Area of Interest Management by Grid-Based Discrete Aura Approximations for Distributed Virtual Environments

Juan E. Jaramillo, Lina Escobar, Helmuth Trefftz

[1] Universidad EAFIT - P.O. BOX 3300, Medellin, Colombia, South America
{jjaram42, lescoba5, htrefftz}@eafit.edu.co

Abstract. *We propose a communication's architecture for scaling distributed environments, aimed at minimizing the number of required multicast groups. The fundamental idea is to approximate the area-of-interest of entities, which ideally would be round, with hexagonal cells, and then associate multicast groups to the cells only when necessary – i.e. when the discrete auras of two or more entities intersect. We meet the goal of minimizing the number of required multicast groups by taking advantage of free processing capacity of the participating entities' hosts, and provide a novel load-balancing mechanism that prevents the hosts' processing capacity from being exceeded.*

Keywords: Distributed virtual environments, load-balancing, area of interest management, scalability, filtering schemes.

1. Introduction

Resource management for scalability and performance is one active area of research in distributed virtual environments. Network bandwidth and information processing are, in particular, two significant resources for a number of reasons. Network bandwidth provides the cornerstone of distributed virtual environments, because it is used to exchange information that entities need to maintain a consistent view of the shared state and engage in real-time interaction. Regarding information processing, it is required to maintain a local copy of the shared state, send and receive updates and generate graphics, among other tasks. (For an overview on resource management techniques see [Singhal and Zyda, 1993].)

Using broadcast communication for large-scale distributed environments can be problematic: as the number of participating entities increases, so does the amount of messages of no interest that entities receive. An entity's host will waste processing capacity when receiving non-relevant messages – even worst, the processing capacity might be exceeded if the amount of messages is large enough, degrading performance.

By using Interest Management, some of these systems now allow entities to receive only the subset of information which is relevant to them [Morse, 1996]. An entity expresses its data interests in terms of application-specific attributes – geographic location being a popular one – to interest managers (IMs), that accept entities' interest expressions (IEs) and use them to provide message filtering mechanisms. Since Interest Management requires more precise direction of message transmissions, unicast and multicast communications are used to support its implementation.

In multicast communication the application transmits each packet to a multicast group by supplying a special multicast address as its destination. The packet is delivered only to the hosts that have subscribed to the multicast group. To receive packets from a multicast group, a host must join the group. In a similar way, to stop receiving packets from the multicast group, a host must leave

the group. Multicasting is efficient, as hosts only need to send a packet once and copies are made along the group's distribution tree, thus saving bandwidth. It has been used in [Macedonia et al., 1995], [Mastaglio and Callahan, 1995], [Calvin et al., 1995], to name a few.

We propose a communication's architecture for scaling distributed environments, aimed at minimizing the number of required multicast groups. The fundamental idea is to approximate the area-of-interest of entities, which ideally would be round, with hexagonal cells, and then associate multicast groups to the cells only when necessary – i.e. when the discrete auras of two or more entities intersect. We meet the goal of minimizing the number of required multicast groups by taking advantage of free processing capacity of the participating entities' hosts, and provide a novel load-balancing mechanism that prevents the hosts' processing capacity from being exceeded.

This paper is organized as follows:

- Related work is reviewed.
- The basic architecture is described.
- Load-balancing concepts, rationale and logic are introduced.
- Some results obtained by simulation are presented and analyzed.
- Finally, we draw some conclusions and suggest future work directions.

2. Related Work

Benford has described a model for the spatial interaction of objects in a large-scale Virtual Environments [Benford et al., 1994]. The spatial model uses different levels of awareness between objects based on their relative distance and mediated through a negotiation mechanism built upon the concepts of aura, focus and nimbus. DIVE [Carlsson and Hagsand, 1993] and MASSIVE [Benford et al., 1994] use this approach.

Benford's approach does not scale well for large numbers of entities, because it requires significant processing resources. Also, as each packet is associated with a custom set of destination hosts, the model cannot take advantage of the packet distribution efficiency provided by multicasting [Singhal and Zyda, 1993].

