A Multigrid Relevance Filtering Technique for Distributed Interactive Simulation

by

Harry Tsai

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical [Computer] Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology
June 1997

Author_____
Department of Electrical Engineering and Computer Science
May 23, 1997

Certified by_____
David Brock
Thesis Supervisor

Accepted by_____
F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

A Multigrid Relevance Filtering Technique for Distributed Interactive Simulation

by

Harry Tsai

Submitted to the
Department of Electrical Engineering and Computer Science

May 23, 1997

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical [Computer] Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Distributed Interactive Simulation (DIS) is an emerging technology for large-scale networking of real-time virtual-reality simulators. Scalability of the network communications system is currently a major stumbling point in its development. This paper opens with an overview of DIS and the scalability problem then surveys the field of solutions, focusing on research in the area of relevance filtering.

In the remainder of the paper, we present a new technique for relevance filtering which significantly reduces data flow by emulating the attenuation of visual resolution with distance. This technique is implemented as an extension to existing grid-based methods and offers a more than 3-fold reduction in data rate for less than a 2-fold increase in multicast group demands.

Thesis Supervisor: David Brock
Title: Research Scientist, MIT Artificial Intelligence Laboratory

Acknowledgments

For their support and guidance, many thanks to Jim Calvin, Dan Van Hook, and Steve Rak of the Lincoln Laboratory Distributed Simulation Systems group.

Contents

List of Figures

Chapter 1

Introduction

Distributed Interactive Simulation (DIS) is a standard protocol which enables groups of computer-based simulators to interact in a common virtual world. At present, the technology is too underdeveloped for anything but military wargames, but potentially, DIS could allow people scattered across the world to share a wide range of experiences via a sophisticated synthetic environment.

However, much work needs to be done before that vision can become a reality. Existing versions of DIS suffer from insufficient communications capacity. "Worlds" are currently limited to the size of small counties, and their "populations" are restricted to a few thousand. Beyond these limits, the amount of data that passes between computers is simply too great for any present technology.

Of course, much effort is being expended to solve this problem. One major approach consists of finding ways to decide what data doesn't actually need to be exchanged between computers. This isn't as easy as it sounds; in the real world, it's a trivial matter not to see everything in the world at once. Computer simulations, however, are much simpler affairs than the real world. We have to explicitly decide what we do and do not want to see, and the more intelligently we consider the decision,

the less computing power we have left for more serious matters.

In the course of our work, we have discovered that a significant portion of the data in the typical DIS exercise is so finely detailed that it can be safely ignored in most cases. We have developed a data-filtering technique that takes advantage of the fact that small motions become less noticeable at greater distances. Our method is based on an previous work which organizes the world along the squares of a map grid. We improve on this method by using grids of differing granularities to organize data by its level of detail. Our *multigrid* filter achieves a 3½-fold reduction in data traffic while suffering less than a 2-fold increase in communications complexity.

For those unfamiliar with DIS, the first section of Chapter 2 provides a general introduction to the subject. In the second section, we discuss the problems limiting the growth of DIS and some solutions to those problems. This discussion culminates with a survey of current research in the field of *relevance filtering* (§2.2.3).

In Chapter 3, we go into the technical details of our multigrid algorithm. Briefly, it uses rethresholding techniques in grids of varying granularity to simulate the attenuation of visual perception at a distance. In the course of this work, we find that a significant fraction (~30%) of the data packets in a DIS exercise are of low magnitude and limited consequence. We also investigate the relationship between the size of one's *area of interest* and the granularity of a grid appropriate to that area.

Chapter 2

Distributed Interactive Simulation (DIS)

2.1 DIS Background

Since their inception, computers have been used to simulate real-world phenomena. The earliest computers were put to work calculating ballistics tables for artillery gunners. In a later era, computers became fast enough to respond to events in real time. This capability led to the development of *interactive simulation*, a technology now familiar in the form of flight simulators and similar devices. Today, the state of the art has advanced to the point where individual computers can exchange data in real time. This *distributed* technology enables people to use simulators in coordinated group exercises.

The main issue in transforming interactive simulation into distributed interactive simulation is devising a scheme for communication between the individual simulators. The difficulty lies in the fact that inter-computer communication channels are more restricted than the internal channels within each computer. Even though every

computer has a sophisticated, highly detailed model of the world, there is simply not enough capacity to communicate all of the details between separate computers.

The *Distributed Interactive Simulation* (DIS) system addresses the issue through a system of abstract descriptions, where information is expressed in purely symbolic terms. This concept can be understood in terms of an everyday analogy. In essence, DIS is very much like a running commentary on a football game. Picture a radio sportscaster, who observes the complex, fast-paced interaction between two teams, then must convey the entire scene with just a few choice words. Players are identified by name or number instead of by appearance; the positions and movements of each play are captured in brief phrases. By the time it reaches the listener, the game exists only as a streaming monologue. The graphic imagery is resupplied by the listener's imagination.

In a strikingly similar fashion, DIS condenses the progression of a war exercise into a relatively compact form. Of course, there are some obvious differences in going from the football field to the battlefield. The playing area is much larger (typically about 50km on a side instead of 100 yards), as are the players (tanks and other vehicles instead of small, solitary humans.) But in spite of the differences, the fundamental concept remains the same.

In DIS, the players are *entities*, where each entity is single, independent vehicle such as a tank, a jeep, an airplane, or, occasionally, a human on foot. Each entity is identified by a short code indicating model, serial number, and other distinguishing characteristics. The movements of the entities translate into vectors denoting position, velocity, and other basic physical parameters. Other actions (such as radio transmission, weapons fire, repair, resupply, etc.) have their own particular codings, as well.

In technical terms, each type of simulation event is encoded as a specifically-formatted data packet, known as a *Protocol Data Unit* (PDU) [5]. The sequence of events in a simulation is encoded as a stream of PDUs which is delivered to all of

the computers participating in a particular distributed simulation. Upon receiving a PDU, each simulator is free to interpret the enclosed data in any manner it wishes. For example, an ordinary vehicle simulator would produce a realistically-rendered image, while a simulated command post might generate icons on a tactical display, instead.

There is one notable discrepancy between the football analogy and the actual workings of DIS. Instead of a single sportscaster summarizing all the action, each entity reports on itself. Furthermore, the entities play the roles of both commentator and listener; the monologue becomes a dialogue, which is what makes the simulation *interactive*. Still, the radio broadcast analogy can be made to work. All parties send their reports to a common channel and listen to the same channel. Anyone tuning in to that channel can take all the individual accounts and piece together the whole of the action.

2.2 DIS Scalability

The radio broadcast analogy maps well to the broadcast transmission mode available in practically all computer networking systems. (The other usual mode is point-to-point transmission, where data is routed from one particular computer to another. This mode is impractical for DIS, since it would require a transmitting entity to repeat its "story" to every single entity joining into the exercise.) In the broadcast mode, computers submit data to be delivered to every other computer on the network. It's a simple, easy scheme, and it worked well enough in the early days of DIS, when exercises were relatively small. There were only a few, tightly-coordinated entities maneuvering in close quarters, a situation comparable in complexity to the average football game.

As time went by, people became more ambitious about DIS, and exercises grew dramatically in scale. Highly realistic battle scenarios demanded thousands of entities

spread out over thousands of square kilometers. In addition, great quantities of auxiliary data were introduced to model various environmental effects (e.g., weather, minefields, terrain damage [4, pp. 44–45]). These demands quickly overloaded all available communications resources. Returning to our analogy, imagine our radio program, but now a thousand people are trying to speak at once; clearly this is an unworkable situation. To make DIS practical on a large scale, the flow of data needs to be seriously constrained [1].

2.2.1 Dead Reckoning

One way to cut back on the communications load is to eliminate PDUs which can be easily anticipated. In our analogy, note that a sportscaster doesn't have to comment on every player on the field at every instant during a game. Players who are just standing around on the field become uninteresting. We simply assume that they'll keep standing there. Similarly, if a player starts walking straight across the field, we don't need to be constantly reminded of the fact for every half-second thereafter. Only when a player makes a significant change in course do we need an explicit update. Then, and only then, the sportscaster makes a comment, and we update our mental picture.

In DIS, a method called *dead reckoning* performs the same function. (see Figure 2-1) Each simulator keeps a "mental picture" of all the entities in the exercise. This picture exists as a collection of *dead-reckoning models* (DRMs), each of which is a simple, linear model of an entity's physical state. A DRM records an entity's last known position and velocity and extrapolates the current position based on that information.[1] As long as an entity's actual position doesn't deviate too far from that of its dead-reckoning model, it doesn't need to send out any updates. However, when the discrepancy exceeds a predetermined *threshold*, the entity sends out a new PDU,

[1] More complex models incorporating acceleration, angular motion, and other parameters exist, but are not yet widely used [6, pp. 147–150].

