uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ihab Kazem

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Visibility-Based Zonal Communication in Massively Multi-User Online Games and Distributed Simulations

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. Shirmohammadi

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. Boukerche

Prof. G.A. Warner

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Visibility-Based Zonal Communication in Massively Multi-User Online Games and Distributed Simulations

by

Ihab Kazem

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Masters in Applied Science degree in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-49220-8
Our file Notre référence
ISBN: 978-0-494-49220-8

Visibility-Based Zonal Communication in Massively Multi-User Online Games and Distributed Simulations

Master of Computer Engineering Thesis
Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

by

Ihab Kazem
September 2007

Abstract

Massively multi-user online games and distributed simulations aim at supporting a large number of users while keeping the communication among the parties synchronous and highly interactive. To achieve collaboration, the virtual world is divided into multiple adjacent hexagonal zones in order to properly organize the entities and efficiently manage their liaison. The data distribution structure is Peer-to-Peer (P2P)-based, to support home users, and is managed by a topology-aware protocol at the application layer rather than the network layer. However, such zoning restricts cross-zonal interactions and exposes the division of the world to the parties. Problems such as crowding one zone among others defeats the very purpose of interest management and makes geographic partitioning alone inefficient for modeling interactions. The objective of the thesis is to introduce the design, implementation, and performance study of an area of interest management approach to manage massively multi-user simulations and online games by shifting messaging from a purely zone-based approach to a visibility-driven one. This makes the partitioning transparent to users. Also, a novel idea is presented for dynamic load balancing to handle transient crowding that can be achieved in real-time without modifying the communication architecture. Implementation and performance measurements are also presented.

Acknowledgements

I would like to express my gratitude to all who gave me the encouragement and power to complete this thesis and all who gave me the necessary knowledge and practical support to achieve the goals set at the start of this journey.

I am highly indebted to my supervisor Dr. Shervin Shirmohammadi. Dr. Shirmohammadi has provided me with an extraordinary opportunity to undertake research in the emerging field of distributed simulations and multi-user online gaming. I thank my supervisor for providing me with all necessary inputs, facilities and, last but not least, his financial support. His project management, support, and guidance, made the realization of this project possible.

Appreciation also goes to Dewan Tanvir Ahmed, Ph.D. candidate, University of Ottawa for his support, guidance, and continuous encouragement.

I gladly remember all my colleagues and well-wishers from the DISCOVER Lab, University of Ottawa, to whom I shall remain thankful always. Special thanks go to Madeh El-Badaoui, undergraduate Software Engineering student, University of Ottawa, for his contributions in implementing the simulation application.

Finally, I dedicate the thesis to my parents, Adnan and Aida, who have provided me with everything possible to achieve my objectives and complete my Masters.

Contents

List of Figures

List of Tables

List of Acronyms

ALM - Application Layer Multicasting

AoIM - Area of Interest Management

CVE - Collaborative Virtual Environment

FCN - Filtering Capable Node

FPS - First Person Shooter

HDSP - Hybrid Distributed Simulation Protocol

HLA - High Level Architecture

IP - Internet Protocol

ISP - Internet Service provider

LAN - Local Area Network

MMOG - Massively Multiplayer Online Game

OSG - Open Scene Graph

P2P - Peer-to-Peer

ROI - Region of Influence

RPG - Role Playing Game

RTI - Run Time Infrastructure

RTS - Real Time Strategy

RTT - Round Trip Time

TCP - Transmission Control Protocol

UDP - User datagram protocol

WAN - Wide Area Network

Chapter 1

Introduction

1.1 Motivation

Multi-user distributed simulations are gaining popularity for their significant contributions in virtual military training and online gaming. These simulation environments are used to avoid risks in the training as well as to assess and visualise real-life scenarios. Military simulations are seen as a useful way to develop tactical and strategical solutions. The scope of simulations has widened to include not only military but also political and social factors.

Simulations offer a virtual world for parties to interact and collaborate in real-time. Defense ministries and military and aviation simulation companies are one users of such technology and are in constant need of overcoming server bottlenecks and the lack of deployment of IP Multicasting in order to make their training environments fully deployable on the Internet in a scalable manner. Massively Multiplayer Online Games (MMOGs) are another kind of such environments and are different from other games in that they aim at supporting thousands of players to share a single game world simultaneously [1].

With the advance of computer graphics, artificial intelligence, and the emergence of the internet as a must in every home around the globe, it is not surprising that networked multiplayer games have quickly gained success and became profitable to their vendors. Everyday, thousands of players are joining online to share the gaming experience with fellow players in the tempting virtual worlds of games such as Sony's EverQuest, Valve's Half-Life, and Blizzard's World of Warcraft. World of Warcraft has over 2,000,000 users with peaks of over 500,000 players interacting at the same time in the digital world [2]. Hsiao and Yuan state that, according to general reports from game companies, the

number of users playing online at a given time is between 6,000 and 10,000, bringing about a powerful US$1 billion in subscription revenue in 2004 and more than double by 2009 [3]. In addition to entertain collaboration activities for players, these games offer "the reward of being socialized into a community of gamers and acquiring a reputation within it [4]." Indeed, MMOGs have, in many cases, become addictive and offer a "social factor" that gamers yearn for. Ducheneaut et al. present a comprehensive case study on how the World of Warcraft is designed to attract subscribers who become an audience for other players' in-game performances, providing an easy and accessible way of information and chitchat [4]. In such cases, the game is like being "alone together" – the player is surrounded by others for "spectator experience" and a sense of social presence. Klastrup goes further to add a psychological factor to these games [5]. He studies how death and the way avatars deal with death help shape the online worlds' social system and norms, and how player culture and player stories enforce a "heroic" approach to death directly and indirectly, serving as a guide to new players, teaching them how to "do better" both as social players and gamers.

Multi-user simulations and MMOGs introduce hard challenges for system designers. The most important challenge is the scaling of the system as it requires maintaining the proper collaboration states and exchanging of too many messages. A fundamental requirement of any real-time collaboration tool is the exchange of frequent update messages among nodes, and it is truly a challenge to keep data rate to a low without affecting the simulation experience. Scalability is a critical issue to consider when designing large-scale simulators and MMOGs, as it is a function of the dependencies among the components of the system. For instance, communication and processing are two different components of a simulation system. The basic definition of scalability is the ability of the system to withstand an increase in the number of users it supports, without degrading the overall performance and interactive experience.

Simulation environments can be collaborative in nature. Collaborative Virtual Environments (CVEs) aim at keeping communication among the parties synchronous. Precise coordination among the parties is affected by the end-to-end delay where the

parties are connected to an intranet or to the Internet from geographically distributed location. Such systems are closely-coupled as it has been recommended that the end-to-end delay should be within 100 msec [6]. Other studies have lowered the condition to 200 msec as an acceptable delay [7]. Moreover, the interactions among parties are highly reactive: a user's response depends upon other parties actions. Therefore, the requirement for frequent updates with a moderate end-to-end delay imposes this hard time constraint on CVEs.

Finally, network lag is another well-known problem in these simulations. It affects the performance of the collaboration. When a user participates in a simulation, its interactions with other nodes must be sent to all the participants over the network such that all entities involved in this collaboration can update their states. Because of the networking limitations and the traffic conditions, some of these "updates" are lost or delayed. Much research has been conducted to overcome the networking limitations to provide better distributed simulators. Some of these studies provide receiver-initiated and selectively-reliable transport protocols [8] that can be used to deliver important messages with a high degree of reliability, while others use sender-initiated approaches to transmit key updates with guaranteed reliability [9]. The IEEE DIS standard [10] has also been successfully used in controlled environment with vast resources, mostly for military simulations. These approaches are all based on IP multicasting, and, although they achieve good results in Intranet environment, they are not readily deployable on the Internet.

Despite the above research, there are many open problems left in this field. In this thesis, an architecture and design is presented where visibility and load balancing issues in such environments are addressed. Let us take a closer look at the specific research problems that this thesis has successfully tackled.

1.2 Research Problem

Currently in MMOGs, the client-server architecture is adopted to achieve synchronous communication between parties. The server is responsible for relaying one client's message to others for collaboration. Since data is maintained and managed by the server, the client-side software is light and simple. But in such collaboration software, the server becomes a bottleneck and could become a single point of failure. This makes scalability both hard and expensive to achieve as vendors try to accommodate the large number of users with server farms and proxies. Also, the server has to send the state updates of each client to every other client, consuming too much bandwidth.

A more efficient solution is the use of IP multicast, whereby one client sends a multicast message to the group of users, and the network intelligently replicates the message such that it arrives at all other clients. Unfortunately, IP multicasting is not deployable at large on the Internet. [11] and [12] well document the lack of applicability of IP multicasting on the Internet. The fact that IP Multicasting is designed for a hierarchical routing infrastructure and does not scale well in terms of supporting large number of concurrent groups, business reasons due to the undefined billing at the source (content provider) and receivers, the deployment hurdles caused by manual configuration at routers, Internet Service Providers' (ISPs) unwillingness to implement IP multicasting, and many other obstacles stand in the way of fully deploying IP multicasting over the internet.

Due to the practical lack of multicasting infrastructure on the Internet and encouraged by the success of P2P file sharing programs such as Napster and Kazaa, an alternative has been proposed to shift multicast support from the networking layer to the end systems and the proxies. This is Application Layer Multicasting (ALM) [13]. In ALM, data packets are replicated at end-hosts instead of routers. The end-hosts form an overlay network, where the end-hosts (or proxies) themselves relay the data to one another. As the routing information is maintained by an application of the end-host rather than at routers, it is more scalable than IP multicasting. As the ALM does not require any special

infrastructure support, it is fully deployable on the Internet. However, ALM does come with tradeoffs: more bandwidth and delay (compared to IP multicasting) for the sake of supporting more nodes and scalability. But it has been shown that ALM-based algorithms can have "acceptable" performance penalties with respect to IP Multicasting and other practical solutions [14].

But even with P2P communication, there will be a bandwidth problem since, with increasing number of clients in this "massive" environment, the traffic increases to a point where not all traffic can be processed by a given client. However, this is not a new problem and has existed, since the 1990s, in IP based simulations of military combat training. To solve it, military simulations use Area of Interest Management (AoIM): dividing the virtual world into separate mini worlds, or zones, where each zone is managed separately. Although this reduces the overall traffic that a given node experiences, problems introduced by such zoning include parties having to migrate from one communication subsystem (server, P2P tree, proxies, etc.) to another when switching zones. Another problem is that the geographical division of the virtual world, which is typically used in simulations and games, isolates message propagation within the individual zones, offering a discrete view for players who are close to the border of two zones, as opposed to the ideal continuous view. Also, since the interest of the parties is ultimately what they can "see" or can interact with, zone-based messaging should be refined to a visibility-driven one. Similarly, heavy concentration of users in certain zones affects the gaming experience and defeats the very purpose of interest management: if a great percentage of users are all in the same zone, then we are back to the original problem as if we do not really have zones.

In this thesis, the two problems of visibility and load balancing are specifically addressed. It will be shown that with proper design and implementation techniques, users can have a continuous view of the world without noticing any "jumping" effects of going from one zone to another. Furthermore, load balancing will make it possible to support a larger-than-usual number of users concentrated in a given zone.

1.3 Research Contributions

This thesis aims at providing an approach to manage the common interest of a large group by promoting zone-based messaging. It also tries to solve the problems introduced by partitioning of the virtual world into zones such as limiting the view of users to a discrete one, the effect of zone switching on overlay networks and the overall simulation or gaming experience, and the crowding of one zone among others hindering user interaction and not evenly balancing servers' resources.

In a nutshell, this work introduces a multicasting approach to manage the common interest of a large group, and includes:

- A topology-aware application layer multicasting protocol
- Zone-based messaging for interest management
- An architecture for inter-zone communication to make the partitioning transparent to users
- The idea of visibility-driven messaging and its assessment
- A novel idea for dynamic load balancing that can be achieved in real-time without modifying the communication architecture
- Proof-of-concept prototype implementation and performance evaluation
- Peer-reviewed publications:
    - S. Shirmohammadi, I. Kazem, D.T. Ahmed, M. El-Badaoui, and J.C. de Oliveira, "A Visibility-Driven Approach for Zone Management in Simulations", Simulation: Trans. of the SCS, (accepted, to appear)
    - I. Kazem, D.T. Ahmed, S. Shirmohammadi, "A Visibility-Driven Approach to Managing Interest in Collaborative Virtual Environments with Dynamic Load Balancing", Proc. IEEE Int. Symposium on Distributed Simulation and Real Time Applications, Chania, Crete Island, October 2007

- I. Kazem, D.T. Ahmed, S. Shirmohammadi, "A Zone Based Architecture for Massively Multiuser Simulations", Proc. SCS/ACM Communications and Networking Simulation Symposium, Norfolk, VA, USA, March 2007
- D.T. Ahmed, S. Shirmohammadi, I. Kazem, "Zone Based Messaging in Collaborative Virtual Environments", Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, ON, Canada, November 2006

1.4 Thesis Outline

After the above introduction, the rest of the thesis is organized as follows: **Chapter 2** presents a background and literature review on topology-aware ALM protocols, AoIM, and load balancing for massively multi-user simulations and online games. **Chapter 3** presents the proposed design for our topology-aware ALM protocol, Zone-Based AoIM, and Load balancing. **Chapter 4** discusses the implementation of a multi-user combat simulation application. **Chapter 5** provides a performance evaluation to verify our design and to show the real time behaviour. It also discusses the possible limitations of the system and possible improvements that can be made for better performance. **Chapter 6** summarizes and concludes the thesis with future work recommendations.

Chapter 2

Background

2.1 Topology-Aware Application Layer Multicasting

In order to achieve efficiency in messaging in large-scale distributed group communication applications and reduce redundancy and packet duplication, IP multicast has long been researched and proved to be efficient [15]. In multicasting, the source sends one message to all members in a group, as opposed to sending a separate copy to each member [16]. Group members are connected in what is called a multicast tree. An underlying routing protocol constructs the multicast tree and takes care of optimizing the communication among its members. As members join or leave a multicast group, the multicast tree is dynamically reconfigured [16]. All of this is done at the networking layer and is transparent to individual clients. Multicasting scales well in massively multi-user simulations as it decouples the size of the receiver set from the amount of data kept at any single node [17].

IP multicasting's dependency on network-layer routers, however, made it infeasible to be deployed on large parts of the Internet [15]. This is also due to the concerns related to scalability (maintaining state information at each router), and network management (higher layer functionality, security, and error control) [16]. Since application-layer multicast does not require any special support from network routers and moves the role of multicast to end hosts, it can be deployed universally. An example is shown in figure 1. In figure 1(a), the routers are responsible for replicating and forwarding the messages to finally reach the end-hosts. In figure 1(b), the end-hosts are responsible for such replication and forwarding, with no concern on what is happening in the routers.

Logically, the end-hosts themselves form an overlay network between each other with the goal of maintaining an efficient data transmission tree [17]. In ALM, members in a

multicast group communicate via the overlay network in which each edge corresponds to a direct unicast path between two group members. All the data packets are sent as unicast packets and forwarded from one member to another. Since data is transmitted between nodes via unicast, we are effectively exploiting all existing security, flow control, and reliable delivery mechanisms that are readily available and mature. It is not surprising that ALM protocols are increasingly being used to implement efficient commercial content-distribution networks.

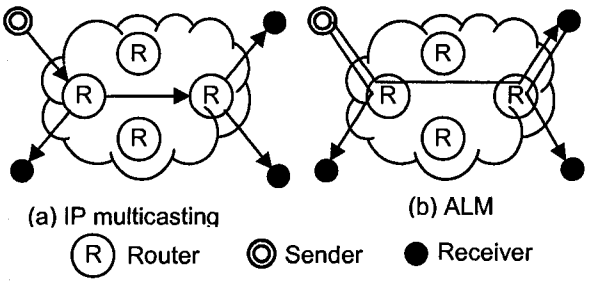


Figure 1. Multicasting and ALM concepts

ALM does have performance penalties: packet delay, bandwidth change, and resilience to change in overlay network. This is because a packet may be replicated and forwarded on the same underlying link more than once. Also, end-hosts may join/leave unexpectedly. In recent studies, application-layer multicast approaches focused on scalability. The main idea of the ALM approach is that each multicast participant maintains only state for partial participants: those connected to it. It makes the approach scalable. However, most approaches pay little effort to ensure that application-layer connectivity is congruent with the underlying IP-level network topology [15]. Zhu et al. state that it is impossible to completely prevent multiple overlay edges from traversing the same physical link. This causes unavoidable redundant traffic on the same link, referred to as stress. Also, since unicast communication between end systems involves traversing other end systems, latency is potentially increased. So it is critical to reduce the end-to-end latency, caused by link stretch, and the per-link bandwidth requirements when compared with IP multicast [18].

Banerjee et al. document the efficiency of ALM protocols with respect to native multicast [17]. ALM overlays are usually measured for their "goodness" using the stress

and stretch criteria. Stress is defined per-link and is the number of identical packets sent by a protocol over each underlying link in the network. Stretch is defined per-member and is the ratio of path-length from the source to the member along the overlay to the length of the direct unicast path. For example, a protocol in which the data source unicasts the data to each receiver (it is a multi-unicast protocol) clearly minimizes the stretch but does so at the cost of O(N) - where N is the number of group members - stress at links near the source. This also requires O(N) control overhead at some single point. More importantly, such protocol is robust and can endure member failures as a single failure does not affect the whole group.

