NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DYNAMIC SCALABLE NETWORK AREA OF INTEREST MANAGEMENT FOR VIRTUAL WORLDS

by

Michael S. Wathen

September 2001

Thesis Advisor:	Michael Capps
Second Reader:	Don McGregor

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Dynamic Scalable Network Area of Interest Management for Virtual Worlds			**5. FUNDING NUMBERS**
6. AUTHOR(S) **Wathen, Michael Scott**			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

A major performance challenge in developing a massively multi-user virtual world is network scalability. This is because the network over which entities communicate can quickly develop into a bottleneck. Three critical factors: bandwidth usage, packets per second, and network-related CPU usage, should be governed by the number of entities a given user is interested in, not the total number of entities in the world. The challenge then is to allow a virtual world to scale to any size without an appreciable drop in system performance. To address these concerns, this thesis describes a novel Area of Interest Manager (AOIM) built atop the NPSNET-V virtual environment system. It is a dynamically sized, geographical region based, sender-side interest manager that supports dynamic entity discovery and peer-to-peer entity communication. The AOIM also makes use of tools provided by the NPSNET-V system, such as variable resolution protocols and variable data transmission rate. Performance tests have shown conclusively that these interest management techniques are able to produce dramatic savings in network bandwidth usage in a peer-to-peer virtual environment. In one test, this AOIM produced a 92% drop in network traffic, with a simultaneous 500% increase in world population.

14. SUBJECT TERMS NPSNET, Multicast, Interest Management, Scalable Virtual Environments, Octree			**15. NUMBER OF PAGES** 71
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DYNAMIC SCALABLE NETWORK AREA OF INTEREST MANAGEMENT FOR VIRTUAL WORLDS

Michael S. Wathen
Lieutenant, United States Navy
B.S., University of Oklahoma, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
September 2001

Author: _____
Michael S. Wathen

Approved by: _____
Michael Capps, Advisor

_____
Don McGregor, Second Reader

_____
Rudy Darken, Academic Associate
Modeling, Virtual Environments, and Simulation Group

_____
Michael Zyda, Chairman
Modeling, Virtual Environments, and Simulation Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A major performance challenge in developing a massively multi-user virtual world is network scalability. This is because the network over which entities communicate can quickly develop into a bottleneck. Three critical factors: bandwidth usage, packets per second, and network-related CPU usage, should be governed by the number of entities a given user is interested in, not the total number of entities in the world. The challenge then is to allow a virtual world to scale to any size without an appreciable drop in system performance.

To address these concerns, this thesis describes a novel Area of Interest Manager (AOIM) built atop the NPSNET-V virtual environment system. It is a dynamically sized, geographical region based, sender-side interest manager that supports dynamic entity discovery and peer-to-peer entity communication. The AOIM also makes use of tools provided by the NPSNET-V system, such as variable resolution protocols and variable data transmission rate.

Performance tests have shown conclusively that these interest management techniques are able to produce dramatic savings in network bandwidth usage in a peer-to-peer virtual environment. In one test, this AOIM produced a 92% drop in network traffic, with a simultaneous 500% increase in world population.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
- **A. THESIS STATEMENT 1**
- **B. MOTIVATION 1**
- **C. DEFINITION 3**
- **D. APPLICATION 3**
- **E. APPROACH 4**
- **F. THESIS GOALS 4**
- **G. THESIS ORGANIZATION 5**

II. RELATED WORK AND BACKGROUND 7
- **A. INTRODUCTION 7**
- **B. PRIOR WORK 7**
  - **1. Client-server systems 7**
    - ***a. BrickNet 7***
    - ***b. RING 8***
    - ***c. SPLINE 9***
  - **2. Peer-to-peer systems 10**
    - ***a. DIVE 10***
    - ***b. NPSNET-IV 11***
    - ***c. Three-Tiered Interest Management Scheme 12***
- **C. NPSNET-V BACKGROUND 12**
  - **1. Entities 12**
  - **2. Protocols 13**
  - **3. EntityDispatcher 13**
- **D. MULTICAST 13**
  - **1. Multicast benefits 14**
  - **2. Multicast drawbacks 14**
- **E. CONCLUSION 15**

III. DESIGN AND IMPLEMENTATION OF THE AOIM 17
- **A. INTRODUCTION 17**
- **B. REQUIREMENTS 18**
- **C. DESIGN OVERVIEW 18**
  - **1. Dynamic geographical regions 18**
  - **2. Send-side filter paradigm 23**
  - **3. AOIM with minimal centralization 23**
  - **4. Implementation details 24**
  - **5. Tools provided by NPSNET-V 26**
  - **6. Dynamic entity discovery 29**
- **D. SUMMARY 29**

IV. PERFORMANCE ANALYSIS 31
A. INTRODUCTION 31
B. TEST OVERVIEW 31
C. TEST CONFIGURATION 32
D. TEST GROUP ONE: INDIVIDUAL COMPONENT TEST 33
1. Test description and parameters 33
2. Test results for test group one 34
E. TEST GROUP TWO: DATA FIDELITY PERFORMANCE TEST 38
1. Test description and parameters 38
2. Test results for test group two 38
F. TEST GROUP THREE: SCALABILITY TEST 39
1. Test description and parameters 39
2. Test results for test group three 39
G. CONCLUSIONS 39
H. RECOMMENDATIONS 41
I. SUMMARY 41

V. CONCLUSIONS AND FUTURE WORK 43
A. INTRODUCTION 43
B. OCTREES 43
C. MULTI-LAYERED APPROACH 44
D. DYNAMIC ENTITY DISCOVERY AND REFLECTION 45
E. FUTURE WORK 45
1. Multi-agent monitoring system 45
2. Look-ahead algorithm 46
3. Multiple octrees 47
4. Entirely new AOIM 47
F. CONCLUSION 48

APPENDIX A 49
A. INTRODUCTION 49

APPENDIX B 51
A. INTRODUCTION 51

LIST OF REFERENCES 53

INITIAL DISTRIBUTION LIST 55

LIST OF FIGURES

Figure 1. NPSNET-V packet distribution process 17
Figure 2. AOIM functionality overview.....18
Figure 3. Octree leaf node 19
Figure 4. Octree after one subdivision.....19
Figure 5. Octree after two subdivisions 19
Figure 6. Graph representation of octrees from Figures 3, 4, and 5 20
Figure 7. Zone containing transmitting entity 21
Figure 8. Zone numbering convention.....22
Figure 9. Client server architecture 25
Figure 10. Example of dead reckoning 27
Figure 11. Example of dead reckoning with smoothing 28
Figure 12. AOIM process and functionality 29
Figure 13. Mean bytes per second vs. Data fidelity 38
Figure 14. UML diagram of the AOIM system (part 1).....51
Figure 15. UML diagram of the AOIM system (part 2).....52

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Zone and population configurations 33
Table 2. Parameter combinations used during data collection.....34
Table 3. Test results for test configuration one 35
Table 4. Test results for test configuration two 36
Table 5. Test results for test configuration three.....37

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS

ANOVA	Analysis of Variance
AOI	Area of Interest
AOIM	Area of Interest Manager
BPS	Bytes Per Second
DIVE	Distributed Interactive Virtual Environment
GUI	Graphical User Interface
HLA	High-Level Architecture
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
LAN	Local Area Network
NAK	Negative Acknowledgement
PDU	Protocol Data Unit
PPS	Packets Per Second
SPLINE	Scalable Platform for Large Interactive Networked Environments
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network
WAVES	Waterloo Virtual Environment System

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Many heartfelt thanks to:

- Dr. Michael Capps for the unenviable task of keeping my grammar straight and for the outstanding turnaround times during the editing process.
- Don McGregor for being a network programming genius. Without his technical assistance, I would still be trying to get that first java file to compile.
- Andrzej Kapolka for always taking the time to answer a programming question.
- Jimmy Liberato for being the worlds greatest network administrator. The network never once stumbled despite my best attempts to crash it.

Lastly, a very special thank you goes to my beautiful wife Melissa and son Jacob. This thesis took up way too much of my time, but you guys never once complained. Thanks!!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THESIS STATEMENT

It is possible to integrate an area of interest management (AOIM) scheme into a peer-to-peer virtual world that supports dynamic entity discovery. Furthermore, by offering dynamic class and protocol behavior control mechanisms, this AOIM can offer improved scalability over a standard interest management scheme.

B. MOTIVATION

Anyone that takes the time to study any of the multi-user virtual worlds currently in use today, such as Everquest™ by Verant™ Software (www.verant.com), or Team Fortress™ by Valve Software™ (www.valvesoftware.com), will undoubtedly come away impressed. These products whisk users away to a fantasyland where they not only get to slay dragons, or defend Earth against an alien hoard, but they can do these things in a world populated with other real people. They can work together as a team, hey can compete against other users for the same goal, or they can go head to head trying to eliminate each other. The possibilities are boundless.

With their impressive 3D graphics, and 3D spatial sound, it is easy to assume that the biggest challenge facing the developers of these worlds is optimizing the graphics and sound programming to maintain an acceptable frame rate, as well as smooth flowing sound. That assumption is incorrect. The computers of today are more than powerful enough to handle real time, near photo-realistic graphics, and spatial audio. In fact, the transistor count of a modern 3D graphics chip alone outstrips those of the most powerful CPUs of only a few years ago. This trend of faster more powerful computers will not end anytime soon.

The single biggest performance challenge to anyone developing a massively parallel multiplayer world is the network programming. The reason for this is that the network over which the entities in the world must communicate can quickly develop into a bottleneck. Entities communicate with each other by sending informational packets across the network. These packets contain everything the other players in the world need to know about the entity sending the packet, such as position and appearance. In the

unicast case, network traffic is $O(N^2)$, where N is the number of entities in the world. As new players join the world, they will begin trying to communicate with all other entities in the world, by sending packets over the network, while at the same time trying to receive this same information from everyone else. It will not take long for this bottleneck to manifest itself in the form of dropped packets.

As an example, if a virtual world that is running at 30 frames per second has 32 players in it, each sending a single position-packet per frame, the net result will be 1024 packets per frame put on the network, which equates to 30,720 packets per second. If the number of players in the world increases from 32 to 300, the number of packets per second will increase to 2,700,000. Should the number of players climb to 2000 (a common number of players found on any of Everquest's many servers at any given time) the packets per second put on the network will increase to 120,000,000. This is just the number of packets; the bandwidth used is determined by multiplying packets per second by the packet size. Even if the packet size is very small, say for example 50 bytes, the amount of data traveling the net will still quickly overwhelm the fastest network. In the Everquest example given above, the 2000 players would attempt to send almost 6 gigabytes of data per second! Obviously, this is not going to happen, and the fact that most users of these online games are still using dial-up connections and 56K modems only magnifies the problem.

Three critical network factors greatly help determine the performance of an online virtual world. They are bandwidth usage, packets per second, and network related CPU usage. In a multi-user virtual world, these factors should be governed by the number of entities a given user is actually interested in, not the total number of entities in the world. As evidenced in the earlier Everquest example, just increasing available bandwidth to individual homes via DSL or cable modems is not enough. Most users in a virtual world will still see their systems have unacceptable performance when the number of users in the world grows above even a small number. The challenge then is to allow a virtual world to scale to any size (or at least to several thousand users) without an appreciable drop in system performance. This is the motivation for area of interest management.

C. DEFINITION

AOIM is a general term used to describe any filtering process applied to data packets to ensure they are used only when and where they are actually required. There are many AOIM schemes in existence which generally fall into one of two categories: send-side or receive-side.

Receive-side filters receive every packet on the network and then determine each packet's relevance. If the packet is pertinent to an entity located on that local host it is passed along to that entity, otherwise the packet is discarded. An advantage to a receive-side interest management scheme is the simplicity with which packets are sent. No *a priori* knowledge of what entities occupy the world or where they are located is required. However, this type of interest manager does not address bandwidth conservation, and that is its major disadvantage.

Send-side filters attempt to send packets only when and where they are required. An advantage to an interest management scheme of this type is a large reduction in bandwidth usage. A disadvantage is the complexity of calculating in real time where to send each packet.

