Dynamic Grid-Based Multicast Group Assignment In Data Distribution Management

Azzedine Boukerche, Amber J. Roy, Neville Thomas

Parallel Simulation and Distributed Systems
Research Laboratory (PARADISE)
Computer Science Department
University of North Texas
Denton, TX 76206
{boukerche, ajr}@cs.unt.edu

Abstract

With the need to do increasingly complex distributed simulations, a considerable amount of effort in the Distributed Interactive Simulation community, including HLA, is concentrating on the data distribution management (DDM) problem. In a large-scale distribution simulations, a large number of entities are simulated and each entity could be a source of data that might be relevant to only few other entities. Thus, DDM is necessary to provide efficient, scalable mechanisms for distributing state updates and interaction information in distributed simulations. In this section, we describe the data distribution management (also known as filtering) mechanisms used for real time training simulations. In this paper, we present a new approach to multicast group allocation in Data Distribution Management, which we refer to as Dynamic Grid-Based Allocation. Our scheme combines positive elements from the two previously studied allocation methods: Fixed Grid-Based (FGB), known for its low overhead and ease of implementation, and Sender-Based, which generally uses fewer multicast groups than the Fixed Grid-Based method. We have implemented our algorithm using the RTI-Kit framework. Our experimental results indicate that (1) our approach is a viable alternative for performing multicast group allocation in large scale distributed simulations, and (2) is scaleable and yields to a significant reduction of message overhead when compared to a FGB strategy.

1. Introduction

Data distribution management (DDM) is an interesting and fundamentally challenging problem for the parallel and distributed community in general, and parallel and distributed simulation community in particular. Sequential simulations can permit direct access between simulated entities since all entities live on one machine and progress consistently in time. On the other hand, in a distributed simulation environment, data access between simulated entities must be closely regulated. Indeed, objects can be at different times¹, and it is difficult to reference remote objects. The aim of DDM is to limit and control the volume of the data exchanged during the simulation, and reduce the processing requirements of simulation hosts by relying events and state information only to those

---

¹ even within one processor

applications which require them, while exploiting the features of the computer network architectures [1].

The fundamental observation leading to this strategy is that practically all of the real-world objects being simulated will be interested in only a fraction of the objects contained in the entire simulation. For example, as illustrated in Fig. 1, a plane is primarily interested only in those ground vehicles, such as tanks, that are close to it. It is probably not concerned with the position of any other tanks, some of which might be thousands of miles away. This real-world phenomenon of limited interest would not be taken into account if a DDM is not omni-present. The host simulating the plane cannot elect to receive only information about certain objects, i.e., the ones that are within the plane's radar range, and gets instead an update message anytime, any tank or any other vehicle or aircraft changes its position. Consequently, this information is always transmitted to all hosts, just in case one should need it.

Figure 1 : Domain Region of Interest

The costs of this irrelevant data transmission can be very high. If a host receives a large amount of transmissions containing data that is irrelevant, the host will waste a significant amount of time and processing resources, receiving and perhaps reading the data, for instance, before discovering it should be discarded. Even if this unneeded data is somehow dismissed by the host in a manner that does not require any computations, it still has contributed to increase the traffic message on the network. Sending data to a host that is not interested in it, puts an unnecessary burden on the sending host (and especially the receiving host), as well as on the network which may be inundated by transmission that are in effect useless, since they will just be ignored when reaching their destination.

In this paper, we describe a new scheme for Data Distribution Management, which we refer to as a Dynamic Grid-Based Allocation. Our scheme is a hybrid approach that uses both the Fixed Grid-Based and Sender-Based strategies, and thereby it has the advantages of low overhead and ease of implementation for which those two strategies are known, respectively.

The remainder of the paper is organized along the following lines. In Section 2, we briefly review previous work and related work. Section 3 describes our dynamic grid-based algorithm. In Section 4, we report the simulation experiments that have been carried out to evaluate our scheme, followed by a discussion of a performance comparison with previous and related algorithms. Conclusions and future work are given in Section 5.