[17] and [19] evaluate ALM protocols according to the following metrics:

- Quality of the data delivery path: measured using metrics such as stress, stretch (end-to-end latency), and node degrees.
- Robustness of the overlay: Since end-hosts are typically less stable than routers, it is important for application-layer multicast protocols to mitigate the effect of node departures. Robustness is evaluated by the scalability of the protocol to handle the potential growth in user population. More importantly, it is measured by quantifying the extent of the disruption in data delivery when different members fail, and the time it takes for the protocol to restore delivery to the other members.
- Control overhead: This should be low to efficiently use network resources. It is an important cost to study the scalability of the scheme in large groups.
- Limited topology information: This is the key to building efficient overlays. End systems have limited or no underlying topology information, unlike network routers.
- Heterogeneity: Since end systems have heterogeneous processing power and interface bandwidth, the result is a mixture of "fan-out" values. This fan-out value is the number of outbound interfaces a node can support.

Furthermore, ALM protocols are divided into centralized and distributed. A protocol is centralized when one entity constructs the overlay network and keeps information of all nodes. In a distributed protocol, there is no dependency on one entity. Distributed

protocols are further divided into tree-first where the tree is built directly when members join and is used for data delivery and mesh-first where the mesh is built when members join and then a tree for data delivery [16]. [15] states that a centralized protocol is based usually on a complete end-to-end delay latency measure to consider the global topology information when constructing the data paths. A decentralized tree-first one, on the other hand, is when each member of the multicast group contacts only partial other members. The centralized approach achieves optimal congruency of the application with the underlying topology but does so at the cost of scalability (each member has to measure end-to-end latency with all other members). The decentralized approach, on the other hand, scales well in large groups.

For these reasons, in order for ALM protocols to scale well and efficiently use network resources, a topology-aware algorithm is to be adopted to build overlays structures (mesh, ring, tree...) in a manner that achieves maximum congruency with the underlying network layer structure.

2.2 Area of Interest Management

Most literature on area of interest management agree on that in order to support a massive number of users, the simulation map is to be divided into multiple zones, where each zone represents the "interest" of the nodes inside of it. [20] discusses that a virtual environment can be quite big and a significant part of it is not relevant to the action context of a user or player at a certain time. It is apparent that the simulation data space should be divided and techniques be used to determine the information relevant for updating the player's current experience. Such "relevant" information is based on the player at a certain moment, based on metrics like virtual proximity, field of vision, line of sight, and action context. Events should be filtered in the communication model according to relevance or interest. This reduces the number of events along with the dependency on latency and network congestion problems [20].

2.2.1 Scalability vs. Network Lag

The original motive behind managing areas of interest, whether for large scale multi-user simulations or MMOGs, is scalability. This, however, has to be achieved without violating the synchronous communication precondition: parties have to exchange states in a smooth interactive manner and are bounded by a maximum tolerable end-to-end delay of roughly 200 ms. This value is necessary to evaluate the performance of our system, and how far we can push the envelope in increasing the number of users we wish to support.

It is essential to understand the application's requirements and multi-user system's needs to know to what extent we wish to scale the system, which components we should give more priorities to, and when we should draw the line in trying to complicate the architecture of one component so as not to affect other components. For example, MMOGs are usually categorized into Role Playing Game (RPG), Real Time Strategy (RTS), and First Person Shooter (FPS). In RTSs, such as Westwood's Red Alert, the user commands the units to move or perform an action. In other words, the user provides the "what" to do. In such scenarios, an end-to-end delay, or the time for response of up to several seconds is acceptable as the player is only interested in the outcome of the action [1]. FPSs, on the other hand, such as Valve's Half-Life and id Software's Quake, usually require a low response time of 180 msec as the player is interested in "how" an action is performed [1]. One good example of such action is when a player uses a sniper rifle as a weapon. The player commands the avatar in every single step from reloading the rifle, to aiming, and finally shooting the target. So for performing a single action, many states are processed and communicated, and it is essential that the player experiences the outcome of all these states. Another variation to these FPSs is having a tank or any other military unit controlled in first person. In such environments, the tolerated end-to-end delay threshold may not be as great as those in RTSs, but it is definitely greater than the avatar using the sniper scenario since hitting the target is not as delay-critical (tank bombing another tank versus sniper shooting another person, for example). Thus, FPSs are the most demanding when it comes to having low end-to-end delay.

Another issue worth considering is that such end-to-end delay is not always a strict condition that, if violated, will render the gaming experience intolerable. The threshold depends on the human perceptivity, which is definitely a characteristic of the player or user and the specific situation he/she is in. Some variance around the threshold can be tolerated. Moreover, if game state consistency and simultaneity is maintained throughout the game, the player can tolerate a network lag above the threshold at one state, as long as the following states are received on time.

After defining the end-to-end delay threshold, achieving scalability can be put into a better context. One of the main reasons behind the trend of shifting from the client-server paradigm to a peer-to-peer one is that the former one achieves scalability by only increasing the server's bandwidth (through server farms) while the latter has proven to support massiveness in terms of users, as file-sharing programs such as Napster and Kazaa have taught us. Some of the proposed protocols for interest management entail having a coordinator node responsible for every zone and several slaves connect directly to this coordinator [21]. In this case, the end-to-end delay is guaranteed as only one level of communication is maintained (server to client) and, in most cases, is almost half of the tolerated threshold (taking 200 ms as benchmark). These protocols do not exploit all of the end-to-end delay value allowed. For this reason, using P2P, we can add another level to the first level of slaves (slaves themselves become master node for other slaves). In this manner, we are exploiting both bandwidth (the number of slaves that a master node can support) and levels (adding another level to the network topology), thus increasing the number of users supported exponentially without violating the end-to-end delay.

2.2.2 Zone Definition

The simulation map is divided into multiple adjacent zones. But on what perspective a zone is constructed is hard to consider. One way to look at it is to consider that in a network where several Local Area Networks (LANs) are connected through the Internet, each LAN represents a zone. A LAN provides bandwidth in gigabytes magnitude, so it

would be easy for the master node or server responsible for the zone to construct the overlay network, maintain its state, and manage new comers and early leavers. However, this is not a sufficient requirement: other factors such as visibility and logical partitions should be taken into consideration when defining a zone.

2.2.3 Zone Shape

The square shape is a poor choice to represent a zone. If we are to implement a multiple-zone layout as opposed to a simple two-zone layout, more connections and disconnections are taking place for the same path traversal scenario. In figure 2, a multiple-zone layout results in more connections and disconnections than a two-zone layout for the same path traversal. Also, the square shape does not apply well to emulating units' range and visibility (defined mostly by a circle). Adopting a circle shape, on the other hand, will create gaps in between the circles, as shown in figure 3. To sum up, we need a shape that is both regular (to avoid gaps) and can emulate visibility as much as possible.

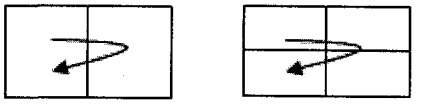


Figure 2. Two-zone vs. multiple-zone square layout

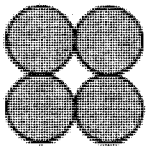


Figure 3. Circular zones with gaps

2.2.4 Problems Introduced by Zoning

Dividing the simulation into multiple zones is needed for interest management; however, such division introduces problems. In order to be objective when doing a

performance analysis of a proposed architecture, one has to compare the benefits of zoning such as the ability to increase the number of users to the problems inherent in having multiple zones. One problem is that when a node moves from one zone to another, it will have to leave the overlay network in the previous zone and join the new one. The disconnection from the previous zone might create complications for the overlay network, such as when the leaving node is a parent to some children nodes which will now become orphans in the overlay tree. Also, connection to the new overlay network should ideally be done instantly so that the joining node will not add a delay and will keep up with the game state messages. In both cases, disconnections and connections during zone switching should not affect the stability of the overlay network. Another problem that is a direct result of the first one is that it seriously affects the stability of the overlay networks when a node is "bouncing" in and out of a zone when frequently moving around the boundary of two zones. This creates several disconnections and connections during a short time interval, which degrades the performance. In figure 4, the path followed by a node at the boundary of two zones creates instability for the zones' overlay networks.

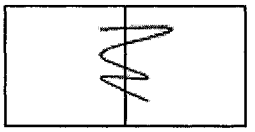


Figure 4. A node moving frequently around the boundary of two zones

2.2.5 Discrete View vs Continuous View

Some proposed architectures, such as Knutson et al's P2P Support for Massively Multiplayer Games [1] and Limura et al's Zoned Federation of Games Servers [22], offer a discrete view of the nodes: a node is only interested in the zone it is in and has a discrete view of the world. This approach simplifies the design and makes it more scalable and robust. However, in reality, simulation requirements or game mechanics require that some nodes have a continuous view of the world; i.e., be able to "see" nodes in other zones. An example is a sensor array, which can see multiple zones at once, or

players that are in two different zones but close to the boundaries and can therefore visually see each other. In figure 5(a), a sensor array (black triangle) has a large visibility range and can see nodes lying in different zones (block dots). In figure 5(b), two nodes residing near the boundary of two zones can also see each other even though they are in different zones. As a result, in general, area of interest management cannot assume a discrete view for all entities and must support continuous view for frequently-occurring cases or special entities.

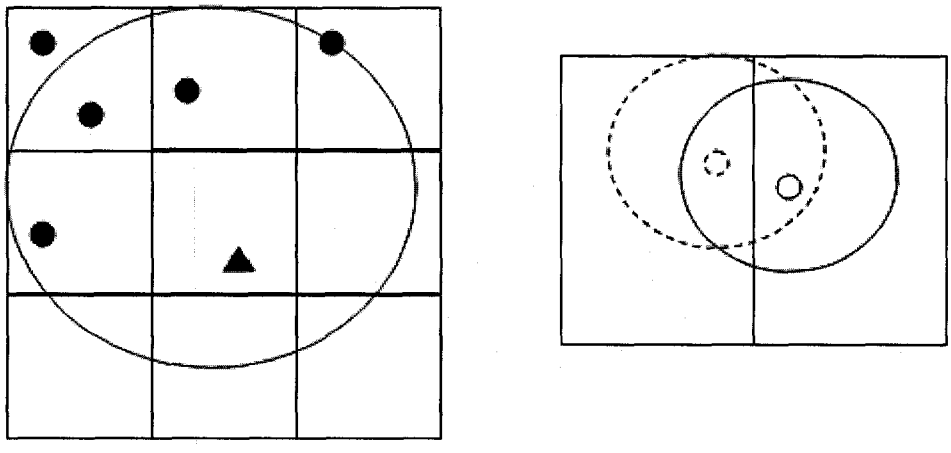


Figure 5. Sustaining a continuous view for nodes

2.2.6 Exploiting P2P Properties

Research that has been done on interest management typically includes a design for the network topology that is adopted in each zone. Many use a P2P network architecture where each node contributes its bandwidth to the overall capacity. However, the P2P characteristic is not at the same level for all approaches. Yu and Vuong's MOPAR [21], for example, suggests an architecture for MMOGs that is a hybrid of distributed hash table and unstructured P2P where a master node, also a player, is chosen to be responsible for each zone and becomes the parent or server for all other players, or slaves, as the paper refers to, in that zone. So each master node supports all of the slaves

inside its zone. Although the architecture is P2P in the sense that master nodes are also connected to each other to manage the inter-zone messaging, the communication architecture inside one zone more closely resembles the client-server approach than P2P. The master node is still responsible for supporting many slaves, and much is relied on the bandwidth and computational power of the master node, which will make it a bottleneck. The approach does not make use of the bandwidth that the slave nodes can potentially offer (S has its bandwidth unexploited), as shown in figure 6. For this reason, it is important to exploit the scalability that P2P architectures get to offer by adopting a structure that really connects peers in a P2P manner such as mesh-based or tree-based structure which lowers the cost of centralized infrastructures and distributes the processing load to where it is benefited from (peer or node participating in the simulation).

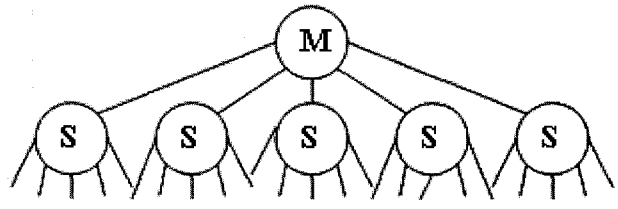


Figure 6. Master node M becomes a bottleneck and slaves

2.2.7 Simulation Issues

To demonstrate their applicability to AoIM, existing work on interest management evaluate their protocols by building a simulation tool to get some tangible results on the number of users that could be supported, message overhead, network stability, and other metrics. However, special care must be taken when doing performance analysis. In particular, one must ensure that the simulation reflects game dynamics, nodes' random behaviour, and the magnitude of states sent and received by each node. El Rhabili et al present the simulation issues at hand [20]. The work points out that the massiveness that needs to be evaluated cannot be carried out by a simple tool as it needs a significant processing capacity. Also, the size of the data that is being sent as messages is huge since

there are thousands of players frequently sending messages in a very large area which makes it hard and complicated to manage data and the simulation model. Furthermore, using such simulation tools makes it hard to imagine real-life scenarios that often happen in a virtual environment that might affect the system's overall performance; e.g., problems introduced by zoning as discussed in Section 2.2.4. This will make the protocols more susceptible to failure when encountering such scenarios and these issues must be carefully addressed in a simulation. For these reasons, the simulation environment should try to address real-life scenarios as much as possible by, for example, implementing a prototype military simulation and predicting the scenarios that could happen in a combat environment. Since it is hard to manage massive data in a single simulation tool, it is best that the communication model is a distributed one, where each peer computes, locally, the virtual arena where it interacts. Even though it is hard to run a massive number of such distributed peers simultaneously, evaluating the massiveness in such simulation environments can be based on the evaluation of simple scenarios run in the virtual world.

2.2.8 Geographic vs. Behavioural Modeling

There are two extremes when modeling the interest of parties' interactions with each other in the distributed simulation. The first is static geographic partitioning, done at the initialization of the game or simulation. This is beneficial when it defines the structure of the virtual world. For example, a virtual world consists of multiple cities. Each city represents a geographic partitioning: it is the area at which most interactions occur, and, in most cases, interacting parties are not interested in what is happening in other cities. Second Life has adopted such approach [23]. To make up for delays introduced by migrations, the virtual world is designed such that boundaries between areas of interest are uninteresting in nature: cities are separated by forests or wilderness, where parties do not tend to stay long. Thus, a migrating party will not experience others' interactions at

boundaries. Although this works in Second Life, it is not an acceptable solution for all virtual simulations.

The second extreme for modeling interest is behavioral. In a military simulation, two different units such as a tank and a fighter jet have different behaviors in terms of how fast they move, how far they can see, and the size of the area they can interact with (a jet launching a missile has a larger area of influence than a jeep patrol). Lu et al. argue that, due to the ease with which mapping processing resources or servers can be applied in geographic regionalization, little effort has been invested in mapping the behavioral approach [24]. The mapping of geographic regionalization to server allocation in a bid to prevent inter-server communications is not the appropriate path to take. This is because geographic regionalization does not afford the greatest potential for player interaction, and the processing overhead of elaborate techniques of allocating processing resources in this manner comes at a high price (process intensive).

Even though behavioral modeling is the ultimate goal for managing the interest of parties, geographic regionalization is not without its merits and can be coupled with behavior-based communications.

2.3 Load Balancing

The idea of load balancing is not new as it has been addressed by previous work. Load balancing is defined as an attempt to efficiently distribute an application's processing requirements across a number of servers [24]. Lu et al. lay out the techniques that are required of a load balancing algorithm [24]. An ideal solution ensures equal distribution of resources to prevent exhausting available processing resources on one server while other servers have spare capacity. The configuration is flexible and does not disrupt user experience. Finally, it does not hinder user experience by promoting player interaction within the virtual world.

Duong and Zhou classify dynamic load distributing algorithms into load sharing algorithms and load balancing algorithms [26]. Load sharing means avoiding the

situations when some nodes are overloaded while others are under loaded. Load balancing also targets solving this situation by attempting to equalize the workload of each node in a distributed system. Load balancing is the way to go to efficiently reduce average response time, but it requires a computational overhead to balance the workload among many servers, hindering the algorithm's effectiveness.

2.3.1 Crowding

Crowding is the problem when many nodes, or players in a game, move into one zone, heavily populating it. Such crowding can violate the quality of service and affect the gaming or simulation performance. Besides, the whole idea behind zoning is to evenly distribute the load of nodes by assigning a server to each zone. The nature of these games, however, demands that nodes interact with each other, navigating through the map and moving from one zone to another depending on their interest. Assuming an even distribution of nodes in the map is impractical. The distribution is in most cases random: some zones maybe completely empty and others completely crowded. That is why there is a need to do load balancing among the servers of each zone: if one zone gets populated, other servers that do not have a heavy load can be utilized to balance the load of the overloaded server.