A more scalable approximation of Benford's model is proposed in [Macedonia et al., 1995]. Naval Postgraduate School NET (NPSNET) is a DIS-compliant simulator testbed developed at NPS. NPSNET uses hexagonal grid cells – which more closely approximate a real-world round Area of Interest (AOI). An entity's AOI consists of a radius of grid cells where the entity is joining new cells at the leading edge and leaving old cells at the trailing edge as it moves forward. Grid cells are pre-assigned to multicast groups in a static fashion. These cells approximate an entity's ideal AOI using multicast addresses that are shared on a per-cell basis with other entities, hereby saving multicast addresses.

Macedonia states that the advantage of belonging to multiple cells mapped to multicast groups is that, as the entity moves through the virtual environment, it is always aware of every entity within its range. The entity in figure 1 must join and leave three multicast groups which are associated with cells at the periphery of its AOI where change is less critical – diminishing the effects of latency caused by joining and leaving new groups. As the entity moves in the virtual environment, it uniformly adds and deletes the same number of cells/multicast groups (3 in this case).

Each cell is mapped to a multicast group. In figure 1, an entity is associated with seven cells that represent its AOI (these cells have gray background). Hence, it is also a member of seven network multicast groups. The entity's host listens to all seven groups but sends messages only to the one associated with the cell in which it is located (with two exceptions that address the problem of late-comers).

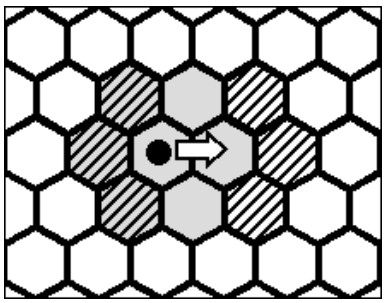

Figure 1: Area of interest for an entity mapped to a subset of multicast groups in Macedonia (the AOI has gray background). Fill patterns with gray background indicate the multicast groups that will be left, and fill patterns with clear background indicate the multicast groups that will be joined.

Macedonia presents a number of reasons to use hexagons. First, they are regular, have a uniform orientation, and have uniform adjacency. An entity's AOI is typically defined by a radius, as in cellular telephony communications. If squares were used, Macedonia points out that more area – and therefore, entities – than strictly necessary would be included in the AOI or, for smaller grids, more multicast groups would be required.

In this communication's architecture there is no need for a server, and only peer-to-peer communication can be used. Yet, a server can be introduced to provide initial state information for the first entity that enters a cell and store final state information when the last entity leaves a cell.

3. Interest Management by Discrete Aura Approximations

We propose a variation of the communication's architecture described by Macedonia. We introduce the use of a server-side Area of Interest Manager (AOIM) that serves as a central authority for dynamically associating multicast groups to cells, and performs load-balancing to make the most of the entities' processing and network resources.

3.1. Entity-to-Server Communication

The AOIM receives discrete position updates from the entities. Entities are aware of the hexagonal grid that partitions the virtual environment, and communicate reliably to the server their current discrete position – i.e. what cell are they currently standing on. If one entity moves inside of one cell, its discrete position doesn't change, and therefore no discrete position update is required. Also, since communication is reliable, discrete position updates are only necessary when an entity moves from one cell to another. If no update is received, the server-side AOIM will know that the entity hasn't changed its position.

3.2. Server-to-Entity Communication

The AOIM analyzes the discrete positions of entities. When it determines that two entities' discrete auras intersect, it will assign multicast groups to the cells the entities currently occupy (one multicast group per cell) and notify the entities' hosts of these assignments with address assignment updates. An address assignment update is a message that communicates the entity of address assignment changes in the cells that map its AOI. Note that this replaces the potentially expensive computation of collision detection and replaces it with non-expensive checks for cell adjacency. Also the assignation of the multicast addresses meets the need of providing communication between the entities. If an entity's AOI doesn't intersect with other entity's AOI, no multicast groups will be assigned because they are not necessary.