Dead reckoning uses simple linear extrapolation to reduce PDU transmission rates. For the curve below, regular sampling produces many data points, (○). A *dead-reckoning model* (DRM) uses position and velocity to generate a local linear approximation to the curve (△). PDUs (●) are then needed only when the dead-reckoned position diverges from the true position by more than an established *threshold*. At that time, the DRM resynchronizes with the true model (resulting in the "sawtooth" edges) then continues normal operation.

In the example above, dead reckoning condensed a 100-point curve into an 8-point approximation. However, this is an unrealistic gain, as we have exaggerated the threshold for purposes of illustration. The example below, shows a more typical usage, where the dead-reckoned curve is virtually identical to the true curve but includes only 19 of 100 points.

Figure 2-1: Dead Reckoning

causing all the simulators to update their DRMs.

This relatively simple scheme proves remarkably effective in practice. Entities typically need to produce only a few (less than 10) updates per second rather than several dozen (full-motion video rate).

Of course, there's always the possibility that nothing will change for a significant period of time, as for example, when a timeout is called. In this case, sportscasters feel obliged to fill the time with background commentary to let the listeners know that they're still there and, also, for the benefit of new listeners who are just tuning in. This has its analog in the DIS "heartbeat." If an entity hasn't sent out any PDUs within a certain timeout period (on the order of several seconds), then it is required to send out an update anyway [12, p. 4]. This makes it possible to distinguish between an entity which have stopped changing course (a *quiescent* entity) and an entity which has malfunctioned or lost its network connection.

Thus far, dead reckoning is the only data management technique actually included in the DIS standard[6, pp. 147–150], but research is under way in several other areas, as discussed in the sections below.

2.2.2 Bandwidth Improvements

Network bandwidth[2] is the scarce resource in DIS economics. Efforts to increase the available bandwidth in DIS network systems have met with some limited degree of success. For instance, data compression can be used to eliminate redundant data in the PDU stream, and improved network hardware can increase the underlying channel capacity [11]. While effective to a point, these methods offer only incremental improvements to the overall situation. Furthermore, it has been determined that, at

[2]The term *bandwidth* is used loosely here as an all-encompassing term for overall communications capacity. In actuality, capacity encompasses many and varied components including throughput, latency, packet rate, bit rate, etc.

this time, the DIS bottleneck hinges on the issue of PDU delivery rate, rather than on overall bandwidth [12, p. 19]. So for the time being, efforts are being concentrated on techniques which focus on PDU-level traffic and which offer more than simple scaling gains.

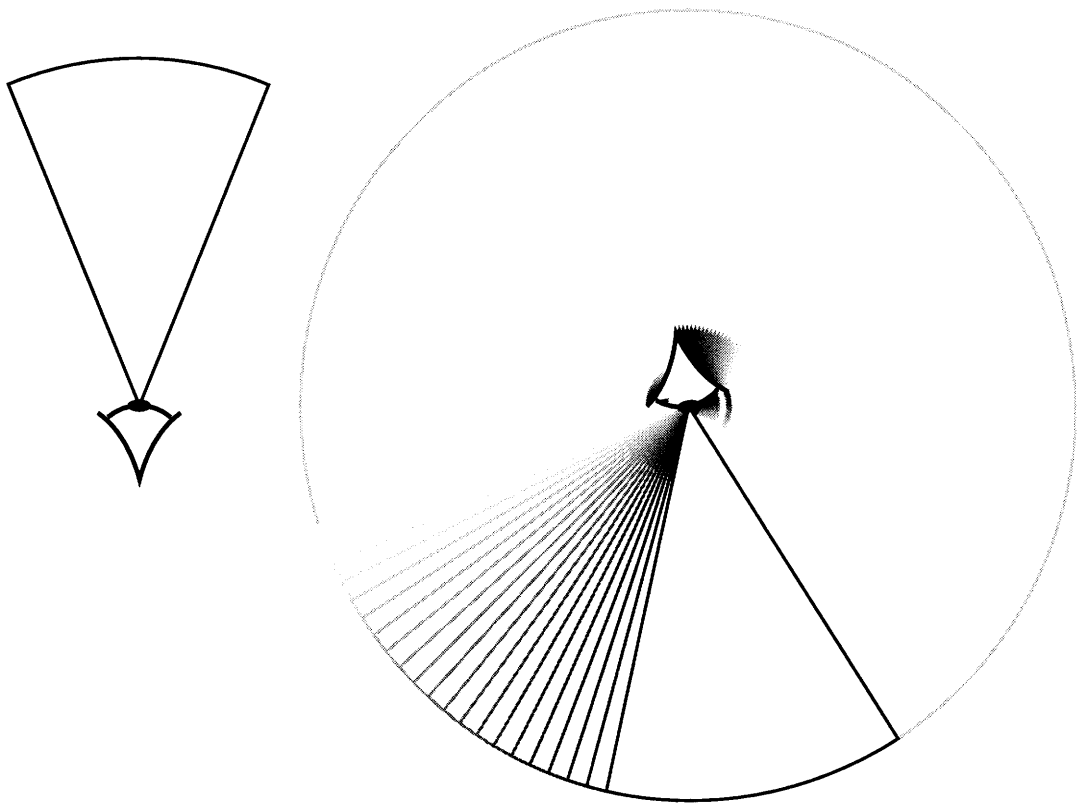
2.2.3 Relevance Filtering

Relevance filtering describes a general class of techniques which seek to filter out all PDUs which are irrelevant to the simulation "dialogue." The fundamental principle is that no individual simulator truly needs to know the entire state of the entire world at any time. For example, a single tank cannot observe a 50km battlefield from edge to edge. Even if it could make out other vehicles at that range, it would be fruitless to try to keep track of the individual movements of each and every one of the thousands of entities inhabiting the battlefield. An ideal filter would, therefore, adjust the operation of the network system so that the tank simulator would receive only the PDUs that were relevant to its current situation.

As a specific example, a tank driver can realistically detect vehicles at a range of about 4km [9, p. 2]. Furthermore, human vision is restricted to a forward-facing cone. Combining these two constraints gives us the filed of view illustrated on the left of Figure 2-2.

You may be concerned that our illustration only shows a two-dimensional field of view, while our simulations take place in a full three-dimensional space. The truth is that practically all current applications of DIS are limited in this manner. For reasons of practicality, DIS exercises tend to be limited to ground-based vehicles (and, on occasion, ground-support aircraft.) Airplanes are simply too fast and far-reaching to be accommodated by current technology.[3] Accordingly, most DIS simulator programs

[3]Note however, that the DIS protocol itself has no such limitations. This limitation is imposed only by the state of the art.



Relevance filtering defines an *area of interest* (AOI) for each simulation entity then limits the entity's reception to the PDUs that fall within its AOI.

Ideally, your current AOI is the exact region you can see at the present moment in time. For a human eye, this is a roughly conical region. Ideally, the cone extends infinitely, but it is typically truncated to some "maximum visual range," or *radius of interest* (ROI). We also use a flat, planar AOI since practically all DIS simulators are limited to ground-based action.

In practice, AOIs are circular rather than conical, since current technology can't handle the speed at which a sensor rotate. A practical AOI covers all the fields of view which could possibly be reached in the next several seconds; i.e., a 360° sweep extending to to the ROI, which is slightly increased to include any possible vehicle motion.

Figure 2-2: Relevance Filtering and the Area of Interest (AOI)

orient themselves to a ground-based coordinate system with a limited dimension of altitude. We follow current practice with our planar representations.

Practicality intrudes again when we define our actual *area of interest* (AOI). An AOI represents the region of the world for which a relevance filter should deliver PDUs to a particular entity. Ideally, the typical entity's AOI should simply be the field of view presented above, but in practice, computer network systems are too slow to keep up with such a rapidly-changing AOI. In particular, a tank driver takes only half a second to turn and look in a new direction, but a network routing system can take half a minute or more to reconfigure its data paths. To work around this problem, a practical AOI must cover all the fields of view that could possibly be reached during the network reconfiguration delay. The typical AOI is produced by sweeping the ideal field of view in a full circle about an entity and slightly increasing the ROI to account for vehicle motion, resulting in the circular AOI on the right of Figure 2-2.