D. APPLICATION

The main challenge in implementing interest management into a peer-to-peer virtual environment is data consistency and AOIM coordination. In a true peer-to-peer system, each client has its own copy of the world at large (interest database), or at least that portion of the world with which it is currently concerned. In effect, it is a large distributed database that must be kept consistent. If the AOIM of one client in the world decides to make a change, that change must quickly propagate to every other AOIM in the world to ensure no loss of data. In a virtual world that allows for dynamic entity discovery and loading, this is doubly difficult because new clients and entities can leave or join the world at anytime.

NPSNET-V is a component based, dynamically extensible, scalable framework for creating cross-platform, persistent virtual worlds. NPSNET-V eschews the more common client-server architecture in favor of a mostly serverless peer-to-peer approach. NPSNET-V fully supports dynamic entity discovery, and area of interest management

has yet to be implemented in this type of environment. This makes NPSNET-V a perfect candidate for performance testing of an area of interest manager supporting dynamic behavior control mechanisms. The area of interest management scheme developed in this thesis, and integrated into NPSNET-V, is primarily a send-side system.

E. APPROACH

This thesis shows that the application of an area of interest manager greatly reduces the amount of network bandwidth used by a peer-to-peer virtual world supporting dynamic entity discovery, which in turn translates directly into much more scalability.

There are three main ways this AOIM accomplishes its goals. The first is through extensive use of dead reckoning, which is the process of calculating an entity's current position based on its last known position, speed, and heading, as well as elapsed time since the last known position. The second is by utilizing variable resolution data packets that allow the area of interest manager to make level of detail adjustments of data packets on the fly. The third is by using multicast and dynamically sized geographic zones, each assigned one or more multicast addresses, allowing an efficient method of sender side packet filtering.

F. THESIS GOALS

- Develop a complete, dynamic, multi-function, area of interest management system that utilizes dead reckoning, variable packet resolution, and dynamically sized geographic multicast regions.
- Implement this AOIM scheme into NPSNET-V, such that it is capable of managing dynamic entities and dynamic entity types.
- Gather benchmark data to show this AOIM scheme allows a much higher level of scalability than an AOIM without support for dynamic loading.
- Provide future students with a baseline set of Java classes defining an AOIM that can be adapted to schemes other than geographic zones.

G. THESIS ORGANIZATION

The organization of chapters in this thesis is as follows:

- Chapter I: Introduction. Contains the thesis statement, an overview of the problem and some possible solutions, a few important definitions, and lists the goals this thesis strives to obtain.
- Chapter II: Background. Review of prior and current work in this area. Describes other area of interest management systems, listing some of their strengths and weaknesses. Also discusses NPSNET-V, describing in detail its unique features as they pertain to area of interest management. Finally this chapter discusses IP multicast as it applies to interest management.
- Chapter III: Approach and architecture. Discusses the approach this thesis takes in the development of a send-side area of interest management scheme, as well as the hurdles that were encountered and overcome. Also discusses in detail all algorithms used for the area of interest manager, whether borrowed from another source, or developed specifically for this thesis.
- Chapter IV: Experimentation and analysis. Compares NPSNET-V performance, scalability, and bandwidth usage both with and without the area of interest management scheme developed in this thesis. Discusses in depth each specific test and the network conditions under which they were run.
- Chapter V: Conclusions and future work. Discusses the strengths and weaknesses of the interest management scheme developed by this thesis. Discusses specific changes to the system that could improve performance. Lastly, this chapter discusses several thesis level modifications that could be made to this system.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK AND BACKGROUND

A. INTRODUCTION

This chapter provides background information for several previous area of interest management systems. It also discusses NPSNET-V, including many important definitions used throughout the rest of this thesis. Lastly, this chapter discusses IP multicast as it applies to AOIM systems in general.

B. PRIOR WORK

Many virtual environment systems were developed before the inception of NPSNET-V. These can be roughly broken-up into two distinct groups: client-server and peer-to-peer.

1. Client-server systems

As the name implies, the following systems primarily use client-server architectures. NPSNET-V is mostly peer-to-peer, and the client-server systems discussed here are included for completeness.

a. BrickNet

The Bricknet toolkit was originally designed to facilitate the rapid development of networked virtual worlds. The worlds would operate on networked workstations and form a loosely coupled system. The designers envisioned that Bricknet would be used to construct multi-user games, concurrent engineering systems, and other asynchronous GUI environments.

The entity communication process in BrickNet is as follows: Clients pass packets via UDP to their server. The server then decides where those packets need to go. If other clients on that same server need the packets, the server will transmit them directly there. If clients that are being serviced by other servers need the packets, then the server will pass the packets via UDP to the appropriate server and they in turn will pass them on to their respective clients.

BrickNet implements interest management by having each client register with the server when it connects to the net. As part of this registration process, the client

passes a list of objects the given entity is interested in to the server (Singh 1994). The server will then pass the appropriate packets along as necessary. For an example of how this may work, consider a player joining the world in an aircraft. The player's client will register with the appropriate server and may express an interest in other aircraft only. As the player flies around the world and reports to its server any actions it takes, such as turning, climbing, or firing, the server will pass the information on. It is important to note, however, that the information is only passed to other entities that have expressed an interest in aircraft. If the player were to fly over a large field containing several soldiers, he would not receive any packets from them and would consequently not know that they were there. However, if the soldiers had expressed an interest in aircraft, they would receive the aircraft's packets, and would therefore be able to detect it. Again, this system is completely client-server.

The advantage to this system is simplicity. The client-server paradigm is well understood. The biggest disadvantage is scalability. To scale to the size that NPSNET-V aspires to requires something other than a client-server architecture.

b. RING

RING is another client-server system that provides for multi-user virtual worlds. It was designed to support densely occluded virtual environments, such as buildings or cities, with a large number of users. RING is a good choice for building a walkthrough, but would not lend itself well to a virtual world with wide-open spaces, such as a flight simulator.

RING uses UDP datagrams exclusively, avoiding Multicast altogether. The area of interest management approach taken by RING is that as entities change state they send update packets to their respective server. The server in turn performs line of sight calculations, and then passes the packets along to only the entities that it has determined can actually see the recently changed entities. This system works best in a "large, densely occluded environment." (Funkhouser 1995) In other words, if everything is in the line of sight of everything else, then there will not be much packet culling taking place.

The biggest drawback to this system is its reliance on a server hierarchy and the associated problem of reliability/survivability. If one server crashes, the whole system crashes. Another, albeit lesser problem with the server hierarchy is that a data packet may have to pass from the sending entity through multiple servers, each one performing their own calculations, before finally being transmitted to its destination, thus adding latency.

c. SPLINE

SPLINE provides an architecture for creating multi-user interactive environments based around a shared world model. It was intended for building large-scale social environments with many simultaneous users.

SPLINE is a region-based system that makes extensive use of servers. In SPLINE the world is divided into non-uniform areas called locales. "The concept of locales is based on the idea that while a virtual world may be very large, most of what can be observed by a single user at a given moment is nevertheless local in nature" (Barrus 1996). There are three key characteristics shared by all locales: separate communication channels, local coordinate systems, and arbitrary geometry and relationships.

SPLINE, like NPSNET-IV, implements communication by assigning individual multicast addresses to locales. Unlike NPSNET-IV, however, locales can vary in size and shape. This allows locales to exactly fit the world at large with no wasted space. It also allows for a very efficient use of multicast addresses (a scarce resource). If a virtual world is not uniformly populated, then all of the unpopulated portion of the world can be represented as one locale. An example of this is an aquarium simulation where most of the fish are schooling in one part of the tank. In this respect, SPLINE is one of the most efficient systems reviewed here.

The single apparent weak spot with SPLINE is the fact that it relies heavily on communication servers. In fact, every time a new locale is created, a new locale server is created right along with it. Any SPLINE process that becomes interested in a locale will query its server to receive the most up-to-date information about the objects currently in that locale. It is important to note, however, that entity-to-entity

communication is via multicast. The servers are only used to allow for latecomers and world consistency. As in any system that relies heavily on servers though, scalability is likely to be of concern, as well as reliability since servers are single points of failure.

2. Peer-to-peer systems

The following systems are primarily peer-to-peer, but may make use of bootstrap servers. NPSNET-V falls into this category.

a. DIVE

Like SPLINE, DIVE was designed as a fully distributed system to enable large numbers of users to interact in a shared environment. A further design goal was that each user also be able to interact with the environment itself.

"DIVE focuses on peer-to-peer multicast communication as opposed to a 'classic' client-server model" (Frécon 1998). However, the system does make use of an important server called the "Collision Manager" in its implementation of area of interest management.

DIVE makes use of two concepts called focus and nimbus. Focus is defined as a geographic area around an entity within which that entity is able to detect other objects in the world. If something enters an entity's focus, that entity will be able to see and hear it. A nimbus is also a geographic area around an entity. Objects in an entity's nimbus are able to see and hear the entity. This interesting approach allows for situations where an airplane is inside a radar site's nimbus, while at the same time; the radar is outside of the airplane's nimbus. This allows the airplane to detect the radar, without the radar detecting it. This is very much how things work in the real world.

There is, however, a drawback to this system, and it is a big one: scalability. The collision manager constantly calculates the focus and nimbus collisions for every entity in the world. Since these calculations are $O(N^2)$, they are clearly the limiting factor in the scalability of any DIVE application. More importantly, this server is a single point of failure; without it, the system cannot function.

Reliance on the collision manager not-withstanding, entity-to-entity communication in DIVE is peer-to-peer. When an entity needs to make a request, or

realizes that it is missing a needed data packet, it sends a request over the appropriate multicast channel. Utilizing a proprietary algorithm to estimate packet round-trip-time and calculate an appropriate timeout, the entity that is closest to the requesting entity will respond with the needed information. This response is sent over multicast as well, thus ensuring that other entities considering responding to the request, will realize that another entity already has, and will not themselves respond. This prevents an explosion of responses for each single request that is sent (Frécon 1998).

b. NPSNET-IV

NPSNET-IV, the precursor to NPSNET-V, was designed primarily to support virtual battlefield simulations over large expanses of terrain. As such, there is no concept of height or depth. A virtual world built on NPSNET-IV will essentially be flat. This limitation would likely preclude using NPSNET-IV for the development of a flight simulation or any similar system.

NPSNET-IV makes extensive use of multicast in implementing area of interest management (Macedonia 1995). In this system, the world is broken up into a grid of hexagonal cells. Each of these cells is associated with a multicast address. When an entity first enters a new cell, it subscribes to the appropriate multicast address. Next, the entity will send a join PDU to the cell, and the "oldest" entity in the cell will respond with state information for every entity currently in the cell. This response is made over a TCP/IP connection to ensure reliability. When an entity moves into yet another cell it will send a leave PDU to the old cell, and the process repeats again.

The biggest drawback to this system is that it is fairly application specific. The reason for this is that the hexagonal cells used to subdivide the world are not dynamically resizable. Their size is determined at compile time. Much experimentation went into the determination of the optimal size to make the cells for the initial NPSNET-IV demonstration, and any reuse of NPSNET-IV for a different application would require similar experimentation. If every entity in the world were to move into one cell, this system could not adapt.

c. Three-Tiered Interest Management Scheme

The 'Three-Tiered Interest Management Scheme' is another AOIM developed at the Naval Postgraduate School (Abrams 1999). This system uses dynamically sized zones, represented by multicast addresses, as the first tier. The second tier uses data received from the first tier to, "create a protocol independent perfect match between a client's interests and the environment." The third tier takes the data passed through the first two tiers and filters it based on specific protocols. The single largest drawback to this system is that the second tier was implemented by assigning a separate multicast address to each entity in the world. Since multicast groups are a scarce resource, this clearly causes some serious scalability problems.