Chen et al. point out that "flocking", when many players move into a zone managed by one server, is a MMOG pattern that cannot be ignored [25]. This is because some areas in a game are more interesting than others. Depending on the type of the game, also, like war games, an area that was secluded for some time, may suddenly experience warriors in big numbers (armies) crowding the area for a battle. Such hot spots are unpredictable, as they depend on in-game events such as following a quest, or players chasing each other. For this reason, static partitioning performs poorly when handling such scenarios.

Lu et al. also discuss how crowding, or congregating in the same area of a virtual world, inhibits interest management causing a slowdown in game play [24]. In the worst

case, it creates a complete inability to perform player interaction. They conclude that crowding creates the exact problem that regionalization aims to solve in the first place: avoiding the heavy population of one zone by a large number of players managed by one server.

2.3.2 Locality Awareness

There is a trade-off in any dynamic remapping algorithm between balancing the server load by dividing existing zones across more servers and decreasing inter-server communication by preserving the locality of adjacent zones [25]. The inter-sever communication that [25] addresses is the one responsible for communicating messages for nodes logically in different zones but having an interest to see across their own zone. Such communication channels are designed to solve temporary unique cases when there is a need to propagate messages between different zones, but they should not be relied on too much, as it defeats the original idea of interest management using zonal partitioning: messaging between parties should reflect locality. In short, the remapping of hot spots zones from an overloaded server to under loaded servers may increase the number of partition boundaries, disrupting zone locality. This negatively affects inter-zone communication for cross-partition interaction and inter-server player hand-offs as players move from one zone to another.

2.4 Other Literature Review

In addition to the above literature, other research in the field is briefly reviewed below.

2.4.1 ALM Protocols

Kim and Chon present a scalable and topologically-aware application-layer multicast approach designed for large-scale distributed applications [15]. The data paths are constructed based on a topological clustering of multicast group members. Even though, usually, topology-aware protocols require network topology information, their protocol requires the relative information of members using landmarks. The clustering of members is based on the ordering of their close landmarks. Their results show that data paths that reflect the underlying topology reduce unnecessary high latency and redundant network resource usage. They compare the performance of their protocol with a centralized one that is based on a complete end-to-end delay latency. [15] bases its design on a decentralized tree-first one because it scales well in large receiver sets.

Banerjee et al. present the NICE protocol, a scalable application-layer multicast protocol, specifically designed for low-bandwidth data streaming applications with large receiver sets [17]. The design is based upon a hierarchical clustering of the application-layer multicast peers and can support a number of different data delivery trees with desirable properties. NICE aims at providing an efficient, scalable, and distributed tree-building protocol, reduce the worst-case state and control overhead at any member to O(log N) (N is the total number of members), maintaining a constant degree bound for the group members, and approaching the stretch bound possible with a topology-aware centralized algorithm. The control topology is hierarchically-connected and the data delivery path is implicitly defined on the hierarchy. End-to-end latency is also used here as the distance metric between hosts: members that are "close" to each other are mapped to the same part of the hierarchy, which results in trees with low stretch. What

distinguishes NICE from other protocols is that it chooses the data delivery path to be a simple loosely-connected tree (loop free structure) and the control path to be clique, a strongly connected structure to reduce traffic, quickly detect changes and restore invariants. The result is a faster converging protocol.

[17]'s performance analysis provides an inclusive study on the different metrics needed to evaluate the "goodness" of a multicast protocol in massively multi-user groups. The IP multicast protocol and multi-unicast protocol are used as a benchmark for performance evaluation. The quality of the data path is evaluated in terms of tree degree distribution, link stress, and stretch. More importantly, the ability of the protocol to recover from host failure is studied by assessing how the data delivery path changes as members join and leave. This is because a member join might create multiple paths and cycles. A member non graceful leave, in contrast, causes a fraction of hosts to incorrectly receive the data packets sent from the source. Last but not least, the control traffic at end hosts and routers is evaluated.

[17] finally compares its performance with that of Narada's, a mesh-first application-layer multicast approach, designed primarily for small multicast groups. In Narada, the initial set of peer assignments to create the overlay topology is done randomly. While this initial data delivery path may be of "poor" quality, over time, Narada adds "good" links and discards "bad" links from the overlay. Narada has $O(N^2)$ aggregate control overhead because of its mesh-first nature: it requires each host to periodically exchange updates and refreshes with all other hosts.

Sobeih et al.'s VRing is another ALM protocol that establishes a virtual ring as an overlay network among the multicast group members in a self-organizing and distributed manner [16]. A spare ring overlay structure that improves connectivity among group members is proposed to reduce the routing delay of the original ring overlay network. Both the original ring and the spare ring are used for forwarding data packets, along with a duplicate suppression mechanism at the spare ring. VRING is counted as a distributed protocol that is mesh first by building both the original ring and spare ring and then uses both for data delivery. The choice of the ring as a structure is because it ensures a O(1)

node out degree for each member. Besides, a secure, ordered, and reliable delivery of messages is guaranteed because of token and flow control provided in a ring structure. This, however, comes at the cost of having a potentially large routing delay, and this is why a spare link is proposed to improve connectivity.

The performance of VRing is compared to NICE because the latter is the only protocol that explicitly addresses fault tolerance. [16] claims that, as the overlay size increases, VRing outperforms NICE in terms of the average data delivery ratio. This is a natural result of having a low member out degree (a failure of one member disrupts the data delivery to only a few other members).

Hybrid Distributed Simulation Protocol (HDSP) is a P2P-based ALM protocol that uses the tree as a structure for the overlay network. It is Hybrid because it has multiple functionalities: it forms and participates in the overlay tree on the Internet and, at the same time, can use IP multicasting to connect to LANs and act as a bridge between intranets and the Internet, if needed [27]. A hybrid node is the root of the overlay tree; it is also the server that manages and builds the tree.

HDSP uses a simple node joining strategy that might not reflect the physical topology onto the constructed overlay structure. This is due to that fact that the incoming join request is satisfied by a node having available bandwidth without exploiting any measuring technique. It does not consider any physical measure to improve the quality of the constructed overlay. Zhou et al. argue that most simulations for the P2P-based Communication Model for MMOG's assume that the network is homogenous when testing the message level [28]. However, such homogeneity assumption seriously affects the simulation results with respect to the network latency. Nodes in the simulation environments running on the network usually have different network locations, bandwidths and offline rates. Chambers et al. also present a redirection service for on-line games based on the geographic location of players relative to servers [29]. They show how such a service "better meets client demand, saving each client, and the Internet as whole, thousands of miles of networking inefficiency."

It should be noted here that our work is based on HDSP: we use the ALM tree structure designed and implemented in [27]. Our contribution to the protocol is detailed in Section 3.1.

2.4.2 AoIM Architectures

El Rhalibi et al. discuss the possibility of deploying MMOGs over a P2P architecture [20]. The target model is distributed and manages shared data. The model addresses issues common to MMOGs, such as interest management, scalability, and communication. Interest is managed by partitioning based on a hierarchal multi-level algorithm which enables contextual communications between peers. They note that special care should be taken when modeling communication to scale the system or reduce the amount of information sent. For example, the decision to apply compression of data transmitted should not just take into account the quality of the compression but also the relation between the bandwidth and latency gain in communication and the time and computational resources spent on the compression/decompression stage. Also, the benefits of adopting a P2P architecture for communication is clouded by the complexity of maintaining the consistency and persistency of the virtual world.

[20]'s P2P topology starts off from an initial centralized server configuration for joining and leaving to maintain control and authority over the players account information but grows into a P2P in-game communication as soon as the peer discovers its peer-group and a new zone is required. Binary Space Partitioning, octrees, or Potential Visible Set data structures could be used to control peers group and their relations in a dynamic way. An individual P2P network is formed for each partition managed by a super node. Subscription lists are dynamically managed by means of the spatial partitioning techniques. Data filtering is enabled based on the aura (which describes an area of the virtual world enclosed by a sphere [24]) detection and line of sight to refine interest. A new level of partitioning is applied to create a sub-zone, and the recursive properties of the spatial partitioning of the game world are exploited to create different

levels in a hierarchy of communication, where full multicast is only used at the lowest level of the hierarchy.

Tumbde and Venugopalan propose an approach to support massively multiplayer online games on a peer-to-peer system [30]. The model is based on the idea that players are more interested in their zone of visibility and influence than in the other zones. A computational geometry technique is used to partition the game space into zones, called Voronoi diagram. Such partitioning reflects closeness and locality. Voronoi partitioning of a 2D virtual world sub-divides the plane into polygonal zones where each zone is a set of points that are closer to some input vertex than to any other input vertex. An input vertex in this context is a player's position at a certain time. Managing the interest of players according to their relative closeness results in a scalable system.

The original idea behind the geographic partitioning according to Voronoi is that each node or player has a Region of Influence (ROI), and it is best to only communicate the state updates among players according to these ROIs. The server managing each partition is called a coordinator, which is also a player. The partition it is managing is actually its ROI. A periodic refining of the partitioning is done to reflect the current state of the game and maintain closeness and relevance. This is needed when a coordinator departs from its original zone. The major limitation of this architecture is that zone switching results in smaller Voronoi zones, which increases inter-zone communication overhead. Also, the dynamic constant repartitioning of the game map as players move from one zone to another, changing their ROIs, incurs a O(N logN) processing time which can be impractical during the game. Moreover, there is still no distributed model for applying Voronoi partitioning among the peers [30].

Knutsson et al.'s decentralized system partitions the world and synchronizes game states with best-effort multicast capabilities [1]. The design is implemented on a P2P overlay, and issues related to P2P networks such as addressing and replication are discussed. Persistent user state is handled by a central server. Such server delegates the bandwidth and processing intensive game state management to the peer-to-peer network formed by clients participating in the game. Temporal and spatial localities of players are

exploited to manage interest and localize communication. The P2P model is based on Pastry [31] and Scribe [32].

A. Yu and S. Vuong's MOPAR is a hybrid scheme of Distributed Hash Tables and unstructured P2P architecture to manage interest in MMOGs [21]. The neighbor discovery scheme is stable and consistent. A hexagonal zoning approach is used for partitioning. A master, home, slave node scheme is used for handling the dynamic characteristics of MMOGs, where the master node governs a set of zones. A mechanism to provide a continuous view of each participant of the virtual world is possible because the Area of Interest of each participant is smaller than the covering area of the master nodes.

Lu et al.'s interest management scheme is a predictive behavioural approach, as player interactions are associated to player expressiveness, as apposed to static geographic regionalisation of the virtual world [24]. They propose a model for communicating player interactions using player expressiveness and define the concept of an aura for determining when players should exchange messages. A Predicted Area of Influence identifies the extent of a player's aura over a period of time given the distance it may travel in a straight line in any direction. The scheme is suitable for peer-to-peer deployment.

Macedonia et al. partition virtual environments for large scale distributed simulations by associating spatial, temporal, and functional classes with network multicast groups [33]. They incorporate both the IEEE 1278 Distributed Interactive Simulation (DIS) application protocol and the IP Multicast network protocol for multi-player simulation over the Internet in context of the NPSNET simulator. The world is broken into hexagons, each representing a multicast group. Each entity sends state information to a multicast group corresponding to the local hexagon it is in, while at the same time subscribing to surrounding hexagons via their multicast groups.

Finally, the Defense Modeling and Simulation Office of the Department of Defense developed the High Level Architecture (HLA) to provide a highly interoperable and software architecture for the construction of component-based distributed interactive

simulation applications. The Run Time Infrastructure (RTI) is the middleware software that conforms to HLA Interface Specification by providing services to support the HLA-compliant simulation. Distributed interactive simulations have recently acquired a real-time nature for real-world dynamic modeling processing. This is why there is a need to have a real-time RTI in order for it to behave predictably within a required bound of response time [34].

Liu et al. argue that interest management systems such as Data Distribution Management service of the HLA focus on providing message filtering mechanisms but do so at the cost of Central Processing Unit cycles overhead [35]. This can be unsuitable for real-time applications. A scalable interest management algorithm is presented for HLA Data Distribution Management which employs a specific existing collision detection algorithm for fast interest matching which is also suitable for dynamic multicast grouping.

2.4.3 Load Balancing Algorithms

Lu et al. present a load balancing technique for Massively Multi-user Online Role-Playing Games that depend on clustered server solutions for scalability [24]. Since the technique is behavior based, it affords more opportunity for introducing rich interaction into game play than a geographic approach to interest management. Servers are allowed to communicate player actions to each other when needed. The model is based on approaches that rely on a Network Address Translator, or software equivalent, to allocate clients to servers efficiently using a number of load balancing techniques (e.g. round robin).

Vleeschawer et al. introduce a micro cell approach to balance load [23]. Instead of only managing one zone or cell, each server is responsible for a cluster of micro cells. Different algorithms are presented and tested to show how micro cells can be clustered to reflect player distribution in a map. In this way, an optimal configuration of server to micro cells cluster is reached to perfectly balance load. However, this clustering is only

applicable when the game first loads. Thus, taking a few seconds for the algorithms to reach an optimal solution is not a problem. Still, micro cell clustering needs to be done in real-time, and it is time consuming and impractical to run such algorithms dynamically in the simulation. Needless to say that, to reach an optimal mapping of clusters to servers, one should assume global knowledge of load and events in the map, which is inappropriate in distributed environments [25].

Chen et al. propose a locality aware dynamic load management through adaptive zone to server remapping as a solution to the load management problem [25]. They use a parallel incremental graph partitioning algorithm to divide the load among distributed servers. The Distributed Virtual Environment (DVE) system is mapped onto a graph G = (V, E), in which V represents all avatars in the system and E contains all edges. An edge represents that there is communication between two avatars. Participants in the DVE will be migrated according to the result of the graph partitioning algorithm.

[25]'s proposed algorithm can be classified as a load balancing algorithm: it sheds the load of an overloaded server to a neighbouring or remote server by doing a search to find the best candidate with the lightest load. Their algorithm involves a third party scheduler that decides which server gets the load, dispatches the decision to both the original and new server, and coordinates the sequences of events that need to be carried out to successfully complete the shedding process.

Chapter 3

Design of the Zone Management Architecture

3.1 A Topology-Aware ALM Protocol

For the reasons discussed in Sections 2.1 and 2.4.1, we introduce the concept of mapping the underlying network's parameters (bandwidth, end-to-end delay...) to an ALM overlay network. We base our protocol on HDSP (Section 2.4.1), use its ALM tree structure, and significantly expand it into a topology-aware protocol.

3.1.1 Fault Tolerance Using Backup Parents

To keep the simulation experience smooth and without any perceived delays or "glitches", a node maintains a spare connection with a secondary parent, called the backup parent. This backup parent is selected at the same time as selecting the primary parent, using the criteria of second-best end-to-end delay (or hops). This way, the node can quickly switch to the back-up parent if needed (e.g., the parent computer crashes) to maintain the stability of the tree and the flow of the application.

3.1.2 Mapping Bandwidth to Node Out-Degrees

End-hosts form the structure of the application layer multicast tree. The feasibility of carrying collaboration data over the ALM depends on whether or not there is available bandwidth at the end hosts. Usually, end hosts have asymmetric downloading and uploading speeds. Moreover, the heterogeneity of outgoing bandwidth of end hosts forces

protocols to consider realistic degree assignment. It may happen that a user has zero out degree (number of descendents it can support), i.e. pure receiver. But our case is quite different. These ALM structures usually carry short but frequent messages. From a practical perspective, the asymmetric bandwidth fact cannot be ignored and is taken into consideration when assigning the out degrees to nodes during implementation. It reflects the maximum bandwidth a node can provide.

3.1.3 Mapping Node Joining to End-to-End Delay

Node joining should be a function of end-to-end delay because of the real-time nature of the application. But due to the synchronization problems at nodes, one can use hop count to select a parent. Every joining node makes a request to the hybrid node. Each hybrid node keeps track of the entire logical topology of a particular zone. So, a hybrid node can send a response to the joining node containing a list of potential parents without consulting all of its descendants (figure 7). It is up to the new comer to select the best parent based on either hop count or end-to-end delay. The parent of the joined node informs the hybrid node about this decision. This centralized architecture may face the problem of scalability depending upon the size of the zone (in terms of number of nodes in a zone).

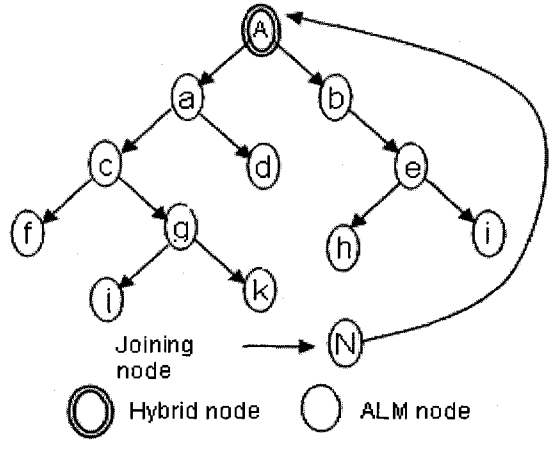


Figure 7. Node joining phase

3.1.4 Maintaining the Mapping

At the beginning, a new comer would not mind a short delay in joining the system and this makes mapping of node joining to end-to-end delay possible. During the simulation, however, nodes are always navigating the map and changing zones, thus having to disconnect from the ALM tree in their previous zone and connecting to the one in the new zone. If, upon connecting to a new ALM tree when switching zones, the node does not do any end-to-end delay measurement, then the ALM trees will seize to reflect the end-to-end delay overtime. This will adversely affect the performance. So we propose that such mapping be maintained throughout the game; i.e., when a node disconnects from a zone and connects to a new zone, it should connect to the best parent according to end-end delay with potential parents.

