# A Visibility-Driven Approach for Zone Management in Simulations

**Shervin Shirmohammadi, Ihab Kazem, Dewan Tanvir Ahmed, Madeh El-Badaoui, and Jauvane C. de Oliveira***

Distributed & Collaborative Virtual Environments Research Laboratory
School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada
{ shervin, ikazem, dahmed }@discover.uottawa.ca, melba037@uottawa.ca
*Computer Science Department, National Laboratory for Scientific Computing
Petrópolis, Brazil
jauvane@lncc.br

**Keywords:** Area of Interest Management, Virtual Environments, Application Layer Multicasting, Multi-user Collaborative Simulations, Massively Multiplayer Online Games, Peer-to-Peer Networking

**Abstract**

Massively multi-user simulations aim at supporting a large number of users while keeping the communication among the parties synchronous and highly interactive. In this paper, we present a collaborative virtual architecture that supports a large number of users by dividing the virtual environment into multiple adjacent hexagonal regions in order to manage the interest of the entities. A master node, called a hybrid node, constructs a Peer-to-Peer (P2P) overlay network to connect and manage nodes that lie in its region. Messaging is done at the application layer rather than the network layer, and a node joining algorithm is proposed to reflect the underlying network physical topology onto the data distribution pathways among the end-hosts to enhance the system performance. In addition, the introduction of a buffer zone between adjacent zones reduces the number of connections and disconnections that occur when a node frequently moves at the boundary of the two zones and provides more resilience to the system. We also attempt to shift the messaging among parties in one region from a zone-based method to a visibility-driven method to refine their interest by enabling message filtering. The effectiveness of this collaboration architecture is tested through a prototype implementation and a high level application.

## 1. INTRODUCTION

Multi-user simulations are gaining popularity for their significant contributions in virtual military training and online gaming. These environments offer a virtual world for parties to interact and collaborate in real-time. Defense ministries and military and aviation simulation companies 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 one kind of such environment and differ from other games in that they aim at supporting thousands of players to share a single game world [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. [2] state that, according to general reports from game companies, the number of users playing online 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. In addition to entertainment and collaboration activities for players, these games offer "the reward of being socialized into a community of gamers and acquiring a reputation within it [3]." Indeed, MMOGs have, in many cases, become addictive and offer a "social factor" that gamers yearn for.

MMOGs and other multi-user simulations environments introduce hard challenges to the 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.

Collaborative Virtual Environments (CVEs) aim at keeping communications among the parties synchronous. Precise coordination among the parties is defined through 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 [4]. Other studies have lowered the condition to 200 msec as an acceptable delay [5]. 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 collaborative virtual environments.

Currently, 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 that could otherwise be saved through multicasting. In P2P architectures, the importance of a server decreases as peers communicate with each other directly, enabling us to improve collaboration in many ways. If a peer connects with other peers, it can share information and collaborate immediately.

The network lag can become a problem in collaborative environments as it affects the performance of the collaboration. When a user participates in a simulation, its interactions with other nodes must be sent to all participants over the network such that all entities involved in this collaboration can update their states. But the networking limitations and traffic conditions cause these "updates" to be lost or delayed. To overcome the networking limitations for a better distributed simulation, research studies propose receiver-initiated and selectively-reliable transport protocols [6] 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 [7]. The IEEE DIS standard [8] has also been successfully used for military simulations; as such controlled environments have vast resources. These approaches are all based on IP multicasting. Although they achieve good results in an Intranet environment, they are not readily deployable on the Internet.

[9] and [10] 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, marketing 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' unwillingness to implement IP multicasting, and many other obstacles stand in the way of fully deploying IP multicasting over the internet.