2. Previous and Related Work

Since the mid 1990's, there has been many research done on finding strategies for implementing Data Distribution Management. A key concept in most of these schemes is the notion of using multicast communication to efficiently transfer data between a host producing data and the many hosts who are consumers of that data. In this paper, the term *multicast* should be understood as the concept of single-source, multiple-destination transmission, which is fundamental to all DDM implementations and it can be implemented by various protocols, such as IP multicast, TCP/IP sockets, etc. Similarly, systems that do not use DDM methods, but instead use a DIS can be seen as using "broadcast" paradigm, that means producers send data to all hosts, regardless of which hosts are consuming that particular data. Thus, the problem of implementing DDM can be seen as the problem of allocating multicast groups to each routing space and assigning hosts as members. Among the two most common approaches to multicast group allocation in DDM, that we consider in this paper, are (1) fixed grid-based and (2) sender-based. We also note the multi-level grid and clustering approaches are basic optimizations to the fixed grid-based and sender-based methods. We shall summarize the strengths and weaknesses of these strategies.

a) Fixed Grid-Based

Fixed Grid-Based allocation of multicast groups scheme was used in the initial implementation of the HLA²-Run Time Interface (RTI), the RTI-s prototype. The DDM algorithm used the following scheme: "It associates multicast groups with cells defined by a grid system overlaid on the terrain." [2]. This algorithm has also been used by the light-weight RTI developed at George Mason University [3] by a system that was built with the DIS protocol. [4, 5]

To illustrate how this scheme works, we will describe a simple scenario. In Figure 2, we show a grid overlaid on the terrain to define cells, and a two-dimensional grid system (although a grid can have greater than two dimensions). Figure 3 shows how the grid cells are then populated with subscribing and publishing regions (publishing regions are also known as update regions). The plane is subscribing to the terrain within its radar

---

² DoD- High Level Architecture

range, which has been mapped to thirteen cells that are distinguished in the figure by gray shading. Each tank is publishing to a terrain area mapped to one or two cells, which are

Figure 2: Grid Overlaid

Figure3: Grid Populated

illustrated in the figure by placing a tank icon in them. The four cells representing both the subscribing region of the plane and the publishing region of a tank have both gray shading and a tank icon.

After publication and subscription regions are mapped to cells that have each been associated with a multicast group, i.e., all units within a specific grid cell are assigned membership to that multicast group[2]. In Figure 4, a number in a cell indicates that the cell has been assigned to a multicast group. Thirty multicast groups are used in this example, although only four groups (in cells 14, 19, 20, and 26, labelled 1, 2, 3, 4) are used to transfer information between a publisher and subscriber.

Figure 4 : Fixed Grid-Based Grouping

In the Fixed Grid-Based (FGB) approach, cell size, and the one-to-one mapping of cells to multicast groups, is fixed at the initialization step of the system. The advantage of this

approach is that it minimizes the inter-federates communication needed to perform DDM because a federate does not need any information from other federates in order to set up multicast group assignments for its units. Using the fully distributed grid algorithm implemented for the RTI-s routing spaces meant that determining the destination address was fully deterministic [6]. In this context, *routing spaces*³ refers to grids. Therefore, at run-time a federate can assign its units to multicast groups simply by checking the cell-to-group mapping already established during the initialization step.

A disadvantage of this strategy is that it requires large number of multicast groups to cover a wide terrain. Experimental studies [2] of this algorithm have found that the optimal cells size is about 2 to 2.5 km. Thus, for a large battlefield, the cell size might easily require thousands of multicast groups. Associating multiple cells to the same multicast group would reduce the number of groups needed but it decrease the accuracy of DDM filtering, and it would require additional filtering at the source. This tradeoff might not be acceptable in terms of increased cost and complexity.

Subscription Agents, which are architectural constructs designed to keep track of the multicast memberships on behalf of the simulators (i.e., federates), have been used to ease the computational burden on the hosts [7]. Agents can help the hosts handle the large number of multicast groups involved in this method of DDM, but they can not decrease the number of multicast groups that are needed since that number is determined statically, based on the grid layout and the cell size.

b) Multi-Level Grids.

