A Zone Based Architecture for Massively Multi-user Simulations

Ihab Kazem, Dewan Tanvir Ahmed, Shervin Shirmohammadi
Distributed & Collaborative Virtual Environments Research Laboratory
School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada
{ ikazem, dahmed, shervin }@discover.uottawa.ca

Keywords: Area of Interest Management, Virtual Environments, Application Layer Multicasting, Multi-user Collaboration, Massively Multiplayer Online Games

Abstract

Massively multi-user simulation requires synchronous communication among the parties. In this paper, we present a multi-user collaboration architecture that divides the virtual world into multiple adjacent hexagonal regions in order to properly organize the entities and efficiently manage their liaison. A special node, named hybrid node, is in charge of each hexagonal region and constructs a data distribution tree at the application layer rather than the network layer. While constructing the data distribution pathways among the end-hosts, the protocol focuses on reflecting the underlying network physical topology onto the overlay network to enhance the system performance. To control the excessive message overhead, necessary messages of a foreign region are only imported when needed to a particular region through a hybrid node. Dynamic adjustment of check-in and check-out marks reduces frequent connections and disconnections between a hybrid and an ordinary node and provides resilience to the system. The effectiveness of this collaboration architecture is tested through the implementation.

1. INTRODUCTION

Multi-user simulation environments have many significant contributions in various aspects ranging from entertainment to virtual training. These simulation environments are used to avoid risks in the training as well as to interpret the real or the simulation results. Massively multiplayer online game (MMOG) is a kind of such collaboration and is different from other games in that it aims at supporting thousands of players to share a single game world [1]. These simulations 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. To achieve this, many collaboration tools use the client-server architecture, where the server relays one client's message to others for collaboration. This has the advantage of maintaining data only at the server, thus client-side software is light and simple. But in such collaboration software, the server becomes a bottleneck causing scalability problems for the collaboration application and could be a single point of failure. 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. P2P architecture enables us to improve collaborations in many ways. The importance of a server decreases as peers communicate with each other directly. If a peer connects with other peers, it can share information and collaborate immediately.

In Collaborative Virtual Environment (CVE), synchronous communication among the parties is essential. Sometimes in a closely-coupled system, precise coordination among the parties is defined through the end-to-end delay where the parties are connected to the Internet from geographically distributed locations. It has been recommended that in such environments, the end-to-end delay should be within 100 msec [2]. Other studies have lowered the condition to 200 msec as an acceptable delay [3]. The necessity for synchronous communication imposes this hard time constraint in collaborative virtual simulation environment. Moreover, these interactions are highly reactive. A user's response depends upon other parties actions. Therefore, it requires frequent updates with a moderate end-to-end delay.

One of the well-known problems in these environments is the network lag. It affects the performance of the collaboration. When a user participates in a simulation, its interactions with other nodes must be sent to the 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 a better distributed simulators. Some of these studies provide receiver-initiated and selectively-reliable transport protocols [4] 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 [5]. The IEEE DIS standard [6] 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.

The lack of applicability of IP multicasting on the Internet has been well documented in [7] [8]. The reasons include scalability, 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, the deployment hurdles caused by manual configuration at routers and Internet Service Providers' unwillingness to implement IP multicasting, marketing reasons due to the undefined billing at the source (content provider) and receivers, and many other obstacles.

Due to the practical 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. The end-hosts form an overlay network, where the end-hosts (or proxies) themselves relay the data to one another. IP multicasting and ALM are demonstrated in Figure 1. 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 [9]. [10] proposed a peer-to-peer communication architecture, and this work is a continuation of the issues left unsolved in their architecture. [10]'s implemented protocol combines both best-effort and reliable message exchange methods based on the type of data packets. Best-effort delivery is used for frequently occurring messages, while reliable delivery is used for important or "key" messages. The protocol, however, does not scale well when it comes to supporting MMOG's as message exchange between nodes does not reflect the interest of these nodes. In this paper, we present an application layer multicasting approach to manage the common interest of a large group. We call it Area of Interest Management (AoIM). The proposed AoIM strategies have been implemented and deployed over both the Intranet and Internet. Various issues related to the AoIM, such as zone definition, zone shape, check-in and check-out policies, and inter-zone communication for visibility purpose are covered in Section 2. We present an application layer multicast approach for intra-zonal communication in Section 3. Implementation details and performance analysis are demonstrated in Section 4. We evaluate our work and talk about our future work in Section 5. Finally, a conclusion is drawn in Section 6.