3.1.5 Node Departure

When a node decides to leave a zone, it has to disconnect from its parent and also leave its descendants parentless. Departures can be divided into graceful departures and ungraceful ones. In a graceful departure, a leaving node notifies all of the peers it is connected to, so the nodes can update their states and reconstruct the tree. In an ungraceful situation, the node just leaves without any notifications. For the latter, "keep alive" messages between a parent and its children are used in conjunction with a timer. If the timer elapses and the node does not receive any acknowledgment, it can assume that the other party has departed.

3.1.6 Underlying Transport Protocol

Two transport protocols available are Transmission Control Protocol (TCP) and User datagram protocol (UDP). Our protocol focuses on collaborative simulations that need

short frequent messages. Losing one of these frequent messages every now and then does not hamper the performance of the protocol significantly since the state synchronizes itself with the next message, as long as losses are not continuous and for long periods. It is therefore best for our protocol to use UDP since it is fast and satisfies the protocol's objective. Acknowledgement-based reliable UDP is used for key messages that require guaranteed delivery, similar to [27]. Although TCP takes care of such reliability, it imposes unnecessary overhead on the performance due to its session initialization and congestion control functionalities.

3.2 Hexagonal Zoning: An Efficient Approach to Defining Interest Area

Geographical zoning is specifically used to define interest in this thesis. Hexagonal zoning has been widely used in game applications and adopted by AoIM papers ([21]'s MOPAR and [33]'s NPSNET). It is also used by cellular networks where each cell, or hexagon, has a base station that the cellular units in that cell connect to. Each of these zones gets allocated a hybrid node which constructs an overlay network and manages the nodes that are in its zone. This will achieve a high degree of message isolation between zones. What is unique about hexagons is that they are regular shapes, which makes it possible to have a multiple zone layout without any gaps. Also, considering a node with a radius of visibility, the maximum number of different zones covered at a time is a 3, as opposed to 4 for squares and 6 for triangles (figure 8).

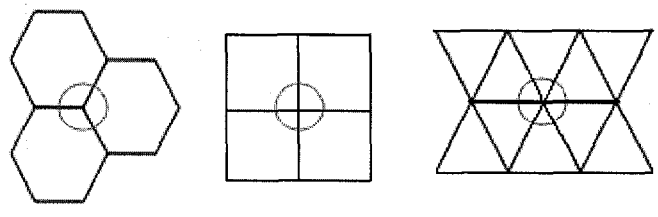


Figure 8. Number of zones covered by a visibility circle in a hexagon, square, and triangle layout

3.2.1 Check-in and Check-out radii

In a military simulation, for example, a tank unit might often be moving back and forth at the boundary of two zones. In Valve Software's Half-Life Counter-Strike online game, also, a FPS avatar might recurrently enter and leave a combat zone to shoot and avoid to be shot respectively. This is the "bouncing" problem discussed in Section 2.2.4. Depending on the application, nodes can have very high speeds and can therefore move very frequently at the boundary of two zones. Very LargeVirtual EnvironmenTs' Area of Interest Management [36] can be implemented to avoid the problem of a node's frequent movement around a zone. This approach uses a check-in (inner) boundary and a check-out (outer) boundary, as shown in figure 9. A moving node connects to a given zone when it crosses the check-in boundary of the zone, and disconnects from the zone when it crosses the check-out boundary. Increasing the difference between the inner and outer radii makes the "common area" (area bounded by the radii) larger, thus decreasing the number of times a node disconnects and connects to a hybrid node. In this manner, when dividing the map into hexagons, we should introduce a "buffer zone" between adjacent zones to take care of this common area. In figure 9, from zone 1's perspective: a node moving from point A to zone 2 and then back to point B will only join zone 2 after it checks out and will join back to zone 1 after it checks in.

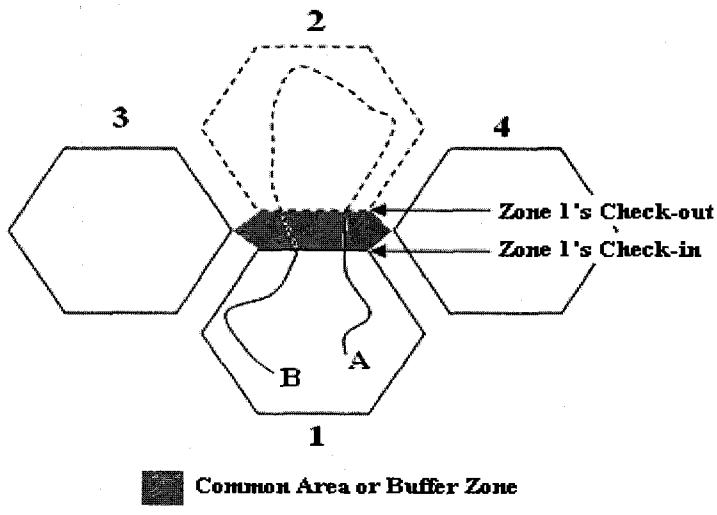


Figure 9. Hexagonal zones with check-in and check-out radii

3.2.2 Intra-zone P2P Communication

We mentioned before that each zone gets allocated a special node called a hybrid node. This node is a non-player; kind of a stand alone server strategically placed by the simulation provider or the gaming company. This is also referred to as the "coordinator" in Knutson et al's P2P Support for Massively Multiplayer Games [1], or "master node" in Yu and Vuong's MOPAR [21], but is actually a player in those architectures. Being a player creates a problem when the master has to switch zones itself and a reelection process needs to take place. Although these protocols use dynamic hash tables to do the election process quickly, there is zone switching and slaves have to reconnect to the new master node, which makes for a delay. The performance is further decreased by the fact that all players inside the zone are doing this context switching. Our hybrid node, however, is a non-player and eliminates the zone switching and reelection problem, although at a cost of dedicating a hybrid node for each zone.

The hybrid node accepts requests from players joining the zone and builds an ALM tree: each node relays message in a multicast fashion. The tree topology is chosen because it shifts the architecture from a server-client one to a P2P one. When a node switches from one zone to another, only its children will have to reconnect, and the tree remains stable. In figure 10, all nodes except for 11 and 12 are connected in a P2P manner. If node 2 leaves, only node 4 and 5 will have to reconnect (sub-tree for which 4 is the parent remains intact).

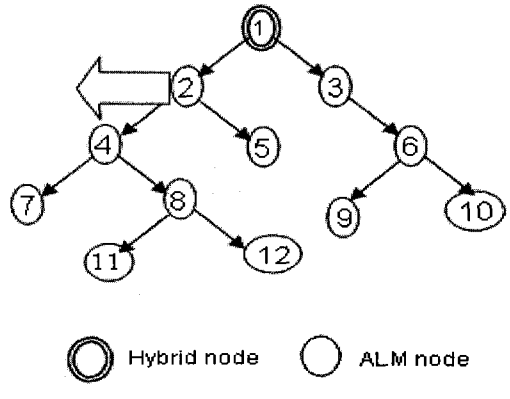


Figure 10. P2P-based tree structure

Another important point is the number of levels that can be supported in the tree. Each level adds an end-to-end delay to the message propagation among peers. As discussed in Section 2.2.1, the number of levels that can be supported is directly related to the network lag threshold that the virtual environment requirements impose. The performance analysis section will show our results on the number of levels that we could support.

3.2.3 Visibility-Based Inter-Zone Communication

How nodes communicate in a zone (intra zone communication) is explained in Section 3.2.2. As mentioned earlier, users should have a continuous view of the virtual world; i.e., dividing the map into zones should not restrict the view of a node. If one is to give more variations to units, like giving a tank radar capability to see beyond other tanks or a plane flying at an altitude having a larger visibility, then in such cases, the nodes need to communicate with other zones. Other visibility issues arise when the nature of the simulation map imposes restrictions. For instance, even though a unit's visibility is to a certain range, and in the actual map there is a geographical restriction, such as a mountain, or a river, then it is unnecessary to send massages to a node which is behind such restrictions. Thus, visibility itself changes dynamically due to map geography. This requires an appropriate physics engines for handling computer graphics, which MMOGs today have. We consider a simple graphical model and leave the geography restriction to the application. Here, we propose making node-specific connections with other hybrid nodes. So if a node has a visibility more than its own zone, it receives necessary messages through a specific communication channel linking it to the other zone's hybrid node. In figure 11, H1 is the hybrid node responsible for zone 1; H2 is the hybrid node responsible for the zone 2. Node X in Zone 1 has a visibility that exceeds its own zone. The dashed line is the connection with H2 to get messages from nodes it is interested in (black dots). In this figure, node X is considered a "foreign" node for hybrid node H2, and therefore not really a member of the ALM tree of zone 2.

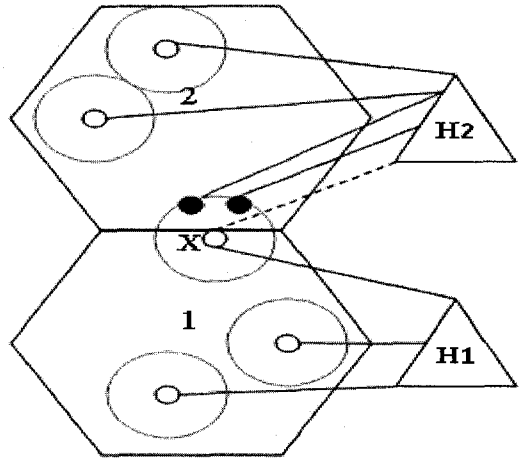


Figure 11. Node specific communication

The significance of such communication is that one is sure that the hybrid node will not leave its zone (non player), so such communication channel is reliable. It serves as a temporary channel until the node decides to join the adjacent zone (node becomes inside zone 2 in figure 11). Another characteristic of such channel is that it is only connected to the hybrid: the hybrid node will not consider it as a member of the ALM tree it is managing, so it will not be a potential parent for nodes newly joining zone 2. This is important because we are not sure that the node is totally interested in zone 2.

The initial approach to design visibility-driven messaging, or offer nodes a continuous view, is to have a hierarchal architecture. The first level of the hierarchy is the ALM tree maintained by a hybrid node in each zone, and on top, the hybrid nodes themselves form the top level of the hierarchy. If a node's visibility exceeds the border of its zone, then it can send an explicit message to its hybrid node, which will decide to which hybrid node it needs to propagate the message based on the logical position of the node and its radius of visibility.

Taking a closer look at this communication channel, we extract the following properties:

- The hybrid node does receive messages from the foreign node, but it does not relay these messages simply because its ALM members are not interested in the node. If one of the members is interested, because it is close enough to the foreign

node, then it has to contact the foreign node's hybrid node and form a similar connection.

- The messages that the foreign node is getting from the hybrid node will not be relayed to the foreign node's peers, again due to lack of interest.
- The foreign node should ideally receive only messages from the foreign ALM tree that it is interested in, and not all messages in the foreign zone as the latter would be an unnecessary overhead. For this reason, we introduce the filtering concept explained next.

3.2.4 Filtering Unwanted Traffic

To avoid unnecessary message propagation, we use a simple filtering policy. Depending on the size of a zone, it might be impractical for each node to get all of its peers' messages. If we are to move into a more visibility-driven approach, ideally, the messages that each node gets should be from the nodes that it actually "sees", even within the same zone. So, a node which is parent to other nodes can have information about their positions and radii of visibility. In this manner, the parent can decide whether or not to forward the messages based on the visibility interest of the destination node. This is what we call filtering.

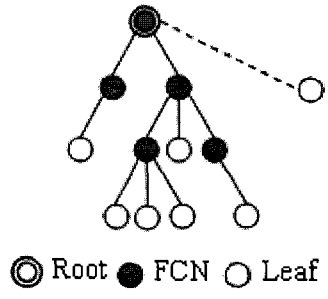


Figure 12. Position of FCN

A Filtering Capable Node (FCN) is one that can filter out specific messages based on the interest of its descendents. However, not all messages will be filtered out because a

parent may not know about the interest of the nodes two levels below it (not direct descendents). Any node that has a leaf is considered a FCN (figure 12). The dashed line in figure 12 represents specific hybrid connection with a node outside the zone but whose visibility crosses the zone. One can also see that a hybrid node that has the node-specific communication channel with a foreign node can be counted as a FCN, because it does not totally admit that node to its tree and is categorized as a leaf.

This scheme greatly reduces unnecessary message propagation in the ALM tree. This is really effective even for a balanced binary tree, where around 50% of nodes are leafs [37]. It is also necessary when a hybrid node is relaying messages to foreign nodes. Since the hybrid node is the root of the tree, it can have the information about all of the nodes in its zone, thus differentiating between those that lie in the visibility of the foreign node or not. This will make sure that not all messages from one ALM tree will be propagated to another ALM tree upon the node-specific connections between hybrids and foreign nodes.

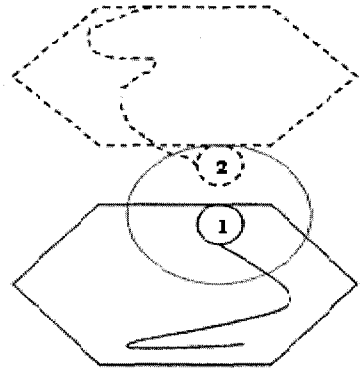


Figure 13. Hidden node problem

3.2.5 Hidden Node Problem

Consider figure 13 in which Node 1 denotes the node which is approaching the dashed zone but still connected to the solid zone. Its visibility, however, crosses the check-in radius of its own zone as explained in Section 3.2.1. Node 2 denotes a node in the adjacent zone that still has not checked out of its own zone. Node 1 needs to see node

2 and perhaps (but not necessarily) vice versa. The buffer zone can therefore become a margin for nodes to be "hidden" from each other. To solve this problem, node 1 needs to make the connection with the adjacent zone's hybrid node as soon as its visibility exceeds its own zone's check-in radius. This way, all foreign nodes in the buffer zone, which are still looked after by their own hybrid nodes, can be seen.

3.3 A Layered Partitioning Approach to Dynamic Load Balancing

[23]'s micro cell clustering approach explained in Section 2.4.3 needs to be done in real-time, and it is time consuming and impractical to run such algorithms dynamically in the simulation. Needless to say that, to reach an optimal mapping of clusters to servers, we should assume global knowledge of load and events in the map, which is inappropriate in distributed environments [25]. Furthermore, whenever computation is involved and coordination is needed to make a decision on balancing among a set of distributed servers, we face the old problem of parallel computing: how to choose who should do the computation, and if there is a need for a third party that coordinates between the servers. The cost of such computations and how badly they affect the performance should be considered. Whatever distributed architecture adopted should be justified for its validity. Our architecture targets simplicity and practicality to adhere to the real-time requirements in a distributed environment.

3.3.1 Service Level Agreements

Hot spots in the simulation, as explained in Section 2.3.1, disturb the quality of service of servers or peers in the ALM tree in our architecture, and response times become unacceptable. There is a need to define Service Level Agreements for such scenarios: what is the quality of service tolerated? [25] assumes 1 second for load balancing and 2 seconds for aggregation (algorithm used to restore locality) as an acceptable delay for the load management to stabilize. Such a delay is too much to endure

given the real-time requirement in CVEs. For our case, the service agreement should be in the range of the original end-to-end delay threshold (200 ms, if taken as a benchmark) [38].

3.3.2 Load Cost

The maximum number of nodes a tree can support is a function of the node out degree (related to bandwidth) and the number of levels supported. If more nodes are added to a tree that cannot sustain connecting more peers (adding an extra level or exceeding the bandwidth of the individual peers), then the delay is violated and we reach an overload. At this point the load balancing mechanism is triggered. The number of nodes in a tree is equal to the average node out-degree (since nodes have asymmetrical out-degrees) to the power of the number of levels in the tree plus 1, minus 1 [39], minus 1 (the hybrid node is not a user, so it is not counted in the overall load). So the load cost is given by:

$$N = D^{L+1} - 2$$

where D is the average node out-degree (since nodes have asymmetrical out-degrees) which translates to bandwidth, and L is the number of levels supported in the tree, which depends on the end-to-end delay threshold that we wish to support, and N is the maximum number of nodes supported in the zone's tree. The hybrid node can keep track of the number of nodes in its tree. If a zone reaches a number of nodes greater than N, then it is in overload mode and triggers a load balancing mechanism.

