Received 31 October 2013; revised 9 April 2014; accepted 9 June 2014. Date of publication 12 June, 2014;
date of current version 10 June, 2015.

Digital Object Identifier 10.1109/TETC.2014.2330520

A Dynamic Networking Substrate for Distributed MMOGs

**MOHSEN GHAFFARI, BEHNOOSH HARIRI,
SHERVIN SHIRMOHAMMADI, (Senior Member, IEEE), AND DEWAN TANVIR AHMED**

Distributed and Collaborative Virtual Environment Research Laboratory, University of Ottawa, Ottawa, ON K1N 6N5, Canada
CORRESPONDING AUTHOR: S. SHIRMOHAMMADI (drshervin@gmail.com)

ABSTRACT This paper proposes a distributed and dynamic networking architecture for massively multiplayer online games (MMOGs). The MMOG networks deal with the challenge of update message exchange among a large number of players that are subject to both mobility (constant change of virtual location) and churn (joining and leaving the physical network at will). Ideally, a player's update messages should be multicasted to the player's area of effect, which is a neighborhood around the player. But, this requires the system to have a centralized indexing service that keeps record of players residing in each region, making the system less scalable as the number of players increases. The use of geometric routing helps alleviate this requirement by exploiting location addressing and thus eliminating the need for IP-search queries. However, geometric routing comes with a number of convergence and performance issues that require solutions for reducing hop-count and minimizing overall delay while providing guaranteed message delivery. In this paper, we propose a geometric routing overlay for message exchange among a large number of MMOG players that not only provides reduced delay and guaranteed delivery, but also supports player mobility and churn. In addition, we enhance our greedy routing method to more efficiently support long distance messages. We demonstrate the effectiveness of our proposed scheme using both theory and simulations.

INDEX TERMS MMOG, Delaunay triangulation, geometric routing, P2P networks.

I. INTRODUCTION

Massively Multiuser Online Games (MMOG) are virtual worlds where a large number of users distributed all over the Internet can interact with each other in real-time. Such environments are very popular with tens of millions of players [1] spending significant time and money interacting with one another in games. Among many challenges, MMOGs must deal with real-time exchange of update messages among a large number of players in order to provide them with a common sense of time, place, and game state. Thus MMOG network design involves distribution of update messages in order to meet strict end-to-end delay constraints, between 100 to 1000 msec [2] among a very large group of users, in the millions. Real-time scalable support for such magnitude of players hints at the usage of either fully distributed architectures where there is no central switching point and thus no potential scalability bottleneck, or hybrid architectures where a centralized system such as a cloud is further enhanced with a distributed delivery mechanism such as P2P networking for even higher scalability [3]. In either architecture, there will be a distributed networking component, which entails solving the following main problem.

In a distributed architecture, routing of the update messages becomes challenging because MMOGs require each player (a node in the network) to send updates to a specific area in the game known as that node's area of effect, in a real-time manner such that players in that area of effect are aware of the node's latest state. The use of IP routing in this scenario is inefficient as it would require a node to send frequent location queries to discover who is located in its area of effect. On the other hand, the use of geometric routing is one way to alleviate the need for such frequent location queries. Geometric routing is a type of distributed routing where updates can be sent to specific coordinates instead of specific IP addresses. This allows an MMOG node to send update messages to a specific coordinate in the virtual space (its area of effect) without the need for querying the IP address of the nodes in that area or having any global knowledge of the topology [4], [5],

2168-6750 © 2014 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

significantly reducing complexity and overhead. As such, geometric routing is an attractive solution for MMOGs in distributed or hybrid modes.

Among many geometric routing protocols, greedy routing is the most commonly used due to its speed, simplicity, and convergence rate. However, greedy routing has the shortcoming that message delivery cannot be guaranteed when routing is performed over a generic overlay. Even though several modifications have been proposed to fix this problem, guaranteed convergence has been gained only at the price of much longer convergence time and hence higher end-to-end delay that is not desirable in MMOGs.

In this article, we present a solution to achieve guaranteed delivery in greedy routing without sacrificing latency. We do so through careful design of the routing overlay and by keeping the routing scheme simple and shifting most of the complexity to the overlay itself. We also solve the challenge of player churn and mobility by proposing a distributed and dynamically updating overlay system which removes the need for a complex routing scheme that would slow down the game.

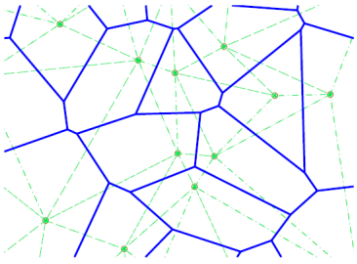

FIGURE 1. Voronoi diagram (solid) and delaunay triangulation (dashed).

We have previously proven [4] that greedy routing can only provide guaranteed delivery over a routing substrate *iff* that substrate satisfies the Delaunay Triangulation constraint (DT, see Figure 1). DT is defined as follows: for a set P of points in the plane *Delaunay Triangulation*, DT (P) is a specific triangulation of the points such that no point in P is inside the circumcircle of any triangle in $DT(P)$. However the generation of a Delaunay graph requires global knowledge of the network topology and involves a high computational overhead for large number of nodes, which is the case in MMOGs. In this article, we also address this problem and design a distributed Delaunay support graph for MMOGs such that it satisfies two conditions: first, we require a totally distributed solution where there is no central point of failure, and second, as network dynamics constantly change, the overlay graph should be repeatedly updated against any changes that affect the Delaunay support constraints.