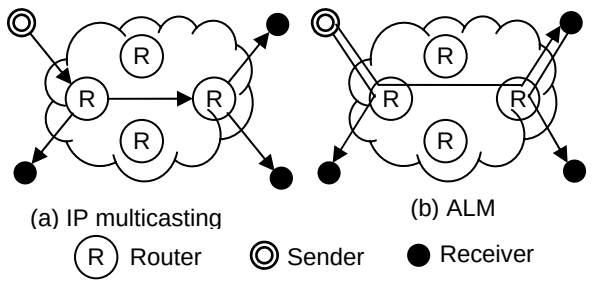

Figure 1. Multicasting concepts

2. AREA OF INTEREST MANAGEMENT

Different issues relating to the area of interest management are explained in the following sub-sections.

2.1 Zone Definition

The simulation region is divided into multiple adjacent zones. Each of these zones gets allocated a special node, called a hybrid node, which constructs an ALM tree and manages the nodes that are in its zone. 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 hybrid node to construct a tree, maintain its state, and manage new comers and early leavers. Other factors, however, should be taken into consideration when defining a zone: visibility and logical positions. The logical position of a node and its visibility on the simulation map is what is intuitively meant by a zone. Nodes that are logically close to each other should be in the same zone. So, when defining a zone, one should consider optimizing the communication among the nodes by looking at how they are connected (LAN or Internet). This will improve the performance in a zone, which is quite desirable. But it is of greater priority to define the zone according to the logical positions of nodes and their visibilities (what they "see"), as their interest is more of their locality and what is visible to them on the map, even though they might not be of close proximity on the underlying network.

2.1.1 Multiple zones

Multiple zones definition is adopted when defining the map on the network. The protocol targets to handle a large set of nodes. If we are to accommodate more nodes, the map should be logically divided into multiple zones. Here each zone encompasses the nodes that are in the same vicinity. Henceforth, we are to define how nodes get disconnected from one hybrid node and join to another when moving from one zone to another. Looking at a primitive two-zone layout, it is easy to see that we are only dealing with the disconnection from one hybrid node and the connection to another. If we are to implement a multiple-zone layout, however, more connections and disconnections are taking place for the same path traversal scenario (Figure 2).

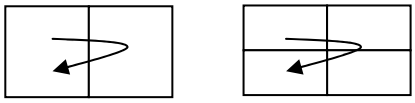

Figure 2. Two-zone vs. multiple-zone layout

2.1.1 Zone shape

Hybrid Distributed Simulation Protocol (HDSP) [10] takes for granted that a square is the mere representation of a zone. However, this might cause a problem when studying the visibilities of the nodes, which is discussed in a following section. But looking at visibility from a zone-perspective,

consider the following scenario (Figure 3(a)): a unit is located right near the center of a zone, defined by a square; the visibility is defined by the circle around the unit. If 1 fires a missile at 2, even though 2 is not in the visibility range of 1, it will get the message that it is being fired at by 1. So 1 will not see whether or not it hit 2, even though 2 is in the same vicinity as 1. This is a problem as one node actually sends a message to another node (1 fires at 2), which are in the same zone but does not observe the effect of the message, i.e. explosion. To solve this situation we unite the concepts of visibility and zone in one concept. This can be achieved by choosing zone shape that best resembles a circle. A hexagonal can be a fair choice as it does resemble a circle to an extent, but still maintaining a regular shape to enable us to stick several zones together without having any gaps, which would be the case if we choose circles as a zone shape (Figure 3 (b)). This representation of zones is not new as cellular networks are designed in this way. In such networks, the user is connected to the base station responsible for the hexagonal it is in. As soon as it leaves its zone, it connects to the other base station.