To combat the problem of using large numbers of multicast groups, the use of multi-level grids has been proposed. This approach is based on the observation that the limit on the number of multicast groups and their associated maintenance overhead is most critical at the wide area network level [8]. A hierarchy of grid-based filters is used, and the cell resolution increases from the wide area network to local area network layers so as to decrease the number of multicast groups needed at the wide area network level while still obtaining the benefits of finer filtering at the local area network level. Another multi-tiered filtering algorithm uses a grid for the first tier of filtering and the Sender-Based strategy for the second tier [9]. However, these approaches are significantly more complex than the fixed grid-based approach.

c) Sender-Based

Instead of performing a cell-to-group mapping at system initialization, the RTI 1.3 allocates multicast groups dynamically. Like its predecessor the RTI-s, the RTI 1.3 uses a grid to determine intersections of subscription and publication regions. However, multicast groups are allocated based on which federates need to send data that they are publishing, not on which cells are part of the intersection [10]. For each publisher, a subscriber set is derived that specifies which federates are subscribing to cells in which that publisher is publishing. In [11], the authors described how to implement a similar scheme using a list of *listeners*, maintained by each sender.

---

³ A two dimensional space

The approach used by the RTI 1.3 is similar to the distribution list algorithm, where each mover maintains a list of sensors that want information about it. In [12] a distributed grid-based approach is used, as in the RTI 1.3, to determine the intersection between the area in which a mover is operating and a sensor's coverage area. Movers can be likened to publishers, sensors to subscribers, and distribution lists to subscriber sets. In [13], the authors use a similar algorithm, where a distribution list is created for each cell of each routing space. Distribution lists are suitable to synchronize federates using DDM, since lists corresponding to various logical times can be kept in memory.

The assignment of multicast groups based on subscriber sets, or distribution lists, can be done by dynamically assigning a multicast group to each publisher created in the simulation, and then have each subscriber in the subscriber set joining that group. Subscribers to the group will change as the publisher and subscribers regions of interest change. A possible optimization designed to reduce the number of multicast groups used involves assigning multicast groups only to those publishers that have a non-empty subscriber set.

d) Clustering

Assigning multicast groups dynamically, based on the number of publishers that have non-empty subscriber sets, may have the disadvantage of requiring large number of multicast groups if there are many publishers in the simulation. Clustering is a technique that has been used to reduce the number of multicast groups needed by combining multiple publication regions into a cluster, which is then treated as a single unit and assigned to one multicast group. In the RTI 1.3, clustering is done by Subscription Agents [14]. Other clustering algorithms have also been studied [15].

Clustering publication regions whose cells are in close proximity in the routing space has the advantage of reducing the number of multicast groups that a subscriber may need to join. This is true, since the subscriber might join the one group assigned to the cluster instead of the many groups associated with multiple publishers. However, clustering publication regions for publishers that have disjoint subscriber sets would cause several subscribers to receive data in which they had no interest, which somewhat defeats the purpose of DDM. Nonetheless, the subscriber can of course filter out the unneeded data. The cost of this filtering must be compared to the cost of implementing a better clustering algorithm that would reduce the transmission of irrelevant data that must be filtered. One such algorithm can assign cells from the same publication region to different clusters and can "split" clusters that exceed a threshold size, as illustrated in [8].

e) Comparison

The fixed-grid based approach is the easiest to implement and requires the lowest overhead, although it has the shortcoming of inefficient use of multicast groups since they are pre-allocated to cells. The sender-based approach attempts to optimize multicast group usage by using information about the state of the publishing or subscribing federates when doing dynamic allocation of multicast groups. In a simulation with a sizable number of objects, the sender-based approach may also use a high number of multicast groups, however, necessitating clustering. Multi-level grids and Clustering can

decrease multicast group usage, but they also add to the cost and complexity of the DDM system.

3. Dynamic Grid-Based Algorithm

Our hybrid approach, which we refer to as Dynamic Grid-Based Multicast Group Allocation, combines (1) the simplicity and easy of implementation that is associated with the Grid-Based method and (2) the reduction of multicast groups used that is connected with the Sender-Based strategy. Thus, our scheme would be advantageous in a system where meeting these two requirements were high priorities.

As in the Grid-Based method, we define cells by overlaying the terrain with a grid. However, multicast groups are not statically assigned to all of the cells in the grid. As in the Sender-Based approach, we dynamically allocate multicast groups based on the current publication and subscription regions in the system. Our strategy is unique because only those cells in which there is at least one federate publishing and at least one federate subscribing are assigned to a multicast group. In other words, a multicast group is allocated to each cell that has been determined to be part of the intersection of a publication region and a subscription region.