3.3.3 Two-layer Partitioning

The basic idea is that if we increase the granularity of the zones, we decrease the individual zones' sizes. By decreasing the sizes, the probability of having high user density decreases in each zone. This makes each zone responsible for a smaller number

of users (smaller load). Consider figure 14. Zone 1 denotes a hot spot. Once the hybrid server managing zone 1 detects a violation of the overload threshold, it triggers a load balancing mode for its zone. Each zone in itself has a zonal partitioning also, following a fixed partitioning configuration (fixed number of partitions). This is what we call a two-layer partitioning: partitioning of the map and partitioning of the individual zones. All the partitioning is done when the game first loads, so there is no need for in-game redrawing processing. Having a fixed configuration for the second layer partitioning, the adjacent hybrid servers know which partitions they are responsible for in case of overload. When a load balancing mode is triggered from zone 1, hybrid servers 2, 3, 4, 5, 6, and 7 quickly assume responsibility for managing nodes inside the partitions. Having this pattern in the partitioning is the key point of the design's contribution: everything is set for hot spots scenarios; the only dynamic thing occurring is the hot spot zone triggering the load balancing mode, and nodes subsequently migrating to their new servers. Since a big part of the hot spot is now managed by the adjacent zones, the load will be evenly shed among 6 adjacent servers. Furthermore, locality is preserved: only adjacent zones accept the shed load, so there will not be any need for too many node-specific connections.

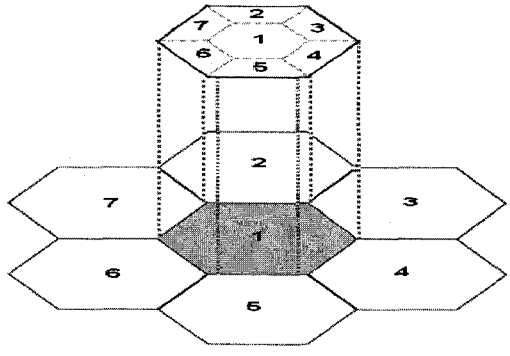


Figure 14. Two-layer partitioning

3.3.4 Rejecting a Load

One of the problems introduced by such architecture is the case when an adjacent server cannot accept to share the load from the overloaded server. Maybe the adjacent servers themselves are experiencing hot spots and cannot tolerate additional load. If a zone is a hot spot, and one of its adjacent zones triggers a load balancing mechanism for itself, it will simply reject the request and will not assume responsibility for its part.

It is important to discuss to what extent shedding of load to adjacent servers disrupts these servers. First, the adjacent servers are only sharing a small portion of the load: 1/8 of the area of the hotspot zone. Even though we cannot assume even distribution of nodes to zones, we still consider such a portion relatively small. Second, even if the load shed is small, but when added to the original load creates an overload, then the zone becomes a hot spot itself and can trigger load shedding to its adjacent zones. This might create a domino effect, but at one point, it will settle by adjacent zones which do not have a big load taking on the load. At the end of the day, a direct result for these hot spots is having zones that are partially or completely empty. So we can assume that an overload in one zone will only have a small disruption on adjacent overlay networks. Moreover, such scenario - when adjacent zones accept load and become overloaded themselves and then shed load to their adjacent zones - is a good picture of how hot spots should be managed: the original zone configuration will change and converge to manage the hot spot, the center of attention of the game or virtual environment at the moment.

3.3.5 Multiple Hot Spots

A single hot spot scenario is fairly simple to deal with as discussed before. A multiple hot spots scenario, on the other hand, becomes complicated and it is important to explain how the architecture works in this situation. Consider figure 15. Zones 5, 6 and 7 represent hot spots. The order in which each zone became a hot spot is as follows: 5, 6, then 9. When 5 becomes a hot spot, each of its adjacent servers assumes responsibility for

managing its partition as explained in the previous section, including servers 6 and 9. When 6 becomes a hotspot itself; however, it abandons managing its partition in zone 5, and server 5 reassumes management of the partition adjacent to zone 6. Same case applies to when zone 9 becomes a hot spot: server 9 abandons managing its partitions in zones 5 and 6, and server 5 and 6 reassume management of the partitions adjacent to zone 9. Figure 15 clearly shows the end result of the server to partition mapping in a multiple hot spot scenario. One can see that the overloaded servers are now managing smaller areas and how the original partitioning configuration adapts to the multiple hot spots situation. Notice how the original partitioning configuration converges into the second layer partitioning, keeping the same uniform shape. This regularity in partitioning is a critical point of the architecture and an important result of hexagonal zoning.

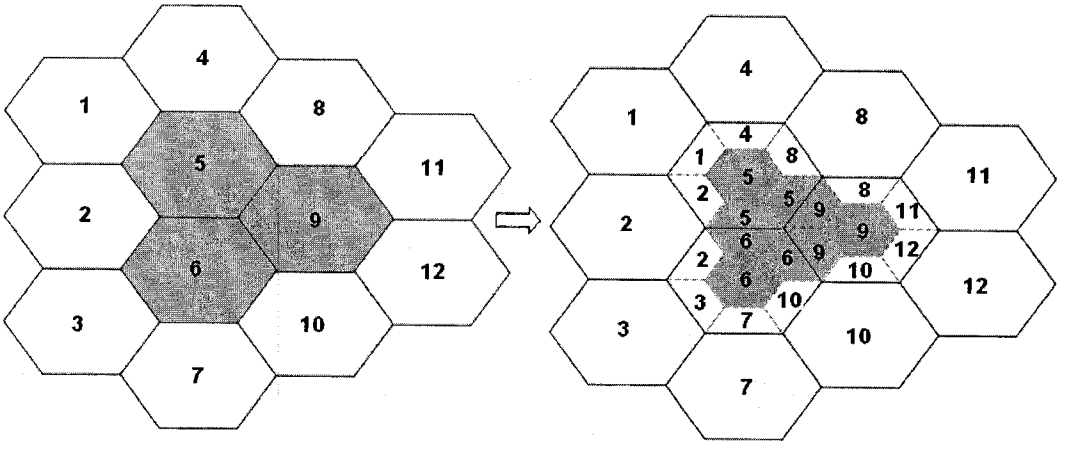


Figure 15. Multiple hot spots scenario

3.3.6 Undo Mode

It is intuitive that there should be a mode when the original partitioning is restored once a hot spot seizes to exist. This is important to preserve the regularity in the first-layer partitioning and reassign servers to zones evenly. We propose that once the hybrid server is back to less than 70% of its overload threshold, it can trigger an undo mode, so that nodes can migrate back to their original hybrid servers. The 30% margin is given to

prevent oscillations: it makes sure that the hot spot case will not reoccur in a short time (The 30% is a suggested value. This percentage can be changed depending on implementation and performance analysis of the undo mode). Since we cannot assume global knowledge of the load, hybrid servers frequently share lists of the nodes they are managing (position of node, which zone it belongs to...). When a hybrid server triggers a load balancing mechanism after its first layer partition becomes a hot spot, it frequently checks the cost of managing the other second layer partitions (restoring its original first layer partition). If, after adding this cost, the total load is less than 70%, it triggers the undo mode.

3.3.7 Discussion of Architecture

[24] discusses that there is a contradiction in the trend of research concerned with dynamic load management in MMOGs. On one hand, server side inter-communication is minimized to promote scalability and balance load. On the other hand, inter-server communications is relied on to alleviate process exhaustion due to crowding. It is very important, therefore, to choose an appropriate zone size to minimize inter-server communication, or node-specific connections in our case. [30] also argues that the Voronoi partitioning can lead to formation of very small zones, in case of clustering of nodes. This can lead to formation of zones which are smaller than ROI of a player. Such a small fragmentation is unnecessary and it may lead to additional communication overhead. For these reasons, zone sizes should surpass nodes' visibility or interaction ranges, so that inter-zone communication is temporary and minimal. When choosing the sizes for the our layered partitions, we try to balance the process of adjacent servers taking part in the load in the hot spot area, and preserving locality and isolation of nodes in one zone from another to reduce inter-server communication. This promotes message isolation and keeps interest well managed.

Another problem that our architecture has to deal with is the migrations of nodes from the original server managing the hotspot to the adjacent servers. Such problem is

considered a natural result of doing load balancing. Still, a predictive approach can be considered to triggering the load balancing mode before it happens to avoid accepting nodes in the first place and reduce migrations.

The architecture presented achieves four objectives: simplicity, practicality, flexible configuration, and preserving the ALM based tree architecture. No additional third party processors are added, such as schedulers that get to decide who accepts the load in an overload scenario. The architecture presented is only dynamic in switching modes: zoning is layered and only the switch between the zonal information is dynamic. Locality is preserved when only adjacent zones assume responsibility of the hot spot zone. So, in contrast to other complicated load management algorithms, our technique is simple and can achieve desirable results in real-time.

Chapter 4

Implementation of a Multi-User Combat Simulation Application

As proof of concept, a multi-user military simulation application was implemented. The simulation has 3 different units with different speeds, accelerations, and ranges of visibility: tanks (normal moving objects with limited visibility), planes (fast moving object with large visibility), and missiles (very fast moving objects). This differentiates units according to their behaviour in order to assess their impact on the simulation environment when collaborating with other units. The prototype allows us to assess how messaging performance is affected and scalability is enhanced when many nodes join, as well as subjectively testing the users' experience. Figure 16 represents a snapshot of the application, where the current user is flying an airplane over a set of tanks that are controlled by other users.

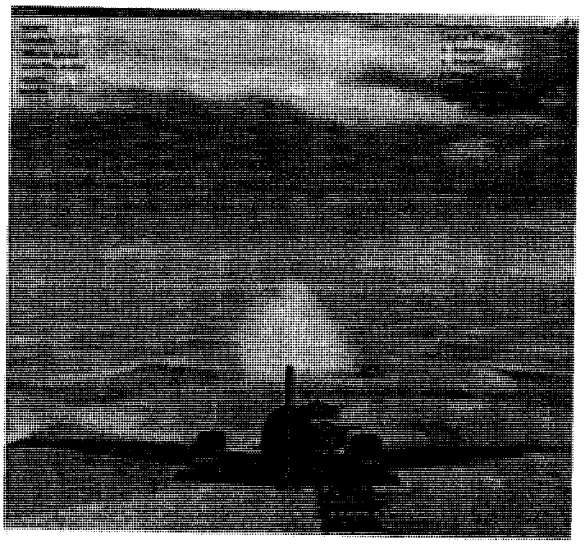


Figure 16. Snapshot of the simulation application

The application is implemented in C++, which provides an object-oriented platform for easy coding, reusability, and promoting scalability, combined with efficient execution in run-time. It is divided into a Communication Layer and a Graphics Layer. The Communication Layer takes care of sending and receiving messages. Since a node is a connected in a P2P manner, the node is both a server to its descendents and a client to its parent. This produces a complex multithreaded environment where threads communicate among each other by common data structures, such as arrays. The integral part of managing this layer is providing a synchronized and race conditions free environment for threads to communicate. The Communication Layer receives massages from other peers and stores them in a data structure that is shared with the Graphics Layer. These messages can then be interpreted by the Graphics Layer and translated to state updates. Finally, the message format has to have a coherent and standard structure to implement the ALM protocol designed earlier and to ensure the correct flow and interpretation of control messages that build and maintain the ALM trees and have centralized information at the hybrid node.

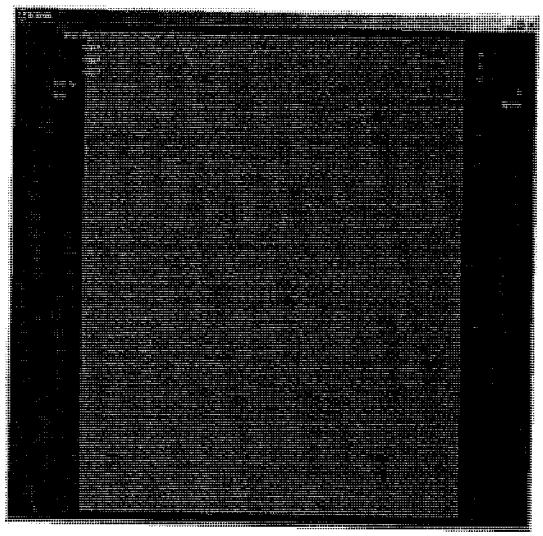


Figure 17. Bird's eye view screenshot of application

The implemented Graphics layer uses OpenSceneGraph (OSG), an open source high performance 3D graphics toolkit, to simulate the graphical virtual world in multiple adjacent hexagonal zones (figure 17). Based on the concept of SceneGraph, OSG

provides an object oriented framework in C++ on top of OpenGL and additional utilities for rapid development of graphics application. The usage of OSG frees the developer from implementing and optimizing low level graphics calls. Its key strengths are performance, scalability, productivity, and portability gains. In terms of performance, OSG supports easy customization of the drawing process, such as implementation of Continuous Level of Detail (CLOD) that was used in the creation and dynamic adjustment of the terrain. For productivity, the scene graph has a set of Node Kits which are separate libraries that were compiled and loaded in this application at runtime. Those libraries added support for particles systems (osgParticle), high quality anti-aliased text (osgText), special effects framework (osgFX), large scale geospatial terrain database generation (osgTerrain), and navigational light points (osgSim) that was used in the flight simulation. Being portable also adds value to the application, where the core scene graph has been designed to have minimal dependency on any specific platform. The multi-threading and database optimization provided by OSG enhances scalability when handling a large number of concurrent users. Thus, the manipulation and updates performed on the graphical form of entities are more efficient and will help in the achievement of a high quality simulation.

Although the application does not consider the services provided by HLA/RTI, it can be used to implement such services. This will put the application in better context when evaluating highly interoperable distributed simulations.

4.1 ALM Protocol

Let us take a look at the implemented ALM protocol in terms of message format, node joining, node departure, and tree structuring.

4.1.1 Message Format

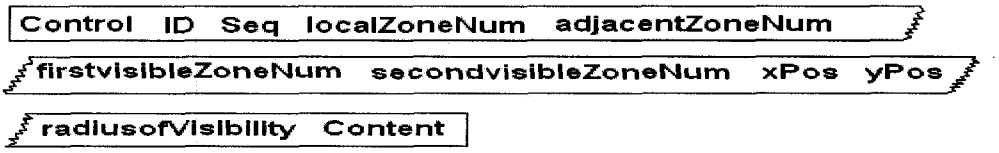


Figure 18. Message Format

Figure 18 represents the message syntax exchanged when nodes communicate their update messages. The semantics of the message are:

- Control: This is the header of the message. Since many control messages are needed in the protocol, this field is important for nodes to differentiate the type of request the client is sending. For instance, JOIN_REQUEST control means that the client is asking a node to accept it as its descendent. CHILD_LEFT control means that the sender is telling its parent that one of its descendent has disconnected.
- ID: Each node in the simulation is given a unique Identity number to differentiate nodes from each other.
- Seq: This is the sequence number of each message. It gets incremented with each message sent. It is used to keep track of the number of messages sent and identify a message.
- localZoneNum: This is the zone number a node is currently in. Based on the geographical position of the node (x, y, z coordinate system), a node is aware of the zone number it is in.
- adjacentZoneNum: This the number of the zone adjacent to the zone a node is currently in. After the introduction of a buffer zone between adjacent zones, it is essential for a node to also be aware when it is in the buffer zone of two zones, thus having information of both its original zone, and the one adjacent to it.
- firstvisibleZoneNum: This is the zone that the visibility or line of sight of a node covers outside its own zone. The concept of visibility will be explained in detail in section.....

- secondvisibleZoneNum: Since the visibility of a node can cover two zones maximum other than its own zone, there is a need to have another visible zone number.
- xPos, yPos, radiusofVisibility: These are self explanatory and are used when applying filtering on forwarding messages. The details will be explained in section....
- Content: This encapsulates all the information needed for a state update for a tank or plane's position, tank's turret orientation, speed....

It is important to have such a message format for better standardization as different graphical applications can use the same communication layer. Such a format is a contribution and a first step into promoting reusability and defining communication protocols and their flow control.

4.1.2 Node Joining

The proposed protocol tries to reflect the underlying network's physical topology onto the ALM trees. This improves the end-to-end delay on the separate unicast channels between nodes which makes it possible to support more levels in the tree structure.

In order to implement the mapping of node joining to end-to-end delay, a new comer will prompt the hybrid node for a list of potential parents in the tree, which are essentially nodes with available out degrees (available bandwidth). The joining node does a Round Trip Time (RTT) hand-shaking with the list of potential parents in parallel. Since the RTT measurements with each of the potential parents are done in the parallel, the time to join the tree is equal to the worth RTT. The joining node will then connect to the parent with which it has the smallest end-to-end delay.

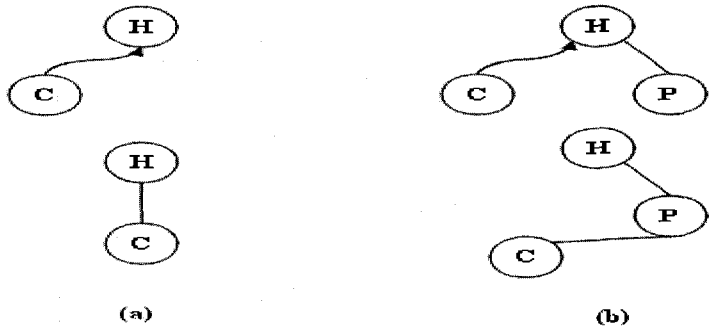


Figure 19. Node joining scenarios

In figure 19, assume that the Hybrid node has an out-degree of 2 (accepts at most 2 children). In figure 19(a), a new comer client C asks the Hybrid node H to join. Since H can accept children, C becomes a child to H. In figure 19(b), a new comer client C asks the Hybrid node H to join the tree consisting of H and a potential parent P. Since H can also accept one more child, it includes itself in the list of potential parents along with P. The new comer C becomes a child to P, assuming that it has a smaller end-to-end delay with P than with H. The message exchange for both scenarios is depicted in the sequence diagrams in figures 20 and 21.