To overcome the lack of multicasting infrastructure on the Internet, 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). In ALM, data packets are replicated at end-hosts instead of routers. An overlay network of end-hosts is constructed, where the end-hosts (or proxies) themselves relay the data to one another. IP multicasting and ALM are demonstrated 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. Since routing information is maintained by an application of the end-host rather than at routers, it is more scalable than IP multicasting. Also, ALM does not require any special infrastructure support, so it is fully deployable on the Internet. However, the disadvantages of ALM are more bandwidth and delay (compared to IP multicasting) for the sake of supporting more nodes and scalability. But ALM-based algorithms have shown "acceptable" performance penalties with respect to IP Multicasting and other practical solutions [11]. [12] proposed a peer-to-peer communication architecture for CVEs. Their implemented protocol combines both best-effort and reliable message exchange methods based on the type of data packets. [12]'s architecture has some side effects as it is not bandwidth efficient to broadcast the messages from one node to all other nodes in the virtual world. Also, since message exchange between nodes does not reflect the interest of these nodes, the protocol does not scale well. In this paper, we present an approach for Area of Interest Management (AoIM) in such environments, shifting messaging from a zone-based approach to a visibility-driven one. The proposed AoIM strategies have been implemented and deployed over both the Intranet and Internet. The rest of the paper is structured as follows: background, related work study, and different AoIM issues are presented in Section 2. We propose our design for AoIM and visibility-based messaging in Section 3. Implementation details and performance analysis are demonstrated in Section 4. We evaluate our work and results in Section 5. Finally, we draw conclusions and discuss future work in Section 6.

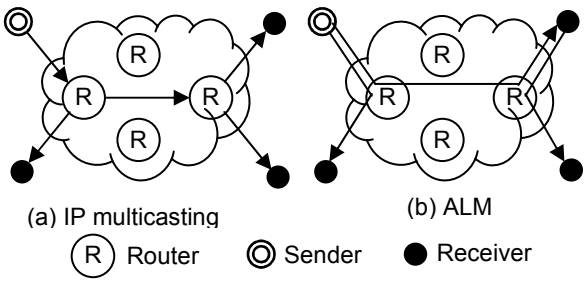


**Figure 1.** Multicasting concepts

## 2. PROBLEM BACKGROUND AND RELATED WORK

In this section, various components of the problem are presented and some of the related work are covered. Later in Section 3, our solution to the problems shown here will be presented and elaborated on. In this section, we discuss the background and related work under three separate subsections: Scalability and Network Lag, Area of Interest Management, and other simulation issues.

### 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 necessary 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 on 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 [13]. 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, another level may be added 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. Area of Interest Management Issues

Most literature on area of interest management agree on that in order to support a massive number of players, the simulation map is to be divided into multiple regions, where each region presents the "interest" of the nodes inside of it. However, when defining zone shapes and related issues, it is important to look at the simulation environment's needs and requirements. The following issues summarize area of interest management's main problems.

#### 2.2.1. Zone Definition

The simulation region 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 LANs are connected through the Internet, every 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.

#### 2.2.2. Zone shape

The square shape is a poor choice to represent a region. 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 (figure 2). 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 (figure 3). To sum up we need a shape that is both regular (to avoid gaps) and can emulate visibility as much as possible.

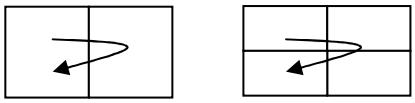

**Figure 2.** Two-zone vs. multiple-zone layout

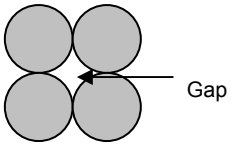


**Figure 3.** Circular zones with gaps

### 2.2.3. 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, depending on the overlay tree dependencies of other nodes to the node leaving. Also, connection to the new overlay network should ideally be done instantly so that the joining node will not add a delay and 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.

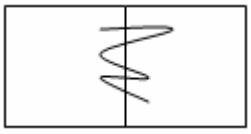

**Figure 1.** A node moving frequently around the boundary of two zones

### 2.2.4. 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 [14], offer a discrete view of the nodes: a node is only interested in the region 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. Therefore, 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. 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.