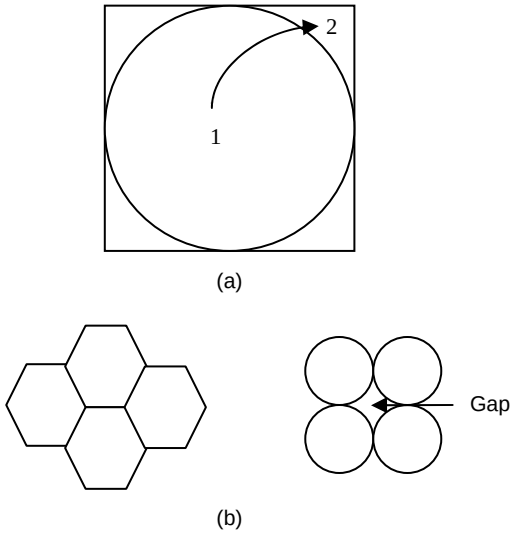

Figure 3. (a) Zone - visibility conflict (b) Hexagonal zones without gap but circular zone with gap

2.2 Check-in and Check-out radii

VEry LargeVirtual EnvironmenTs' (VELVET) Area of Interest Management [11] can be implemented to avoid the problem of a node's frequent movement around a zone (Figure 4). This protocol also points out that how increasing the difference between the inner and outer radii makes the common area (area bounded by radii) larger, thus decreasing the number of times a node disconnects and connects to a hybrid node. However, no suggestion was given to choose the inner and outer radii. Assuming that radii values are chosen in a way to make the common area large enough to avoid bouncing in and out of a zone, still the developer does not have a prior knowledge of how a node is behaving, as it is strictly up to the player to decide on what velocity a node is moving (speed and orientation). The node can still bounce in and out of this area depending on its behavior. What this paper suggests is that the values of check-in and check-out radii are dynamically set based on the behavior of the node. Thus, a history of the node checking in and checking out is maintained at the application level to estimate its behavior and set radii values accordingly (Figure 5).

2.3 Inter-zone communication - Visibility Issues

How nodes communicate in a zone (intra zone communication) is explained in Section 3. This intra-zone communication deals with various aspects such as constructing an ALM tree, reliable and non-reliable message handling, etc. Still, situations may occur when a node needs to send a message to a node in another zone. If one is to give more variations to units at hand, like giving a tank with radar capabilities to see beyond its fellow nodes (nodes in the same zone) or a plane flying at an altitude, thus having more visibility, then, in such cases, the nodes still have their intra zone communications with their fellow nodes but need to communicate with other zones. Here, we can make node-specific connections with other hybrid nodes. So if a tank has a visibility more than its zone, it receives necessary messages from other hybrid nodes through its local hybrid node. 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, the concept of visibility is also unstable as it might be a characteristic of the unit itself (tank with a radar), or restricted by map geography.

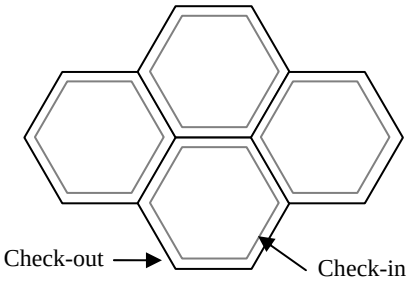

Figure 4. Hexagonal regions with check-in and check-out radii

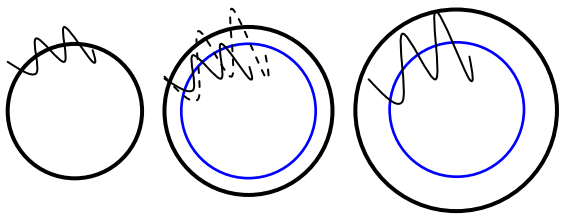

Figure 5. Choosing check-in and check-out radii according to unit's behavior for circular regions. The dashed line represents the unpredictable behavior of a node.

There is an intrinsic problem with having the node connect to several remote hybrid nodes at the same time. Considering a multiple hexagonal zones layout, if a node sits on the intersection of the three hexagons, then it will have connections to three hybrid nodes at the same time, at least logically. This is a big overhead as every message that the node needs to send will be propagated to three ALM trees at the same, even though many nodes in these trees are not even interested to get the messages. Ideally speaking, only the nodes that are in the visibility range of the node need to get the message. To avoid the multiple connections scenario, and propagating messages to uninterested nodes, we make the use of 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 second 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. However, the idea of an "explicit" message being sent to a specific address (hybrid node) is redundant to the ALM architecture already established. Messages currently propagate based on node-to-node unicast connections. So every node is only responsible for sending or forwarding a message to whatever nodes it is connected to in the ALM tree. For these reasons, we propose that the logical position and the radius of visibility get encapsulated in the content of the message. So when a hybrid node gets a message, it determines whether node's visibility exceeds the border of the zone by looking at its logical position and radius of visibility. If this is the case, it informs the foreign hybrid node to handle such a situation. The necessary messages are imported through the reverse path to solve the visibility issues.