C. NPSNET-V BACKGROUND

The NPSNET research project was begun in 1990 at the Naval Postgraduate School. The project is a continuing test bed for advanced applied research in networked virtual environments (McGregor 2001). NPSNET-V is the fifth and latest incarnation of this research. The goal of NPSNET-V is to be "a framework for fully distributed, component based, persistent, networked virtual worlds, extensible at runtime and scalable to infinite size on the Internet." NPSNET-V is a large and complex architecture and describing it in its entirety is far beyond the scope of this chapter. The interested reader is encouraged to point their web browser to http://www.npsnet.org/~npsnet/v/index.html or to read the paper: *NPSNET-V. A New Beginning for Dynamically Extensible Virtual Environments* (Capps 2000), for complete information on the NPSNET-V architecture. There are, however, three specific components of NPSNET-V, the understanding of which, are critical to the AOIM developed in this thesis. They are entities, protocols, and entity dispatcher.

1. Entities

Entities in NPSNET-V are defined as any "active" component in the virtual world. These could include obvious things such as vehicles or avatars, but less intuitive things such as terrain elements or physical forces can also be entities. There are three types of entities possible in NPSNET-V: masters, ghosts, and shared.

Entity masters, as the name would imply, are the actual representation of an object in the world. There will only ever be one entity master per object in the world, and it will reside on the machine of the user controlling it. They contain the most accurate, up to date information concerning the object they represent.

Entity ghosts represent their master. They approximate their respective master as closely as possible. If a user creates a new entity master in the virtual world, every other user in that world that sees "it", will actually be seeing an entity ghost.

"Shared entities approximate ideal shared state" (McGregor 2001). Good examples of shared entities are terrain, or physical models.

2. Protocols

When an entity changes state, that change needs to be made known to the virtual world at large. Protocols are the component responsible for this. They communicate all state changes over the network, via the entity dispatcher. "Current protocols are lightweight and composable; each transmits a single element of state" (McGregor 2001). Consequently, there will likely be multiple protocols per entity. Some examples of protocols in NPSNET-V are TransformProtocol and AnimationProtocol.

3. EntityDispatcher

This is the NPSNET-V object that deals with network communication. There is only one of these per machine in the virtual world, and protocols make use of methods in this class to send data packets over the network. Figure 1, in the following chapter, shows the relationship between entities, protocols, and entity dispatcher.

D. MULTICAST

Multicast, which is based on UDP, is a form of network communication that sends out data non-redundantly across a network graph. Like UDP, multicast is unreliable; unlike UDP, however, multicast packets do not go to a specific Internet address, but are sent to a group address instead. The process is similar to opening a broadcast network connection, with the additional requirement that the user must also subscribe to the specific multicast addresses they are interested in. Any packet sent to a multicast address will result in that packet being sent to every other computer that is currently subscribed. Data reception works exactly the same way. Subscribing to a

multicast address will result in the subscribed computer receiving every data packet sent to that particular multicast address. There is no concept of "send only", or "receive only" where multicast is concerned. If a computer subscribes to a multicast address it will immediately begin receiving every data packet sent to that address (even its own). It is a simultaneous one-to-many and many-to-one relationship.

1. Multicast benefits

There are several benefits to using multicast. The first of these is that there is no need for a sender to keep track of subscribers. A user just transmits data packets once, and anyone currently subscribed, including the transmitter, receives them; subject of course to the fact that multicast is unreliable. This is a powerful feature, but the single largest benefit to using multicast is that data filtering is pushed onto the network hardware layer and away from the application layer. As network hardware improves, becoming faster and more sophisticated, the user will see the benefits immediately and automatically.

2. Multicast drawbacks

There are also some drawbacks to multicast. There are 268,435,456 multicast addresses available on today's Internet. They range from 224.0.0.0 through 239.255.255.255. At first glance, this may not seem like a problem, but as more and more virtual worlds pop up, each with possibly thousands of users, using hundreds or thousands of multicast addresses, the very real possibility of collisions presents itself. As an example, consider two massively multi-player worlds trying to use the same block of multicast addresses. Game-two's data packets arriving at machines running game-one and vice versa will cause problems. At the very least bandwidth will be wasted, and in the worst case one or both games could crash or have unexpected behavior.

Another problem with multicast is that it is not widely supported on the Internet. Many routers have their multicast feature disabled, or only support a limited number of addresses. In addition, PC network cards currently support only a few simultaneous multicast groups in hardware (Abrams 1999). More than this requires pushing some of the processing back up to the operating system layer, thereby partially negating the single biggest advantage to using multicast in the first place.

E. CONCLUSION

This chapter has outlined several virtual world implementations both peer-to-peer and client-server, along with their associated interest management schemes. It has provided the necessary background on NPSNET-V, the architecture that this AOIM is integrated with, to understand the rest of this thesis. Lastly, multicast was described along with it benefits and limitations.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN AND IMPLEMENTATION OF THE AOIM

A. INTRODUCTION

This chapter presents the underlying theory, design, and implementation of the geographical area of interest manager. It discusses in detail the requirements that an AOIM must meet or exceed to be successful, the design choices made to meet those requirements, and lastly the actual implementation. In short, this chapter is the why, what, and how of a successful geographical AOIM.

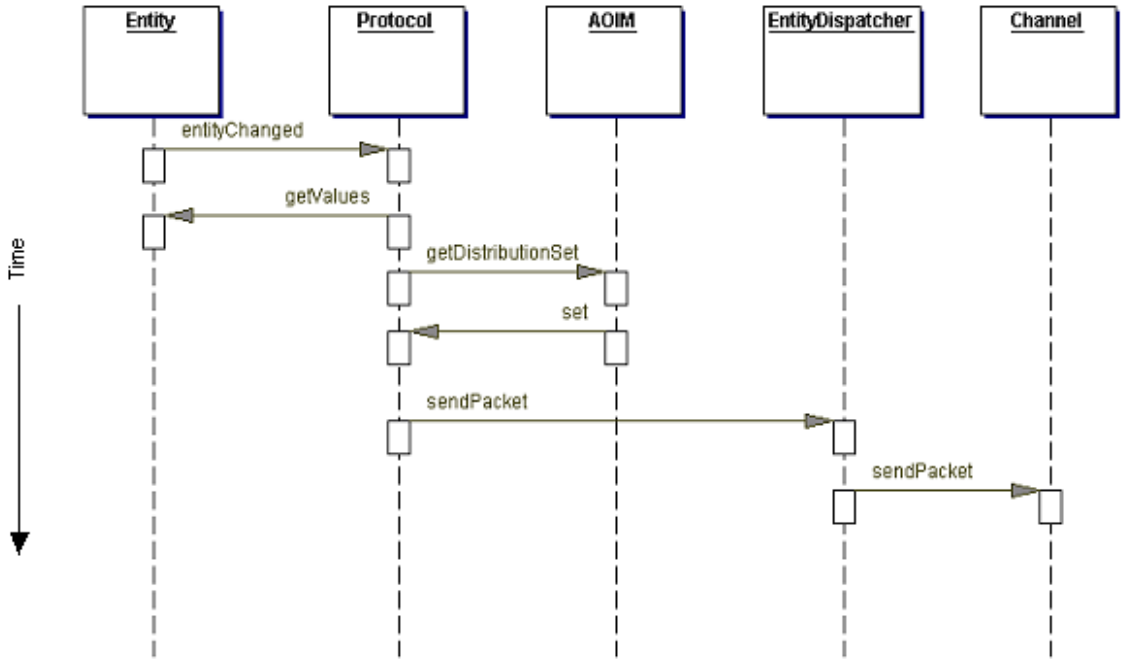

Figure 1. NPSNET-V packet distribution process

The following is an overview of the NPSNET-V packet distribution process shown in Figure 1. Whenever an entity in the world changes, it notifies one or more of its protocols. The protocol in turn queries the entity for the needed data regarding its state change. The protocol then asks the AOIM for a distribution set of multicast groups where the update data needs to be sent. Next, the AOIM ensures the entity has the correct data fidelity and data resolution settings. Finally, the AOIM calculates where the data needs to go, builds the appropriate distribution set, and returns

it to the protocol. The protocol then passes the distribution set, along with the data packet being sent, to the entity dispatcher, which in turn sends it on its way.

Figure 2 shows a high level breakdown of the AOIM functionality provided by the system developed in this thesis. Figure 12, later in the chapter, further expands this diagram, and provides more details of the AOIM process.

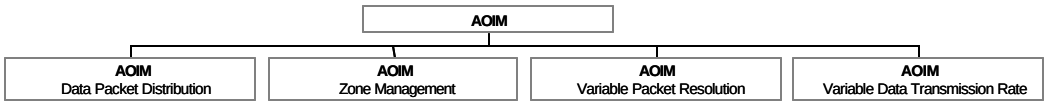

Figure 2. AOIM functionality overview

B. REQUIREMENTS

To be successful an AOIM must consume resources proportional to the number of entities with which it is actually interested. Bandwidth usage, packets-per-second, and network-related CPU usage should be independent of the total number of entities in the world (Abrams 1999).

C. DESIGN OVERVIEW

The AOIM designed in this thesis is a dynamic, object oriented, geographical region based, sender-side, peer-to-peer (mostly), interest manager that supports dynamic entity discovery. Every geographical region is represented by one or more multicast addresses. The AOIM also makes use of external tools specific to the NPSNET-V system, such as variable resolution protocols, and variable data transmission rate.

1. Dynamic geographical regions

There are many options available for subdividing a virtual world, ranging from simply dividing it in half, to more complex polygonal tessellations. The scheme implemented in this work utilizes octrees. This allows the system to collapse or expand any or all regions in the world, on the fly, in an extremely fast and efficient manner.

An octree, as the name implies, is a tree type data-structure, with one simple rule: every node in the tree has either zero or eight children. A node with no children is called a leaf node. An octree lends itself very well to the representation of three-dimensional space. Initially, the octree will consist of a single leaf node representing the entire world.

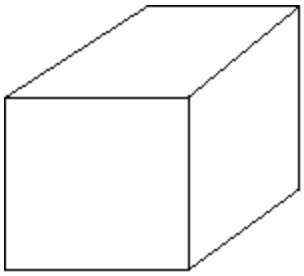

Figure 3. Octree leaf node

When the AOIM determines that the world has become too crowded, it can tell the octree to subdivide. The leaf node then breaks into eight new octrees, each of which is now a leaf node in the tree.

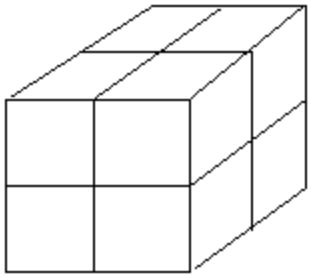

Figure 4. Octree after one subdivision

If the AOIM determines one of the new zones has become too crowded, it will simply tell that leaf node to subdivide as well. The resulting octree after two subdivides might look like Figure 5.

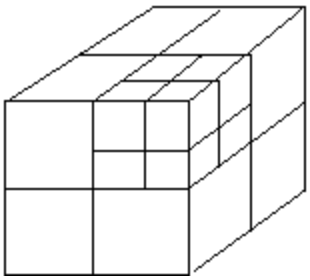

Figure 5. Octree after two subdivisions

In the event a subdivided node becomes less crowded after a time, the AOIM can just as easily collapse a node. This process lends itself very well to recursion, since each new leaf is a duplicate of its parent in form and function. Figures 3, 4, and 5 show the

geometric representation of the octree data. Figure 6 shows the underlying tree structures that the octree class uses for each of the geometric ones.