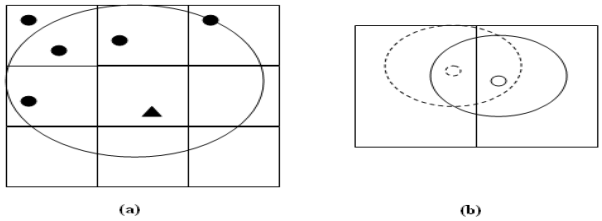


**Figure 2.** Sustaining a continuous view for nodes

### 2.2.5. 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 [13], for example, suggests an architecture 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. There is no overlay tree formed among peers in a given zone, and the approach does not make use of the bandwidth that the slave nodes can potentially offer, 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).

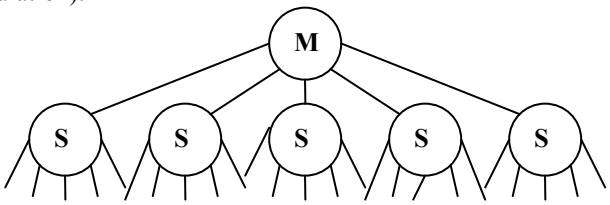


**Figure 6.** Master node M and slaves S

### 2.3. 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 behavior, and the magnitude of states sent and received by each node. El Rhabili et al well document the simulation issues at hand [15]. The work points out that the massiveness that we wish to evaluate 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 we have 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 2.2.1. 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.

### 2.4. Data Distribution Management

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 [16].

The Data Distribution Management (DDM) service is one of the six services provided in the RTI. Its purpose is to perform data filtering and reduce irrelevant data communicated between federates. Currently, DDM has two proposed schemes for RTI, region-based and grid-based. These aim at sending as little irrelevant data to subscribers as possible but only manage to filter part of this information. Some irrelevant data is still being communicated [17].

The services provided by DDM rely on the computation of the intersection between "update" and "subscription" regions. The filtering algorithms which evaluate these intersections have their shortcomings [18]. This is the same as the concept of the overlap between update and subscription regions, called matching [19]. [17] divides filtering mechanisms into region-based and grid-based. The region-based mechanism uses the concept of a routing space which is a multi-dimensional coordinate system through which a region is specified. In a grid-based mechanism, the routing space is partitioned into a grid of cells. Cells are used to efficiently specify the overlapped part between a subscription region and an update region. Much research has been done on both DDM mechanisms. Most of them focus on studying the impact of filtering on performance, optimizing the routing space, and reducing multicast groups

Liu et al. argue that interest management systems such as DDM service of the HLA focus on providing message filtering mechanisms but do so at the cost of Central Processing Unit (CPU) cycles overhead [20]. This can be unsuitable for real-time applications. A scalable interest management algorithm is presented for HLA DDM which employs a specific existing collision detection algorithm for fast interest matching which is also suitable for dynamic multicast grouping.

## 3. THE PROPOSED ZONE MANAGEMENT ARCHITECTURE

In this section, we will present our area of interest approach and discuss how its design best suits multi-user virtual environments and solves the issues discussed in the previous section. Data Management is realized on two levels in this paper. On the first level, geographical zoning is specifically used to define interest. On the second level, data filtering inside one zone is enabled to further sort and communicate only relevant data. This is similar to the region-based data filtering mechanism presented in Section 2.4. Our architecture has no dependency on the services provided by HLA and DDM. We consider a more generic approach that can be tailored to suit such services since it is only a matter of implementation when it comes to having a platform that HLA/DDM based or not HLA/DDM based.

### 3.1. Hexagonal Zoning: An Efficient Approach to Defining Interest Area

Hexagonal zoning has been widely used in game applications and adopted by AoIM papers. 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 special node, called a hybrid node, which constructs an overlay network and manages the nodes that are in its zone. These are called Hybrid because they have multiple functionalities: they form and participate in the P2P 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 [12].

Each hybrid node manages the nodes inside the zone it is responsible for, which 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 7).