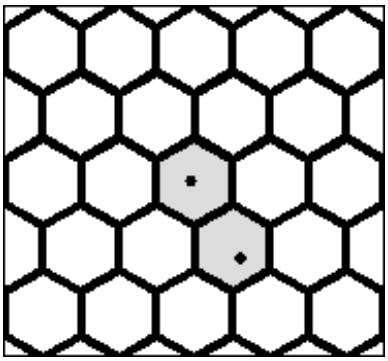

Figure 2: Area of interest for two entities mapped to a subset of multicast groups. The gray background means that the cells have been assigned a multicast group

3.3. Entity-to-Entity Communication

Entity-to-entity communication takes place by means of the multicast addresses assigned by the server. An entity will be told to connect to the multicast groups assigned to cells that comprise its discrete aura. In figure 3, the entity in the center will receive messages from the 2 groups of entities around it – whose auras intersect its aura. The groups of entities on the sides will only receive messages from the entity in the center and will not receive messages from one another. This happens because an entity joins the multicast groups that map its AOI but only sends messages to the multicast group that corresponds to the cell it stands on.

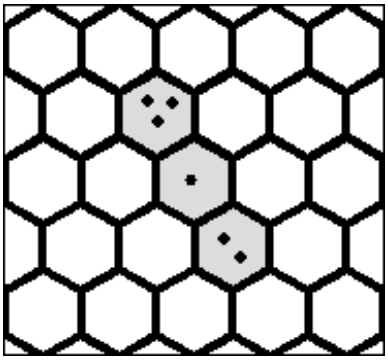

Figure 3: The entity in the center will receive messages from the 2 groups of entities around it – whose auras intersect its aura. The groups of entities on the sides will only receive messages from the entity in the center and will not receive messages from one another.

The architecture's allocate-on-demand policy has the benefit of minimizing the number of multicast groups, but comes at a price: joining and leaving rates for multicast groups are higher in most cases (see section on Simulation and Results.)

4. Introducing Load-Balancing

So far, we have presented the basics of our communication's architecture. Now we introduce load-balancing as a means to make the most of the entities' processing and network resources.

4.1. The Need for Load-Balancing

Up to this moment, we have been concerned with providing an entity with relevant data – i.e. messages coming from entities that lie on its AOI. By doing so, we ensure that its networking and processing

resources are being employed in its best interest. However, if these resources have free capacity, entities can afford to receive some non-relevant messages, if this can contribute to minimize the number of total active multicast groups that support the interaction among entities. This is done in the best interest of the global virtual environment. By using fewer multicast groups we are preserving a network resource and potentially enabling the interaction of a higher number of entities in the distributed virtual environment with the same number of multicast groups.

4.2. Sharing Multicast Addresses Among Cells

In Macedonia's architecture each cell is mapped to a multicast group statically. However, if two adjacent cells have a number of entities that are interested in one another, why don't these two cells use the same multicast address? Also: If one cell is occupied only by one entity and this cell is assigned a multicast address, couldn't this multicast address be shared?.

The first question suggests that two cells should be more likely to have the same multicast address if they are adjacent. The second question suggests that even if two cells are not adjacent, they can share its address if they contain a small number of entities.

A pattern of neighboring hexagons can provide alternative tessellations of a plane. By joining a number of neighboring hexagons we can create different tessellations that provide a less fine-grained partition. The maximum number of multicast addresses that will map an entity's AOI will diminish. For example, the maximum number of addresses will be 4 (instead of 7) when joining 3 cells. (See figure 4.)

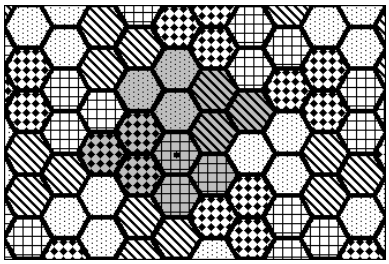

Figure 4: Example of joining 3 adjacent cells together. The maximum number of multicast addresses that will map an entity's AOI is 4 addresses when joining 3 neighboring cells. The cells with gray background show cells that would approximate the area of interest of the entity in the center

The potential benefits of joining cells are:

- Membership to multicast groups will last longer, therefore alleviating the router's join/leave group request processing. (There will be less joins/leaves per time unit).
- Fewer multicast addresses will be used as users will need to connect to less multicast addresses.

