Three-Tiered Interest Management for Large-Scale Virtual Environments

Howard Abrams
abramsh@acm.org

Kent Watsen
watsen@acm.org

Michael Zyda
zyda@siggraph.org

Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118 USA

1. ABSTRACT

As virtual environments have grown in size, increasing attention is being brought to the issue of filtering data that is of no interest to a given client. This filtering is known as interest management. Typically, interest management is thought of as a one step process: data flows in from the network, and either is rejected or accepted.

This paper outlines a three-tiered approach to interest management, utilizing dynamic multicast group assignment based on a load-balanced octree design. Initial prototyping suggests that it is possible to create virtual environments with the number of entities interacting in the environment at least an order of magnitude larger than previously demonstrated.

2. INTRODUCTION

As virtual environments grow in size and grow in number of clients, it has become increasingly important to filter unneeded data before it arrives at a client for processing. This filtering process is known as interest management.

In the past there have been several approaches taken. In NPSNET [1], the world is broken into hexagons, each representing a multicast group. Each entity sends state information to a multicast group corresponding to the local hexagon it is in, while at the same time subscribing to many surrounding hexagons via their multicast groups. This approach, while not exact, works well when entities are distributed evenly within the virtual environment, but fails if all entities are clumped within the same cell.

In Spline [2], the world is broken up into 'Locales' which can be any size or shape. This allows the designer of the environment to partition the world so as to try and avoid this clumping problem. Although this helps, clumping can still occur because the designer can never know in advance how many entities will be in one place. Furthermore, spatially large virtual environments typically must be generated by an automated process, not by hand. If the environment is too large to be processed by hand, most likely the partitioning of the space into 'Locales' must also be automated. It could be very difficult to create an optimal automatic partition of the space, and therefore would reduce the original benefit of designing the 'Locale' to any size or shape.

Interest management has typically been a one-step process. Data would flow in from the network, and then an area-of-interest manager (AOIM) would roughly filter it, hopefully passing what would be mostly relevant data, on to the client. In [3], ten interest-management systems are outlined. Of the ten outlined, only Proximity Detection filters in more than one step. Proximity Detection uses a two-step approach, where the first step involves breaking the world into grids, similar to NPSNET or Spline. An entity uses point-to-point unicast to transmit its information to each entity within the current grid. In the second step, each entity itself performs a more accurate filtering based on what it actually wants, but still actually receives the data before throwing it away.

This paper outlines a three step, or tiered, approach to interest management.

3. DESIGN

The first tier works similarly to NPSNET, Proximity Detection, or Spline, where the world is broken up into manageable pieces, or regions. However, these approaches are extended by allowing the regions to change size dynamically, thereby eliminating the clumping problem. The information sent to these regions is at a low rate and low fidelity. The low fidelity information is only used for an approximation of where an entity is located, and where it is going. High fidelity information is then gathered for entities of interest in the upper two tiers.

The second tier uses the data from the first tier to create a protocol independent perfect match between a client's interests and the environment. This is similar to Proximity Detection, although this second pass is done in a broad and protocol independent manner.

The third tier, building on the first two, adds protocol dependence, allowing the client to receive only the data from the protocol it needs. At the same time, by separating out the protocol from the core interest management, we can allow multiple protocols to simultaneously exist within the same environment, while using the same underlying filtering mechanism.

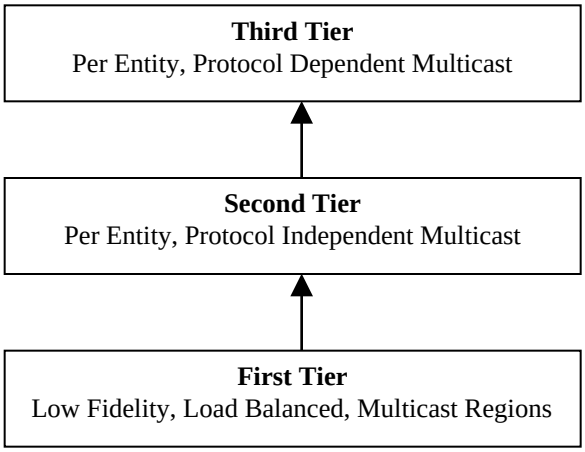

Figure 1 – Data flow in the three tier design

It is thought that, together, these tiers create the best possible match between what the client wants to receive and what it actually receives, while conserving network bandwidth and CPU cycles.