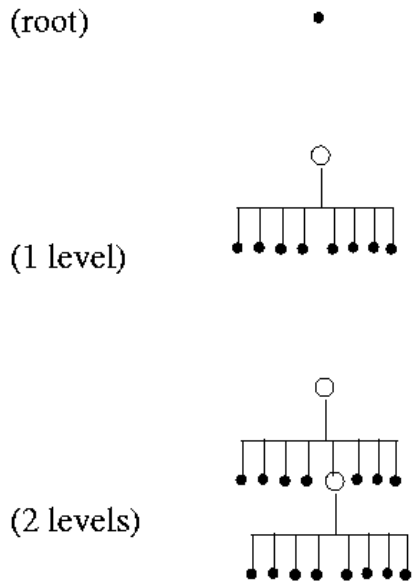

Figure 6. Graph representation of octrees from Figures 3, 4, and 5

Another benefit to using an octree is the speed with which searches are performed. Searching an octree is analogous to performing three binary tree searches for each level of the octree that needs searched, one for each of the three axes X, Y, and Z. For example, if an entity needs to send an update packet and queries the AOIM for where to send the information, the AOIM will perform an octree search based on the entity's position in the world. If the current state of the world is represented by the octree in Figure 7, and the entity is in the upper rightmost zone, then the search will need to go three levels deep to find the multicast address the entity should transmit on. The first pass of the search will determine that the root node is sub-divided. The second pass will determine that the appropriate child node has also been sub-divided, and the third pass will return the needed information from the correct leaf node three levels deep.

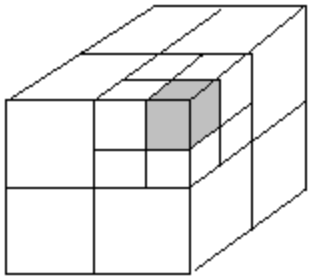

Figure 7. Zone containing transmitting entity

The way the search is conducted is as follows: The AOIM will call a method (`Octree.whichZone(x,y,z)`) in the root octree node, passing along the entity's position (x y z coordinates) in the world. The first thing this method does is check f the root node is a leaf; if so the search is complete. If the root node is not a leaf, it recursively calls an internal method (`Octree.getZone(x,y,z)`) that performs a search that returns the child node containing the entity. Then that child node has its own `Octree.whichZone(x,y,z)` method called. This process continues recursively until the correct leaf node is reached.

This search is very fast. If there are four levels of subdivision in the world, there could be up to 8^4 (4096) separate zones in existence. This search algorithm, however, would only need four queries to determine the correct leaf node. In the general case, the number of possible zones in an octree is 8^N where N is the number of levels in the tree, and only N queries are necessary to drill down to the correct node.

The best case search performance is when the tree is full. The performance of the search algorithm in this case is O(log(H)), where H is the height of the tree. The worst-case performance is when the tree only has one non-leaf node per level. In other words, at each level in the tree, there is only one node with children (see figure 6). In this case performance is O(H), where again, H is the height of the tree.

The octree implementation used in this AOIM only stores two pieces of information at each node: a String containing the name of the node, and a Boolean variable representing whether or not that particular node is a leaf. The node naming convention the octree follows is simple and has several advantages. The root node is always named "0". When the root node subdivides, each of its children will take that

name and append a single digit to it representing the child node's index in the array of children. For example after a single sub-division the root node will still be named "0", and it will have eight children named "00", "01", and so on through "07". If node "07" were to subdivide, then its name would stay the same and it would get eight children named "070", "071", through "077". There are many advantages to this scheme including automated name generation, and guaranteed unique names for each zone. In fact, just knowing the name of a zone tells you immediately where in the octree the zone is, which greatly aids in the debugging effort.

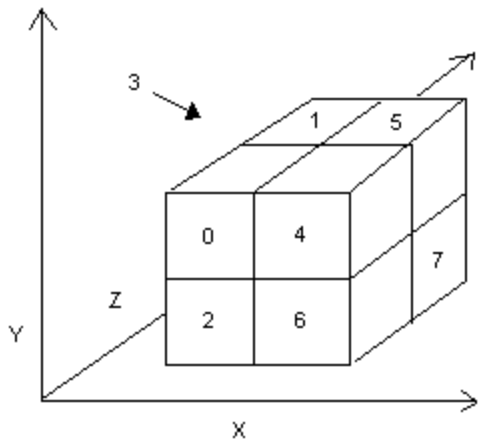

Figure 8. Zone numbering convention

The AOIM makes use of hashtables to store all multicast addresses and ports over which it may need to transmit. The key used by the hashtables is the name of the octree zone. The decision to store the multicast addresses and ports separately from each other and the octree was made to allow for various degrees of resolution. It is possible to have many different hashtables, all containing multicast addresses keyed on octree zone names. This allows the AOIM to use one octree and its associated naming scheme to send packets when and where it sees fit. One hashtable can be used for the highest level of resolution so that packets are only sent to the zone the entity actually resides in, and another can be used in a broadcast mode so that packets are sent everywhere. The octree remains unaware of these intricacies, only concerning itself with which zone any given entity resides.

2. Send-side filter paradigm

The decision to go with a send-side filtering scheme was driven by the design goal of NPSNET-V to be useable over a 56K modem connection, and the need for bandwidth conservation that goes along with that requirement. The AOIM attempts to send an entity's data packets only where those packets are actually needed; so only minimal receive-side filtering is necessary. To do this not only requires the AOIM to calculate where in the world every entity is when it is transmitting, but requires that it do it in real time as well. This requirement reinforces the choice of the octree/hashtable combination described in the previous section. Both data structures offer extremely fast searches even when they grow very large, allowing for virtually unlimited scalability, and both data structures grow and shrink quickly and easily, allowing for fast, on-the-fly, changes to the world.

3. AOIM with minimal centralization

The NPSNET architecture is mostly peer-to-peer and this causes some problems for a send-side AOIM. A few of the more difficult challenges are: maintaining world consistency among all the players in any given world, keeping track of players joining and leaving the world at runtime, and the infamous latecomer problem.

The latecomer problem occurs in peer-to-peer environments because of the lack of a central server with a well-known address. In true peer-to-peer systems, the state of the world is distributed across the machines currently in that world. As users leave the world, any world-state information they may have been maintaining shifts to other machines on the network, thus maintaining world consistency. The problem occurs when new users try to connect to the world. They need the entire state of the world, but there is no one place to get it, and just sending a general request for the needed information over the network can easily slow it to a crawl as all the current occupants of the virtual world respond to the request, flooding the network with redundant information. The more populated a virtual world becomes, or the more user turnover there is, the worse the problem becomes.

The solution employed in this thesis to meet these challenges was to build a minimal AOIM server. The AOIM server resides at a well-known network address, thus

solving the latecomer problem. It maintains the master copy of the current world configuration (octree and hashtables), thus solving the problem of maintaining consistency among the players. The AOIM server also registers players as they enter the world and de-registers them as they exit.

The registration process consists of simply storing the address of new users when they join the world. These addresses are stored in a Vector, which allows for both scalability and ease of use. When the AOIM server needs to pass something to the individual AOIMs, such as a command to expand or collapse a zone, it simply iterates through the Vector to do it. De-registration is simply the removal of the appropriate address from the Vector.

While these capabilities sound very much like client-server, and not at all like peer-to-peer, use of the client-server portion of the AOIM is kept to a minimum. The AOIM only utilizes the client-server portion of the system in three situations: when first connecting to the world, when departing the world, or when there is a configuration change to the world (these changes are infrequent). All data flow among entities remains exclusively peer-to-peer.

4. Implementation details

Each user in the world will have one AOIM running on their machine as part of NPSNET-V. When a user runs an NPSNET-V enabled program, one of the first things it does is to read an XML configuration file. This file contains, among other things, the type of AOIM to use, as well as the IP address of the corresponding AOIM server. The AOIM then establishes a TCP/IP connection with the server and registers itself. The AOIM server responds by logging the IP address of the new joiner, and sending back the current octree and all associated hashtables. The AOIM then spawns a new thread that waits for the AOIM server to connect to it via a second TCP/IP connection. This gives both the server and the AOIM the ability to initiate communications.

In a typical client-server setup the client will initiate communication, the server will hear the request, respond to the request, and then enter a wait loop waiting for the next client request. The problem with this is that the server cannot initiate

communication on its own. Having two different connections allows the server to push information to the client. This ability to push is critical to the AOIM's design.

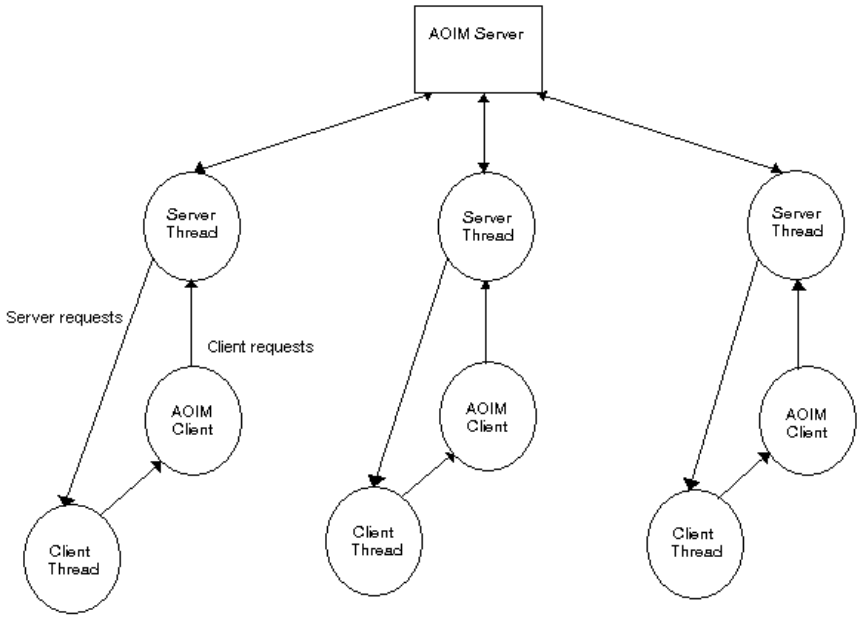

Figure 9. Client server architecture

Once everything just described has taken place no further interaction between the AOIM and the server will occur until the configuration of the world changes, or the AOIM program terminates. Again, all data communication among entities is truly peer-to-peer. In fact, if the AOIM server crashes, the impact is minimal on the world at large. The configuration of the world (the number and arrangement of zones, to be specific) is frozen in its current state, and no new players are able to join the world until the server is brought back online. However, all the entities already in the world will continue to communicate as before, and there will be no indication of a problem to any players already in the world.

When one or more AOIMs decide a zone needs to expand or collapse, they send a request to the server. The server responds by locking out all other requests for the same zone change (additional requests are simply discarded). If the request is for a zone expansion, the server checks to make sure it has eight new multicast addresses available. If it does, it expands the appropriate zone, loads the new multicast addresses and ports

into the appropriate hashtable, and then pushes a message to every registered AOIM telling them to expand the zone. The AOIMs respond by requesting the new hashtables and expanding their respective octrees. In a collapse request the server pushes a message, and the clients collapse the appropriate zone. No further action is necessary for a collapse. The hashtables maintained by each AOIM will still contain multicast addresses keyed on zone names that no longer exist, but this causes no problems. The AOIM's hashtables are only updated after an expansion. The trade off here is that by continuing to store some unneeded information, the AOIM can avoid downloading a copy of the hashtables every time a zone collapses. Again, this is all driven by the desire to conserve bandwidth.

5. Tools provided by NPSNET-V

The two external tools available to the AOIM to help conserve bandwidth are variable entity data transmission rate and variable resolution protocols.

Variable entity data transmission rate or, data fidelity as it is sometimes called, is simple to understand and challenging to implement. The idea is to send exactly the number of data packets needed to allow every entity ghost to closely or exactly represent their respective entity master. The way this works is as follows:

All NPSNET-V Entities implement the `NetworkTunable` interface which provides two methods: `increaseFidelity()` and `decreaseFidelity()`. These methods work just like a volume knob on a radio. Every time `increaseFidelity()` is called, the amount of time that passes between each heartbeat will decrease, until eventually the entity will send packets at frame rate. This is clearly the highest fidelity possible. Conversely, each time `decreaseFidelity()` is called, it will increase the time between heartbeat packets and the ghost will have to rely more and more on dead reckoning. This is obviously a very powerful tool the AOIM can use in the fight to conserve bandwidth, and the key to implementing it is dead reckoning.