The potential drawbacks of joining cells are:

- The processing and networking resources of an entity's host might get exceeded in its capacity because of non-relevant messages.

The goal of load-balancing for this scenario is to maximize the benefits of joining cells (longer multicast group membership time, lower number of multicast addresses required) and minimizing its disadvantages (degradation of an entity's virtual environment performance because of non-relevant messages).

4.3. Load-Balancing Algorithm

The goal for the load-balancing algorithm is to compute an ideal partition of the world. By ideal partition we mean answering the following question for every group of cells that share one multicast address: Is it better to assign one separate multicast address to each cell or to give the whole group of cells the same multicast address?

We assume that each host allocates some processing capacity for the sole purpose of receiving and processing messages from other entities. When some part of this capacity is free, the entity's host can tolerate some non-relevant messages. When it is saturated, it cannot. The remaining processing capacity is used by the host to perform other tasks as graphics rendering, network packet generation, input handling, etc. We also assume, for simplicity in our discussion, that we have one entity per host.

We coin the term "noisy cell." A noisy cell is one that contributes to exceed the processing capacity of other entities' hosts with noise – messages that are not relevant. A noisy cell will always belong to a multicast group that is being shared between 2 or more cells. The "complaints" about the noisy cells will come from entities that have joined the multicast group because they are interested in the messages produced by one or more cells – different from the noisy cell – that belong to the same group as the noisy cell. The weight of a complaint is proportional to the number of entities in the noisy cell, as we expect noise to be higher if the noisy cell has a large number of entities.

The steps for the algorithm are:

1. Compute the noisiest cell – the cell that contributes the most to exceed the hosts' processing capacity – by adding up the weighted complaints from all the hosts.
2. Assign to each cell in the noisiest cell's group one separate multicast address.
3. Repeat steps 1 and 2, as long as noisy cells exist.

When computing the ideal partition, we assume that all the groups of cells in the world are not partitioned and then we decide on which ones should and should not be. This way we ensure that we detect the cells that produce the most noise – or at least a good approximation, as step 2 also assigns a separate address to the other cells (different from the noisiest one) in the noisiest cell's group.

The output of the algorithm is an ideal partition of the world that can be used to modify the current partition of the virtual world.

5. Simulation and Results

We developed a software simulation of Macedonia's architecture and of the proposed architecture and configured it to measure average active multicast groups count, and joining/leaving rates for multicast groups per minute. The simulation software generates random socialization sequences of computer-simulated entities and computes statistics for both architectures – at the same time, based on the same socialization sequence. A number of tests were run increasing the number of entities to check for scalability.

Each entity moves independently from all other entities, their initial position being randomly generated in the beginning. Given these conditions, entities tend to occupy with uniform density the world. Note that for linear increments in the number of entities, the percentage of occupied cells grows logarithmically – for 600 entities the average percentage of occupied cells was already 90% (see figure 8.)

We simulated a world – that from an upper view looks like a rectangle-shaped area – whose area is covered with a grid of 240 hexagons. Hexagons have a radius R, and entities move with a maximum

speed of 2.14R/hour. We set the number of entity updates per minute to 300 messages, and the host processing capacity per minute to be 50 times the number of entity updates per minute – meaning that an entity can receive information at a rate of 300 messages per entity per minute for 50 entities without exceeding the processing capacity allocated to receive and process messages from other entities.

For Macedonia's architecture, we know that a maximum of 240 multicast groups will be used – and that is the number of addresses actually needed as address pre-assignment is a requisite. In the statistics for his architecture we cite the number of active multicast groups instead of the number of multicast addresses, which is 240.