3.1 The First Tier

The reason data is often only roughly filtered by interest managers is that for a large number of clients, it is too costly to calculate an exact intersection of a client's interest expression (IE) and all of the data coming in over the network.

If the roughly filtered data was used as a first pass towards this intersection, instead of an approximation, it would be possible to compute exactly the data needed for a given client by using only this subset of the total data. Therefore the size of a simulation would only be limited by the number of entities a given client is interacting with, not the number of entities in the entire simulation.

One element present in most IEs is the distance from a client. Typically this area of interest (AOI) can be represented by a sphere with a radius equal to the maximum distance of interest. As described in the introduction, several approaches have used spatially based multicast groups to reduce network and CPU load at a hardware level. Although successful, these approaches are limited by the size of a group in three-space. If a region was too small, a client would have to subscribe to too many groups, and if the region was too large, a client would have to listen to other clients it did not care about.

Figure 2 shows an example of a case in which the regions are too large in size. It is easily seen that there are many clients 'clumped' in region 4, but few clients in region 1, 2, or 3. If a client, with an AOI as shown by the circle, were interested in only a small corner of region 4, it would be overwhelmed with data from clients it cares little about.

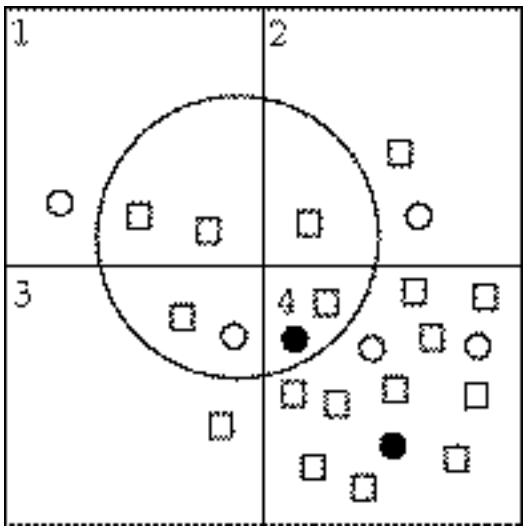

Figure 2 – Example of 'clumping'

For the military simulation STOW-E, it was concluded that a multicast group size of 2 to 2.5 km provided significant reduction in total host download, and that sizes less than 2 km provided only marginal additional benefit [4]. It was also concluded that "if the multicast grid could be dynamically re-sized and re-aligned locally, relative to the areas of highest activity, a significant reduction in total host download would be achieved." [4]

One simple solution to this problem is to use an octree to load balance these regions, and therefore the number of entities within a multicast group. If too many entities fall within one region, simply subdivide it into eight regions. If too many entities leave a group of regions, merge eight regions into one larger region. By load balancing the multicast groups in this manner, it is impossible to encounter the clumping example described above. Figure 3 shows the same distribution of clients as in figure 2, but with load balanced regions. Notice that a client with an AOI as shown by the circle, would only receive information about 12 entities, instead of 24 as in figure 2.