Now we have an AOI, but how are we going to explain it to the network system? To sensibly implement relevance filtering, we need a network transmission mode which is subtler than broadcasting, yet more versatile than point-to-point transmission. This need can be filled by most network *multicast* schemes. Return once again to our radio analogy. Broadcast transmission corresponds to a single station being received by a universal audience, while point-to-point corresponds to everyone having a walkie-talkie tuned to a different channel. Multicast transmission corresponds to multiple stations operating on different channels, with listeners free to pick and choose amongst them. Each channel corresponds to a *multicast group* — transmitting on the channel corresponds to sending to the multicast group, and tuning in to the channel equates to *subscribing* to the group.[4]

[4]This analogy actually describes *one-to-many* multicasting, where a single multicast address maps to multiple individual receivers. (i.e., There is one commentator but many listeners for each channel.) This happens to be the only model supported by the now-standard TCP/IP protocol. Other technologies support the *many-to-one* (a group of commentators with an audience of one) and *many-to-many* (a group of commentators speaking to many listeners) modes of multicasting, which may be of interest in the future.

With this underlying infrastructure, it is now possible to construct a complete relevance filtering scheme. The essential problem is to find a method which can correlate locations of events with areas of interest and produce a corresponding system of multicast sendings and subscriptions. Many approaches are possible:

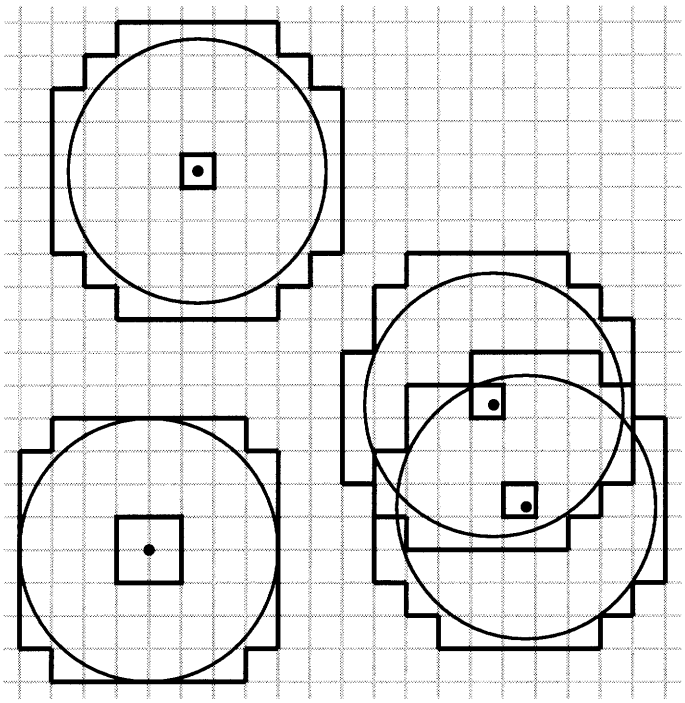
Organizational Relevance

In organizational schemes, PDUs are sorted based on the affiliation of the entity producing them. Each affiliation (team, platoon, army, etc.) gets its own multicast group to send to, and entities subscribe only to groups in which they are interested. For example, a tank making precision maneuvers might be interested only in keeping formation with its own platoon. That tank would subscribe exclusively to the platoon's group. Then, the network system would deliver only those PDUs and could dispense with any from other sources.

As another example, note that in all but the rarest cases tanks are completely oblivious to the actions of air combat fighters, and vice versa. If each type of vehicle has its own multicast group, then neither tank nor airplane needs to see the other's PDUs at all.

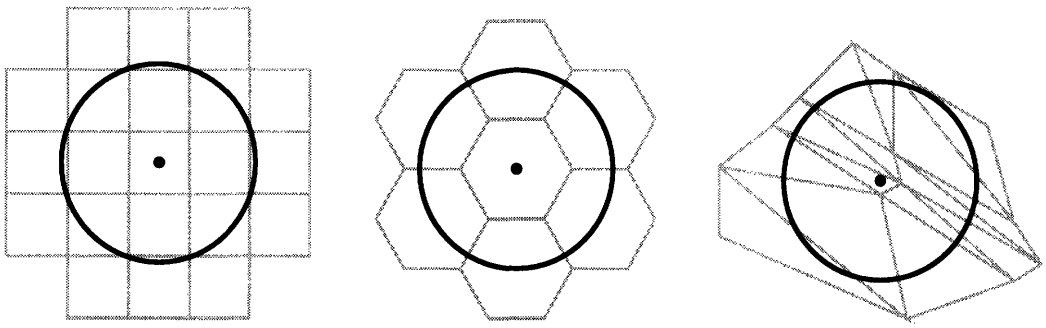
Geographical Relevance

In the geographical scheme, multicast groups are tied to physical regions of the world. As a simple example, picture a map with a grid overlay. Each square in the grid is assigned a different multicast group. When an entity enters a grid square it sets its "transmitter" to the channel corresponding to that square and sets its "receiver" to the same channel (and perhaps a few of the surrounding squares as well, depending on the entity's sensory range.) Thus, all the PDUs sent by the entity are limited to a single channel, or grid square, and are delivered only to other entities who are "in the neighborhood" and have specifically subscribed to that grid square's channel.



Ideally, an AOI describes a circle extending to the ROI. In practice, free-form circles are too complex to manage efficiently, so we approximate them with discrete grid cells.
In the diagram above, each entity (•) has a circular AOI which is approximated by an irregular square outline. Every grid cell within the outline corresponds to a particular *multicast group* to which the entity *subscribes* in order to receive updates on events within the corresponding cell.
The smaller, rectangular outlines surrounding each entity's location indicate the area in which an entity produces updates. Each entity transmits PDUs to the grid cells within this outline. When one entity lies within the AOI of another, their transmission and subscription groups overlap and they receive PDUs from one another.

Figure 2-3: Grid-Based Relevance Filtering



Square grids are popular for their simplicity, but grids may also take on many other forms, such as the hexagonal and triangular examples shown here.

Figure 2-4: Alternative Grids for Relevance Filtering

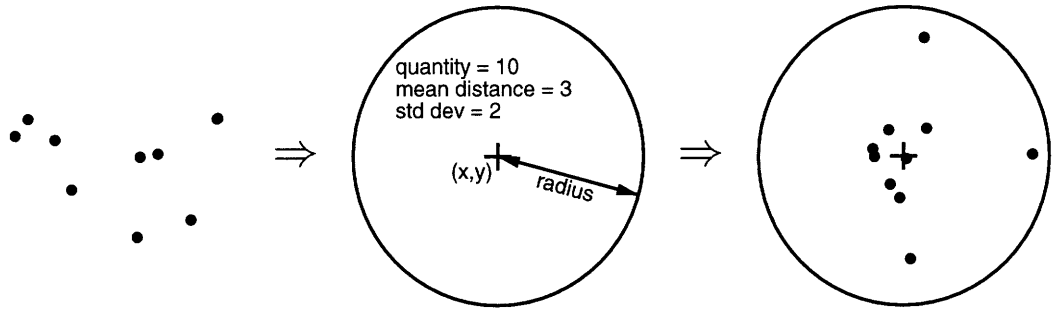
Of course, square grids are only one option. Hexagonal grids have been used in some systems, and irregular grids are also a possibility. The only guideline is that the grid cells partition the data in a useful way.

Organizational/Geographical Relevance

We have seen that both organizational and geographical relevance schemes have their uses. How does one choose between the two? The Distributed Systems Group at Stanford says you don't have to choose; the two systems can be combined [10]. In their *projection* system, organizational and geographical divisions are mapped against each other to form new subgroupings. In other words, if you have "Team A" and "Team B", moving through "Area 1" and "Area 2," then the projection system would combine these parameters to form four multicast groups.

	Team A	Team B
Area 1	Group A1	Group B1
Area 2	Group A2	Group B2

If you want to use organizational relevance, you can subscribe to Groups A1 and A2 for Team A, or to B1 and B2 for Team B. If, on the other hand, you opt for geographical relevance, you can subscribe to Groups A1 and B1 for Area 1, or to A2 and B2 for Area 2. Furthermore, you can pick and choose groups to get more specific



Aggregation is a more complex method of data management, which actually reinterprets data instead of simply filtering it. The basic assumption of an aggregation system is that, we will often just be interested in group behavior rather than in the actions of individual entities.
In that case, we can summarize the individual behaviors, replacing them with a single pseudo-entity which contains a statistical model of the entities within the group. At the receiving end, the super-entity can be reexpanded into a reasonable approximation of the original group.

Figure 2-5: Aggregation

areas of interest. For example, if you only wanted to pay attention to Team A *and* you only wanted to monitor Area 1, then you would subscribe only to Group A1.