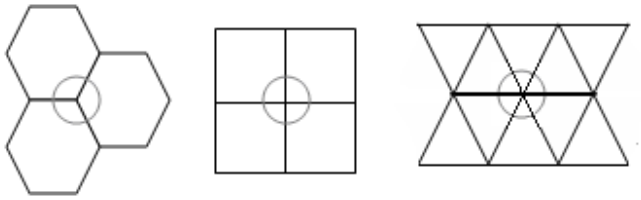

**Figure 7.** Number of zones covered by a visibility circle in a hexagon, square, and triangle layout

### 3.1.1. 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 2.2.3. 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 [21] 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 8, 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.

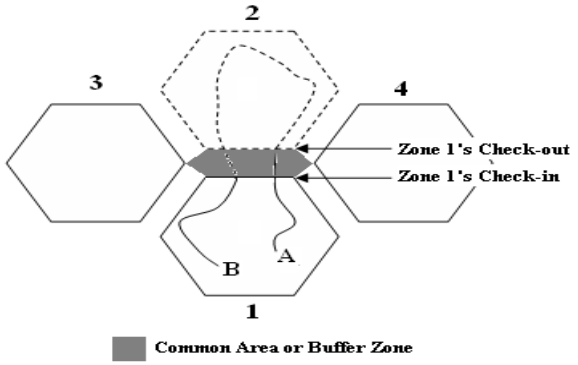


**Figure 8.** Hexagonal zones with check-in and check-out radii

## 3.2. 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 [13], 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 application layer multicasting tree: each node relays message in a multicast fashion as explained in Section 1. 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 9, 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).

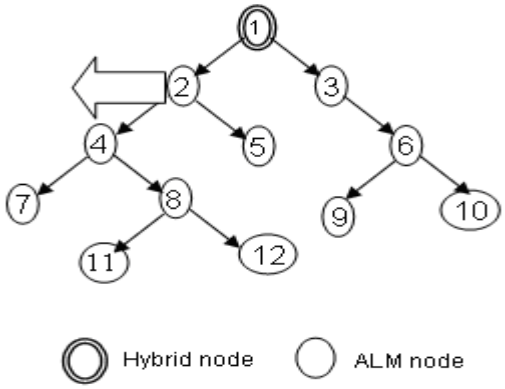


**Figure 9.** P2P-based tree structure

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 2.1, 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.

### 3.2.1. Node Joining

[22] argues that most simulations for the P2P-based Communication Model for MMOGs assume that the network is homogenous when testing the message level. However,

such homogeneity assumption seriously affects the simulation results with respect to the network latency. In reality, nodes in the simulation environments have different network locations, bandwidths and offline rates. [23] also present a redirection service for on-line games based on the geographic location of players relative to servers. They show how such a service "better meets client demand, saving each client and the Internet as a whole, thousands of miles of networking inefficiency [23]." For these reasons, we introduce the concept of mapping the underlying network's parameters (bandwidth, end-to-end delay) to the ALM overlay network, as shown in the next two subsections.

#### 3.2.1.1. Mapping Bandwidth to Node Degrees

End-hosts form the structure of the application layer multicast tree. In order for end hosts to carry data over the tree, they need to have available bandwidth. Usually, end hosts have asymmetric downloading and uploading speeds. Moreover, the heterogeneity of outgoing bandwidth of the end hosts is an important consideration. Some users could have a zero out degree, i.e. pure receiver. From a practical perspective, the asymmetric bandwidth nature cannot be ignored and should be taken into consideration when assigning *out degrees* for nodes: the maximum bandwidth a node can provide. For example, if a node has an out degree of 3, it means it can support at most 3 children.

#### 3.2.1.2. 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 a hybrid node. Each hybrid node keeps the 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. 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.

### 3.2.2. 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. This means that over time, the ALM trees will seize to reflect the end-to-end delay, which 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.