Taking a closer look at such a coordinate system, every logical position can also have a position to take the alleviation of a unit into consideration (thus we can also have flying units in the simulation game in addition to tanks). The visibility is hence defined as a sphere for flying objects and a semi-sphere for land objects.

3. APPLICATION LAYER MULTICASTING FOR ZONAL COMMUNICATION

HDSP [10] 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. [12] argue that most simulations for the P2P-based Communication Model (PCM) for MMOG's 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. Nodes in the simulation environments that we are interested to support running on the network usually have different network locations, bandwidths and offline rates [12]. [13] 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 [13]." For these reason, we introduce the concept of mapping the underlying network's parameters (bandwidth, end-to-end delay…) to the ALM overlay network.

3.1 A Mapping: Bandwidth to Node Degrees

End-hosts form the structure of the application layer multicast tree. The feasibility of carrying collaborated 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, i.e. pure receiver. But our case is quite different. These ALM structures usually carry short but frequent messages. From practical perspective, asymmetric bandwidth fact cannot be ignored and is taken in consideration to assign *out degrees* to nodes during implementation. It reflects 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 A Mapping: Node joining to end-to-end delay

This paper suggests a node joining process as a function of end-to-end delay. End-to-end delay is a good choice for real-time application. But due to the synchronization problems at nodes, one can use hop count to select its parent. Every joining node makes a request to a hybrid node. We consider two approaches, namely centralized and decentralized, for node signup purpose. These are explained in following subsections.

3.2.1 Centralized approach

In centralized architecture, each hybrid node keeps the track of the entire logical topology of a particular zone. So, a hybrid node has an option to send a response to the joining node containing a list of potential parents without consulting all of its descendants. It is all up to the new comer to select the best one based on either hop count or end-to-end delay. The parent of the joined node informs the hybrid node about the incident. 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). For a moderate zone size, it works quite well.

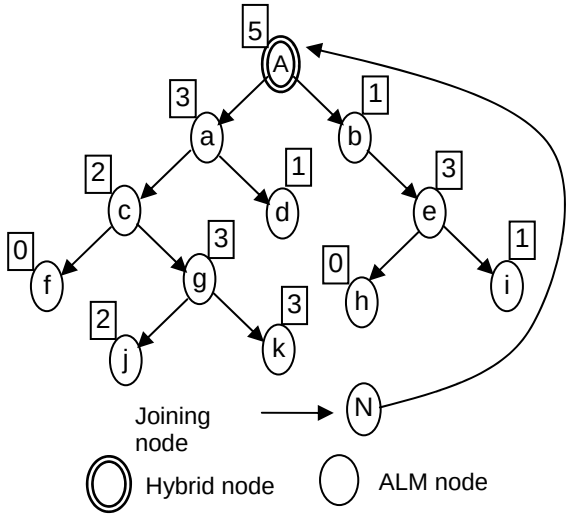

Figure 6. Node joining phase

3.2.2 Decentralized approach

Each new member sends a $JOIN\_REQ$ message to its nearest hybrid node along with the maximum depth to limit the scope of the parent searching process on the multicast tree. Upon receiving the join request, the hybrid server delegates the request to its children to take care of the new node. The server also reports to the requester if it has free out degree. All children do the same as the server does. This parent searching process continues within the depth limit. The requester gets the responses from potential parents. It chooses one of them as a parent based on the smallest end-to-end delay. If it does not receive any response, it can re-send $JOIN\_REQ$ with an increased maximum depth limit. This time, it has to increase waiting period just like the exponential back-off mechanism. Figure 6 is an example of the joining process. Numbers within squares represent the maximum out degree of that node. Here, Host N is a newcomer and sends $JOIN\_REQ$ to server A with a maximum depth limit set to 3. According to the Figure 6, Node N gets responses from Node $A, a, d, e, g,$ and i.