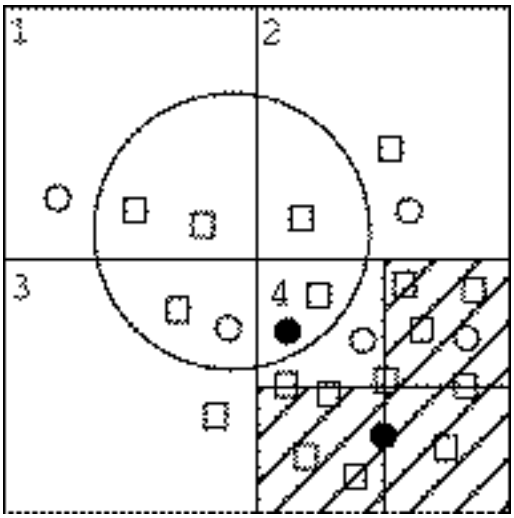

Figure 3 –'Clumping' with dynamic load balancing

The dynamic subdivision of the octree creates a new problem. If there is a very high density of entities, subdivision can occur to the point where entities are continuously switching regions and adding overhead when it is needed least. Take for example the case of a man standing next to an anthill with 10,000 ants. Because of the high density of ants, the octree will subdivide until it meets some criteria of entity density. This is exactly what the ants need, but exactly what the man does not. If the man were to move across the hill, he would go through tens, possibly hundreds, of regions with each step.

To solve this problem, we introduce the concept of a smallest region. An entity calculates the smallest region that it deems reasonable given its size and speed. When a region that it is in divides, an entity simply checks to see if it is below its minimum size requirement. If so, it stays within the current region instead of switching to one of the eight new leaf nodes.

The outcome is that entities are found throughout the octree, not only in the leaf nodes, and they are distributed not only by location, but also by size and speed. This has the added benefit for additional filtering based on size, and the ability to do efficient aggregation. Again take the example of the man and the ants. It may be that the man is running through a field, and happens to pass by the hill. If so, he would not be interested in things the size of ants, and as such, need not subscribe to the ants' regions. An aggregate version of the ants may be present in one of the larger regions, giving him the impression that the ants are there as he passes by. But if he stops to examine the hill, he could simply subscribe to the smaller regions temporarily, to see the ants in all their detail.

One limitation of using multicast groups is that the time to join a group may be on the order of a half second. This problem can be accounted for when deciding on an AOI. For example, if an entity can move within the virtual environment at a speed of 100 meters per second, the radius of its AOI can be extended by 50 meters to account for a 0.5-second lag in the time it takes to join a group. By adding to the AOI, it is possible that the client may receive information about more entities than is wanted, but remember, this is only a first pass, and the second tier will decide if indeed an entity's higher fidelity information is needed.

3.2 The Second Tier

Traditionally, the use of multicast groups was only used at a broad level because the number of addresses available limited the implementation. Under IPv4, the address space for multicast addresses is limited to just over 8 million addresses [5]. Even more limiting is the number of addresses a single interface can subscribe to and the number of multicast routes a router can keep track of. As multicast matures, and becomes more widely used throughout Internet, these hardware limitations will become less restricting. Already IPv6 is being used on the Internet. Under IPv6, the address space used for multicast address is currently over 32 bits, but has allocated space for over 112 bits to be used once routing hardware catch up.

If an entity were to have its own multicast address, clients could subscribe to each other on a per entity basis. A client could use the information gained from the first tier filtering to limit the list of possible candidates to choose from, thereby limiting the amount of network and CPU resources needed for interest management.

The manner in which data is gathered in the first tier need not be protocol specific. In fact, it has already been demonstrated how multiple protocols could be dynamically inserted into a simulation at runtime [6]. An added benefit to having a network stream per entity is that they can be protocol specific, and can bypass the AOIM layer altogether.

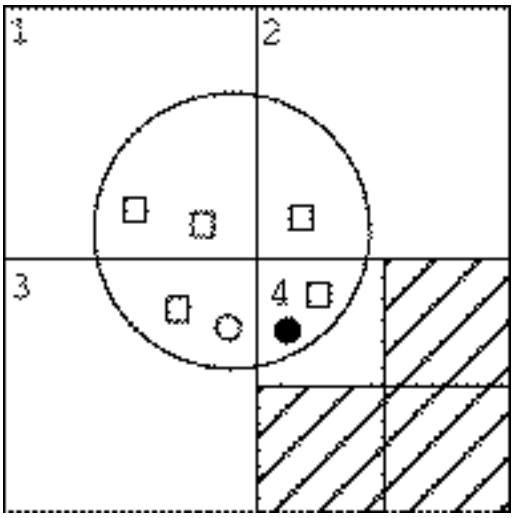

Figure 4 – Load balancing with AOI filter

Figure 4 shows the same AOI and region intersections from figure 3, but now the client is interested only in entities that are within the AOI.