The idea of dead reckoning is as follows: given an entity's last known position, course, speed, and the elapsed time since the last known position was reported, the entity's current position can be estimated fairly accurately. For example, if an entity at position (0, 0, 0) heading straight up the Y axis at a speed of 1 unit per second sends out a

data packet, and never changes course or speed again, it will never have to send another packet (Figure 10). This is because every time a ghost entity needs to update its position, and there are no new data packets available, it can just calculate where it should be.

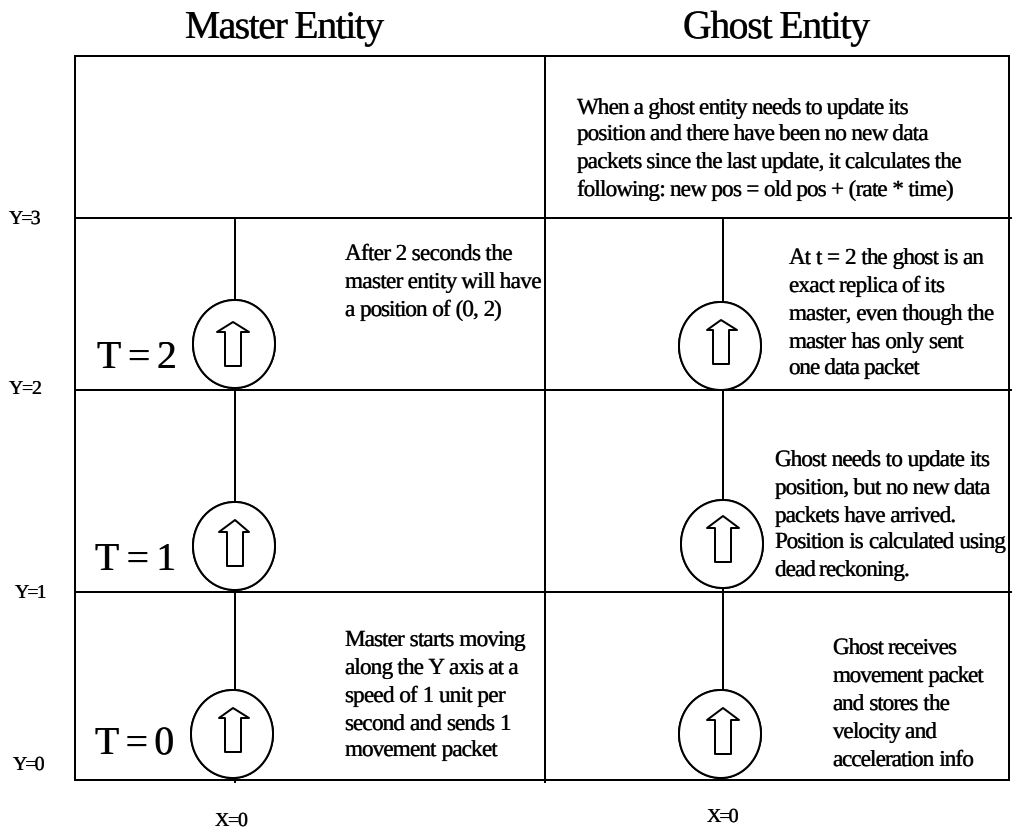

Figure 10. Example of dead reckoning

A problem arises however if an entity master changes speed or direction, sends a new position packet and one or more ghost entities do not receive that packet. In this case, the ghost will not be an accurate representation of its master. NPSNET-V employs a two-part solution to this problem (Figure 11). The first part is to have each master send a periodic position packet (heartbeat) whether it has changed course and speed or not. The second part is on the ghost side. When a ghost receives a position packet from its master, and the ghost's position does not match that of its master, it will employ a smoothing algorithm to gradually move to the correct position instead of just jumping directly there. This is more for aesthetics, or the user experience, than anything else. The rule NPSNET-V ghosts employ is to move from their current position to the correct one

in 60 frames (approximately 2 seconds). In this way, a person watching a ghost will not see a sudden, discontinuous jump.

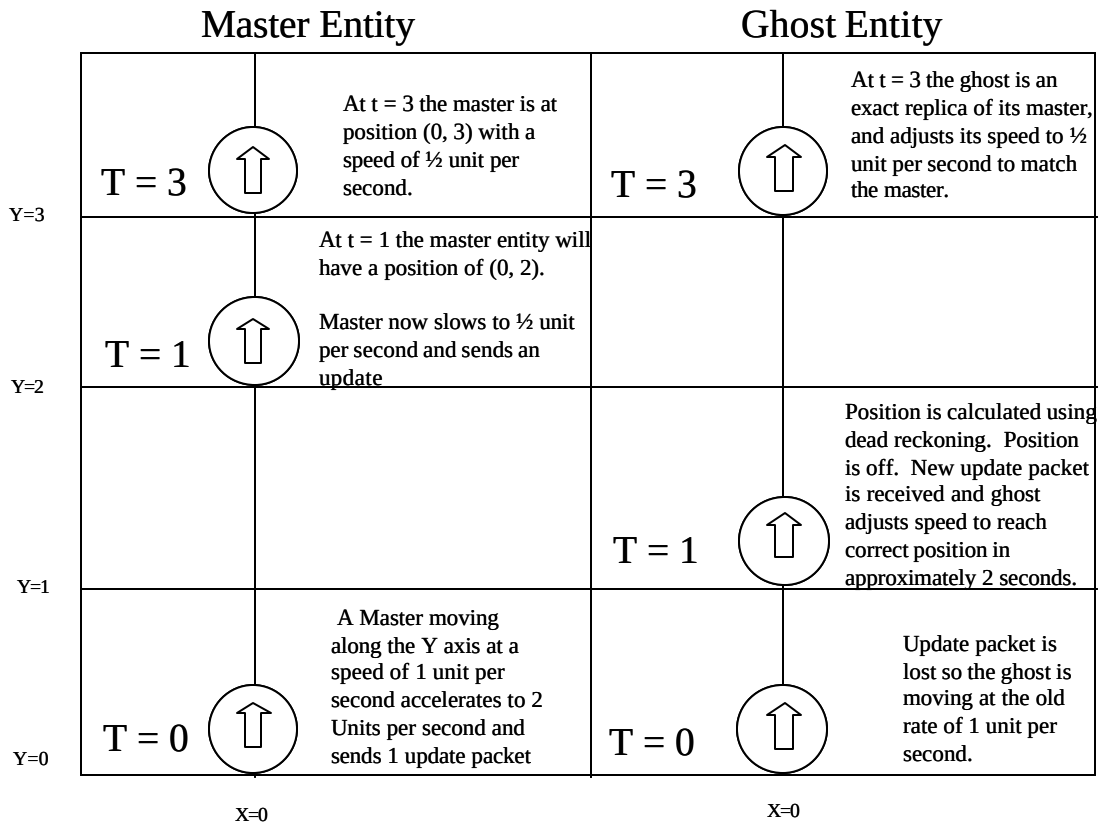

Figure 11. Example of dead reckoning with smoothing

The second external tool available to the AOIM is variable resolution protocols. NPSNET-V provides a component class called EntityDispatcher. This class has a method called `setResolution()`. This method accepts a Byte as its only parameter. This method functions a little differently than the methods discussed in the preceding paragraphs. It is analogous to a gearshift in a car. Passing 1, as a parameter to the method, shifts all protocols to a low resolution, and passing a 2 will shift them into high-resolution mode. In effect, this method simply changes the precision of the data packets that are being sent. In high resolution mode, all the information transmitted in the data packets will be double precision. In low resolution mode the information will change to single-precision floating point from double-precision, or from a long, to an integer representation as appropriate. The trade off here is bandwidth for accuracy. In low-fidelity mode, data packets will be much smaller, but ghosts will be less accurate

approximations of their masters. Any entities that attempt to interact with these ghosts will always be a little off.

6. Dynamic entity discovery

One of NPSNET-V's key features is that it supports dynamic entity discovery. In other words, a user can create a new entity from scratch, and enter it into a virtual world populated with users that have no pre-existing knowledge of this new entity. The new entity will look and behave exactly as it should on everyone's machine, even though it has never before existed.

What enables NPSNET-V and the AOIM to support dynamic entity discovery is the Java programming language and a capability it provides called reflection. This feature allows a program to query an Object, at run time, about its name, available methods, and implemented interfaces. The AOIM makes extensive use of this capability. When AOIM changes the data fidelity setting, it uses the `instanceof` operator (Holzner 2000) to check if each entity supports dynamic data fidelity. If the entity supports it then the appropriate change is made, if it does not, then nothing happens, and no exception is thrown. Without reflection the AOIM would not have been able to fully support dynamic entity discovery.

D. SUMMARY

The area of interest management functionality imbedded in this system is summarized in Figure 12.

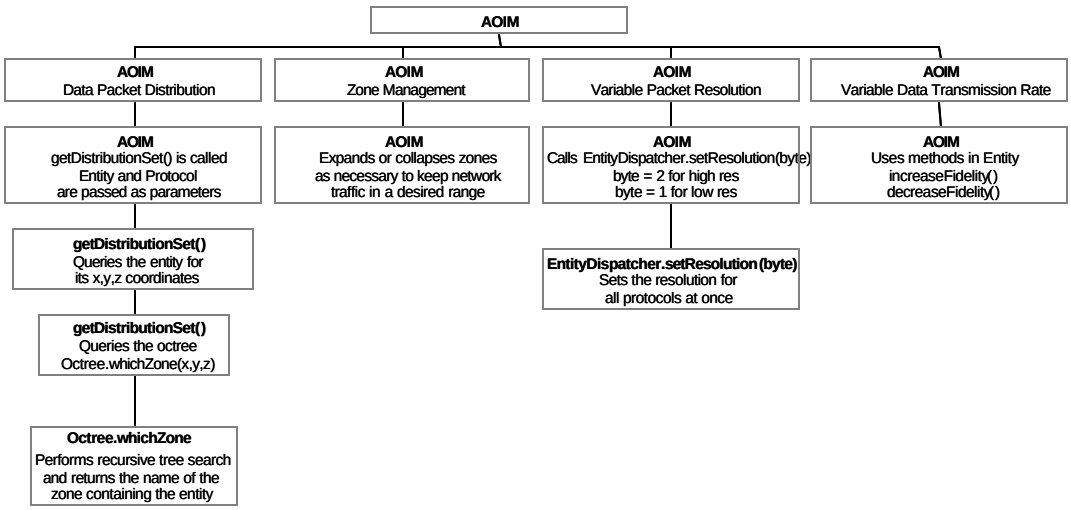

Figure 12. AOIM process and functionality

The AOIM described in this thesis can adapt to a constantly changing environment, at runtime, to maintain network bandwidth usage within acceptable parameters. It performs this task in real time with no adverse effect on system performance as a whole. The AOIM does this by employing highly efficient data structures and fast search algorithms that allow for rapid size changes and even faster queries. The AOIM allows NPSNET-V to make extensive use of the peer-to-peer paradigm, and while there is a client-server component to the system, its use is kept to an absolute minimum.

The AOIM also makes extensive use of external tools provided as part of the NPSNET-V architecture: variable resolution data packets, and variable data transmission rate (fidelity).

Every tool available to the AOIM is useable at runtime to change how the system uses network bandwidth. The changes these tools make occur seamlessly, with the users in the world blissfully unaware they have even occurred.

IV. PERFORMANCE ANALYSIS

A. INTRODUCTION

This chapter describes the performance and reliability testing to which the AOIM was subjected. It describes in detail, the test environment and parameters used. Finally, the results are analyzed and specific recommendations are made, that suggest how the AOIM should respond at runtime to changing network conditions.

B. TEST OVERVIEW

There are three distinct groups of tests performed in this chapter. The first group is designed to isolate and test the performance of the individual components of the AOIM. Performing these tests will allow each component to show their individual strengths, working alone, and in conjunction with the other components of the AOIM. This will be very helpful in designing the control algorithm that will drive the AOIM at run time (see future work section in Chapter V). It is expected that each component, working alone, will cause a significant reduction in total bandwidth usage. It is further expected that using the components in concert will show additional reductions in bandwidth usage over the individual results.