Using the same example described in Figures 1-3, and our strategy, the major differences, is that instead of assigning thirty multicast groups—one per cell as in Figure 3, only four would be allocated, as shown in Figure 5. In large scale simulations, our strategy would result in a dramatic decrease in the number of multicast groups that would be allocated.

Figure 5 : Dynamic Grid-Based Grouping

a) Description of the Algorithm

The basic idea of our algorithm is shown in Figure 6. As we can see, when a federate publishes or subscribes to a region, the following sequence of steps is performed for each cell overlapped by that region. If both publisher and subscriber were previously in that

cell, then there is already a multicast group assigned to that cell, and therefore the federate simply joins that multicast group and performs the final step, which is explained later. Otherwise, we must check if the publication or subscription being made by the federate creates an intersection between publisher(s) and subscriber(s) in that cell. If there is no intersection then we shall proceed to the final step, otherwise we shall allocate a multicast group to that cell. That federate, as well as all other federates publishing or subscribing in that cell, must then join that group. The federate then performs the final step.

Figure 6: Basic steps of the Algorithm

The last step in the algorithm is to record the federate's publication or subscription to the cell, since even if that cell may not currently be assigned to a multicast group, it may be assigned in the future.

b) RTI-Kit

It is important to recall that our Dynamic Grid-Based Data Distribution Management algorithm has been designed in the context of the RTI-Kit, which has been developed at Georgia Tech. The RTI is basically a set of libraries designed to support the development of Run-Time Infrastructures (RTIs) for parallel and distributed simulation systems, especially federated simulation systems running on high performance computer platforms. It is envisioned that these libraries will be embedding (sic) into existing RTIs, such as adding new functionality and/or enhancing the performance of the simulation system while exploiting the features of the computer network architectures and its

capabilities. Alternatively, the libraries can be used in the development of new RTIs [16]. The major components of the RTI-Kit are as follows.

- **KIT** – Initializes the components of the RTI-Kit and gives them cycles to process messages.
- **FM** – communication layer software, uses an API based on the Illinois Fast Messages (FM)
- **COMMON** – utilities dealing with sockets and message handlers
- **SESSION** – provides session management
- **TIME** – provides time management services
- **MCAST** – provides group communication services

We have designed and implemented an additional component (GRID) which can be integrated into the RTI-Kit:

- **GRID** – provides dynamic grid-based group assignment

c) Algorithm Details

The GRID library has been used to implement our dynamic grid-based multicast group allocation , which basically performs the following tasks⁴.

(T1) Creating a distributed grid.

In our algorithm, all cells are evenly distributed among the nodes participating in the simulation. The node to which a cell is distributed is said to be the *Owner* of that cell. For simplicity, a card-dealer scheme is used in our scheme to map the cells into the nodes, i.e., $Owner = Number \bmod Nodes$; where *Owner* is the cell owner, *Number* is the cell number, *Nodes* is the total number of nodes participating, and *mod* is the modulo operator. The Cells are numbered 0, 1, . . . N where N is an integer that represents the product of the sizes of each dimension in the grid. Note that our design supports the use of multiple independent grids, which we intend to use to represent different routing spaces. Each grid cell numbering begins from zero. Thus both the cell number and the Grid ID must be provided to specify a cell. Each node keeps track of the number of grids being used, and the number of dimensions in each, the dimension's sizes. Note that All information that keep track of which nodes are publishing or subscribing in a particular cell is only stored on the node which owns that cell.

(T2) Keeping track of federates that are publishing and subscribing to each cell.

For each cell owned by a node, there is a bit-array that represents which nodes are publishing to that cell. If node i is publishing to that cell, the i^{th} bit in the array is set.

---

⁴ In the sequel, we will use node and processor interchangeably.

Unpublishing clears the bit. Similarly, for each cell owned by a node there is also a bit array that represents which nodes are subscribing to that cell.

When a node publishes or subscribes to a cell that it owns, the appropriate bit can be set directly. When a node publishes or subscribes to a cell owned by another node, our GRID mechanism sends a message to the owner node requesting that the publishing or the subscribing node to be added to the appropriate bit-array. Messages are sent and received by calls to the FM library. Unsubscribing and unpublishing is done in an analogous manner.

(T3) Allocating groups to cells where publisher/subscriber intersection occurs.

