Exploiting Reality with Multicast Groups: A Network Architecture for Large Scale Virtual Environments," submitted to the 1995 VRAIS Conference.

[Mil92] Miller, J.T., "Strawman Static Multi-state Objects," 7th Workshop on Standards for the Interoperability of Defense Simulations, Vol. I, pp. A-183 - A-191.

[ShB92] Sherman, R. and Butler, B., "Segmenting the Battlefield," 7th Workshop on Standards for the Interoperability of Defense Simulations, Vol. I, pp. A-27 - A-35.

[StW94] Steinman, J.S. and Wieland, F., "Parallel Proximity Detection and the Distribution List Algorithm," ELECSIM 1994, R. Smith, Editor, smithr@mystech.com, pp. 1 - 11.