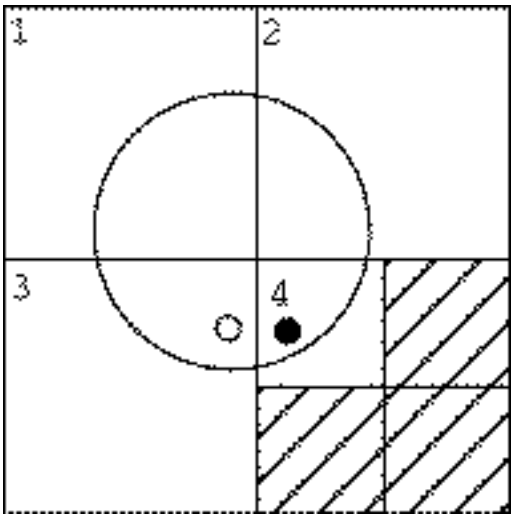

Figure 5 – AOI and protocol filter

Figure 5 shows the same AOI and region intersections from figure 3, but now the client is interested only in entities that are within the AOI and using the circle protocol.

3.3 The Third Tier

As stated above, the first two tiers can be implemented in a protocol independent manner, so that a simulation consisting of multiple protocols can exist with the interest management only computed once per client, not once per protocol per client.

This introduces a problem. If the AOIM is unaware of specific attributes of a given protocol, than these attributes cannot be contained in a client's IE. By adding interest management specific to a protocol into the protocol module itself, we create a third tier to allow an almost perfect filtering of data.

Continuing the example from figure 5, figure 6 again shows the same AOI and region intersections, but now the client is interested only in entities that are circles, and have the protocol specific property of being solid in color. This yields only one entity, rather than 24 as in figure 2.

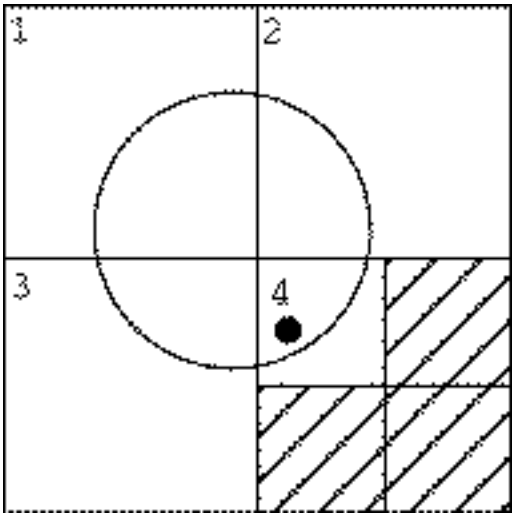

Figure 6 – AOI and protocol specific filter

The second tier simply hands up a socket to a client to the correct protocol specific layer, and then that layer decides if it should subscribe to that entity or not. In fact, a protocol could have multiple multicast addresses per entity, each transmitting a specific type of data, or at a different rate. It is in this manner that a client would receive exactly the data it needs.

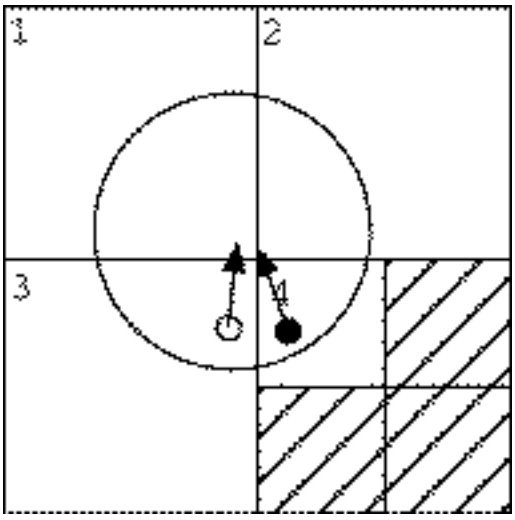

Figure 7 – AOI with varying fidelity

Figure 7, also continuing our example from figure 5, but this time the client is interested in any color circle, but at increasing fidelity as it approaches the center of the AOI.