### 3.2.3. Fault Tolerance Using Backup Parents

To keep the simulation experience smooth and without any perceived delays or "glitches", a node maintains an active connection to 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.

### 3.2.4. 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.

### 3.2.5. Underlying Transport Protocol

Two transport protocols available are TCP and 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 longer 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 [12]. Although TCP takes care of such reliability, it imposes unnecessary overhead on the performance for its session initialization and congestion control functionalities.

## 3.3. Visibility-Based Inter-Zone Communication

How nodes communicate in a zone (intra zone communication) was explained in Section 3.2. 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 appropriate physics engines for handling computer graphics, which all MMOGs today have. In this paper, 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 10, 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.

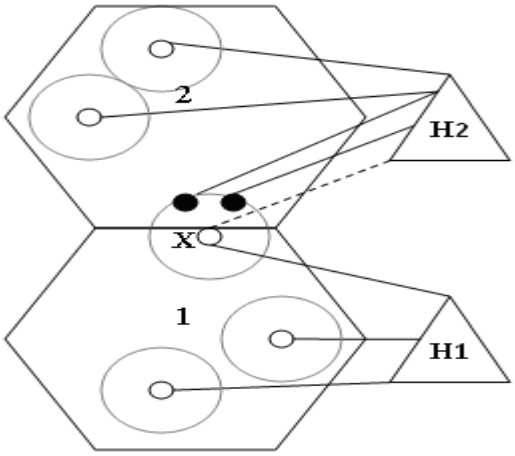


**Figure 10.** Node specific communication

The significance of such communication is that we are 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 10). 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.

Our 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:

1. 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.
2. 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.
3. 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.

### 3.3.1. 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.

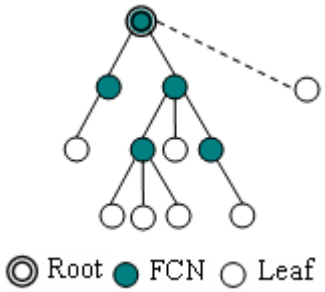


**Figure 11.** Position of FCN

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 11). The dashed line in figure 11 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 [24]. 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.

### 3.3.2. Hidden Node Problem

Consider figure 12 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. 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.

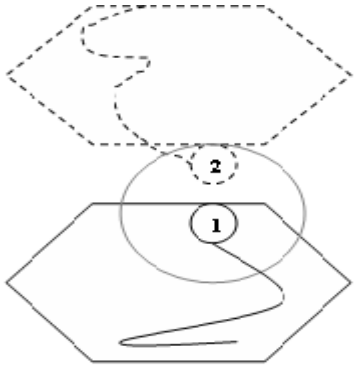


**Figure 12.** Hidden node problem

## 4. IMPLEMENTATION AND PERFORMANCE ANALYSIS

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. Dividing the simulation map into multiple zones, it is intuitive that the number of nodes supported on the map is whatever 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 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.

As proof of concept, a multi-user military simulation application was implemented. 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 13 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 13.** Snapshot of the simulation application

### 4.1. Graphics Implementation

The implemented environment uses OpenSceneGraph (OSG), an open source high performance 3D graphics toolkit, to simulate the graphical virtual world in multiple adjacent hexagonal regions. Based on the concept of SceneGraph, OSG provides an object oriented framework 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.

### 4.2. Area of Interest Management

Area of interest management was implemented by dividing the simulation map into several hexagonal zones. The number of zones can be manually set, and their configuration best fits the map to avoid unwanted gaps. As mentioned earlier, portioning the map into zones comes with a cost of having to deal with nodes moving from one zone to another and potentially causing instability for trees as they try to restructure themselves.

**Figure 14.** 4-zones lay-out

**Figure 15.** 9-zones lay-out

**Figure 16.** 16-zones lay-out

To study this, we ran a random path traversal on the same map with different zone configurations, as shown in Figures 14, 15, and 16, where the black line presents the path, and the white lines present the boundaries of the 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 1 summarizes the results.