The load-balancing algorithm is executed every minute. Its output – the ideal partition of the world – is applied right away to the virtual world, meaning than if in minute m = x we determine that a group of cells should be assigned separate multicast addresses, we perform the assignation; and if in minute m = x + 1 we determine that the group of cells should be put together again, we do it. To provide some additional information about the effect of load-balancing on the architecture's behavior, we also simulated the proposed architecture without load-balancing algorithm. We'll call the architecture with load-balancing "balanced discrete aura," and without load-balancing "simple discrete aura" – simple because a hexagon is always mapped to a multicast group. The balanced discrete aura architecture uses a pattern of 3 neighboring hexagons, as appears in figure 4.

The results obtained from the simulation follow:

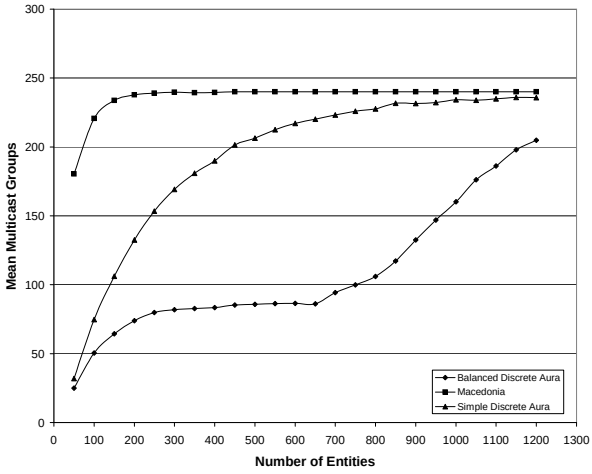

Figure 5: Mean Number of Multicast Groups in Macedonia vs. Mean Number of Multicast Groups in Discrete Aura

Note that for a number of entities of 650 and less, the data in the figures 5, 6 and 7 follows a trend. The mean number of multicast groups for the balanced architecture is minimally 3 times smaller for the balanced discrete aura architecture. Join and leave rates are also smaller, meaning that the results are overall positive: Not only we used a lower number of multicast groups, but also each host performed a lower number of joins and leaves on average. Up to this moment, load-balancing is not necessary and the interaction among users is supported by groups of cells that share the same multicast group together, verifying our initial guess that joining and leaving rates would be lower when cells were joined together.

From a number of 700 entities on, the mean number of multicast groups for the balanced architecture is no longer 3 times smaller in the balanced discrete aura architecture. This means that the load-balancing algorithm started detecting "noisy" cells and some groups of cells therefore no longer shared the same multicast group. Unfortunately, join and leave rates no longer smaller: they grow almost

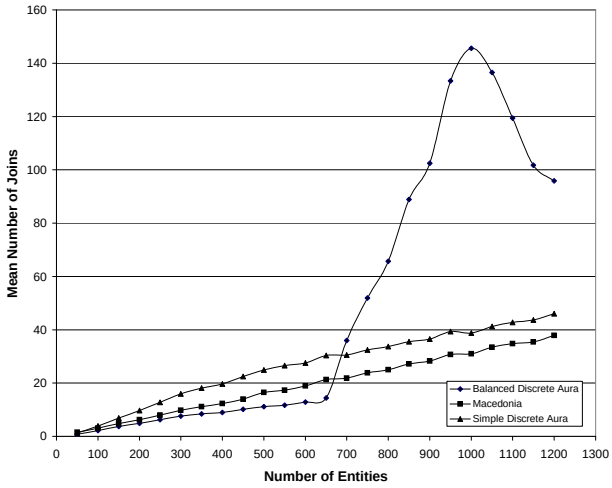

Figure 6: Mean Number of Total Joins Per Minute in Macedonia vs. Mean Number of Total Joins Per Minute in Discrete Aura