When the owner of a cell adds a publisher or subscriber to that cell, our dynamic grid-based scheme is performed, as illustrated in Figure 6. Furthermore, a call to the MCAST library creates the group when “Allocate Mcast Group to Cell” is performed (see Fig. 6).

(T4) Notifying federates when they need to join or leave the groups.

The call to the MCAST library performed in item (T3) returns a handle to the new group that was allocated. This handle is then sent, as part of a message via a call to the FM library, to each node that has a bit set in the publisher or subscriber array for the cell for which the new group has been allocated.

(T5) Trigger federates to join or leave the groups.

When a node receives the message sent in (T4), which instructs it to join or leave a particular multicast group, the node uses the handle as an argument to the MCAST library call that subscribes that node to the group. Thus, as illustrated in Figure 6, the other Federates Publishing/ Subscribing in Cell Join Group, and the federation joins the group for Cell steps are performed. The MCAST calls these subscribers to the group so that they will send a message to the node that is the owner of the multicast group in order to update the group membership list managed by the MCAST library because they have joined the group. In our implementation, the node that owns the multicast group is the node that is the owner of the cell associated with that group.

d) Cell Owner Pseudocode

The basic pseudocode algorithm is presented in Figure 6. Note that if there is no Publisher/Subscriber intersection⁵ the function returns immediately, without sending any messages or creating any multicast groups. The function in Figure 7 carries out (T3) and

---

⁵ also known as a match,

(T4) as described above, i.e., allocating multicast groups to cells where publisher/subscriber intersection occurs, and trigger federates to join or leave groups.

```

/*****
/* Add a publisher on node=NodeID to      */
/* cell=CellID, done by cell owner at    */
/* request of publisher                   */
*****/

OwnerAddPub (GridID, CellID, NodeID)
{
    LocalID = GetLocalIndex (CellID);
    prev_match = PublishersHere(LocalID) && SubscribersHere(LocalID);
    /* after checking for previous intersection, add this node as
       publisher in cell */

    AddPub(GridID, LocalID, NodeID);
    curr_match = PublishersHere(LocalID) && SubscribersHere(LocalID);
    if (!curr_match)    return; /* no intersection */
    if (prev_match) {
        /* we have group already assigned,
           look it up and tell Pub to join */
        GroupsOwnedLookup (GridID, CellID, Handle);
        OwnerTellJoin(GridID, CellID, Handle, NodeID);
    } else {
        /* new intersection has been created */
        GroupName = MakeName(GridID,CellID);
        CreateMcastGroup(Group Name,&Handle);
        /* put new group's handle in table */
        GroupsOwnedInsert (GridID, CellID, Handle, GroupName);
        /* tell other pub/sub to join */
        for (i=0; i<NUM_NODES; i++) {
            if ( PublisherHere(LocalID, i) || SubscriberHere(LocalID, i))
                OwnerTellJoin(GridID, CellID, Handle, i);
        }
    }
}

```

Figure 7: Cell Owner Pseudocode

4. Simulation Experiments

In this section, we will present our experimental results. First let us, discuss the parameters, we choose to evaluate the overhead of our scheme. The additional overhead incurred by our Dynamic Grid-Based group allocation, when compared to the Fixed Grid-Based method, is due to the need to create multicast groups as the simulation progresses and triggers federates to joins those groups. This approach is different from the Fixed Grid-Based strategy, which creates groups during the initialization phase of the system, where the federates join directly, without the need to communicate with other federates. To assess and measure the overhead of our dynamic group allocation algorithm, we have selected the following two parameters.

(i) DDM Time – which represents the time needed to publisher or subscribe to a cell or group of cells (region), including the time needed to create any multicast groups, trigger federates to join, and update all tables that are needed in our scheme. It also includes the time consumed to trigger the federates to join the appropriate group.

(ii) DDM Messages – represents the total number of messages generated to perform the task of publishing or subscribing to a cell or group of cells (region). This number is the sum of the following two components: (1) the number of messages generated by the GRID library, and used to notify the cell owner that a node wishes to be added as a publisher or subscriber to a specific cell, and trigger federates to join a newly created group; and (2) the number of messages generated by the MCAST library, used to orchestrate the creation of a multicast group and update the membership list.