**Table 1.** 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 |

### 4.3. Node Joining

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 degree. It will then connect to the parent that with which it has the smallest end-to-end delay by doing a Round Trip Time (RTT) hand-shaking with the list of potential parents. 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 new comer calculated the delay 10 times. The average delay is shown in Table 2. The new comer successfully joined the parent with the smaller end-to-end delay.

**Table 2.** Reflecting the End-to-end delay on the ALM overlay network

| | Possible Parent 1 | Possible Parent 2 |
|---|---|---|
| End-to-end delay (ms) | 20 | 38 |

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

### 4.4. Buffer Zones

Section 3.1.1 proposed how introducing a buffer zone (common area) between adjacent zones decreases the frequency of disconnections and connections upon the movement of a user at the boundaries of a zone. We ran a test 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 17, 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 18 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.

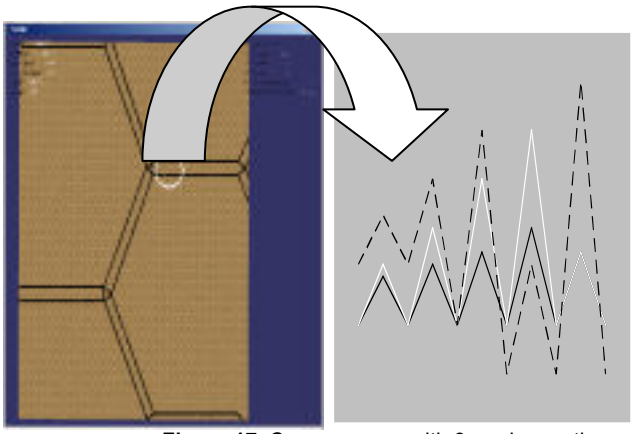

**Figure 17.** Common area with 3 random paths

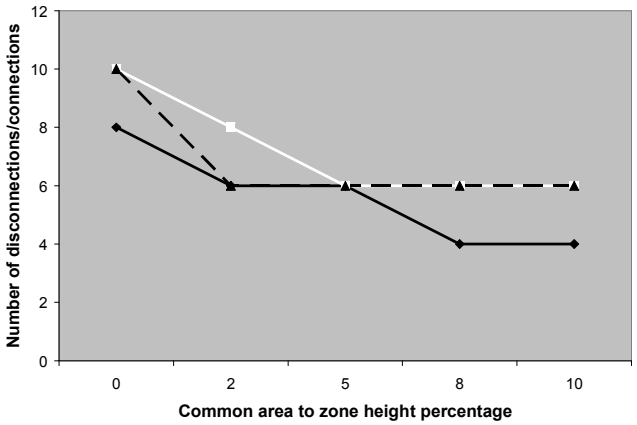


**Figure 18.** Common area performance for random paths black, white, and dotted

### 4.5. Message Filtering

To test message filtering, three users joined the simulation and they connected according figure 19's tree structure. User 1, being a parent of the other two leaf users, was marked as a FCN and given filtering abilities.

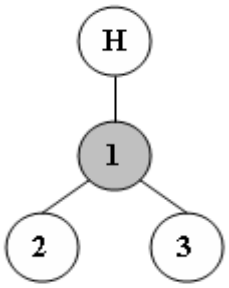


**Figure 19.** Tree structure to test filtering. The grey node represents the FCN.

All users had initial positions in the same zone, as shown in figure 20, such that the distance between each and everyone is the same. 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:

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

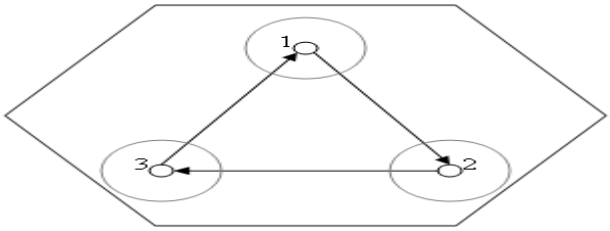


**Figure 20.** User's positions and paths run

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 3 summarizes the results.