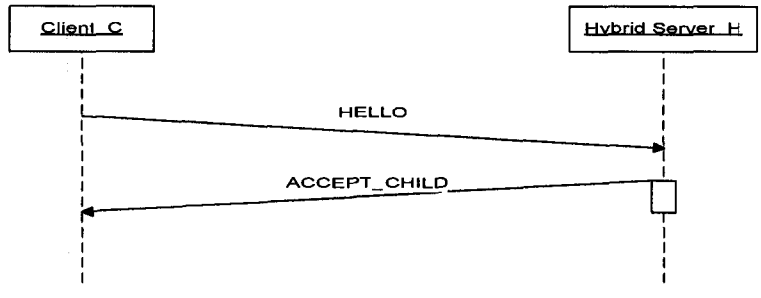


Figure 20. Sequence diagram for figure 19(a) node joining scenario

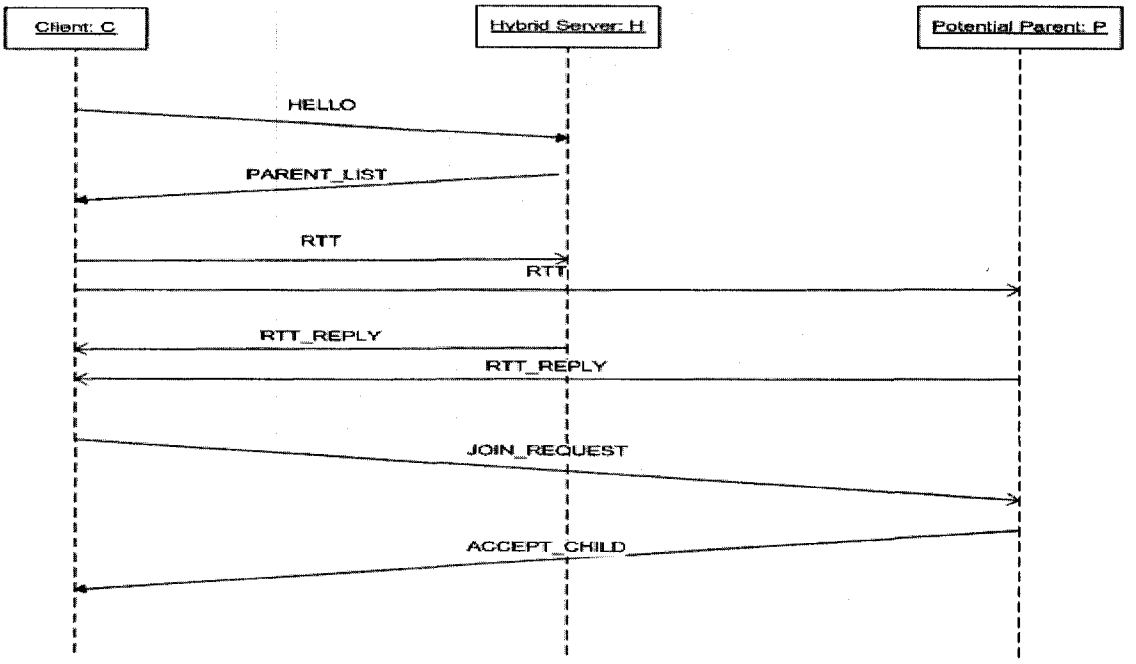


Figure 21. Sequence diagram for figure 19(b) node joining scenario

4.1.3 Maintaining the Mapping

In order for ALM trees to keep reflecting the end-to-end delay between users as nodes move from one zone to another, thus disconnecting from the original zone and connecting to the new zone, the mapping is maintained throughout the simulation. Upon the connection to a new ALM as a node moves into a zone, it performs the same node joining scheme explained above.

4.1.4 Node Departure

Graceful departures are implemented in the simulation. This happens when a node switches zones. Upon switching, the parent of the node is aware of such departure. The children of the leaving node are also aware of such event. This is because the zone information, its number, is found in the message. Three different scenarios had to be

implemented for different tree structures to ensure perfect tree reconstruction upon a node departure.

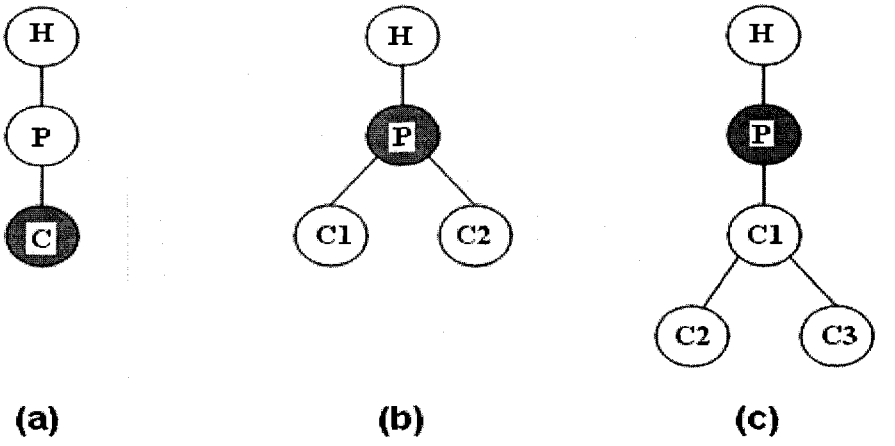


Figure 22. Node departures scenarios

Figure 22 represents the different node departure scenarios. The grey node denotes the departing node. H denotes the hybrid node. P denotes a parent. C denotes a child.

In figure 22(a), the departing node C is a leaf. This is the easiest case to handle because the node is not a parent to other nodes. The only thing that had to be implemented is update the information at the hybrid node H (departing node is not a member of the tree anymore). Since the parent P of the departing node may not be the hybrid node (as in figure 22(a)), the update of information at the hybrid node is possible because P sends a notification to H about the departure. No matter how many levels are in the tree, the parent will send a notification that will eventually reach the hybrid node. The message exchange for this scenario is depicted in figure 23.

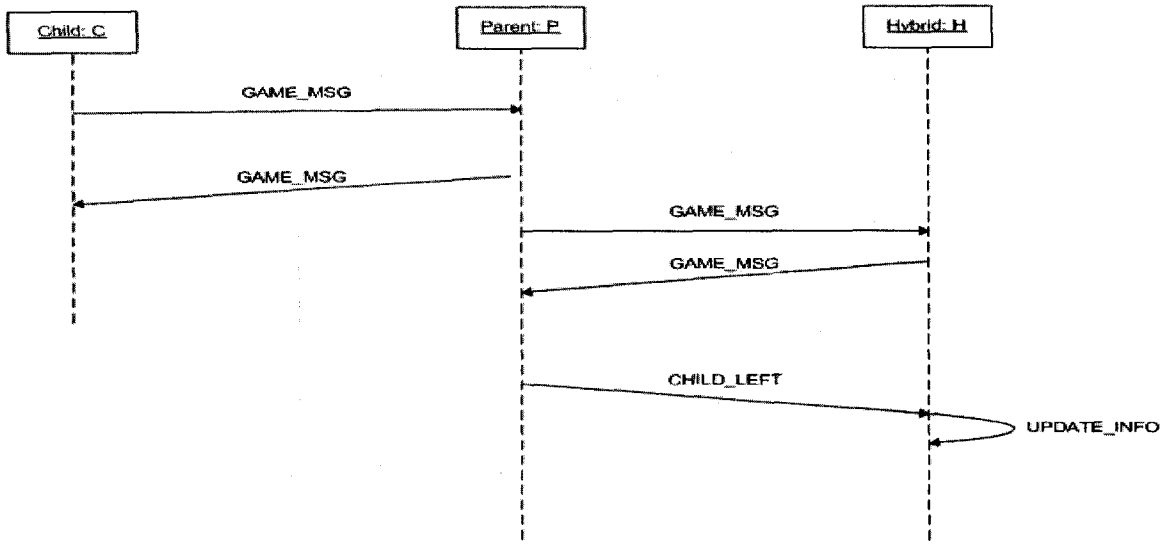


Figure 23. Sequence diagram for figure 22(a) node departure scenario

In figure 22(b), the departing node is parent P to two nodes C1 and C2. Before it departs, it notifies its children of the event (graceful departure). Each of the children, then, reconnects to the tree. The message exchange for this scenario is depicted in figure 24.

Lastly, in figure 22(c), only the direct child C1 of the departing node has to reconnect to the tree as explained before: the sub-tree (C2 and C3) of which the child C1 is the root remains intact. This is an important feature of tree structure, as in many cases, a departing node does not affect all of the children. The message exchange for this scenario is depicted in figure 25.

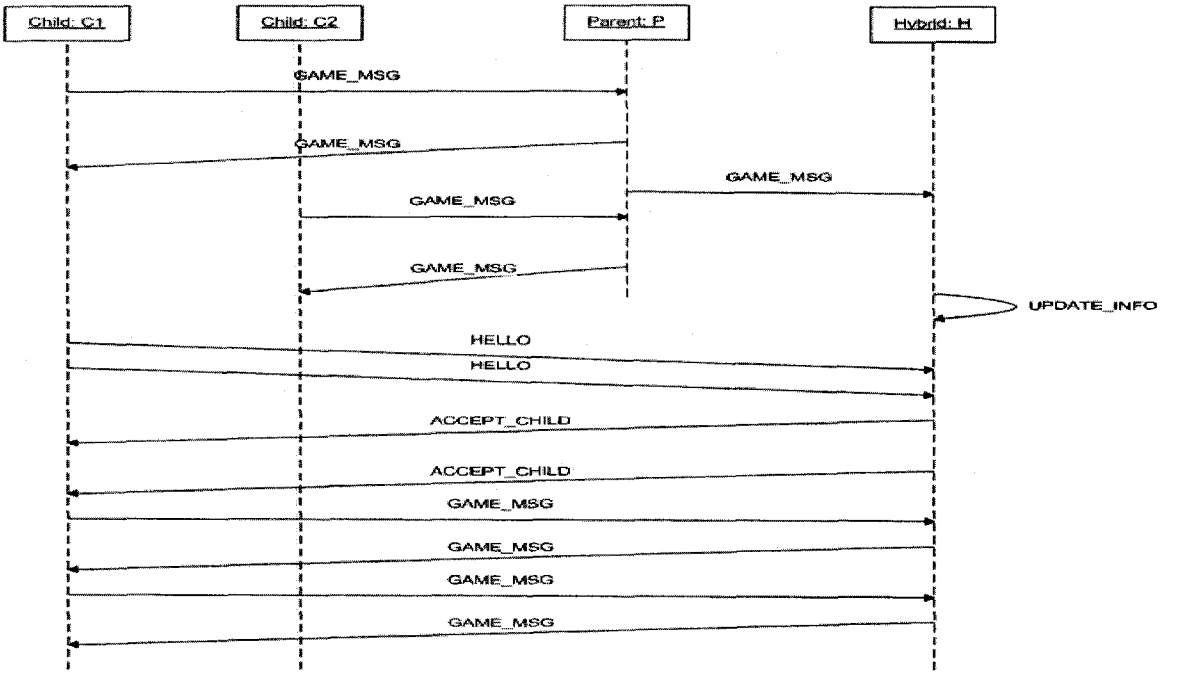


Figure 24. Sequence diagram for figure 22(b) node departure scenario

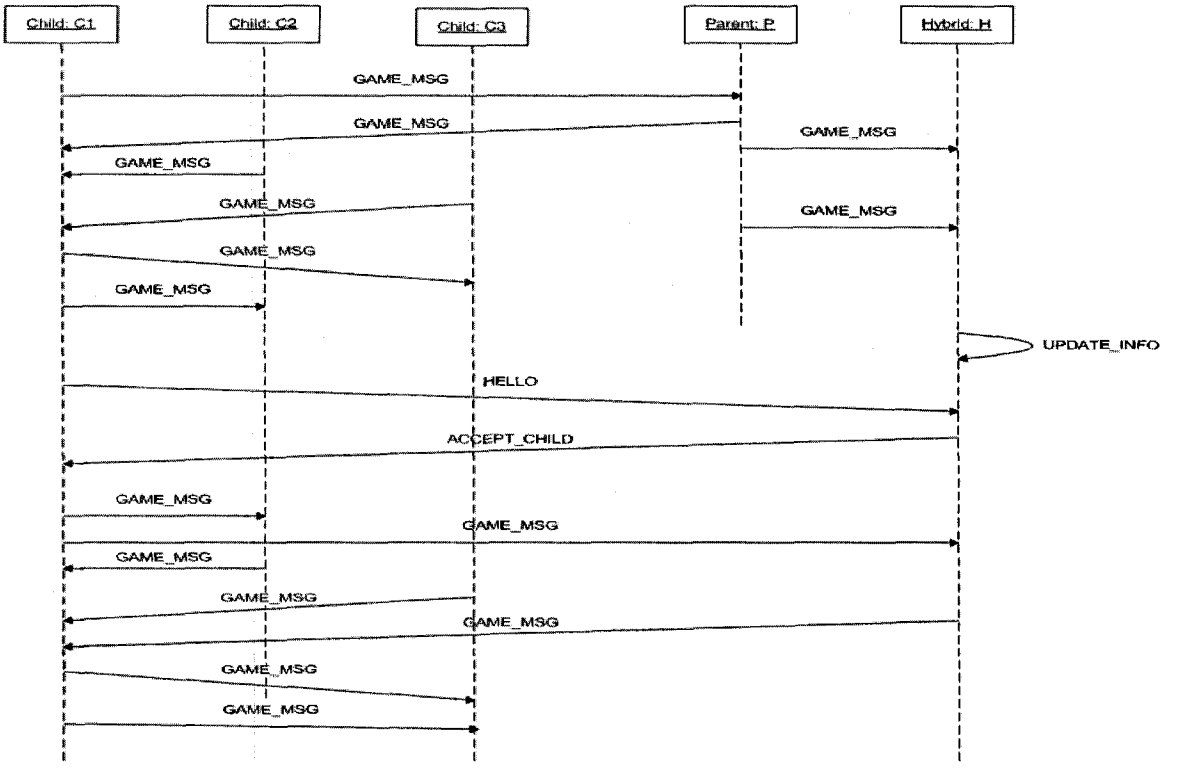


Figure 25. Sequence diagram for figure 22(c) node departure scenario

In an ungraceful departure, the node just leaves without any notifications. The effect of the departure on the tree was implemented as graceful departures (tree reconstruction explained above). Since both the parent and the children of the departing node are not aware of the event, keep alive messages are sent every second between peers. A timer is also used at each node. If the time elapses 3 times (3 keep-alive messages are missed), then a node detects a departure and acts accordingly. It was important, however, to ensure that the parent of a departing node detects the event before the departing node's children. Race conditions can occur between the parent and children, and it may happen that the children reconnect to the tree before the parent does. Since we are maintaining the mapping of end-to-end delay to node joining throughout the simulation (node joining algorithm is implemented upon node departures and reconnections), the parent has to reconnect first in order for the hybrid node to include it in the parent list when the descendents reconnect. For this reason, the elapsed time for the parent to detect an ungraceful departure was less (2 missed keep-alive messages).

4.1.5 Fault Tolerance Using Backup Parents

As explained before, a backup parent is selected at the same time as selecting the primary parent, using the criterion of second-best end-to-end delay. The joining keeps an inactive connection with the backup parent. If the primary parent disconnects, the child activates the connection with the backup parent. This is essential as it reduces the time needed to rejoin the tree (no need to redo the RTT handshaking for node joining), also maintaining the end-to-end delay mapping. Although such node joining time can be tolerated in a LAN environment, the end-to-end delay in a WAN environment can be close to 100ms, which can seriously affect the flow of the application as nodes switch zones and trees reconstruct themselves.

4.2 Zone Based Messaging

4.2.1 Hexagonal Partitioning

The simulation map is divided into zones in order to effectively organize the entities and efficiently manage their liaisons. The number of zones can be manually set, and their configuration best fits the map to avoid unwanted gaps.

The map is divided into multiple, adjacent, hexagonal zones. The division of the map is performed by an algorithm that is based on spatial vectors concepts, such as Cartesian coordinates, spherical coordinates, polar coordinates, and other geometric rules. The algorithm is designed as general as possible and could be applied to create any polygon with number of sides equal or greater than 3. In our case, the polygon is a hexagon and the number of sides is 6. This algorithm can also be implemented on a square map (as in our implemented map) or rectangular map. It takes the number of the zones as an input. Each hexagonal zone is assigned a unique number.

In figure 26, the red square represents the map. Notice how a different number of hexagons give a different map to hexagon size ratio. Since the map is a square shape, some zones are not complete hexagons. For example, if the input to the algorithm is 6 zones, the outcome is 2 full zones, 2 semi-zones, and 2 small zones (figure 25(a)).