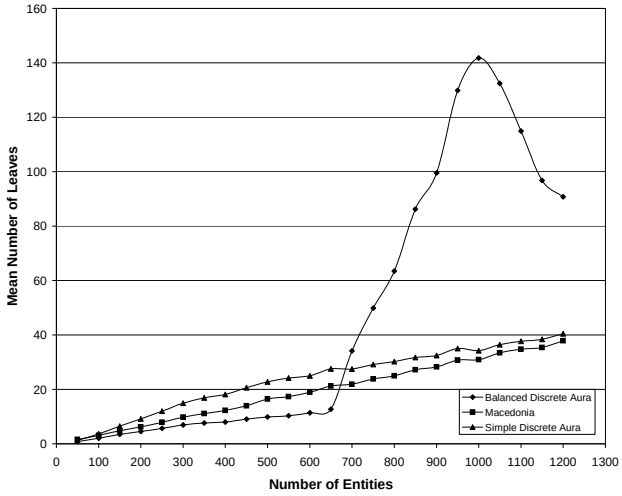

Figure 7: Mean Number of Total Leaves Per Minute in Macedonia vs. Mean Number of Total Leaves Per Minute in Discrete Aura

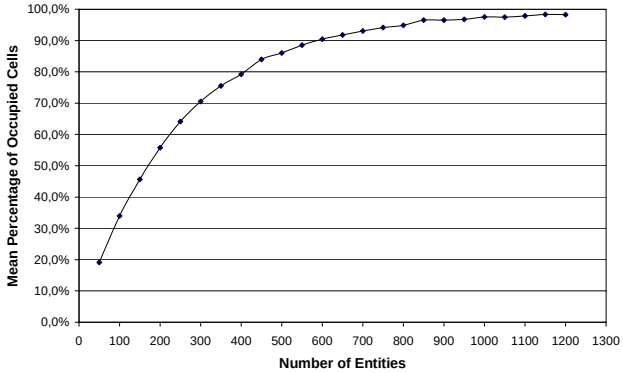

Figure 8: Mean Percentage of Occupied Cells

linearly to a maximum at 1000 entities.

This makes us wonder, what causes the join and leave rates for each architecture? For Macedonia's architecture, join and leave rates for an entity clearly depend on the entity's crossing of cell boundaries. If no boundary is crossed, the multicast groups will remain the same. For the simple discrete architecture, join and leave rates for an entity depend both on the entity's and its neighboring entities' crossing of cell boundaries, because as entities' discrete auras intersect, they are assigned multicast groups, and as they stop intersecting, they multicast groups are claimed by the server. In figures 6 and 7 we see that the join and leave rates and slightly higher for the simple discrete architecture if compared to the ones of Macedonia's architecture.

For the balanced discrete architecture, join and leave rates for an entity depend on three factors: the entity's and its neighboring entities' crossing of cell boundaries, and the load-balancing algorithm. This means that the huge difference between Macedonia's architecture and the balanced architecture join and leave rates is due to application of the ideal partition to the virtual world, since we were able to determine above that the second's factor significance – the neighboring entities' crossing of cell boundaries – is not so high. Why is this? As we decided to apply immediately the ideal partition to the virtual world, multicast groups can last as little as one minute, penalizing join and leave rates. The application of the ideal partition should cause an increment in the join and leave rates, but how high this increment is depends on how such an application is performed.

Finally, we can anticipate from figure 5 that the balanced architecture will converge to Macedonia's architecture at some point, when the number of entities is large enough. This happens because, as entities tend to distribute with uniform density, all groups of cells will have to be partitioned at some point – when every cell becomes a "noisy" cell. We argue that this is not so likely to happen in a real-world distributed virtual environment, as common sense suggests that users will not distribute with uniform density, as nothing forces them to do so. For specific military simulation data, Macedonia found out in [Macedonia, 1994] that the peak percent of cell occupancy for hexagonal cells' radius of 1km, 2km, 3km and 4km was 13%, 19%, 27% and 30% respectively. Also in military simulation, individual vehicles may move very fast, but they won't do it so because they fight as part of units in which movement must be coordinated [Macedonia et al., 1995], meaning that the different regions units occupy will contribute to a global non-uniform density. Finally, virtual world analogies to real world physical structures – or sets of structures – like dungeons, rooms [Curtis and Nichols, 1994] and cities [Ingram et al., 1996] are commonly used, and they stimulate non-regular entity density.