We have designed a prototype system to test our scheme using a cluster of eight 300 MHz Pentium II computers, connected by 100Mb Ethernet. In our experiments, we selected one node to be a subscriber, while the rest of the nodes as publishers, in a region of size S cells, thereby S multicast groups are needed in our Dynamic Grid-Based allocation scheme. We varied the size of the cell (S) from 4 to 256, and the number of node from 2 to 8, in all our experimental sets. The GRID system was tested to assess its overhead and characteristics. Then, its performance was compared to the Fixed-Grid scheme, which we also implemented and incorporated in our system.

Lets us now turn to our results.

4.1 GRID System Evaluation

Our goal in the evaluation of the GRID system was to determine the overhead, using the DDM time and DDM messages parameters described earlier, with respect to the number of groups allocated and the number of nodes. We also investigated the overhead of DDM messages generated by each of the GRID and MCAST. This might help us in the future to understand where the overhead is really coming from, and thereby enhance further our scheme. As illustrated in Figure 8, our results indicate that as the number of mulitcast groups allocated increases, the DDM Time increases. However the DDM time obtained is quite reasonable. Similarly, in Figure 9, we notice that the number of message increases

as we increase the number of multicast groups. Here again, the message overhead obtained is very encouraging. Figure 10 portrays the DDM message overhead obtained from each library, i.e., GRID vs. MCAST libraries. As we can see, the GRID library generates about 60% of the total DDM Messages, and the MCAST library produces the rest. We also observe that this ratio remains constant when we use eight nodes, and vary the number of multicasting groups. Similar results were obtained for 2, 4, and 6 nodes. This results is quite important, since it provide useful insights to understand better the behavior of our scheme, and thereby help us to understand how to enhance and reduce further the total DDM messages overhead (MCAST vs GRID messages. In the future, we plan to investigate ways to reduce the messages induced by the MCAST library, so that the GRID and MCAST libraries work jointly to reduce further the total DDM messages overhead.

DDM Time vs. Number of Multicast Groups

Figure 8

DDM Messages vs. Number of Multicast Groups

Figure 9

Contribution of MCAST and GRID Messages To DDM Message Volume

Figure 10

4.1.1 Comparison to Fixed Grid-Based Approach

In our next set of experiments, we wish to compare the overhead, both in terms of number of multicast groups used and volume of DDM Messages generated, of our scheme to the Fixed-Grid approach. Using a grid of 2000 cells, we measured the number of messages needed to create the multicast groups, and join them if necessary, as described by both approaches. Recall that the Fixed Grid-Based approach creates 2000 multicast groups at the initialization step of the system, whereas the Dynamic Grid-Based approach only creates multicast groups when there is a Publisher/Subscriber intersection in a cell.

Figure 11 indicates clearly that in all our tests the Dynamic method reduces significantly the number of multicast groups used. In our experiments, we also varied the number of cells that were part of a Publisher/Subscriber intersection from 2.5% to 12.5% of the total grid area. Our results (see Figures 12-15) show that for intersections of 5% or less, the Dynamic method had lower or slightly great DDM Message overhead than the Fixed allocation scheme. We also observed this phenomenon for intersection of 7.5% when the number of federates that were publishing or subscriber in each cell was less than six. Our results also indicate that using six or more federates per cell, and greater than 5% cell intersection, the overhead of triggering each federate to join the multicast group associated with that cell causes our Dynamic scheme to have a much higher DDM Message overhead when compared to the Fixed Grid based method, where federates join groups directly without needing to receive and acknowledge trigger messages.

Figure 11

Finally, the results obtained indicate that when there few publisher/subscriber intersection or few federate per cell, the Dynamic Grid-Based multicast allocation scheme has lower overhead than the Fixed Grid-Based approach in terms of DDM Messages generated. Furthermore, it has a lower and/or equivalent overhead with respect to the number of multicast groups created.