There is one glaring flaw in the projection scheme, namely its order of growth. The number of multicast groups required increases with $O(mn)$, where m is the number of teams and n is the number of areas. When m and n are doubled, the number of groups required quadruples; and as m and n grow even larger, the situation becomes progressively worse. The Stanford system actually has another component keeps this expansion under control. The second component is known as *aggregation* and is described in the next section.

Aggregation

Aggregation is a more sophisticated method of filtering than the ones we have covered thus far. Instead of simply picking out existing PDUs, the aggregation system actually modifies them. Specifically, the system picks out certain groups of entities and replaces their collective PDU streams with a single new PDU stream which summarizes the original data. Of course, some detail is lost when the many original PDUs are condensed into one, but in many cases, the loss is unimportant. If properly executed, aggregation should produce a functionally equivalent, though not identical,

record of events.

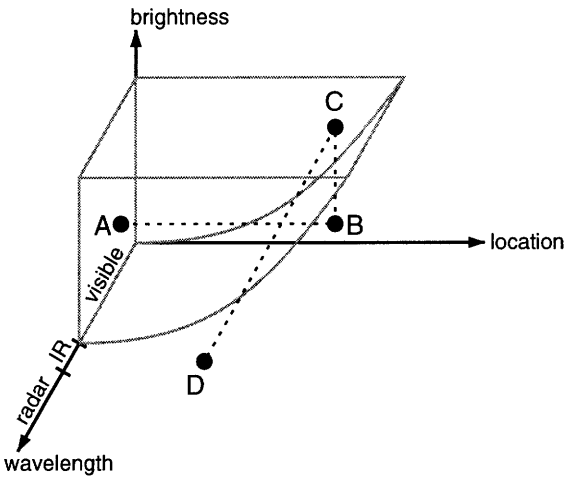
Aggregation works by replacing a group of entities with single *aggregated entity*. Obviously, this doesn't work in all cases, only those in which the behavior of the whole group is more important than the behavior of the individual members. For example, a platoon of tanks may be traveling in formation, using no personal initiative, acting only as a coherent whole. Alternatively, a group of vehicles may be so far away that the exact positions of the individuals are inconsequential. In such cases, the entities in the group may be *aggregated*.

When a group of entities is aggregated, the individual entities disappear (as far as the receiving simulators are concerned) and an aggregate entity appears in their stead. This aggregate has all the properties of a regular entity but has, in addition, a number of parameters which describe the original entities from which it was composed. For example, the aggregate entity of Figure 2-5 records the number of original entities, their centroid position, their mean distance from that position, and their standard deviation that distance. These parameters may subsequently be used to construct a reasonable facsimile of the original entities.

Aggregation is a promising, but challenging, area of research. Existing systems which utilize aggregation include the *projection aggregation* project[10] at Stanford and the observer-based multiresolution architecture[7] at MIT.

Miscellaneous Relevance Parameters

Relevance filtering can be based on almost any criteria imaginable. For example, all sensors have lower limits on their sensitivity — in order to be detectable, objects must fall above a certain minimum brightness, cross-section, etc. Sensors respond to different types of energy, such as ultraviolet (UV), visible, and infra-red (IR) light, radar, and sonar. Some sensors, like Doppler radars, are sensitive to the relative speed of a target. In short, the possibilities are endless.



Routing space allows arbitrarily-formed AOIs to be defined within an abstract space whose dimensions are formed from arbitrary bits of PDU data. This diagram illustrates an AOI which models a typical optical sensor operating in the visible light range. The model follows the inverse-square law by raising the bottom of the AOI as distance from the observer increases. Points A, B, C, and D illustrate the limits of the AOI. See text for details.

Figure 2-6: Routing Space

Future DIS systems will have to be far more versatile in their handling of relevance criteria. To this end, the design specifications for the next generation of DIS technology includes an interface for specifying arbitrary filter criteria within a generalized *routing space*.

Routing space is an abstract construct, a "space" whose dimensions consist of arbitrary bits of DIS data. Anything that can be extracted or calculated from the contents of a PDU is fair game. In the example of Figure 2-6, we have chosen [electromagnetic] wavelength, brightness, and location.

Once we have defined this space, entities are free to subscribe to AOIs of arbitrary shape within the space. In our example, we have a chosen a non-trivial form which represents a typical optical sensor. On the wavelength axis, the AOI spans the spectrum of visible light. Along the location-brightness plane, the AOI's sensitivity to brightness tapers off with distance; i.e., the further an object is from the observer (located at the origin), the brighter it must be to remain visible.

Finally, entities may restrict the distribution of their PDUs to specific regions

within the routing space. In a manner analogous to AOI subscription, entities define arbitrary areas in which they wish to *publish* their PDUs. At present, these regions of publication have no formal nomenclature; but for symmetry, we shall refer to them as *areas of presence* (AOPs). In our example, the four points A, B, C, and D represent simple, point-shaped AOPs. Their interpretation will be discussed with the next point.

Once the network system[5] has been informed of both AOIs and AOPs, it correlates them to determine where data needs to be transported. Wherever an AOP and AOI overlap, data is transferred from the former to the latter. In our example, point A represents an nearby object of moderate brightness. It falls easily within the AOI. However, if the object moves out to point B, the distance is too great for it to remain visible. Only objects as bright as point C can be seen at that distance. Finally, consider an object at point D, which is as bright as point C, but can only be seen by radar. (Imagine an unlit airplane at night, for example.) This object, again, does not fall within the AOI.

The overall benefit of routing space is that entities have the ability to declare AOIs and AOPs to suit any need. Furthermore, once they have declared their AOIs and AOPs, they have no further worries over data delivery, and the network is fully informed when it gets around to making its routing decisions.

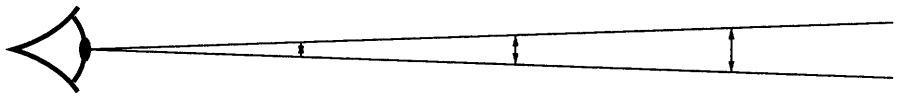
[5]To be specific, routing space is implemented within the new DIS component known as the High-Level Architecture Run-Time Interface (HLA RTI) [2]. The RTI serves as an interface layer separating the simulators from the actual network mechanisms.

Chapter 3

A Multigrid Algorithm for DIS Relevance Filtering

3.1 Multigrid Principles

The *multigrid* relevance-filtering algorithm developed in this paper is based on the grid-filtering work done at Lincoln Laboratory [9]. That system uses square grids to approximate circular AOIs about each entity. Our multigrid scheme further limits the AOI through a model of decreasing resolution with increasing distance. In other



In our model, we assume that visual acuity is limited only the angular resolution of the eye (or other sensor). Thus, as distance from the eye increases, the size of the smallest discernible detail increases proportionately. Values for these parameters may be found through simple trigonometry, as illustrated below and performed in the text.

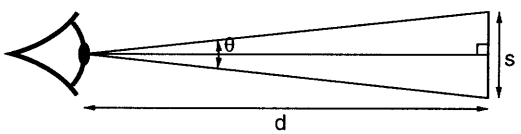


Figure 3-1: Visual Acuity

words, the more distant an object is from an observer, the less finely its details can be perceived.

3.1.1 Visual Acuity Model

We begin with a simple[1] model of visual acuity. The only assumption is that the eye (or any other sensor, for that matter) has a lower limit to its resolving power which can be expressed as a constant angular measure. If we follow that angle from the eyepoint out, we see that it describes a cone in space, as illustrated in Figure 3-1. By tracking the width of the cone, it is apparent that details must be larger at greater distances in order to be visible.

The relationship between these parameters can be found by simple trigonometry, as shown in the lower diagram in Figure 3-1. The cone forms a pair of congruent right triangles where the inner angle is half the arc resolution ($\theta/2$), the near side is the distance from the eye (d), and the far side is half the size of the smallest discernible detail ($s/2$). Using the tangent relationship (tangent = far side / near side), we obtain the following relationship:

$$\begin{aligned} \tan\frac{\theta}{2} &= \frac{s/2}{d} \\ \theta &= 2\tan^{-1}\frac{s/2}{d} \end{aligned} \tag{3.1}$$

θ = arc resolution
s = size of detail
d = distance to detail

Equation 3.1 is not a useful form, since the arc resolution θ of the human eye is actually a constant with the generally-accepted value of 1 arc minute ($\frac{1}{60}$ degree) [8, p. 50]. By solving for s, we obtain a more useful relationship.