3.2.3 Maintaining the mapping throughout the game

Our initial approach was that a mapping of node joining to end-to-end delay is possible when the game first loads because the user would not mind a time delay until the joining is completed. The nature of the game, however, implies that 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 the mapping is only applied at first (when the game loads), then after a few minutes of the game, the ALM trees will seize to reflect the end-to-end delay. This will affect the performance improvement achieved by such mapping. So we propose that such mapping is maintained throughout the game. However, if, upon testing, the joining time turns out to be imposing a delay that cannot be sustained to maintain a smooth gaming experience, this approach should be abandoned, and we should stick with a simple minimal time node joining.

3.3 Fault Tolerance – A Backup Parent Technique

We already mentioned that in application layer multicast, routing has been shifted from the network layer to the application layer. It is the end-host that plays the key role of maintaining the overlay structure. For the sake of fault tolerance, we keep one more active connection to another ALM node. We call it backup parent. The policy of choosing a backup parent is still in the design phase. If we make a rule for the node to reserve the second-best parent when first joining, then it can quickly switch to the back-up parent and activate the connection with it so as to maintain the stability of the tree.

It is noteworthy to point out here that the time needed for the ALM tree to restructure itself upon a node moving from one zone to another, which we call recovery time, should be considerably small so as not to add an extra delay on the original message delay when nodes are sharing game states. The back-up parent policy could improve such recovery time.

3.4 Node Departure

When a node decides to leave a zone, it has to disconnect from its parent, as well as leaving all of its descendants parentless. Departures can be divided into graceful departures and ungraceful ones. In a graceful departure situation, a leaving node notifies all of its communication partners, so the nodes can update their states and reconstruct the tree. In an ungraceful situation, the node just leaves without any hints. In such scenario, a node departure detection mechanism has to be deployed. This mechanism is similar to fault-detection algorithm, where a "keep alive" message is being sent to a communication partner, and a timer is started, say for 5 seconds. If the timer elapses, and it does not receive any acknowledgment, then it can be sure that the partner has departed.

Figure 7. Graphical view of the simulation

3.5 Underlying Protocol

Two communication approaches are available for the ALM namely TCP and UDP. The protocol focuses on collaborative virtual environment that needs short frequent messages, where a few missing packets does not hamper the performance of the protocol as long as it does not miss too many. So our ALM protocol uses user datagram protocol – UDP, as a communication technique since it satisfies the protocol's objective. Acknowledgement-based reliable UDP is used for key messages that require guaranteed delivery TCP, on the other hand, does take care of such reliability, but the question is do we need all the features provided by TCP for our application? If not, then we might be imposing unnecessary overhead on the performance, not to mention the context switching between UDP messaging and TCP messaging. Also, TCP provides congestion control which should definitely be avoided in collaborative games. Reliability has to be carefully designed to only deliver what the application demands, but at the same time deploying steadfast techniques, maybe more complicated than timer techniques.

4. IMPLEMENTATION AND PERFORMANCE ANALYSIS

Our 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. With today's bandwidth (which translates to a node's out-degree), one can imagine how any more level supported will significantly increase the number of nodes that can be sustained by a single ALM tree in a zone. 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 automatically 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 a node moves from zone to another.

We implemented a multi-user military simulation environment. It serves as a simulation for the protocol and shows how messaging performance is affected and scalability is enhanced when many nodes join. Figure 7 represents a snapshot of our network game.

First, 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 (nodes with available bandwidth; free out degree) following the centralized algorithm explained in Section 3. It will then connect to the parent that it has the smallest end-to-end delay with 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 on home internet users, both using Rogers© ISP. The node successfully joins the parent which has a smaller end-to-end delay.

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

	Home user 1	Home user 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. Second, 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 the cases when nodes move from one zone to another. In these cases, the node will have to disconnect from its original tree, and connect to the new one. This will create instability for trees as they try to restructure themselves.