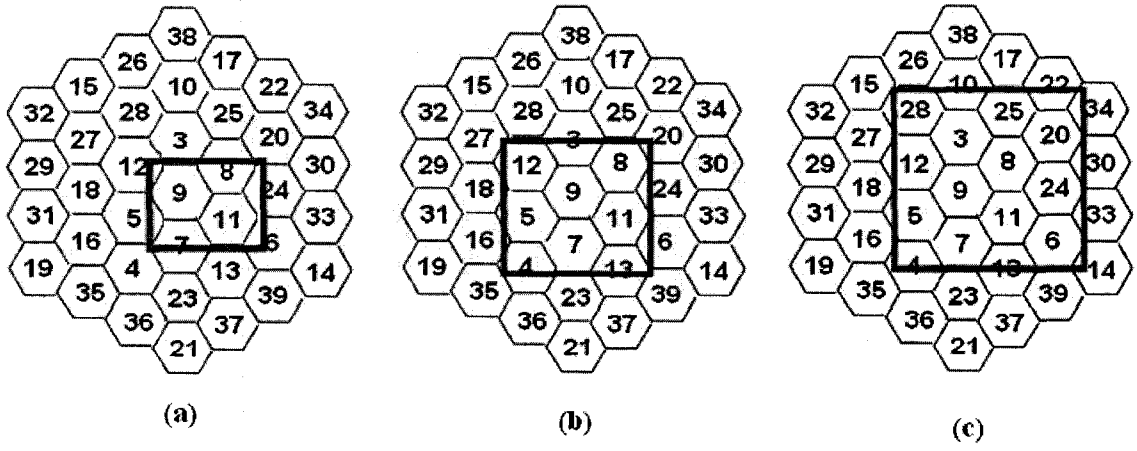


Figure 26. Different number of hexagons in the simulation map.

4.2.2 Position of a Node with respect to a Zone

To detect if a point is in a specific hexagon, the hexagon is divided into 4 triangles (figure 27). The position of an entity in terms of the current zone number is determined by using a triangle algorithm and the Cartesian analogy. Given the sides of the triangle, whether or not the position of a point is inside the triangle can be determined. This calculation requires a low number of arithmetic operations, improving efficiency the algorithm. This is desirable as an entity is constantly determining which zone it lies in. If the entity is in one of the triangles, then the entity is in the hexagon; if the entity is not in any of the triangles, then the entity is not in the hexagon.

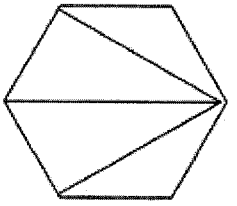


Figure 27. Hexagon is divided into 4 triangles.

The algorithm can be applied on all polygons with number of sides of 3 or more. As the number of the polygon sides increases, the resulting number of triangles increases. This makes the algorithm general and usable in different contexts.

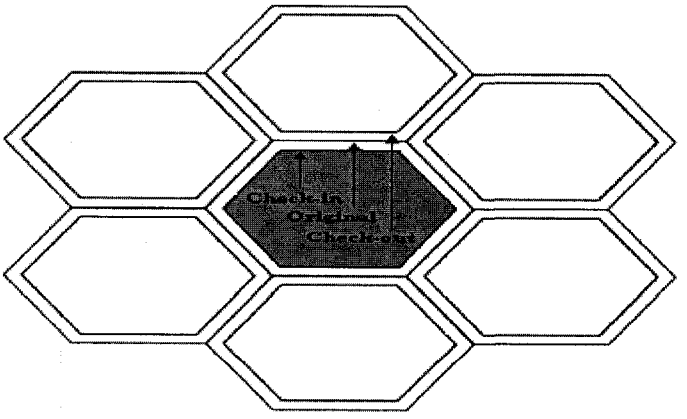


Figure 28. Buffer zone introduced between zones

4.2.3 Buffer Zone

Introducing a buffer zone between adjacent zones reduces the frequent connections and disconnections of a node navigating at the boundary of two or three zones. Consider the grey zone in figure 28. The original boundary is the boundary created when the map is divided into hexagons. A check-in and check-out boundary are introduced. Nodes coming to the grey zone only join the overlay network inside that zone when they cross the check-in boundary. Nodes leaving the grey zone only disconnect when they cross the check-out boundary. The check-in boundary of one zone is the check-out boundary of the adjacent zone, and vice-versa. The original boundary is not taken into consideration anymore as it is replaced by the buffer zone between the check-in and check-out radius.

4.3 Visibility Based Messaging

4.3.1 Visibility

Entities in the simulation are introduced a visibility range, which represents the line of sight of the entity. These ranges can be manually set, and entities with different behaviors are given different visibilities (a plane is given a visibility range larger than a tank). Assuming that units cannot see behind them, the visibility was implemented as a semi-circle. The semi-circle is then divided into four equal arcs producing five points on the semi-circle that are evenly separated from each other (figure 29). Each of these points is called a "visibility point".

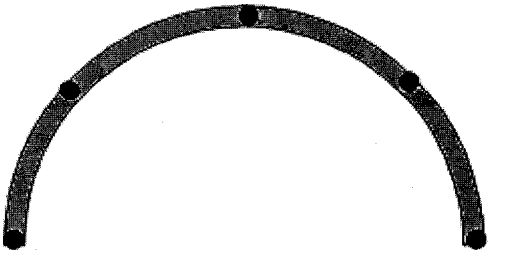


Figure 29. Visibility semi-circle

By checking the geographical position of each of these five visibility points and whether any of the points is located in a zone other than the local zone (zone the node belongs to), one can determine if a node can see beyond its zone, thus forming the node-specific connection with the adjacent zone's hybrid node. Doing the check for only five points gives accurate information on the line of sight with low processing overhead.

To combine the concept of the visibility range explained above and the range at which objects actually get rendered in the application (can be "seen"), OSG's *VisiblityGroup* class was introduced to define objects with a sphere as a bounding volume. All objects outside this bounding volume do not get rendered automatically by the OSG layer, which provides optimal performance in terms of rendering: only objects that are in the line of sight of a node get rendered.

Once a node's visibility range exceeds its own zone, it opens a temporary communication channel with the adjacent zone's hybrid node and starts receiving messages form nodes that it can actually "see". As soon the node moves entirely into the adjacent zone, the temporary channel is destroyed with the hybrid node and the node joins the overlay tree as explained in the node joining section

4.3.2 Hidden Node Problem

In order to solve the hidden node problem, a node forms a node-specific connection with the adjacent hybrid node as soon as its visibility range exceeds its own zone's check-in radius. If the node forms this node specific connection at the check-in boundary, all nodes in the buffer zone that are still connected to adjacent zones' trees can be seen (it starts receiving update messages from the adjacent hybrid node).

4.3.3 Message Filtering

Since filtering cannot be applied at any level of the tree (only the level that has leaf descendents), filtering was implemented as a prototype for selected nodes with leaf descendents. Since a node relays messages from its descendents in a multicast fashion, it stores its descendents' IDs and geographical positions (xPos and yPos message fields) and radii of visibility (radiusofVisibility message field) in a structure that is updated with every message relayed. When relaying a message from a certain node, the position of this node is checked whether it lies inside the visibility of the other descendents. Based on this check, the relaying node, or parent, decides whether or not to forward the message.

Chapter 5

Performance Measurement

Dividing the simulation map into multiple zones, it is intuitive that the number of nodes supported on the map is the maximum number of nodes a single tree could support multiplied by the number of zones. Thus, scalability is achieved when implementing the area of interest approach. But, such scalability comes with a cost; mainly the density of nodes in a zone and the instability that the individual zone trees will endure as nodes move from one zone to another. Implementing the backup parent policy improves the joining time upon nodes' departures and the overall trees stability. Implementing node specific communications with foreign hybrid nodes for nodes whose visibility crosses their own zone provides a continuous view for users. Problems such as a node frequently moving at the boundaries of two zones and the hidden node problem were taken into consideration when implementing the system to ensure a smooth user experience. Finally, filtering was implemented for FCNs to change the message propagation from a zone-based one into a more visibility-driven one, to prevent unnecessary propagation of messages from different zones (for node-specific connections with foreign nodes), and to decrease the messaging overhead. The following tests were run to measure the effectiveness of our ALM protocol. Also, specific simulation scenarios were stressed upon to study the specific problems discussed in Chapter 2 and solutions designed in Chapter 3.

5.1 Results of the Performance Tests

5.1.1 Node Joining Algorithm Study and End-to-End Delay Measurement

A simple scenario was setup when a node had to choose between two parents which were home internet users, both using cable access (Rogers® ISP). The home computers all had 512 Mega Bytes Random Access Memory (RAM), as it has been shown that the RAM can have an effect on how fast a message is processed. The new comer calculated the delay 10 times. The average delay is shown in Table 1. The new comer successfully joined the parent with the smaller end-to-end delay.

Table 1. Reflecting the End-to-end delay on the ALM overlay network

	Possible Parent 1	Possible Parent 2
Average End-to-end Delay (ms)	20	38

Although the above table does not show the massiveness that is to be supported, it does show that the protocol works in terms of reflecting the end-to-end delay on the tree structure.

5.1.2 Node Joining and Backup Parent Activation Time Measurement

As mentioned earlier, introducing zones into the simulation map creates a problem of nodes having to disconnect from one overlay network and join another as they switch zones. This is what is called a graceful node departure, and it happens often in a simulation. Such departure destabilizes the tree and causes it to reconstruct with peers having to reconnect to resume exchanging state updates. Such reconstruction entails two events: the departing node has to join the new overlay network, and the orphaned descendents have to rejoin their overlay network. Such joining is implemented as the node joining algorithm proposed in the design since we have established that end-to-end

delay to node joining mapping is to be maintained throughout the simulation. This is time consuming as there is a need to prompt for the parent list, do RTT handshaking with each of the potential parents, and then request a join to the best parent. The first event, when the departing node has to join the new overlay, is tolerable as it is only the departing node itself getting affected. The nodes depending on the departing node, however, such as its descendents, can be quite many, so it is more important for these to have a minimal node joining time. Since the backup parent policy is implemented, such node joining time can be reduced, and the overall simulation flow improved.

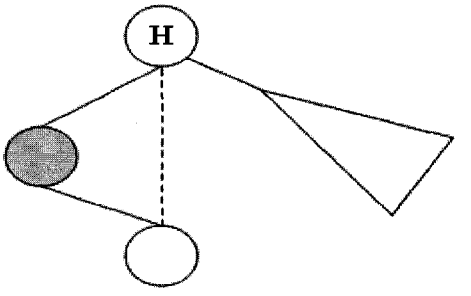


Figure 30. Backup Parent Policy.

A scenario was run to measure the node joining time and backup parent policy. This gives an idea on how bad it is to the trees when nodes switch zone, and how well it is improved by having a backup parent. In figure 30, the grey node is a departing node. The triangle represents a sub tree of the original overlay tree connected to the hybrid node.

The descendent of the departing node has to reconnect to the tree. In the first case, the descendent rejoins the tree as in a regular node joining scenario. The time at which the node departs to the time at which the descendent resumes receiving state update message is recorded as Node Joining Time. In the second case, the descendent keeps an inactive communication channel with a backup parent (hybrid node in our case). The time at which the node departs to the time at which the descendent activates the communication channel and resumes receiving state update messages is recorded as Backup Parent Activation Time.

The above two scenarios were run 10 times in a LAN environment, and 10 times in a WAN (Rogers® internet home users – same ISP) environment. The average times are shown in Table 2.

Table 2. Backup Parent Policy

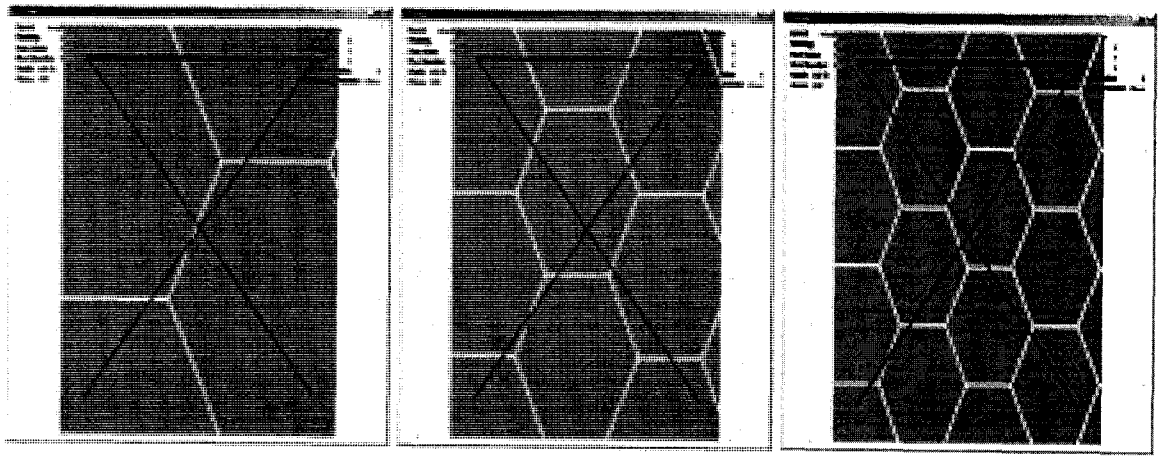
	LAN	WAN
Average Node Joining Time (ms)	41	93
Average Backup Parent Activation Time (ms)	16	32

The results show that in a LAN environment, the joining time without back parent policy is 41 ms, which can be quite tolerated in the simulation, taking the 200 ms as threshold. In a WAN environment, however, the delay is 93 ms, which, when added to the original delay it takes for a state update message to be communicated, can easily exceed the 200 ms threshold. This negatively affects the performance of all of the descendents of departing node.

The Backup Parent policy improves the joining time by almost a third. This is a direct result of having to communicate only one message to activate the communication with the backup parent (Hello message), as opposed to communicate 3 messages in a regular node joining (Hello message, RTT message, and Join Request message).

5.1.3 Zone-Based Messaging Study

To study this, we ran a random path traversal on the same map with different zone configurations, as shown in figure 31, where the black line presents the path, and the white lines present the boundaries of the zones. This is to stress on the scenario where a node traverses several zones. Let N be the maximum number of nodes supported in one zone. It is a function of nodes' out degree and number of levels supported in the tree. Table 3 summarizes the results.



(a) (b) (c)

Figure 31. (a) 4-zones lay-out. (b) 9-zones lay-out. (c) 16-zones lay-out

Table 3. Number of nodes supported vs. number of disconnections and connections.

	Number of nodes supported	Number of disconnections/ connections
4-zones layout	4N	5
9-zones layout	9N	9
16-zones layout	16N	13

5.1.4 Buffer Zone Study

A test was run to study how the size of the common area affects the number of disconnections and connections. Three random traversals were conducted at the boundary of adjacent zones, as shown in figure 32, where the black, white, and dotted curves represent the random traversals. The height of the common area was then varied as a percentage of the hexagonal zone's height. Figure 33 summarizes the results. The graph clearly demonstrates how the buffer zone significantly decreases the times the user has to disconnect from one zone and connect to another as it moves at the boundary of two adjacent zones. Also, the larger the height of the buffer zone (10% of the zone's height in our case), the less frequent disconnections and connections happen.

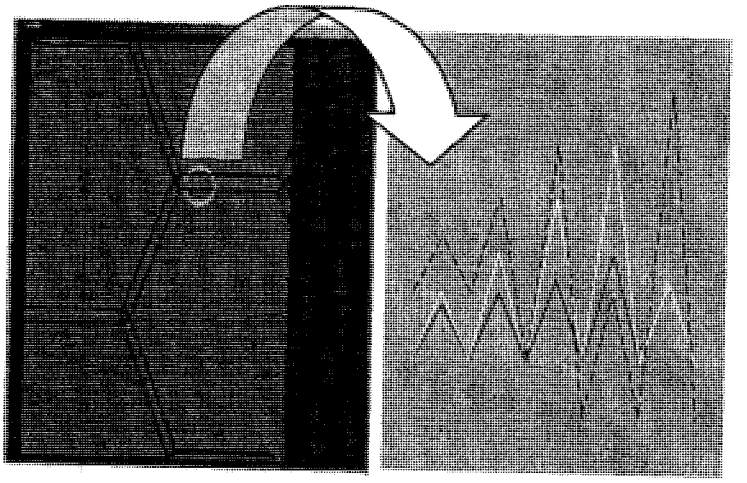


Figure 32. Common area with 3 random paths

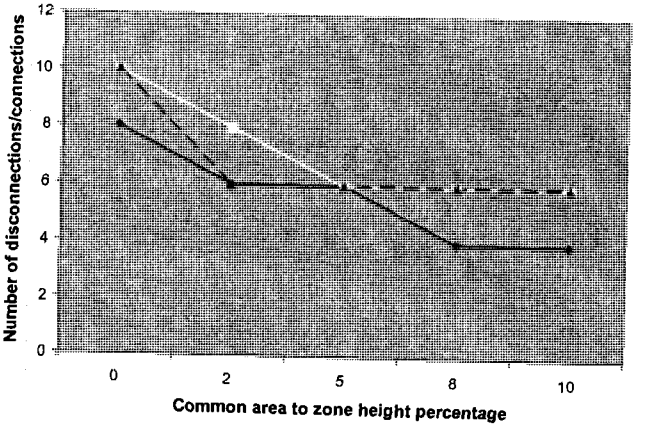


Figure 33. Common area performance for random paths black, white, and dotted

5.1.5 Message Filtering Study

To test message filtering, three users joined the simulation and they connected according figure 34's tree structure (the grey node represents the FCN.). User 1, being a parent of the other two leaf users, was marked as a FCN and given filtering abilities.