[1]The model is "simple" because it takes only the geometric consideration into account. We disregard inverse-square light propagation, atmospheric attenuation, and all other optical effects.

$$
\begin{aligned}
s &= 2\,d\,\tan\frac{\theta}{2} \\
s &= 2\,d\,\tan\frac{0°1'}{2} \\
s &= 2.91 \times 10^{-4}\,d
\end{aligned}
\tag{3.2}
$$

s = size of detail
d = distance to detail

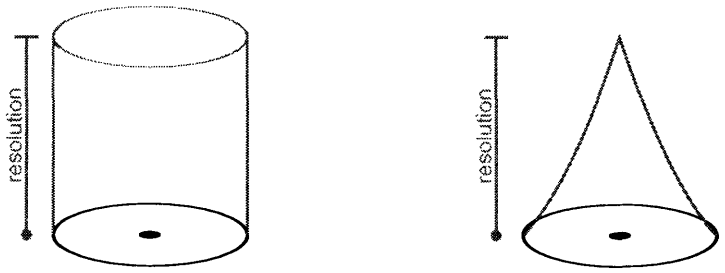
Equation 3.2 shows that the size of the smallest discernible detail is directly proportional to distance. For example, we can make out a 1-inch detail at about 290 feet. At twice the distance (580 feet), we can only make out details twice as large (2 inches), and at half the distance (145 feet), we can see details twice as fine (½ inch). At the typical ROI of 4km, the smallest discernible detail is 1.16m (~3.82ft)

3.1.2 Multi-resolution AOI

Now that the visual-acuity model is complete, we can begin work on our AOI. In Figure 3-1 and Equation 3.2, we saw that the "threshold" of detail size increases linearly with distance from the eyepoint. If we now define [linear] "resolution" to be the reciprocal of the threshold ($1/s$), then we can draw the picture in Figure 3-2.

On the left is the ideal AOI for an ordinary grid-based scheme. The base of the AOI extends from an entity's position out to the entity's ROI. Resolution is constant across the entire range, giving us a cylindrical form in geography-resolution space. The volume of this cylinder can be interpreted as a rough estimate of the amount of data that will be received by this AOI. Increasing the base area brings in more data by including more of the surrounding entities, and increasing the resolution brings in more data by giving the dead-reckoning mechanism less leeway to suppress PDUs.

Now consider the right side of the figure, which shows the form of our desired multi-resolution AOI. At the center of the base, resolution is at a maximum, but it



The traditional AOI maintains a constant resolution from the center all the way out to the ROI. We believe this technique to be wasteful and propose a multi-resolution AOI in which resolution tapers off as distance from the observer increases.

Figure 3-2: Multi-resolution AOI

tapers off as distance increases, eventually reaching zero at the ROI. It is apparent that this shape has a much smaller volume than the traditional cylinder. It was our hope that this reduction in abstract volume would be reflected in the volume of PDU traffic.

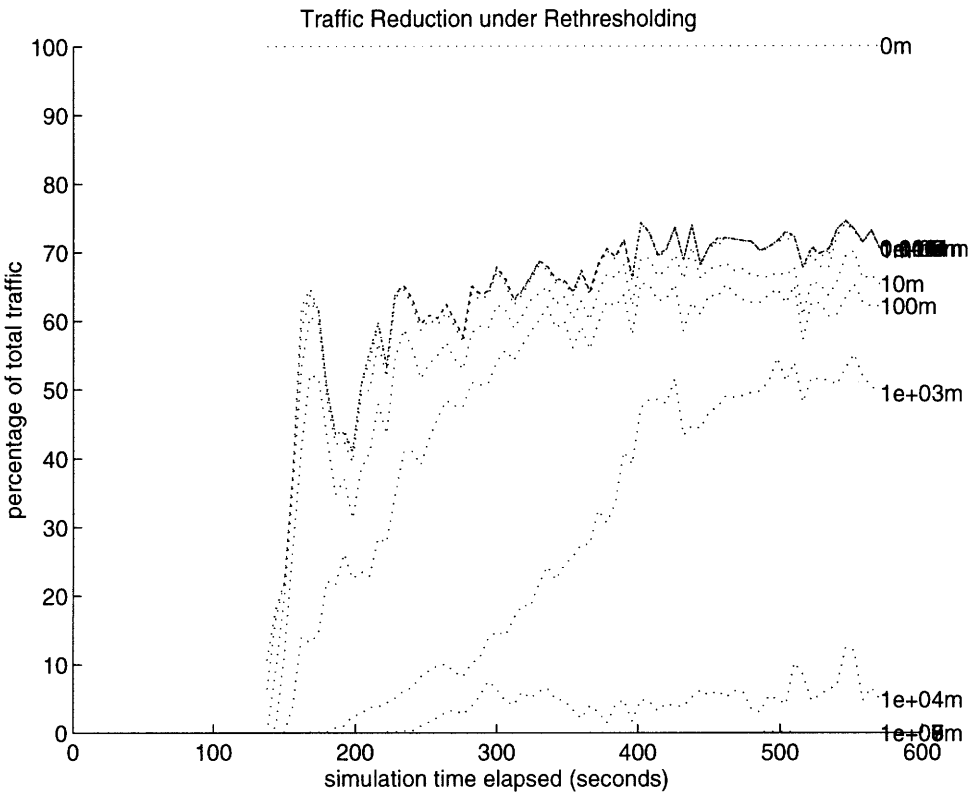
3.1.3 Rethresholding

To test our ideas, we obtained a PDU log file from the large-scale DIS engineering demonstration test known as ED-1A. This test involved a series of exercises operating at the limits of DIS technology — each exercise included on the order of 5,000 semi-automated[2] entities distributed across seven sites nationwide [12, pp. 1–2]. Many experimental technologies were in operation, including the Lincoln Laboratory grid filtering system.

Our data sample consisted of a representative 10-minute segment from the unclassified portion of the test. We wrote our own software to replay the log file. The resultant PDU stream was passed through an idealized network model, and periodic statistics were collected on PDU rates and multicast subscription levels.

[2]In staging large-scale exercises, it is often difficult or impossible to find enough human pilots to operate all of the required entities. *Semi-Automated Forces* (SAF) technology employs artificial-intelligence techniques to allow computers to substitute for many of the human controllers [3].

Humans are still required at the command level, since the computer is not intelligent enough to perform complex tasks. However, a single human can supervise many more entities when the lower-level tasks are all handled by the computer.



Significant portions of PDU traffic can be filtered out by rethresholding (i.e, applying an extra pass of dead-reckoning filtering.) We experimented with thresholds ranging from 10^{-9}m to 10^{9}m by powers of 10. We find large gap in traffic levels for any non-zero threshold. That is, in the range of thresholds between 0 and 1m, traffic drops to 70% of the original level. The rest of the graph has little utility to us; see text for details.

Figure 3-3: Data Reduction by Rethresholding

The results of the first relevant analysis are presented in Figure 3-3, a graph suggesting the effectiveness of *rethresholding*, a technique which thins out a PDU stream by passing it through an additional layer of high-threshold (low-resolution) DRMs. Rethresholding was used in the Lincoln Laboratory work as an aid to data collection [12, p. 3]; for completeness of the logs, PDUs from outside the AOIs were included, but only after being subsampled at a low rate. In our work, we apply rethresholding in the other direction, instead of including PDUs from outside the AOI, we will eliminate them from inside it.

To produce the graph, we applied many different levels of rethresholding to the log data. Specifically, we resampled the data at thresholds which ranged from 10^{-9}m to 10^{9}m by powers of 10. We also included a threshold of 0, for reference. The

horizontal axis of the graph denotes time elapsed into the simulation — we monitored the progression of the exercise to ensure that there were no transient anomalies. We found none, so all subsequent analyses will use time-averaged values instead.

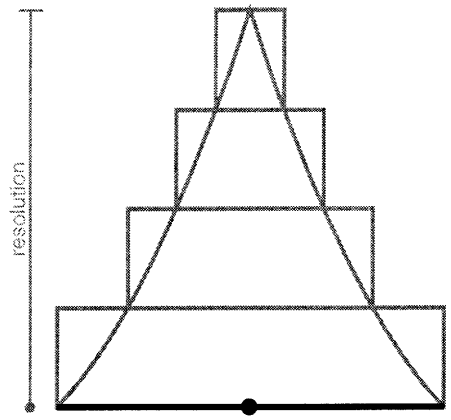
Now note the vertical axis, which denotes the percentage of total PDU traffic which is not rejected by rethresholding. The top graph line sits at the 100% traffic level. This is not surprising, since its threshold is 0, so it filters out nothing at all. The second line down is more interesting. It seems that any non-zero threshold results in an immediate drop to about 70%, which implies that about 30% of the PDUs involve absolutely no change in entity location. In other words, a significant amount of PDU traffic is spent in describing relatively minor events.