The second group of tests is designed to probe the full range of performance of the data fidelity component of the AOIM in a fixed world configuration. The purpose of these tests is to determine if bytes per second (BPS) decreases in a linear fashion as data fidelity decreases (all other things held constant), or if there is an obvious point of diminishing returns where further reductions in fidelity will no longer have a significant effect. This will again be useful in writing the AOIM controller module. It is expected that each time data fidelity is set to a lower value there will be a corresponding drop in bandwidth usage.

The third test group is designed to compare the scalability of the AOIM to a comparable world with no AOIM. This test will aid in determining the upper limit of the number of entities that can be in the world at once before bandwidth usage exceeds network capacity.

In every test, data was collected from one designated computer. The computer used for data collection had only one entity master created on it for each individual test. Every other entity on the test machine was an entity ghost representing an entity master on one of the population machines.

All other computers on the network were populated with multiple entity masters to simulate real world network interaction. This setup, with several machines, each having many entities, very closely approximates a real world scenario in which there are many machines, each with a single entity. This would correspond to a virtual world in which each user on the network controlled one entity on one computer. The main difference is that a single machine with many entities (40 entities for example) will use more CPU cycles for entity administration, and therefore will send data packets at a lower rate, than would 40 computers each with a single entity.

In an ideal situation, these tests would have been run on a network containing hundreds of machines, with only one entity master per machine. However, due to limited laboratory resources this compromise was used, and for showing the AOIM's ability to reduce network traffic significantly, the compromise worked well.

C. TEST CONFIGURATION

The AOIM performance tests were conducted in the Naval Postgraduate School Virtual Reality Lab. All tests were conducted during periods of low network traffic to avoid large variations in testing which could have occurred due to uncontrolled traffic. The network in this lab consists of the following equipment:

- 100 Mbps switched fast Ethernet LAN
- Dell Dimension 4100 computers
    - 1 GHz Pentium III's
    - 512 MB System memory
    - NVIDIA GeForce II video cards with 64 MB of RAM
    - 3Com EtherLink XL 10/100 Ethernet cards
    - Windows 2000 Professional operating system

D. TEST GROUP ONE: INDIVIDUAL COMPONENT TEST

1. Test description and parameters

In the first test group, three different zone and population configurations, depicted in Table 1, were tested.

Config	Number of Entity Masters Loaded		Interest Management Zones
	Test Machine	Population Machines (total)	
1	1	15	1
2	1	15	8
3	1	30	8

Table 1. Zone and population configurations

For each of the three test configurations, several parameter settings were used (one change per run) and a running tally of BPS received by the test machine was recorded. After running for one minute, the statistical package used contained an accurate one-minute running tally of BPS received by the test machine. This tally was stored at ten second intervals and after thirty samples were gathered, the mean and standard deviation, for BPS received by the test machine, were logged.

The parameters that changed for each new run were: packet resolution and entity data fidelity. The number of entities in the world, and the zone configurations were held constant in each separate test. Table 2 depicts the parameter combinations used.

Setting Number	Parameter	
	Packet Resolution	**Data Fidelity**
1	High	High (1)
2	High	Medium (20)
3	High	Low (40)
4	Low	High (1)
5	Low	Medium (20)
6	Low	Low (40)

Table 2. Parameter combinations used during data collection

As explained previously in Chapter III, data fidelity can range from 1, which is the highest possible fidelity, to 40, the lowest possible fidelity. For these tests, however, only three settings were used: High (1), Medium (20), and Low (40). Again, fidelity affects packets per second, while resolution is float vs. double etc.

2. Test results for test group one

The results from test one, displayed in Table 3, show that of the two variable parameters, data fidelity has a much bigger impact on BPS than packet resolution does. On average, changing from high fidelity to medium fidelity resulted in a 74% reduction in mean BPS received at the test machine. Switching from medium fidelity to low fidelity resulted in a further 17% average reduction of mean BPS.

Packet Resolution	Data Fidelity	Mean BPS	Standard Deviation	Sample Size
High	High	198,046	3701	30
High	Medium	49,580	862	30
High	Low	41,897	2145	30
Low	High	182,839	1098	30
Low	Medium	47,658	711	30
Low	Low	38,904	1049	30

Table 3. Test results for test configuration one

Visually, there was practically no difference in the behavior of the ghost entities viewed on the test machine as data fidelity dropped from high to medium. When fidelity dropped from medium to low, there was a slight degradation in smoothness of the ghost entities on the test machine. Despite this drop in quality, which was noticeable, the system remained visually appealing. In fact, without having first seen the medium fidelity mode immediately before the low fidelity mode, it is doubtful a user would have known that significant dead reckoning was taking place.

Next, with data fidelity held constant, packet resolution was varied with an average 6% reduction in BPS received at the test machine. There was no visual change in the performance of the ghost entities on the test machine. This result may or may not be significant. Because fish entities were used in all tests conducted on this AOIM, and since fish do not maintain a constant speed, heading, or position, packet resolution probably is not as critical as it would be in other applications requiring more precision. In other words the results were visually plausible without necessarily being accurate.

The results of the test using configuration two are depicted in Table 4.

Packet Resolution	Data Fidelity	Mean BPS	Standard Deviation	Sample Size
High	High	22,114	3,013	30
High	Medium	12,959	7,081	30
High	Low	3,574	978	30
Low	High	15,709	4,124	30
Low	Medium	8,291	5,814	30
Low	Low	4,777	1,592	30

Table 4. Test results for test configuration two

The results of this test, when compared to the previous test, were just as conclusive. Just simply breaking the world into eight zones of interest, and holding everything else constant, resulted in an 89% reduction in BPS received at the test computer versus no interest management at all. Comparing all like parameter settings between configuration one and configuration two shows that breaking the world into eight interest management zones, results in an across the board reduction of BPS at the test computer of 86%.

The data fidelity results of this test were different from the results of the configuration one test where most savings occurred by switching from high to medium fidelity. In the configuration two test, switching from high fidelity to medium resulted in a 44% reduction, and dropping from medium to low fidelity caused a further 57% drop in BPS received at the test computer.

Changing packet resolution from high to low achieved an average BPS reduction of 10% at the test computer.

The results of the test using configuration three (Table 1) are depicted in Table 5.

Packet Resolution	Data Fidelity	Mean BPS	Standard Deviation	Sample Size
High	High	40469	13991	30
High	Medium	6190	2864	30
High	Low	4952	2435	30
Low	High	24934	8404	30
Low	Medium	4520	2165	30
Low	Low	4514	1156	30

Table 5. Test results for test configuration three

The parameters for the last test in this series are the same as in test two except that the number of entities in the world has been doubled. Again, breaking the world into eight geographic zones proved effective. There was an 80% drop in BPS received at the test computer, as compared to the test with half the entities, but no interest management. When all test settings are compared between configuration one and configuration three, the result is an average reduction of 87% of mean BPS received at the test computer. In other words, despite the fact that there were twice as many entities in the world, the test computer saw an 87% reduction in BPS received.

The data fidelity results were similar to those from test one. Switching from high to medium fidelity resulted in an 83% drop in mean BPS, while dropping from medium to low only caused another 10% drop.

Packet resolution proved more effective in this test than it did in any other. Dropping from high to low resolution resulted in an average 25% drop in mean BPS. Again, this savings in bandwidth resulted in no noticeable degradation in entity visual appeal.

E. TEST GROUP TWO: DATA FIDELITY PERFORMANCE TEST

1. Test description and parameters

In this test, which used only configuration one (Table 1), only the high packet resolution setting was used, and data fidelity varied from 1 to 40, one step at a time. At each setting, the statistical package was reset, and as before, BPS was recorded in ten-second intervals. Five samples per setting were collected.

2. Test results for test group two

The results of this test, shown in Figure 13, are interesting. As expected BPS was highest when data fidelity was set to 1. As data fidelity was decreased form 1 to 8, BPS dropped exponentially. As data fidelity was decreased further from 8 to 19, the decrease in BPS became mostly linear and much less pronounced. As data fidelity decreased from 19 to 40, there was no appreciable drop in BPS received at the test computer.

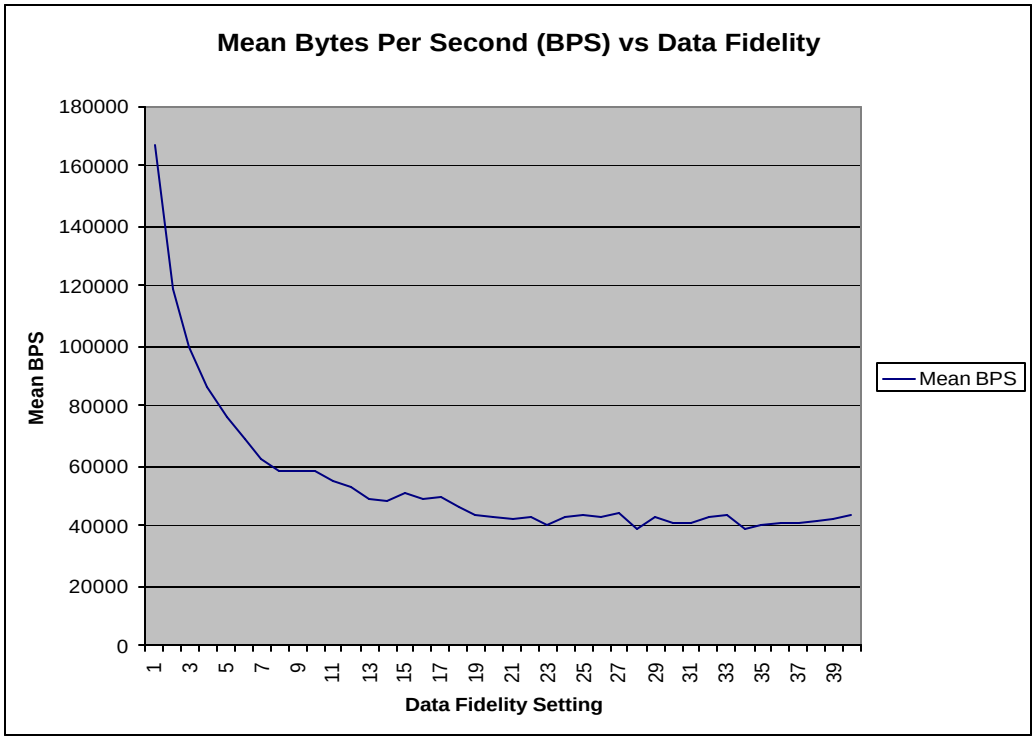

Figure 13. Mean bytes per second vs. Data fidelity

F. TEST GROUP THREE: SCALABILITY TEST

1. Test description and parameters

The last test performed examined the BPS received in configuration one (Table 1) and setting one (Table 2), which is effectively no interest management at all. Interest management was then enabled, and more and more entities were added to the world (population machines only). The statistics package was reset every time the number of entities was increased, and average BPS was recorded at the test machine. The goal of this test was to find out how many entities could be added to the world as a whole, using aggressive area of interest management techniques, before the BPS received at the test machine exceeded that of a system with no AOIM and 16 total entities in the world.

2. Test results for test group three

Table 3 shows that with no interest management techniques applied, and 16 entities in the world, the test computer received an average of 198,046 BPS. These baseline values are what test three's results will be compared to.

The test was begun by inserting 85 new entities into the world, bringing the total number of entities to 101. All of the interest management tools previously described were enabled, and the stats package was given time to stabilize. The results were amazing! The mean BPS received at the test computer dropped to 19,385! With the number of entities in the world increased by over 600%, the AOIM was able to decrease network traffic at the test computer by over 90%.

Though the initial goal was to drive the mean BPS back towards 200,000, and record the number of entities in the world required to do this, it quickly became apparent that it probably was not possible to insert that many entities into the world with the limited number of machines available. Given the results attained, this was a moot point anyway. These results show that the AOIM system can easily scale to allow a very large number of users.