**Fixed vs. Dynamic Group Allocation:
Six Federates Per Group**

Figure 12

**Fixed vs. Dynamic Group Allocation:
Four Federates Per Group**

Figure 13

**Fixed vs. Dynamic Group Allocation:
Six Federates Per Group**

Figure 14

**Fixed vs. Dynamic Group Allocation:
Eight Federates Per Group**

Figure 15

5. Conclusions and Future Work

In this paper, we summarized the fundamental goal behind the use of DDM in large scale distributed simulations. We then observed that the problem of DDM can be reduced to the problem of allocating multicast groups that facilitate connectivity between publishers and subscribers. Using this analogy, we presented a hybrid scheme, which we refer to as Dynamic Grid-Based allocation algorithm that dynamically allocates a multicast group to each cell that is overlapped by a publication region and a subscription region. It combines the simplicity of Fixed Grid-Based strategies and the flexibility of the Sender-Based schemes. We also described its implementation using the RTI Kit developed at Georgia Tech. Our experimental results show that our algorithm is a feasible and a viable alternative for multicasting group allocation in an RTI, is scalable and outperforms the Fixed Grid-Based scheme in certain circumstances.

As part of the future work on this project, we plan to investigate further the capabilities of our scheme, and enhance further our hybrid approach. One could use the Grid Based strategy to designate different regions of the grid. For example, cells in which many federates are expected to be present, such as the airspace close to an airport in an aviation federation, might use the Fixed method of multicast group allocation. On the other hand, groups of cells that were not expected to contain a high number of federates at any given time, such as the airspace between airports, could be configured to use the Dynamic method for multicast group allocation. We are in the process of investigating these variants.

References

- [1] Department of Defense High Level Architecture Interface Specification, Version 1.3,DMSO, 1998, <http://hla.dmsomil>.
- [2] Rak, Steven J. "Evaluation of Grid Based Relevance Filtering for Multicast Group Assignment," 96-14-106, Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations, March 11-15, 1996.
- [3] Moreau, Marc. "Documentation for the RTI." George Mason University, 1997.
- [4] Macedonia,M., M.Zyda, D.Pratt, and P.Barham. 1995. "Exploiting Reality with Multicast Groups: a Network Architecture for Large Scale Virtual Environments." Virtual Reality Annual International Symposium 95).
- [5] Russo, K.L., L.C. Schuette, J.E. Smith, M.E. McGuire. "Effectiveness of Various New Bandwidth Reduction Techniques in ModSAF." Workshop on Parallel and Distributed Simulation, May 1999.
- [6] Rak S., M. Salisbury, and R. MacDonald. "HLA/RTI Data Distribution Managesment in the Synthetic Theater of War," 97F-SIW-119, 1997. Fall Simulation Interoperability Workshop.
- [7] Calvin, James O., Daniel J. Van Hook, "AGENTS: An Architectural Construct to Support Distributed Simulation," 94-11-142, Eleventh Workshop on Standards for the Interoperability of Distributed Simulations, Sept 26-30, 1994.

- [8] Berrached, Ali. "Alternative Approaches to Multicast Group Allocation in HLA Data Distribution Management." 98S-SIW-184. Spring Simulation Interoperability Workshop, March 1998.
- [9] Abrams, H., K. Watsen, and M. Zyda. 1998. "Three-Tiered Interest Management for Large-Scale Virtual Environments." ACM Symposium on Virtual Reality Software and Technology, 1998.
- [10] Van Hook, Daniel, and James O. Calvin. "Data Distribution Management in RTI 1.3." 98S-SIW-206, Spring Simulation Interoperability Workshop, March 1998.
- [11] Reynolds, Paul F. Jr., and Sudhir Srinivasan. "Communications, Data Distribution and Other Goodies in the HLA Performance Model." 97S-SIW-050. Spring Simulation Interoperability Workshop, 1997.
- [12] Steinman, Jeffrey S. "Parallel Proximity Detection and the Distribution List Algorithm." Workshop on Parallel and Distributed Simulation, 1994, pp 3-11.
- [13] Tacic, Ivan and Richard M. Fujimoto: "Synchronized Data Distribution Management in Distributed Simulations," 1998. Workshop on Parallel and Distributed Simulation, May 1998.
- [14] Van Hook, Daniel J., and Steven J. Rak, James O. Calvin, "Approaches to RTI Implementation of HLA Data Distribution Management Services," 96-14-084, Fifteenth Workshop on Standards for the Interoperability of Distributed Simulations, September 16-20, 1996.
- [15] Morse, Katherine L., Lubomir Bic, Michael Dillencourt, Kevin Tsai "Multicast Grouping for Dynamic Data Distribution Management," Proceedings of the 31st Society for Computer Simulation Conference (SCSC '99), 1999.
- [16] Fujimoto, Richard. RTI-Kit V 0.2 Specification. March 18, 1998.