It is also interesting to note, in passing, that the "second line down" is actually composed of ten overlapped lines, corresponding to the thresholds from 10^{-9}m to 1m. The fact that these lines are virtually identical implies that the threshold for the underlying data was about 1m.

Traffic level continues to fall off at 10 and 100 meters, then it makes another jump between 100 and 10,000 meters. Actually, the center of the gap changes as evidenced by the rising inflection of the 1000m line. This is to be expected — as the simulation progresses, more and more vehicles manage to travel across distances of that magnitude. Unfortunately, this gap isn't of much use to us, since 1km details become negligible only at distances exceeding 3500km (~2000mi), which is well beyond the typical 50km battlefield size.[3] Going the rest of the way down the graph, we note that the traffic level falls to zero at 10^5m. This is not surprising, either, given the 50km-size of the battlefield.

We conclude that the zero to non-zero gap in the graph holds great promise for the effectiveness of a multi-resolution AOI. After all, our visual model tells us that zero-size movements should be observable only at zero distance. In practice, this fine-detail traffic should be easily constrained.

[3]Future DIS exercises may reach transcontinental scales, but for now, this observation is useless.



As mentioned in Figure 2-3, smooth curves are computationally impractical. Just as we approximate circular AOIs with square grid cells, we can substitute a series of steps for the smooth slope of the AOI's sides.

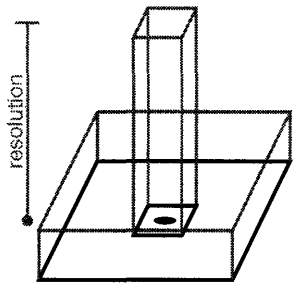
Figure 3-4: Square-grid Multi-resolution AOI

3.1.4 Discrete Multi-resolution (Multigrid) AOI

The first step in implementing our algorithm is to translate the circular, ideal AOIs a more tractable form. While circles are highly appealing for their geometric simplicity, they don't mesh easily with the computer's highly rectilinear coordinate systems. Regular grid squares are much easier to handle than free-floating circles, so we approximate the ideal AOIs using the cells of a square grid, as shown back in Figure 2-3. For every entity, we calculate the intersection of its circular AOI with the cells of the grid and include any squares which overlap the circle.

The standard geographical grid system takes care of our ideal AOI's base area, but for the "vertical" resolution component, we need to come up with another way to approximate the resolution/threshold curve. In a manner similar to the square gridding of the spatial component, we will choose discrete threshold levels at which to form "steps" along the sides of the AOI. Figure 3-4 gives a rough idea of our goal.

Looking back at Figure 3-3, we recall the rethresholding gap just above 0 (and forget the useless one around 1000m.) It would seem most effective to arrange our threshold levels such that the "step" fell into the gap, maximizing the differential between traffic levels at different step levels. Since we only have one gap, we decide



Looking back at the graph of Figure 3-3, we find a threshold gap just above 0. A threshold chosen in this region would represent the lowest "step" of the AOI shown in Figure 3-4.
We also find a gap above 100m, but this second gap is of no use to us. For a 100-meter detail to be unnoticeable, we would have to be much further away than the usual ROI of 4km. Thus, we decide to concentrate on a two-level AOI with one threshold at 0 and the other slightly higher.

Figure 3-5: Two-level Multi-resolution AOI

that we only need one step (not including the 0-threshold step which is always required at the innermost region of the AOI.) Thus, a two-level multigrid will be most suitable, and our AOI will take the form shown in Figure 3-5.

Now we can begin to discuss the actual operation of the multigrid system. So far we know we have grids on two distinct resolution levels. Let's refer to them as the *hifi* and *lofi* grids (abbreviations for *high-fidelity* and *low-fidelity*, respectively.) The *hifi grid* is represented by the tall central spire in Figure 3-5. This grid should have small cells to better accommodate the narrowness of regions it will have to cover. These small cells will carry high-resolution data, which must be confined to small, highly-restricted areas. The *lofi grid* is represented by the broad base in the diagram. This grid should have larger cells to more efficiently attain its greater coverage. Since low-resolution data is generated at slower rates, we don't have to be as concerned about the inevitable "spillovers" caused by the larger cells sticking out beyond the ideal AOI.

In operation, each grid runs DRMs at different levels of resolution (i.e., with different thresholds).[4] Before being sent, every PDU is checked against the lofi grid. If the PDU describes an movement whose magnitude which exceeds the lofi threshold,

[4]Actually, the hifi grid doesn't really have to run DRMs, since its threshold is just zero. But for consistency, imagine that it does.

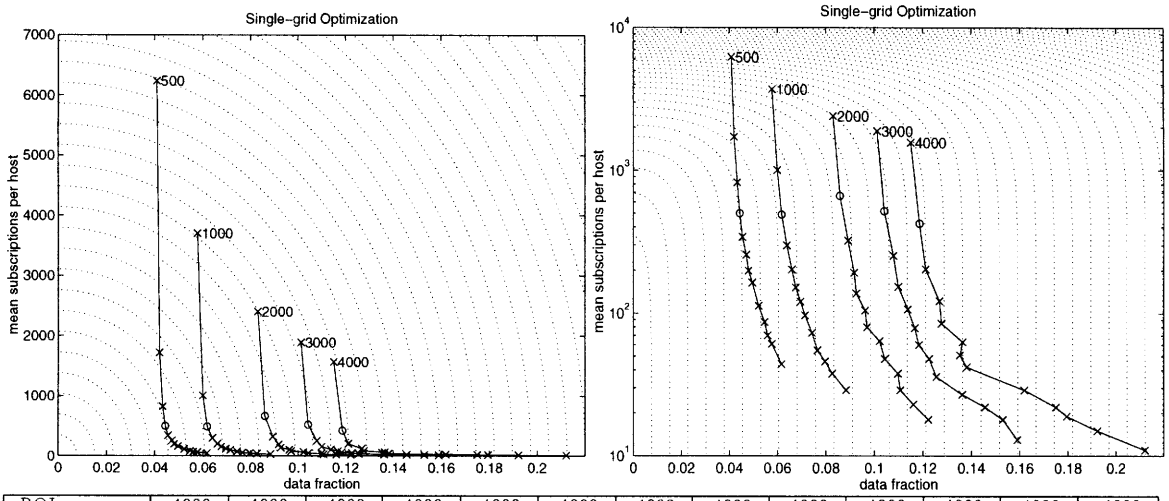
then it is dispatched to the lofi grid. (More precisely, it is sent to the multicast group corresponding to the lofi grid cell containing the originating entity.) Otherwise, the packet describes a "fine-detail" event, and the data is passed along to the hifi grid. In this way, data is sorted into low and high-fidelity categories.

On the receiving end, entities subscribe to the hifi grid for the small region in their immediate vicinity and fill in the rest of their ROI with subscriptions to the lofi grid. Thus, high-fidelity events only arrive from within a limited area, while the more infrequent low-fidelity events arrive throughout the whole of the AOI. Collectively, the low and high-fidelity data form the desired two-level AOI. Note that the hifi grid doesn't actually carry all the data; it suffers occasional "skips" when a packet is rerouted through the lofi grid. So it is necessary to subscribe to *both* grids to get true "full-fidelity" data. In essence, we have two quasi-independent grids, one with high fidelity over a small ROI, the other with low fidelity over a large ROI. We've just arranged things such that combining the data from the two grids happens to give you the standard, full-fidelity DIS data stream.

3.2 Single-grid Design Parameters

The multigrid algorithm sounds straightforward enough, doesn't it? However, there are certain details that must filled in before we proceed. First, we have to deal with what is, essentially, a single-grid issue. As stated in the original grid filtering paper [9], there is a fundamental question of what cell size should be used. Our problem is compounded by the fact that we will be trying many ROIs (We haven't yet decided on the exact ROI of the small-radius, hifi grid.) in addition to the standard 4km one.