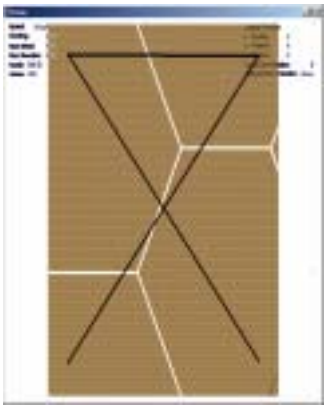

Figure 8. 4-zones lay-out

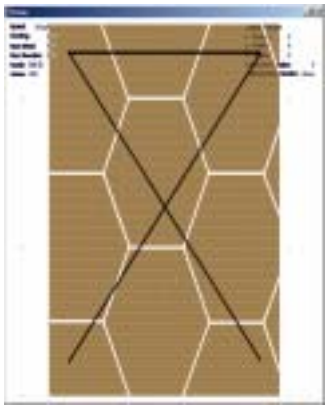

Figure 9. 9-zones lay-out

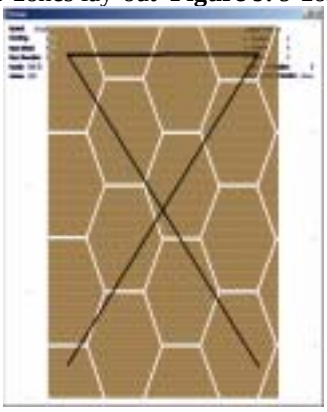

Figure 10. 16-zones lay-out

To study how scalability is improved and the effect of dividing the map into multiple zones in terms of the number of connections and disconnections as a user moves from one zone to another, we ran a random path traversal on the same map with different zone configurations (Figures 8, 9 and 10 – black line presents the path, 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 node out degree and number of levels supported in the tree). Table 2 summarizes the results.

Table 2. 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

Third, section 2.2 proposes 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 boundary of a zone. We implemented the common area between adjacent zones to be statically set (as opposed to being dynamically set as proposed in section 2.2), we ran a test to study how the size of the common area affects the number of disconnections and connections.

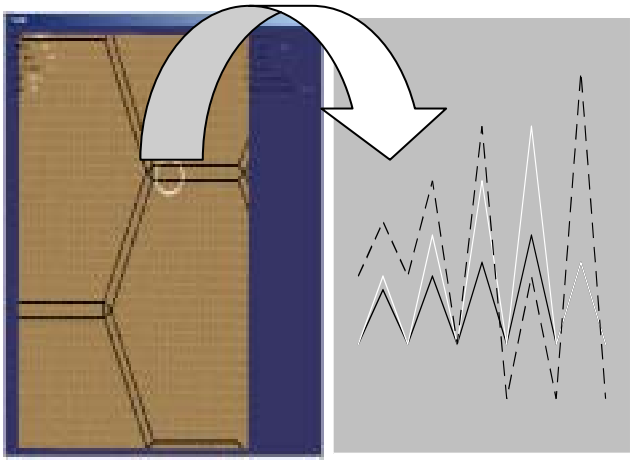

Figure 11. Common area with 3 random paths

Three random traversals were conducted at the boundary of adjacent zones, where the common area was introduced (Figure 11 – black, white, and dotted black curves represent the random traversals). Also, the height of the common area was varied as a percentage of the hexagonal zone's height (keeping the same path traversals). We got the following results:

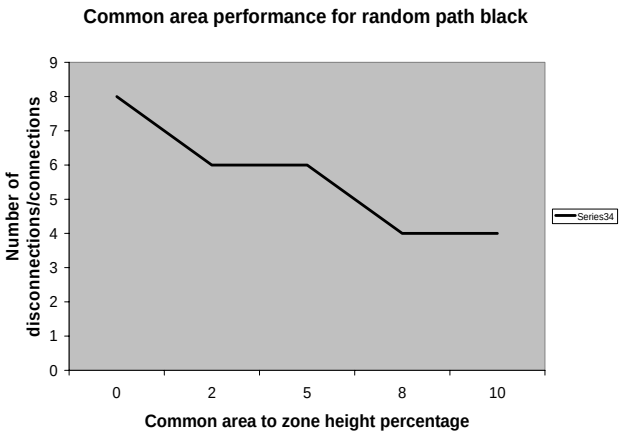

Figure 12. Common area performance for black path

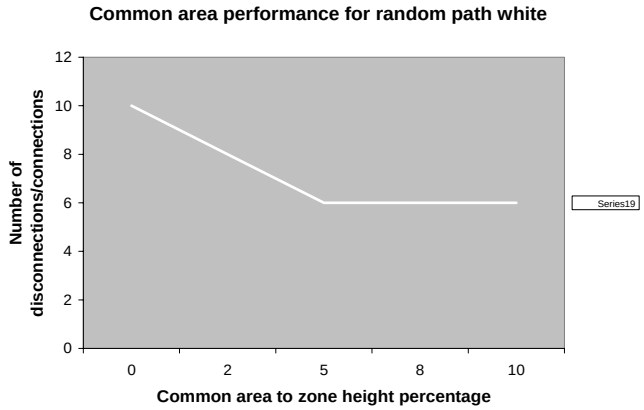

Figure 13. Common area performance for white path

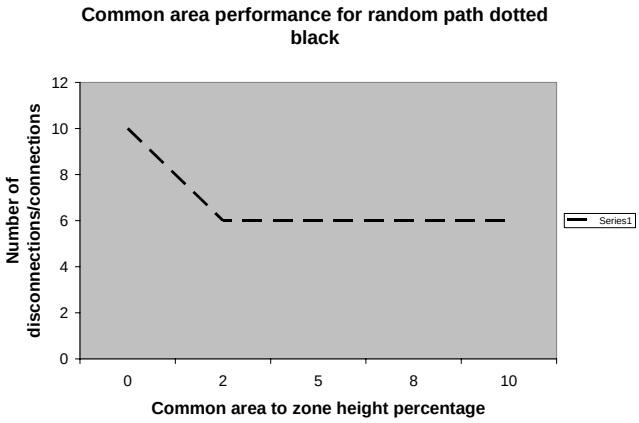

Figure 14. Common area performance for dotted black path

The above graphs clearly demonstrate 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), less frequent disconnections and connections happen.

5. EVALUATION AND FUTURE WORK

HDSP's [10] performance analysis clearly conclude that in case nodes are under the same ISP, a three level tree should be easily supported and the expected delay from the Hybrid node to that 3rd level should be about 120-140 ms. 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. Table 1 also justifies such conclusion by showing 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 ISP's and sorts out the ALM tree's branches according to nodes on the same ISP or LAN's. 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 first one. 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 tolerated in one zone and more so in the map as a whole.

Nodes on different zones have complete message isolation: messages from nodes in one zone only get propagated to nodes in the same zone. Being at liberty of choosing the size and the number of zones in the simulation map, scalability is easily achieved depending on those parameters. However, as Table 2 shows, scalability is improved with the side effect of having to deal with more disconnections and connections as a user moves form zone to another, which will destabilize the ALM tree and will take some time to recover. Our conclusion is that, depending on the size of the map used and the number of nodes that we wish to support, zone sizes are chosen accordingly to make the ALM trees more stable, which will ensure a smooth gaming performance. As mentioned earlier, having successfully divided the simulation map into multiple zones, we automatically solve the problem of having a huge simulation map with massive number of nodes to support since there is message isolation between zones and one tree structured for each zone. However, the density of nodes in one zone is a problem as we have no control on several nodes moving into a zone that cannot tolerate them. In our future work and prototypes, we will try to address this problem. We will also design and implement a state-of-the-art visibility-driven messaging; redefining the interest of a node from the zone it is in to what is visible to it (what it "sees").

Finally, having common areas between adjacent zones might not have completely solved the problem of having disconnections and connections upon a user's frequent movement around the boundary of the zone, but it improved it by a great deal to ensure tree stability.

6. CONCLUSION

In this paper, we present a scalable multi-user virtual collaboration architecture. The structural design divides the fantasy virtual world into multiple adjacent hexagonal regions in order to properly organize the entities and efficiently manage their association. This reduces the communication overhead and keeps the required timing constraints within the limit. A node joining algorithm is presented that reflects the underlying network's topology on the ALM overlay network to improve end-to-end delay. Common areas between zones are introduced to solve the problem of a node frequently bouncing in and out at the boundary of two zones. We implemented the presented collaboration architecture and got a flawless communication platform.

ACKNOWLEDGMENTS

The authors acknowledge the financial support of the CAE. We would also like to acknowledge Madeh El-Badaoui for his graphical contribution in this collaboration architecture.

REFERENCES

[1] Knutsson et al, "Peer-to-Peer Support for Massively Multiplayer Games", INFOCOMM, 2004

[2] B. Blau et al, "Networked Virtual Environments", Proc. ACM SGGRAPH, 1992, pp. 157-164

[3] 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.

[4] 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.

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

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

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

[8] 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.

[9] 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.

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

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

[12] Zhou et al, Pigeon: A Framework for Testing Peer-to-Peer Massively Online Games over Heterogeneous Network, IEEE CCNC 2006 proceedings.

[13] Chambers et al, A Geographic Redirection Service for On-line Games, Proceedings of the eleventh ACM international conference on Multimedia, November 2003