**Table 3.** Message reduction when enabling filtering

| | Without Filtering | With Filtering |
|---|---|---|
| Outbound bandwidth for 1 | 1660 | 223 |
| Inbound bandwidth for 2 | 800 | 112 |
| Inbound bandwidth for 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. EVALUATION

### 5.1. Number of Tree Levels Supported

HDSP's [12] performance analysis clearly conclude 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 2 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, we will increase the number of nodes that can be supported 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, the number of levels supported is almost half of the 4 levels supported in

the same ISP; so our expectation is a maximum of 2 levels supported in practice.

### 5.2. 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 1 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. One solution is to increase the granularity of zone dividing. This decreases the maximum density in one zone to a few players. Also, depending on the virtual environment itself, the architecture can allocate more hybrid servers from less crowded zones to the crowded zones. For instance, combat zones in military simulations are usually more likely to have many nodes inside of them. Also, some techniques can be applied to measure the statistics on the movement of the nodes in each area or the population in each zone and allocate more hybrid servers accordingly.

### 5.3. 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 form 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 we could support from the performance analysis, we 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.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we present a multi-user simulation architecture. Scalability is achieved by dividing the fantasy world into multiple adjacent hexagonal regions in order to properly organize the entities and efficiently manage their liaison. This achieves message isolation between zones and reduces the communication overhead, keeping the required timing constraints within the limit. 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. The problem of a node frequently bouncing in and out at the boundary of two zones is solved by introducing common areas between adjacent zones. A continuous view is maintained for the player to ensure a perfect simulation experience. Message filtering based on the positions and radius of visibility of players is enabled to achieve visibility-driven messaging. We implemented the presented collaboration architecture and got a flawless communication platform.

Our future work will focus on designing and implementing load balancing techniques among hybrid servers to solve the problem of having many users moving into a zone and heavily populating it. We will also refine the interest of users and propose a mathematical model to determine the best node in the best zone with a given interest vector. Node joining is also reconsidered to sort entities according to their behavior as the hybrid server opens multiple multicast channels to reduce structural reformation events among the entities.

## ACKNOWLEDGMENTS

The authors acknowledge the financial support of the CAE and the Natural Sciences and Engineering Research Council of Canada.

## 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] 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

[3] 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

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

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] 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.

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

[13] 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

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

[15] 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

[16] 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

[17] G. Tan, L. Xu, F. Moradi, S. Taylor, “An Agent-based DDM for High Level Architecture”, Workshop on Parallel and Distributed Simulation, Proceedings of the fifteenth workshop on Parallel and distributed simulation, 2001

[18] C. Raczy, G. Tan, J. Yu, “A Sort-based DDM Matching Algorithm for HLA”, ACM Transactions on Modeling and Computer Simulation (TOMACS), Volume 15 , Issue 1 (January 2005), p. 14 – 38

[19] K. Pan, S.J Turner, W. Cai, Z. Li, “An Efficient Sort-Based DDM Matching Algorithm for HLA Applications with a Large Spatial Environment”, Workshop on Parallel and Distributed Simulation, Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation, p. 70-82, 2007

[20] 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

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

[22] Z. Zhou, H. Wang, J. Zhou, L. Tang, K. Li, W. Zheng, M. Fang, “Pigeon: A Framework for Testing Peer-to-Peer Massively Online Games over Heterogeneous Network”, Consumer Communications and Networking Conference, 2006. CCNC 2006. 2006 3rd IEEE Volume 2, 8-10 Jan. 2006 Page(s): 1028 - 1032.

[23] C. Chambers, W.C. Feng, W.C. Feng, D. Saha, “A Geographic Redirection Service for On-line Games”, Proceedings of the eleventh ACM international conference on Multimedia MULTIMEDIA '03, November 2003

[24] http://en.wikipedia.org/wiki/Binary_heap