At this point, we repeat the optimization procedure from the single-grid paper [9]. For each of our candidate ROIs, we test a set of cell sizes, ranging from $\frac{1}{8}$ROI to ROI. Figure 3-6 shows the results. For each ROI, we generate a curve by plotting the number of multicast subscriptions per host computer against the normalized PDU



ROI	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000
cell size	500	1000	1500	2000	2500	3000	3500	4000	5000	6000	7000	8000	10000
data fraction	0.115	0.119	0.121	0.127	0.128	0.136	0.135	0.138	0.162	0.175	0.180	0.192	0.212
subscriptions	1567	425	204	122	85	63	51	42	29	22	19	15	11
ROI	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000
cell size	375	750	1125	1500	1875	2250	2625	3000	3750	4500	5250	6000	7500
data fraction	0.101	0.104	0.108	0.110	0.114	0.117	0.118	0.123	0.126	0.136	0.146	0.153	0.159
subscriptions	1889	521	254	154	107	79	60	48	36	27	22	18	13
ROI	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
cell size	250	500	750	1000	1250	1500	1750	2000	2500	3000	3500	4000	5000
data fraction	0.083	0.086	0.089	0.092	0.093	0.096	0.097	0.102	0.104	0.110	0.111	0.116	0.122
subscriptions	2405	663	323	193	138	105	80	64	48	38	29	23	18
ROI	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
cell size	125	250	375	500	625	750	875	1000	1250	1500	1750	2000	2500
data fraction	0.058	0.060	0.062	0.064	0.066	0.068	0.070	0.071	0.074	0.077	0.080	0.083	0.088
subscriptions	3703	1005	489	299	203	152	121	97	73	55	46	38	29
ROI	500	500	500	500	500	500	500	500	500	500	500	500	500
cell size	62	125	187	250	312	375	437	500	625	750	875	1000	1250
data fraction	0.041	0.042	0.043	0.044	0.046	0.047	0.048	0.050	0.052	0.055	0.056	0.058	0.062
subscriptions	6237	1718	824	500	342	257	199	164	113	87	70	61	44

One major design parameter is the size of the grid cells in relation to the size of the AOI (as represented by the ROI.) We optimize our relationship by minimizing the per-host PDU reception and multicast subscription rates. The most efficient operating points (o) are marked above.
The plotted curves indicate PDU rate vs. number of subscriptions for various cell sizes and ROIs. Each individual curve represents a grid operating at a particular ROI, and the points along the curve represent various cell sizes for that ROI. Our optimization function is represented by distance from the origin (zero traffic with zero subscriptions), as illustrated by the concentric circles emanating from the origin. (In the graph on the left, these circles actually appear circular. On the right side, the circles are distorted, by semilog axes.) For each curve, the point closest to the origin is indicated with a circle.

Figure 3-6: Cell Size vs. ROI Optimization

rate. We have labeled this the *data fraction*, since it represents a fraction of original data volume.[5]

The resulting graph gives us a sort of cost-benefit diagram which we use to optimize our choices. The objective of our optimization is to minimize the data fraction without incurring an exorbitantly high subscription rate. We decided to keep things simple and use an obvious distance metric as a guide. Assuming that the origin is perfect (zero data load using zero multicast groups), we try to find the closest point to it. The dotted lines in the background of the graph indicate the distance isoclines, which are scaled along both axes to fit the range of the graph data. The graph on the left has linear axes and shows a natural rendition of the isoclines. The graph on the right uses a logarithmic y-axis to improve the detail in the curves, but the isoclines appear far less intuitive. Use both graphs to get a proper feel for the data.

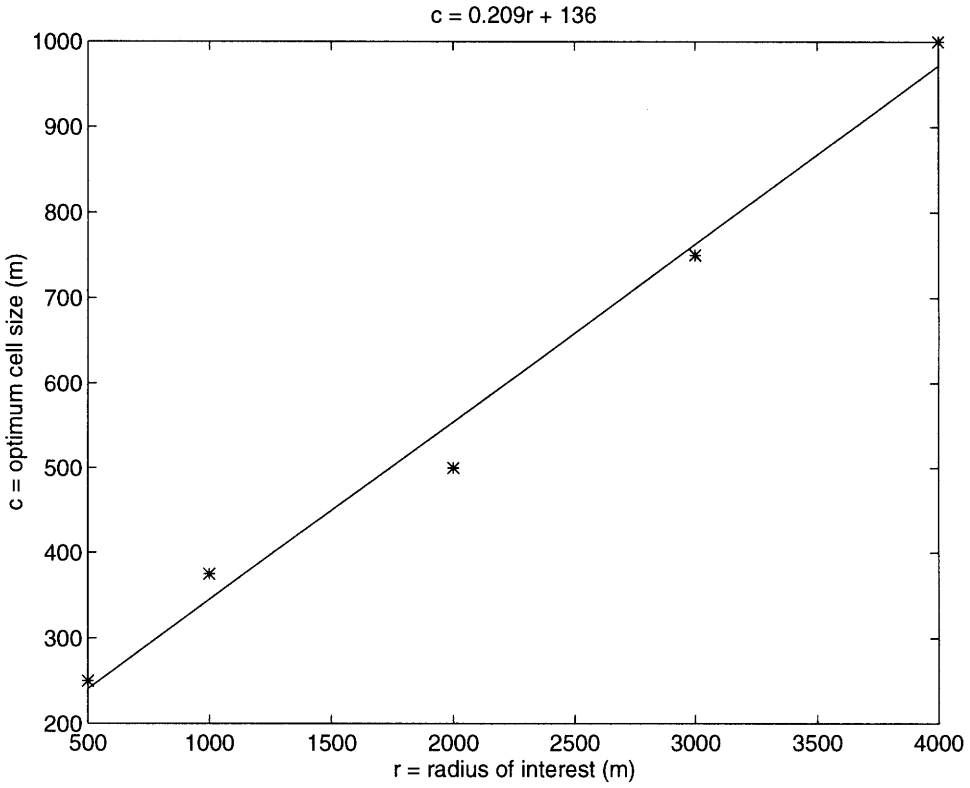
The optimal (closest-to-origin) points are marked with circles on the graph.[6] By going back to the data and extracting the ROIs and cell sizes for those points, we acquired the basis for forming a relationship between optimal cell size and ROI. This data is graphed in Figure 3-7, along with a best-fit line.

For the sake of verification, we note that our cell size for the 4km ROI is in the same ball park as the Lincoln Lab figure[9, p. 5] (1000m vs. 1500m). Given Lincoln's caveat that this figure is highly situation-dependent, we are satisfied with our result. As for the rest of the graph, it shows that the optimal cell size is directly proportional to the ROI in question. This is not surprising, since smaller squares should provide better approximations to smaller circles.[7]

[5]This is equivalent to the *download ratio* used in the Lincoln Laboratory paper.

[6]Interestingly enough, they seem to form a horizontal line (i.e., the optimal points coincide with a particular subscription rate). This may worthy of further study.

[7]It is somewhat interesting that the line doesn't go through the origin, but instead crosses above it. One possible explanation is that at some point, the savings realized through better approximations become outweighed by the savings of entities sharing groups, putting a lower practical bound on cell size.



The data presented in Figure 3-6 leads us to this relation for cell size as a function of ROI. By taking the optimal points from the previous graph, we obtain the best-fit line shown above.

Figure 3-7: Optimal Grid Cell Size vs. ROI

This is the formula we chose for the relation between cell size and ROI.

$$c = 0.209r + 136\text{m} \tag{3.3}$$

c = cell size
r = radius of interest

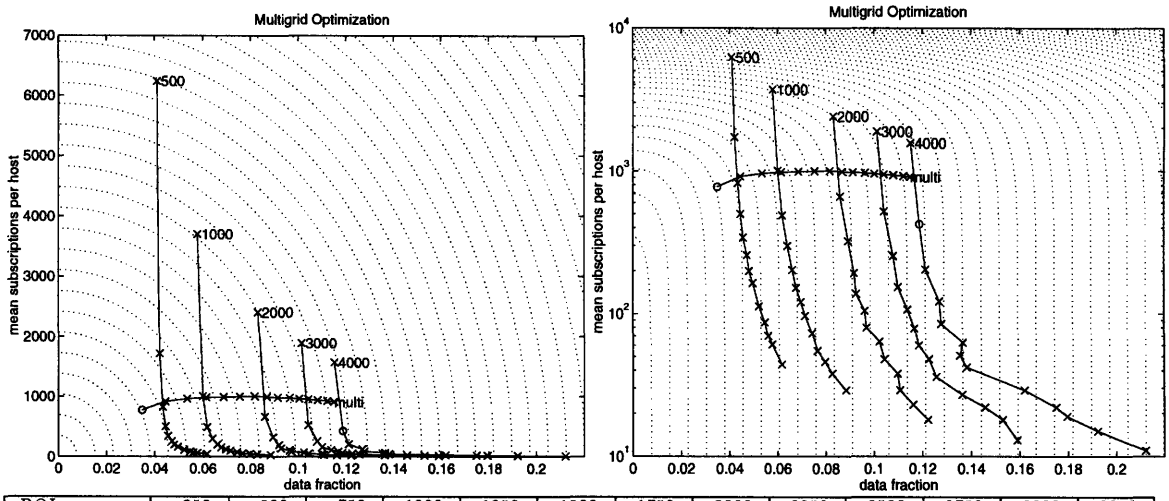
3.3 Multigrid Design Parameters

With the single-grid parameters out of the way, we can start on the main problem — how to set up the multigrid for optimal results. We have one major decision — how should we set the transition between the two levels of the multigrid? There are actually two related quantities involved in this decision: the "intermediate" ROI of the hifi grid and the corresponding threshold for the lofi grid. To link these two values, we go back to our visual acuity model of Equation 3.2. For any choice of intermediate ROI r, the corresponding threshold t is equal to the size s of the minimal discernible detail at that distance $d = r$. Because this threshold is equal to the size of the minimum discernible detail, it is valid to use it for the lofi grid. Since everything closer than the intermediate ROI will be handled on the hifi grid, the lofi grid doesn't ever need to cover details smaller than

$$t = 2.91 \times 10^{-4} r \tag{3.4}$$

t = threshold value
r = radius of interest(intermediate)

Now we are ready to construct the multigrid. We have found ways to calculate all the design parameters except one, the intermediate ROI. But we do know enough to find that value empirically. The following table summarizes all the other parameters going into our design.