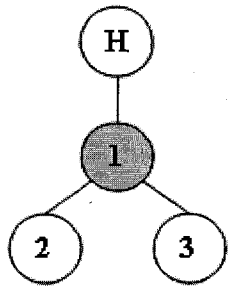


Figure 34. Tree structure to test filtering.

All users had initial positions in the same zone, as shown in figure 35, such that the distance between each and everyone is the same. Note that there is no relationship between a node's position in the tree and its geographical position. Each user has a visibility range, indicated by a circle encompassing the user. The following scenarios were run twice; once with 1 being a FCN, and another time without any filtering:

- User 1 moves from its initial position to user 2's position, then 3, then back to its initial position.
- User 2 moves from its initial position to user 3's position, then 1, then back to its initial position.
- User 3 moves from its initial position to user 1's position, then 2, then back to its initial position.

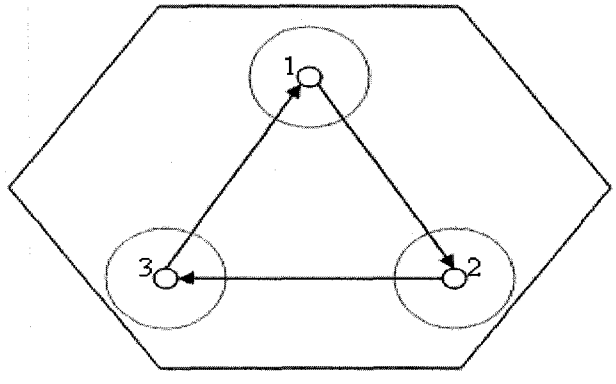


Figure 35. User's positions and paths run

The significance of these paths is that only a portion of each path taken lies in the visibility range of each user. For instance, if user 1 is moving towards 2, it will not send

any update messages to 3 and will only send messages to 2 when it enters its visibility range. Same thing applies for user 2 and 3's path traversal. Since user 1 has filtering abilities, it will save on its outbound bandwidth: the number of messages it sends to 2 and 3. Players 2 and 3, on the other hand, will save on their inbound bandwidth: the number of messages they receive. Table 4 summarizes the results.

Table 4. Message reduction when enabling filtering

	Without Filtering	With Filtering
Outbound bandwidth for user 1	1660	223
Inbound bandwidth for user 2	800	112
Inbound bandwidth for user 3	861	111

Results show that for user 1, there was an 87% reduction in messages sent. Users 2 and 3 had an 86% and an 87% reduction in messages received respectively. Of course, the number of messages sent out by each user varies a bit because we cannot expect the user to take the exact same path as it moves from one point to another point. However, the idea here is to show the message reduction as opposed to number of messages in total.

5.2 Analysis of the Performance Results

5.2.1 Number of Tree Levels Supported

HDSP's [27] performance analysis clearly concludes that in case nodes are under the same ISP, a three level tree with nodes having an out degree of 1 should be easily supported and the expected delay from the Hybrid node to that 3rd level should be about 120-140 msec. A 4th level might also be supported with a slight violation of the 200 msec threshold. In case nodes are under different ISPs, HDSP should be able to support the third level with the maximum average delay around the threshold (200msec) or less. However, a 4th level will likely not be supported. HDSP's conclusions are verified here:

Table 1 shows that nodes on the same ISP show a relatively small end-to-end delay. The node-joining approach explained in the previous section handles the problem when having different ISPs and sorts out the ALM tree's branches according to nodes on the same ISP or LAN. In other words, if a node using ISP1 is to choose between a parent on ISP1 and one on ISP2, it will definitely go with the former. As such, nodes using the same ISP or LAN will be on the same branch of the tree, having smaller end-to-end delay, thus supporting more levels. With each level added, the number of nodes that can be supported is increased in one zone and more so in the map as a whole. In reality, since about 50% of nodes have an out degree of zero (low-bandwidth, pure receivers), the number of levels supported is almost half of the 4 levels supported in the same ISP; so the expectation is a maximum of 2 levels supported in practice.

5.2.2 Message Isolation

Nodes on different zones have complete message isolation: messages from nodes in one zone only get propagated to nodes in the same zone, with the exception of foreign nodes. Being at liberty of choosing the size and the number of zones in the simulation map, scalability is achieved depending on those parameters. However, as Table 3 shows, scalability is improved with the side effect of having to deal with more disconnections, which will destabilize the ALM tree and will take some time to recover. To balance this, one must choose a zone size depending on the size of the map used and the number of nodes to be supported.

Another problem is the density of nodes in one zone as we have no control over several nodes moving at the same time into a zone the overlay tree of which cannot handle more users. The layered partitioning approach to dynamic load balancing serves as a design for future work implementation and performance analysis to solve this problem.

5.2.3 Trees Stability

The backup parent policy serves as a good mechanism to quickly restore a tree's stability upon a node's departure. The results show that in a LAN environment, the node joining time is tolerated to an extent. Given the distributed nature of the application though, one cannot always assume peers only connected on a LAN. In most cases, peers form a heterogeneous overlay network that is a hybrid of Internet and LAN users. This entails peers even connected from different ISPs, making the node joining time even worse, over 200 ms (HDSP [27] claims about 90ms end-to-end delay between nodes on different ISPs). This makes zone switching a serious problem. This can be avoided at the application level by making boundaries uninteresting in nature, so that a user does not perceive any state updates when switching zones. As for the descendents of the departing node, it could be best to consider overlay structures other than trees, to decrease the dependencies among nodes. This can be accomplished by adopting the architecture similar to those present in the Other Literature Review on ALM protocols in Section 2.4.1, such as the NICE protocol which has a strongly connected structure for control messages (node joining for instance), and a less connected structure for data delivery. Another option would be having spare links between peers for fault tolerance, as in VRing, which is the idea adopted in the backup parent policy.

5.2.4 Message Filtering and Other Issues

The suggested architecture for solving the hidden node problem and providing a continuous view for the player was implemented and tested. Hybrid nodes with node-specific connections with foreign nodes filter out the messages according to the node's visibility interest to prevent unnecessary propagation of messages from one zone to another. FCNs are enabled and filtering gave significant results in terms of message reduction and refining the concept of interest of a node from the zone it is in to what it "sees." Of course, FCNs can only filter out messages to its leaf children, so we did not completely achieve visibility-driven messaging. However, looking at the number of

levels that could be supported from the performance analysis, one can see that only the hybrid node will not be a FCN for its children. This means that all nodes in the first level are sending out messages according to the visibility interest of their children, and all nodes in the second level are receiving messages according to their interest. In order to achieve total data interest management (filtering data according to visibility or area of influence), again, other overlay structures could be adopted like server-client or proxy-based architecture on some levels to promote behavioural modeling *inside* one zone.

Chapter 6

Conclusion and Future Work

Scalability is achieved in massively multi-user distributed simulations by dividing the virtual world into multiple adjacent hexagonal zones in order to properly organize the entities and efficiently manage their connection. This achieves message isolation between zones and reduces the communication overhead, keeping the required timing constraints within the limit. It also manages interest according to geographic locality.

Reflecting the underlying network's topology onto the ALM overlay network is taken into consideration when a node joins the network to improve end-to-end delay. This makes the ALM structure congruent with the underlying network topology, improving the "goodness" of the data links in terms of metrics such as stress and stretch.

The problem of a node frequently bouncing in and out at the boundary of two zones is solved by introducing buffer zones between adjacent zones. The hidden node problem is also taken into consideration to sustain a continuous view and prevent cases when nodes can be hidden from each other as a result of introducing the buffer zone.

The continuous view is maintained for the player to ensure a perfect simulation experience by providing node-specific communication channels between foreign nodes interested in seeing beyond their own zones and adjacent hybrid nodes. Message filtering based on the positions and radius of visibility of players can be applied on such channels to only import messages that the foreign node is interested in (can see). Filtering is also applied on nodes in an overlay network of one zone to achieve visibility-driven messaging among peers.

The presented collaboration architecture was implemented in order to imagine real-life scenarios that can occur in a combat environment. The model is distributed such that each peer locally computes the virtual environment it is interacting with and receives updates messages from peers it is connected to. Having such a "thick" client, the

management of data in the massive environment becomes possible. The performance results and their analysis show that the implemented communication platform can have decent results.

The full realization of interest management cannot be achieved without modeling interactions and communications according to the behaviour of parties collaborating in a virtual environment. Regionalization of the world based on locality has its problems in limiting the view of parties, having to deal with migrations from one zone to another, and scenarios where one zone gets heavily populated. This affects the performance and violates the quality of service expected in such environments. Dynamic load balancing, which is both practical and simple, is needed to evenly distribute resources and remap nodes to servers or overlay networks managing areas of interest.

The designed and implemented system has its shortcomings. Even though data can be managed given the distributed nature of the application, it is hard to really evaluate the massiveness that is to be supported and assess to what extent the proposed design is valid. The robustness of the protocol and how well it scales are also at stake here since it is not put to test for message traffic, massive node departures, massive simultaneous node joining, etc. Even though it scales well compared to server-client or mesh-based structures, the tree structure has its disadvantages in creating many dependencies between peers, making the departure of one intolerable to the simulation experience, especially in WAN environments. Inter-server or cross-zonal communication can seriously hinder the message isolation achieved by zone messaging in some cases, making the geographic partitioning too rigid of a choice to manage interest. Finally, filtering messages according to visibility interest can face a scalability problem, such as the processing power required at the filtering node (a parent doing a check for 100 of its children for filtering each relayed message, for example).

Future work should focus on implementing and studying the performance of the load balancing design to solve the problem of having many users moving into a zone and heavily populating it. The interest of users can be refined by proposing a mathematical model to determine the best node in the best zone with a given interest vector. Node

joining should also reconsidered to sort entities according to their behaviour as the hybrid server opens multiple multicast channels to reduce structural reformation events among the entities.

Future contributions should focus on designing and implementing a predictive approach to load balancing to reduce migrations for peers, and to behaviour modeling in general. An interactivity factor can also be taken into account when measuring load and couple our area of interest management with better behaviour modeling. Finally, the tree topology as a overlay structure for connecting peers should be reassessed to move more into a structure that (1) has better control and quick convergence when detecting failure and resorting invariants (node departures for instance) and (2) give more power to model behaviour and reach the ideal interest management: communicating messages according to line of sight or region of influence.

References

[1] B. Knutsson, H. Lu, W. Xu, B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games", INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, Volume 1, 7-11 March 2004

[2] Blizzard Entertainment Press Release. World of Warcraft achieves new milestone with two million paying subscribers worldwide. In [Online] http://www.gamesindustry.biz/content_page.php?aid=9509

[3] T.Y. Hsiao, S.M. Yuan, "Practical Middleware for Massively Multiplayer Online Games", Internet Computing, IEEE, Volume 9, Issue 5, Sept.-Oct. 2005 Page(s): 47 - 54

[4] L. Ducheneaut, N. Yee, E. Nickell, R.J. Moore, "Alone Together?": Exploring the Social Dynamics of Massively Multiplayer Online Games", Proceedings of the SIGCHI conference on Human Factors in computing systems CHI '06, April 2006

[5] L. Klastrup, "Death Matters: Understanding Gameworld Experiences", Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology ACE '06, June 2006

[6] B. Blau, C.E. Hughes, M.J. Moshell, C. Lisle, "Networked Virtual Environments", Proc. ACM SGGRAPH, 1992, pp. 157-164

[7] K.S. Park and Robert V. Kenyon, "Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment", IEEE Virtual Reality (VR '99), Houston, Texas, March 1999.

[8] M. Pullen, "Reliable Multicast Network Transport for Distributed Virtual Simulation", Proc. IEEE Workshop on Distributed Interactive Simulations and Real-Time Applications, Greenbelt, Maryland, October 1999.

[9] S. Shirmohammadi and N.D. Georganas, "An End-to-End Communication Architecture for Collaborative Virtual Environments", Computer Networks, 2001.

[10] IEEE Standard for Distributed Interactive Simulation, Application Protocols, IEEE 1278-1995.

[11] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs", ACM Trans. on Computer Systems, 8(2):85--110, 1990.

[12] A. El-Sayed, V. Roca, and L. Mathy, "A survey of proposals for an alternative group communication service", IEEE Network Magazine, vol.17, no.1, pp.46–51, Jan-Feb. 2003.

[13] M. Hosseini, D.T. Ahmed, S. Shirmohammadi, and N.D. Georganas, "A Survey of Application-Layer Multicast Protocols", IEEE Communications Surveys and Tutorials, Vol. 9, No. 3, 3rd Q 2007.

[14] Y. Chu, S.G. Rao, S. Seshan, and H.S. Zhang, "A Case for End System Multicast", IEEE JSAC, special issue on networking support for multicast, 2002.

[15] Y. Kim, K. Chon, "Scalable and Topologically-aware Application-layer Multicast", IEEE Communications Society, Globecom 2004

[16] A. Sobeih, W. Yurcik., Jennifer C. Hou, "VRing: A Case for Building Application-Layer Multicast Rings (Rather Than Trees)", Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)

[17] S. Banerjee, B. Bhattacharjee, C. Kommareddy, "Scalable Application Layer Multicast", UMIACS-TR 2002-53 and CS-TR 4373, May 2002

[18] Y. Zhu, B. Li, J. Guo, "Multicast With Network Coding in Application-Layer Overlay Networks", IEEE Journal on selected area in Communications, Vol. 22, No. 1, Jan. 2004

[19] S.W. Tan, G. Waters, J. Crawford, "A Multiple Shared Trees Approach for Application Layer Multicasting", IEEE Communications Society, 2004

[20] A. El Rhalibi, M. Merabti, Y. Shen, "AoIM in Peer-to-Peer Multiplayer Online Games", Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology ACE '06, June 2006

[21] A. Yu, S. Vuong, "MOPAR: A Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games", Proceedings of the international workshop on Network and operating systems support for digital audio and video NOSSDAV '05

[22] T. Limura, H. Hazeyama, Y. Kadobayashi, "Zoned Federation of Game Servers: a Peer-to-Peer Approach to Scalable Multiplayer", Online Games. In SIGCOMM'04.

[23] B. D. Vleeschauwer, B. V.D. Bossche, T. Verdickt, F. De Turck, B. Dhoedt, P. Demeester, "Multiplayer game architectures: Dynamic microcell assignment for

massively multiplayer online gaming", Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games NetGames '05, October 2005

[24] F. Lu, S. Parkin, G. Morgan, "Load Balancing for Massively Multiplayer Online Games", Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games NetGames '06, October 2006

[25] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, C. Amza, "Locality Aware Dynamic Load Management for Massively Multiplayer Games", Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming PPoPP '05, June 2005

[26] T. N B. Doung and S. Zhou, "A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games", Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation PADS '06, May 2006

[27] S. Shirmohammadi, A. Diabi, P. Lacombe, "A Peer-to-Peer Communication Architecture for Networked Games", Proc. Future Play 2005, USA, October 2005.

[28] Z. Zhou, H. Wang, J. Zhou, L. Tang, K. Li, W. Zheng, M. Fang, "Pigeon: A Framework for Testing Peer-to-Peer Massively Multiplayer Online Games over Heterogeneous Network", IEEE CCNC 2006 proceedings.

[29] C. Chambers, W.C. Feng, W.C Feng, and D. Saha, "A Geographic Redirection Service for On-line Games", Proceedings of the eleventh ACM international conference on Multimedia, November 2003

[30] A. Tumbde, S. Venugopalan, "A Voronoi Partitioning Approach to Support Massively Multiplayer Online Games", CS 740 Project, The University of Wisconsin, Madison, 2004.

[31] A. Rowstron and P. Druschel. "Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems". In Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November 2001.

[32] M., Michael B. Jones, A.M. Kermarrec, Antony Rowstron, Marvin Theimer, Helen Wang, and Alec Wolman. "An evaluation of scalable application-level multicast built using peer-to-peer overlays". In Infocom'03, April 2003.

[33] M.R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, P. T. Barham, "Exploiting Reality with Multicast Group". IEEE Computer Graphics & Applications, 1995 (September 1995), 38-45.

[34] A. Boukerche and K. Lu, "A Novel Approach to Real-Time RTI Based Distributed Simulation System", Proceedings of the 38th annual Symposium on Simulation ANSS '05

[35] E.S. Liu, M.K. Yip, G. Yu, "Scalable Interest Management for Multidimensional Routing Space", Proceedings of the ACM symposium on Virtual reality software and technology VRST '05

[36] J.C. de Oliveira and N.D. Georganas, "VELVET: An Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs", Presence, 12(6), Dec. 2003.

[37] In [Online] http://en.wikipedia.org/wiki/Binary_heap

[38] I. Kazem, D. T. Ahmed, S. Shirmohammadi, "A Zone Based Architecture for Massively Multi-user Simulations", 2007 Spring Simulation Multiconference of Society for Modeling and Simulations

[39] In [Online] http://en.wikipedia.org/wiki/Binary_tree