G. CONCLUSIONS

The AOIM met or exceeded expectations in every test performed. The only surprising result was the fact that reducing data fidelity below 20 had no discernible effect. One explanation for this is that as data fidelity is set lower, the percentage of full

state packets being sent versus the number of simple steering commands being sent, goes down. Eventually there are so few, full state packets being sent, that they make up an insignificant number of the total packets being sent. Therefore, dropping the number even lower has no effect. It becomes analogous to removing a single white grain of sand from a large pile. There is definitely a point of diminishing returns.

It should be noted that variance and standard deviation were much higher for some of the tests than for others. The reason for this is the type of entities used in the tests, namely fish, are all separate autonomous Agents, each with their own goals and characteristics. In addition, they die, and reproduce utilizing a genetic algorithm that produces new fish with characteristics of both its parents. Their goals can change at run time, they can school (or not), and they can prey and be preyed upon. The end result of this is a large amount of geographic and temporal cohesion. In other words, a fish is likely to be in the same area it is currently in, doing the same thing it is currently doing, in the near future. So, for some of the tests, the fish may have been schooling in one area, and not sending many update packets over the network, and during other tests a shark may have been attacking the school causing the fish to scatter for their lives and in turn causing them to send a large amount of update packets. Some of the tests include data samples from both situations resulting in high variance.

To test how the AOIM would perform in the real world, the decision was made to perform these tests with the entities that actually populate the world, instead of writing new simple ones that just sent data packets at a constant rate. This decision is the reason for the high variance at times, but it is a much more realistic test of the AOIM's capabilities.

H. RECOMMENDATIONS

As of now there is no mechanism in place for the AOIM to autonomously change the various settings tested here (see future work section in Chapter V). Based on the test results just described, however, the following is suggested as a rudimentary algorithm to implement autonomous behavior into the AOIM:

- When BPS exceeds the upper threshold, decrease data fidelity one step at a time.
- When data fidelity reaches 8, and the threshold is still exceeded, set packet resolution to low.
- When the BPS threshold is exceeded once again, continue decreasing data fidelity one step at a time until it reaches 20.
- If the threshold is once again exceeded, then simultaneously subdivide the world into eight new zones, reset packet resolution to high, and reset data fidelity to 1.
- Continue performing these same steps in this order as necessary. If network traffic falls back below the low threshold, simply reverse the steps taken, one step at a time.

The goal of this algorithm is to maintain bandwidth usage at acceptable levels while at the same time maintaining visual appeal. Data fidelity and zone splitting both offer large bandwidth savings, but data fidelity has a much smaller impact on visual appeal, so zone splitting is used as a last resort.

I. SUMMARY

These performance tests have shown conclusively that the AOIM developed in this thesis is able to produce dramatic savings in network bandwidth usage in a peer-to-peer system, in real time, while at the same time maintaining system performance, and visual appeal. The two tools with the most dramatic savings potential are data fidelity, and the ability to break the world into smaller zones of interest. Variable packet resolution, also offers significant, albeit less dramatic, savings. The real power in the AOIM comes from

being able to use these tools in combination to provide additional savings over using the tools separately.

V. CONCLUSIONS AND FUTURE WORK

A. INTRODUCTION

This thesis has demonstrated that it is possible to integrate an area of interest management scheme into a peer-to-peer virtual world supporting dynamic entity discovery. In addition, the support of dynamic class and protocol behavior control mechanisms provides greater scalability than a standard interest management scheme.

Several innovations and techniques made this scalable interest management scheme successful. Using a combination of octrees and hash tables with their associated fast searches enables the system to grow quite large with no noticeable performance hit. An independent multi-layered approach allows the AOIM to constantly adapt to changing network conditions, by using each of its tools individually or in concert. Finally, the use of reflection allows the system to fully support dynamic entity discovery, a critical design goal of NPSNET-V.

B. OCTREES

Scaling interest management to a very large number of users required careful consideration of which data structures to use. There are two main reasons why the octree was the data structure of choice: speed and adaptability. The octree allows for fast changes to the configuration of the world. Zone collapses and zone expansions are easy to perform at runtime, with no noticeable drop in system performance or visual appeal. As for adaptability, octrees are a near perfect choice for representing three-dimensional space.

Another benefit is the low memory requirement of the octree. Since the only information actually stored in each octree node is a String (zone name), and a Boolean (leaf node flag), the octree itself remains quite small, even as the world itself grows very large. This minimizes the delay a new player joining the world will experience while waiting for the octree to transfer over the network.

Where the octree really shines, however, is when it performs a search. The search algorithm employed is effectively a three dimensional, binary tree search, with one query for each axis. Where a binary search can eliminate up to 50% of the possible solutions

with each query, this algorithm can eliminate up to 87.5% of the possible solutions per query (when the tree is full). This allows for extremely large worlds.

As discussed in Chapter III, the worst case search for octrees is O(h), where h is the height of the tree. Assuming a circumference for the earth of 25,000 miles, and a diameter of about 8,000 miles, an octree height of about 18-19, would result in each leaf node being about 100 yards long per side; this assumes that the original octree cube enclosed the entire earth. At an octree height of about 25, this would reduce further to about a 1 yd cube. This means that an O(h) search for even the largest environment is bounded by a reasonably sized h.

Of course, just because the search is reasonable in theory, does not mean a world this large can actually be implemented. A completely full octree with a height of 25 would contain 8^{25} nodes, and would subsequently require 8^{25} multicast addresses. This is a tall order as 8^{25} = 37,778,931,862,957,200,000,000. However, putting a sphere (the Earth) into a cube (initial octree) results in much wasted space, and the areas of the cube that do not contain any of the sphere could be culled. The result would be a vast reduction in the number of octree nodes actually required.

C. MULTI-LAYERED APPROACH

Another major contributor to this system's success is the use of multiple independent layers. Chapter IV showed the effectiveness of the individual parts of the system, but the AOIM becomes even more powerful by being able to use the parts in combination with each other.

For instance, using the Fish World example, if there were fifty fish in the world, and the mean bytes-per-second received at the test computer became too high, one way to deal with the problem would be to subdivide the world. However, if all fifty fish ended up in a zone together after the subdivision, then the net result would be that the bytes-per-second would still be too high. A standard, geographic based AOIM would stumble here, but the system developed in this thesis has additional weapons it can bring to bear. For example, the AOIM could respond further by reducing data fidelity, or packet resolution, while at the same time collapsing the eight new zones just created, since the subdivision tactic was not effective in this case.

The ability of this AOIM to respond to a highly dynamic environment, while still allowing true peer-to-peer communication among the entities in the world, makes it unique.

D. DYNAMIC ENTITY DISCOVERY AND REFLECTION

One of NPSNET-V's key features is that it supports dynamic entity discovery. This ability to accommodate, previously unknown entities, seamlessly at runtime is a highly desirable feature that provides some interesting development challenges. Foremost among these is how to make full use of an entity's capabilities without first knowing what those capabilities are. By utilizing the Java programming language and one of its capabilities known as reflection, these challenges were overcome. Reflection allows runtime queries of objects to determine their name, what methods they have, and what interfaces they implement. This feature allows the AOIM to query entities at runtime to ensure that they implement the `NetworkTunable` interface (necessary for the data fidelity feature to work properly). This lets the AOIM deal cleanly with entities whether they implement `NetworkTunable` or not. Without reflection, the AOIM would not have been able to fully support dynamic entity discovery.

E. FUTURE WORK

It has been said that good research asks more questions than it answers. This thesis is no exception. While this AOIM performs well, many improvements could be made to make it work even better. Some of the possible additions to this system that would greatly improve its performance are: a multi-agent monitoring system, a look-ahead algorithm, and support for multiple octrees. Lastly, due to NPSNET-V's component architecture, an entirely new, non-geographic AOIM could be developed and inserted into the system.

1. Multi-agent monitoring system

As previously mentioned, there is currently no mechanism in place to autonomously change AOIM parameters at run time. All parameters are currently manually controlled. Chapter IV suggested a rudimentary algorithm, based on testing, that should provide acceptable performance, and that could be quickly inserted into the AOIM. However, to fully realize the potential of this system, a multi-agent control

mechanism could/should be developed, with the sole purpose of keeping the AOIM optimally configured for current network conditions.

The AOIM developed in this thesis has the ability to use all of its tools, at runtime, in any conceivable combination. These tools are effectively switches that can be flipped, and knobs that can be turned. No "if-then-else" algorithm could possibly hope to adapt to every possible network condition. The reason for this is simple; for an "if-then-else" algorithm to work, the person that writes it needs to allow for every possible network condition (at compile time), and that is just not possible. A multi-agent system however, with its ability to learn and adapt to changing conditions, would be a prime candidate to flip the switches and twiddle the knobs.

2. Look-ahead algorithm

The AOIM, as currently implemented, only sends data to the zone that the sending entity currently occupies. This solution usually works well, but a problem develops when an entity is moving along a border between zones. As the entity enters the new zone, it begins sending and receiving packets accordingly, but if it quickly turns back into the old zone, and then back into the new zone once again etc, it ends up sending half of its packets to one zone and half to the other. This can cause anomalies such as pop-up.

Pop-up occurs as an entity approaches a new geographic zone. Since the entity is not yet in that zone, it is not receiving any data packets from that zone, and in turn, there will be no other entities visible. The visible portion of the new zone will appear empty. However, as soon as the entity crosses the boundary and begins receiving new data packets, all of the entities already in the zone will magically appear out of nowhere; they just "pop-up".

The fix for this problem is to use a look-ahead algorithm. If an entity gets within visual range of another geographic zone, simply subscribe to that multicast address as well. The entities will still pop-up in the world, but this may now occur beyond visual range of the entity in question, thus improving visual appeal. The drawback to this is the extra, potentially unnecessary, network traffic that will be generated. However, just like all the other tools the AOIM has, it can choose not to use them if necessary. If the

network is being crushed with traffic, the AOIM can decide to allow pop-ups to occur, thus saving bandwidth at the expense of visual appeal; this is just another switch the AOIM can flip.

3. Multiple octrees

One additional feature that could be added to this system to improve functionality is support for multiple octrees. The benefit of multiple octrees is the ability to allow various levels of resolution simultaneously. For example, one octree might be broken down into many small zones of interest, which in the Fish World example would make sense if the water in the aquarium were very cloudy, and visibility very low. Now suppose that an entity enters the world that does not rely on visibility at all, but on something else such as sonar. This new entity can see across multiple zones (maybe all of them at once) and, allowing a different octree to maintain its area of interest would be an elegant solution to the problem. The octrees themselves are so small, and the octree searches so fast, that many different octrees could be in use at once without causing a hit on system performance.

4. Entirely new AOIM

While a geographic AOIM is a good choice for many applications, there are situations where some other type of AOIM might be a better choice. NPSNET-V's component architecture makes switching from one AOIM to another simple. This allows multiple AOIMs to be developed, each one completely different from the others, and the decision of which one to use to be made at runtime. Two possible types of AOIMs that could be developed are functional and temporal.

A functional AOIM is one in which entity type, not location, determines what packets are received. An example of this would be a naval simulation in which ship entities receive packets from other ship entities, and on each ship, the individual sailors receive packets from other sailor entities on the same ship. However, ship entities never receive sailor entity packets. This type of system could work underneath the geographic AOIM to act as an additional packet filter.

A temporal AOIM is a little harder to understand. An example of this type of system could be a world in which there are multiple time scales. The world itself could

be on a geological time scale for things such as seasonal changes, erosion, movement of continents etc. The entities in the world could be on a time scale in accordance with the human perception of how time passes. Lastly, there could be another time scale, call it 'Bullet Time', where things happen in nanoseconds or picoseconds. An example of this is a bomb exploding. As far as the explosion is concerned, everything else in the world is moving so slowly as to really be considered stationary.

F. CONCLUSION

The main goal of this thesis was to show that it is possible to integrate an area of interest management scheme into a peer-to-peer virtual world that supports dynamic entity discovery. A further goal was to show that offering dynamic class and protocol behavior control mechanisms allows improved scalability over a standard interest management scheme. The AOIM developed in this thesis met these goals.