ROI	250	500	750	1000	1250	1500	1750	2000	2250	2500	2750	3000	3250
data fraction	0.035	0.044	0.053	0.061	0.069	0.075	0.082	0.087	0.091	0.096	0.100	0.104	0.108
subscriptions	775	916	963	982	990	998	1003	992	981	973	959	949	937

ROI	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000	4000
cell size	500	1000	1500	2000	2500	3000	3500	4000	5000	6000	7000	8000	10000
data fraction	0.115	0.119	0.121	0.127	0.128	0.136	0.135	0.138	0.162	0.175	0.180	0.192	0.212
subscriptions	1567	425	204	122	85	63	51	42	29	22	19	15	11

The multigrid analysis adds a curve to the graph of Figure 3-6, showing that the multigrid filter achieves much better traffic reduction than the 4km single grid, and it does it with only a modest increase in the number of required multicast subscriptions.
Quantitatively, the multigrid algorithm offers a more than three-fold (3.41, to be exact) reduction in traffic at a cost of less than twice (1.82) the original number of multicast subscriptions.

Figure 3-8: Multigrid Optimization

grid	ROI	threshold	cell size
lofi	4 km	$2.91 \times 10^{-4}\, r$	$0.209\, r + 136$m
hifi	r	0	972m

The multigrid algorithm was run with intermediate ROIs ranging between 0 and 4km. The results are shown in Figure 3-8, which plots the optimization data, overlaid with the optimization data for the single-grid cases. Note that only the 4km and multigrid curves are marked with optimal points. A single grid with an ROI less than 4km is not a valid AOI for a real exercise; the curves are included only for reference.

We conclude that the multigrid is reasonably successful. At its optimal (leftmost) point, it produces $1/3.41$ times the data of the 4km-grid optimal point, while using only 1.82 times as many multicast groups. Scanning across the multigrid curve, we see that each time it crosses another curve, its data fraction is only slightly higher than that of the single grid corresponding to the intermediate ROI. This observation

is easily explained by the fact that the multigrid includes the single grid as its hifi component; the additional traffic comes from the lofi data. On the other axis, we see that the multigrid subscription rate is nearly horizontal, just like the single-grid optimums, but it sits somewhat higher. This, too, is to be expected from the fact that the multigrid is built out of single grids.

3.4 Multigrid Conclusions

To summarize our findings,

- A significant fraction of the data in a typical DIS exercise is spent in describing very small-scale motions; i.e., many of the most frequent events are also the most trivial ones.
- The optimal grid cell size for a particular radius of interest is directly proportional to that ROI.
- The multigrid technique offers a significantly lower data rate in exchange for a moderately higher subscription rate.

There are a few issues that weren't addressed in this paper but will probably be of interest at some point in the future.

- One undiscussed advantage of the multigrid system is the availability of a dedicated low-fidelity channel. (i.e., A receiver may choose to subscribe exclusively to the lofi grid.) In many simulations, tanks and other ground vehicles aren't the only observers present. There might be high-flying surveillance airplanes or, more simply, there might be commanders and strategists watching the action on plan-view displays (PVDs).

In simple terms, a PVD is basically a map-type display with abstract icons to represent entities on the battlefield. Such a display doesn't require close-up, high-fidelity data, but it does have a much larger than normal AOI. A low-fidelity channel is extremely well-suited to providing wide-ranging data to a PVD without swamping it.

- It isn't immediately obvious whether or not the multigrid scheme fully maintains the accuracy of the simulation. That is, it's possible that some of the events we dismissed as "imperceptible at a distance" actually aren't. For example, a tank rotating in place may change its apparent width by several meters, but our scheme would simply notice that it wasn't making any translational movement and relegate the data to the low-fidelity channel. Now, tanks spinning in place aren't common on the battlefield, but perhaps there are similar phenomena which are.

- We focused only on the "per host" aspect of multicast subscriptions. In reality, other considerations come into play, most notably, overall usage of the multicast address space. In other words, there is only a finite supply of multicast addresses which can be assigned to groups. If a grid scheme is too exorbitant in its use of multicast groups, then problems will arise.

  For this investigation, we made an implicit assumption which can be interpreted in at least two different ways. One, multicast address space is plentiful, and we don't have to worry about exhausting it by coming up with too many grid cells. For the sake of simplicity, many current grid schemes simply preassign specific multicast groups to each and every grid cell in an exercise. It saves computation, but more importantly, it obviates the need for the individual simulators to tell each other about a set of constantly changing multicast group assignments.

  The second option may now be obvious to you. If we don't preassign multicast groups, then we have to come up with a system to track and manage them during the simulation run. The difficulties in this approach are in the relative complexity of such a scheme and in the fact that some kind of intercommunica-

tion is necessary to ensure that all the simulators have the same idea of which group goes where. The network traffic associated with this activity could very well negate any benefits of the relevance filtering system.

Bibliography

[1] James O. Calvin and Daniel J. Van Hook. Agents: An architectural construct to support distributed simulation. In *EleventhWorkshop on Standards for the Interoperability of Distributed Simulations*, number 94-11-142, September 26–30 1994.
http://dss.ll.mit.edu/dss.web/94.142.ps.

[2] James O. Calvin and Richard Weatherly. An introduction to the high level architecture (hla) runtime infrastructure (rti). 1996.
http://www.dmso.mil/docslib/briefs/DIS/14DIS/96-14-103.PS.

[3] Andy Ceranowicz. Modular semi-automated forces. In *Electronic Conference on Constructive Training Simulation*, 1994.
http://www.mystech.com/~smithr/elecsim94/modsaf/modsaf.txt.

[4] The DIS Steering Committee. The dis vision: A map to the future of distributed simulation. Technical Report IST-SP-94-01, Institute for Simulation and Training, University of Central Florida, May 1994.
http://ftp.sc.ist.ucf.edu/STDS/docs/vision/index.htm.

[5] Joint Data Base Elements. Dis data dictionary, May 1997.
http://STDS.sc.ist.ucf.edu/dis/dis-dd/index.htm.

[6] Institute for Simulation and Training. Enumeration and bit encoded values for use with protocols for distributed interactive simulation applications. Standard

for Information Technology IST-CR-93-19, Institute for Simulation and Training, University of Central Florida, June 1993.

[7] Daniel Michel and David L. Brock. A 3d environment for an observer based multiresolution architecture. In *Simulation Interoperability Workshop*, Orlando, FL, March 1997.

[8] U.S. Department of Defense Joint Services Steering Committee. *Human Engineering Guide to Equipment Design.* U.S. Government Printing Office, 1972.

[9] Steven J. Rak and Daniel J. Van Hook. Evaluation of grid-based relevance filtering for multicast group assignment. In *Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations*, number 96-14-106, March 11–15 1996.

[10] S. K. Singhal and D. R. Cheriton. Using projection aggregations to support scalability in distributed simulation. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, Hong Kong, May 1996. IEEE Computer Society.
ftp://ftp.dsg.stanford.edu/pub/papers/projections.ps.Z.

[11] Daniel J. Van Hook, James O. Calvin, Michael K. Newton, and David A. Fusco. An approach to dis scaleability. In *Eleventh Workshop on Standards for the Interoperability of Distributed Simulations*, number 94-11-141, September 26–30 1994.
http://dss.ll.mit.edu/dss.web/94.141.ps.

[12] Daniel J. Van Hook, David P. Cebula, Steven J. Rak, Carol J. Chiang, Paul N. DiCaprio, and James O. Calvin. Performance of stow ritn application control techniques. In *Fourteenth Workshop on Standards for the Interoperability of Distributed Simulations*, number 96-14-157, March 11–15 1996.

5466-45