A client could also choose not to subscribe to any individual entities, but only to the regions found in the first tier. This can be very useful for Plan View Displays, where information about many, perhaps hundreds of thousands of entities is needed, but only at a low rate, and at low fidelity.

4. RESULTS

A prototype of the first tier was developed on the SGI IRIX 6.5 platform using IPv4 multicast. This includes a lightweight server, a test client, and a graphical client based on the Bamboo [7] architecture.

The server is implemented in less than 700 lines of C++ code. It is a complete implementation, and includes TCP based communication for region check-in/check-out, and location to region lookup. Also included is region load balancing based on time delayed entity density, and reliable multicast based messaging for octree division and collapse.

The test client is based on a set of C++ base classes designed to make interest management transparent to the programmer of a virtual environment. The classes handle region check-in/check-out communication with the server. For each entity, they periodically transmit a state packet based on low fidelity dead reckoning parameters to the multicast address of the region the entity is within, and filter a list of interesting entities. The client itself is implemented in 15 lines of code, and moves thousands of entities in random circles within a 10000-meter cube.

The graphical client controls a floating observer which can move within the space. It is used for measurement and visual verification of the simulation as it progresses.

The scenario tested was the server running on an SGI Indy, three test clients each controlling 3333 entities running on an SGI Indy, O2, and Onyx. The graphical client ran on an SGI maximum impact 10000. The scenario ran for over three hours and completed without failure.

While moving through the environment with the graphical client, typically over 3000 entities were visible at a time within the 5000-meter AOI. Even though the client is single threaded, refresh rates never dropped below 30 Hz.

Although few experimental numbers were gathered, it can be said that the prototype system handled the 10000 dynamic entities with ease. It should also be noted that the server is used for load balancing only, and most communication is client to client via multicast.

5. CONCLUSION

By using a three tiered approach to interest management, this paper suggests that a client can subscribe to only the data it needs, with a minimum of overhead in network traffic and CPU time. Using this approach, a virtual environment's scale should only be limited by the number of other entities a client needs to know about at a specific moment, not the total number of entities in the environment. Early prototypes of this work are able to handle tens of thousands of entities, with no indication of an upper bound. Further large scale testing is needed, as well as an implementation of the second and third tier.

6. ACKNOWLEGMENTS

This paper outlines an architecture in which development was helped through technical conversations with Kent Watsen, Don McGregor, and Don Brutzman. Furthermore, this work would not have been possible without the support of our sponsors: Advanced Network and Services, the National Tele-Immersion Initiative, the Office for Naval Research, the Defense Modeling and Simulation Office, and the Defense Advanced Research Projects Agency.

7. REFERENCES

[1] Macedonia, M.R., et al. , Exploiting Reality with Multicast Group: A Network Architecture for Large-scale Virtual Environments. IEEE Computer Graphics & Applications, 1995 (September 1995), 38-45.

[2] Barrus, J.W., R.C. Waters, and D.B. Anderson, Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments. 1996, Mitsubishi Electric Information Technology Center America. http://www.merl.com/reports/TR95-16a

[3] Morse, K.L., L. Bic, and M. Dillencourt, Interest Management in Large-Scale Distributed Simulations. 1996. In Preparation.

[4] Rak, S.J. and D.J.Van Hook. Evaluation of Grid-Based Relevance Filtering for Multicast Group Assignment . in 14th Workshop on Standards for the Interoperability of Distributed Simulations. 1996.

[5] Stevens, W.R., Networking APIs: Sockets and XTI . 2nd ed. Unix Network Programming. Vol. 1. Prentice Hall, New Jersey 1998

[6] Watsen, K. and M. Zyda. Bamboo - Supporting Dynamic Protocols for Virtual Environments. in Image. 1998. Scottsdale, Arizona.

[7] Watsen, K. and M. Zyda. Bamboo - A Portable System for Dynamically Extensible, Real-Time, Networked, Virtual Environments. in IEEE Virtual Reality Annual International Symposium. 1998, 252-259. Atlanta, Georgia.