APPENDIX A

A. INTRODUCTION

It should be noted that the AOIM developed in this thesis is not a standalone piece of software. It is tightly integrated into the NPSNET-V system. The complete NPSNET-V system, including all source code and documentation, is freely available in a CVS repository located at:

sse.cs.nps.navy.mil

login = anoncvs

password = Panda3D! (Please note the case)

NPSNET-V is still very much a work in progress, and outside contributors to this project would be greatly welcomed. For more information about the system or for contact information, please point your web browser to http://www.npsnet.org/~npsnet/v/index.html.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

A. INTRODUCTION

This appendix provides UML diagrams representing the entire AOIM system.

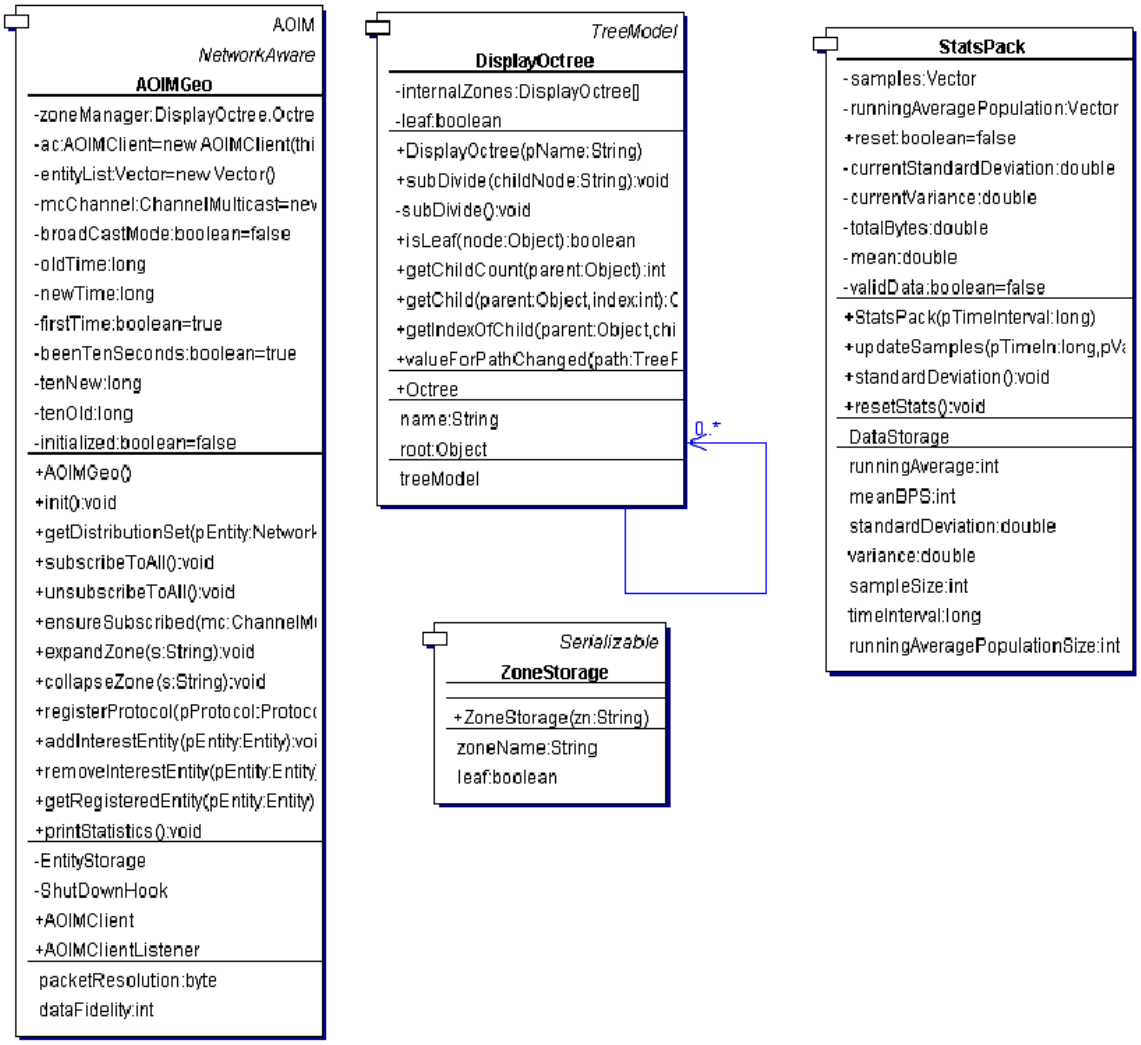

Figure 14. UML diagram of the AOIM system (part 1)

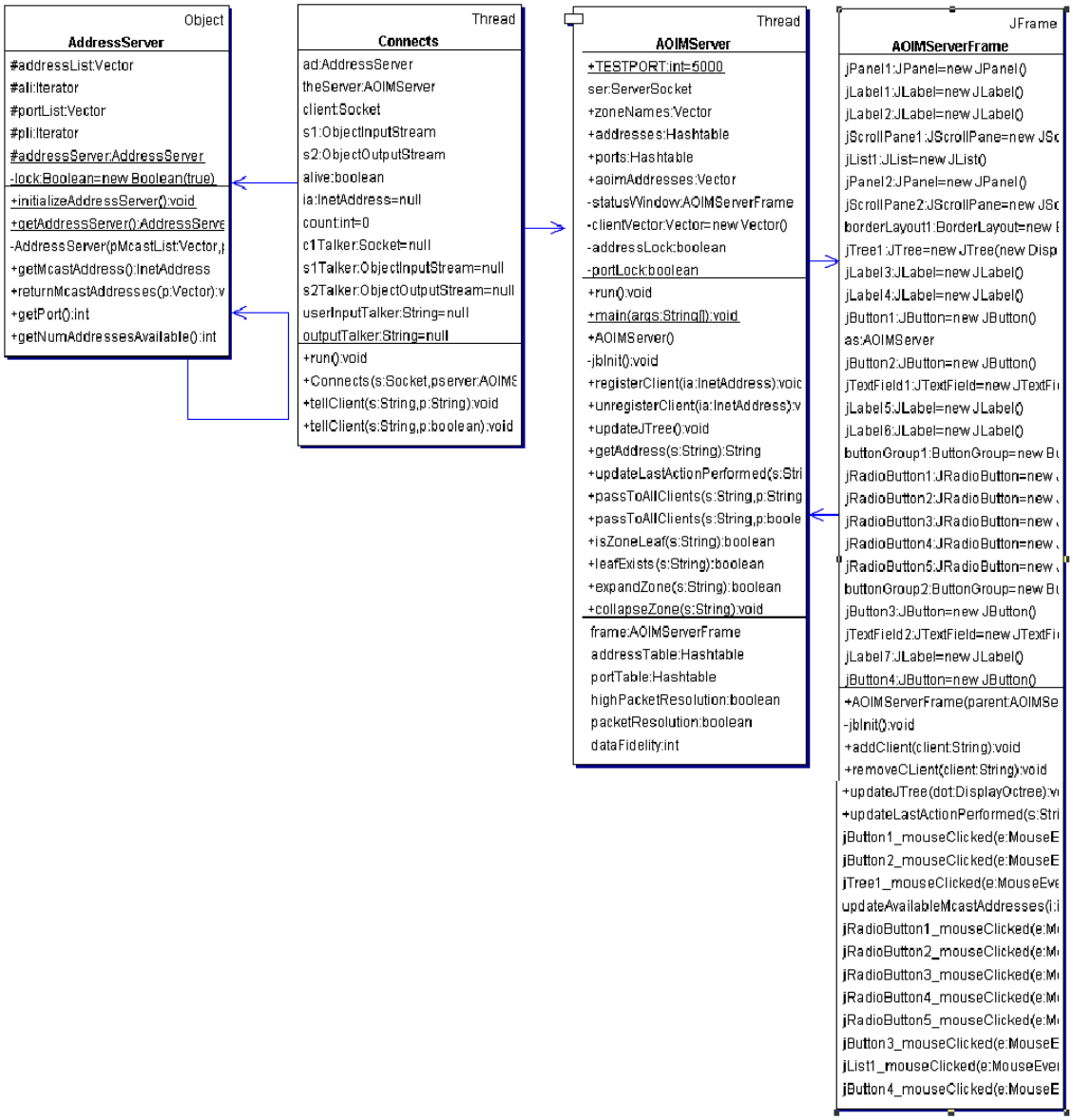

Figure 15. UML diagram of the AOIM system (part 2)

LIST OF REFERENCES

Abrams, H. (1999). *Extensible Interest Management for Scalable Persistent Distributed Virtual Environments*. Doctoral Dissertation, Naval Postgraduate School, CA.

Barrus, J., Waters, R., and Anderson, D. (1996). *Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments*. Mitsubishi Electric Research Laboratory. http://www.merl.com/papers/docs/TR95-16a.pdf

Barrus, J., et al. (1995). *Building Multi-User Interactive Multimedia Environments at MERL*. IEEE Multimedia, Volume: 2, Winter 1995.

Capps, M., et al. (2000). *NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments*. IEEE Computer Graphics and Applications, Volume: 20 Issue: 5, Sept.- Oct. 2000.

Coulouris, G., Dollimore, J., and Kindburg, T. (2001). *Distributed Systems: Concepts and Design*. Essex, England: Pearson Education Ltd.

Frécon, E. and Stenius, M. (1998). *Dive: A Scalable Network Architecture for Distributed Virtual Environments*. Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), Vol. 5, No. 3, Sept. 1998, pp. 91-100.

Fischer, W. (2001). *Enhancing Network Communication in NPSNET-V Using XML-Described Protocols*. Masters Thesis, Naval Postgraduate School, CA.

Funkhouser, T. (1995). *RING: A Client-Server System for Multi-User Virtual Environments*. ACM for 1995 Symposium on Interactive 3D Graphics, Monterey CA., pp. 85-92.

Holzner, S. (2000). *Java Black Book*. Scottsdale, AZ: Coriolis Group.

Macedonia, M., et al. (1995). *Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments*. IEEE Computer Graphics and Applications, Volume: 15 Issue: 5, Sept. 1995.

Mahmoud, Q. (2000). *Distributed Programming with Java.* Greenwich, CT: Manning Publications.

McGregor, D., Kapolka, A. (2001). *NPSNET-V: An Architecture for Creating Scalable, Dynamically Extensible, Networked Virtual Environments.* PowerPoint Brief given at the 2001 MOVES Institute Open House. Monterey, CA: Naval Postgraduate School.

Samet, H. (1990). *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS.* New York, NY: Addison-Wesley Publishing.

Singh, Serra et al. (1994). *BrickNet: A Software Toolkit for Network-Based Virtual Worlds.* PRESENCE: Teleoperators and Virtual Environments, Vol. 3, No. 1, 1994.

Taylor, S., Saville, J., and Sudra, R. (1999). *Developing Interest Management Techniques in Distributed Interactive Simulation Using Java.* Proceedings of the 1999 Winter Simulation Conference.

Washington, D. (2001). *Implementation of a Multi-Agent Simulation for the NPSNET-V Virtual Environment Research Project.* Masters Thesis, Naval Postgraduate School, CA.

Zyda, M. and Singhal, S. (1999). *Networked Virtual Environments: Design and Implementation.* New York, NY: ACM Press.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Professor Michael Zyda
   Code MOVES
   Naval Postgraduate School
   Monterey, CA
   zyda@movesinstitute.org

4. Research Assistant Professor Michael Capps
   Code CS/Cm
   Naval Postgraduate School
   Monterey, CA
   capps@cs.nps.navy.mil

5. Don McGregor
   Code MOVES
   Naval Postgraduate School
   Monterey, CA
   mcgredo@nps.navy.mil

6. Scott Wathen
   Code MOVES
   Naval Postgraduate School
   Monterey, CA
   swathen@hotmail.com

7. Brenda Wick
   Princeton, IN
   brenda_wick@hotmail.com,

8. Mike Wathen
   Newburgh, IN
   mcwathen@evansville.net