Yet, as uniform distribution is a possibility, cells could be appropriately sized to handle the maximum number of users distributed uniformly. Ideally such a case would result in no groups of cells being partitioned – we appreciated this behavior when number of entities is 650 – and not in convergence to Macedonia's architecture.

6. Conclusions and Future Work

The proposed architecture proved to be able to reduce the number of multicast groups used to support the interaction of entities, compared with Macedonia's architecture, taking advantage of free processing capacity allocated by the entities' hosts to receive and process messages from other entities.

Remarkable results were found when joining groups of cells to share a multicast address. Yet, these results will only maintain its applicability in the absence of load-balancing. The presence of load-balancing was found to be dependent on the number of entities. For the particular world size and entity's maximum speed simulated, load-balancing was absent in a range from 1 to 650 entities.

More modest results were found for load-balancing. Although the architecture managed to keep the number of active multicast groups lower than Macedonia's architecture, the price was high: join and leave rates grew. However, we think that the increment in the join and leave rates depends on how the ideal partitions – computed by the load-balancing algorithm – are used to modify the real-world partitions. With the current modification scheme, multicast groups can last as little as one minute, strongly penalizing the overall join and leave rates.

One worst-case scenario was detected: a world populated with uniform density will – only when the number of entities is high enough – make the architecture converge to Macedonia's architecture. We argue that this is not so likely to happen in a real-world distributed virtual environment, but we suggest an approach to deal with this: cells could be appropriately sized to handle the maximum number of users distributed uniformly without load-balancing.

Finally, some variables were given fixed values for the simulation. We set the number of entity updates per minute to 300 messages, and the host processing capacity per minute to be 50 times the number of entity updates per minute. Two interesting questions for future work are:

- What would happen if we let the processing capacity to be set by each host? The architecture might acquire the capability of dealing with heterogeneity to a certain degree. What would be the limitations and advantages of this approach?
- How can we take advantage dynamically modifying the number of entity updates per minute? For example, there is one case that we didn't take into account. Sometimes an entity's processing capacity might be exceeded solely by the messages received from cells that map its discrete aura. In this case partition is not the right solution. An approach to this problem is to have the entities negotiating the ideal number of entity updates per minute for one particular multicast group.

References

Benford, S., Bowers, J., Fahlen, L., and Greenhalgh, C. (Singapore, 1994). Managing mutual awareness in collaborative virtual environments. In *Virtual Reality Software and Technology*, pages 223–236.

Calvin, J., Cebula, D., Chiang, C., Rak, S., and Hook, D. V. (1995). Data subscription in support of multicast group allocation.

Carlsson, C. and Hagsand, O. (Seattle, Washington USA, 1993). Dive - a multi-user virtual reality system. In *Virtual Reality Annual International Symposium*, pages 394–400. IEEE.

Curtis, P. and Nichols, D. A. (1994). MUDs grow up: Social virtual reality in the real world. In *COMPCON*, pages 193–200.

Ingram, R., Benford, S., and Bowers, J. (1996). Building virtual cities: applying urban planning principles to the design of virtual environments.

Macedonia, M. (1994). Dissertation network software architecture for large-scale virtual environments. Technical report, NAVAL POSTGRADUATE SCHOOL, Monterey, California.

Macedonia, M., Zyda, M., Pratt, D., and Barham, P. (Research Triangle Park, North Carolina, 1995). Exploiting reality with multicast groups: a network architecture for large scale virtual environments. In *Virtual Reality Annual International Symposium*, pages 2–10. IEEE.

Mastaglio, T. W. and Callahan, R. (1995). A large-scale complex virtual environment for team training. *IEEE Computer Graphics and Applications*, 25(6).

Morse, K. L. (1996). Interest management in large-scale distributed simulations. Technical Report ICS-TR-96-27, University of California, Irvine.

Singhal, S. and Zyda, M. (1993). Networked virtual environments - design and implementation.