Perhaps the most important obstacle to achieve the above is the lack of any reference point for all geometric operations. Our proposed solution is to partition the overlay nodes into two groups referred to as *Red* and *Black* groups. Each of these groups forms a graph that can be updated using the other graph as a reference. We will also show how the overall topology will be constructed based on the two sub-graphs.

Furthermore, even though the Delaunay overlay helps greedy routing to achieve its best performance, another major shortcoming still persists: greedy geometric routing, primarily introduced in the context of wireless routing, is not intended to perform hierarchical routing due to the limited transmission range in wireless networks. However, such limitation does not exist in MMOGs. Therefore, failure to use hierarchical routing may result in a large number of hops in routing long-distance messages. We will later address this concern by introducing *distant routing*.

The work in this article is the continuation of our work in [6] and has the following contributions:

1) Proposing a new scheme for topology control and maintenance in greedy supporting dynamic overlays for MMOGs where users frequently join, leave, and change their locations in the virtual world.
2) Introducing a distributed state management scheme for shared objects in the MMOG, which was unaddressed in [6].
3) Enhancing the routing infrastructure in [6] by adding a *Distant Routing* mechanism that reduces the number of hops to route messages between nodes that are long distances away from each other in the overlay.
4) Performing extensive new simulations where the newly-introduced distant routing scheme is simulated, analyzed, and shown to be highly effective with a large number of nodes (12,000) over a 1000×1000 area.

Our proposed solution can be applied to both fully distributed and hybrid (cloud + P2P) architectures, though in this paper we concentrate only on distributed networking.

It should be noted that parts of this paper's sections III to VI have some overlap with our previous paper [6]. The reason is to make this paper self-contained and to free the readers from the inconvenience of having to constantly refer to [6] for mathematical and technical details. After discussing the related work in Section II, Section III covers the general problem model and required preliminaries to the proposed solution. Section IV discusses the general architecture, while Section V presents the overlay updating theory and further details of the architecture. Section VI addresses consistency issues, while Section VII describes the infrastructure for state management. Section VIII covers the proposed distant routing architecture, while Section IX presents the simulation results. Finally, the last section includes the summary and conclusion.

II. RELATED WORK

Many distributed MMOG architectures have been proposed in the past decade. At a high level, [7] categorizes these architectures into 6 main groups: world partitioning, DHT, multicast, fully connected, neighbor-list exchange, and mutual notification approaches, while [8] divides them into structured, unstructured, and hybrid, the latter

itself divided into server+IP multicast, server+unstructured, and server+structured. Our work in this paper focuses on mutual notification according to [7], and structured (fully-distributed MMOGs) or hybrid: server+structured (cloud+P2P MMOGs) according to [8]. *Mutual notification* architecture relies on the geometrical properties of its underlying P2P overlay to maintain peer connectivity, while *structured* architecture uses a deterministic protocol to form a specific P2P graph structure that ensures any node can route a message to any other node. In both of these architectures, the peers are responsible to collaboratively build and maintain a graph structure and the connectivity among the players is defined based on their locations in the virtual environment and their areas of effect. But before describing how these architectures are used to build our proposed system, let us take a look at some of the existing related work in this area.

SCORE is a multicast communication protocol for Large-Scale Virtual Environments [9]. It supports multiple multicast groups and multiple agents to a handle large number of concurrent participants. SCORE dynamically partitions the virtual world into spatial areas and associates the areas with multicast groups. For this purpose, it applies a planar point process to determine the proper cell size. Thus it ensures that traffic at the receiver side remains below a threshold with a given probability. On the other hand, Tsai et al present a routing protocol using virtual coordinates for wireless sensor networks named ABVCap that establishes an equator and a number of meridians, and assigns multiple virtual coordinates to sensor nodes in a four-phase process [10]. ABVCap routing guarantees packet delivery with no computation and storage of global topological information. But to be operational, nodes must be static in ABVCap. Thus it is inappropriate for MMOG where nodes are dynamic and move in the virtual space.

Probably the most straightforward approach to geographic routing is greedy forwarding. This is beneficial especially in densely populated average-case networks. Also in worst-case networks, the cost imposed by greedy routing cannot become arbitrarily high. However, greedy forwarding is not guaranteed to always reach the destination, and so a number of workarounds have been proposed to fix that. Face routing, for example, offers guaranteed message delivery with a cost of $O(n)$, where n is the number of network nodes [11]. However, this is unsatisfactory as a simple flooding algorithm will reach the destination with $O(n)$ messages. Kuhn et al. present GOAFR$^+$ [12] for wireless ad hoc networks combining the features of greedy forwarding and Face routing, guaranteeing that the destination will be reached, but leading to a far less efficient performance. Bose et al. propose another algorithm which is competitive with the shortest path between source and destination on Delaunay triangulations [13]. The latter authors also demonstrate *Delaunay Triangulation* (DT) as one of the structures that supports greedy routing; i.e. greedy routing on DT guarantees message delivery. Due to this property, many researchers have tried to come up with a distributed DT construction algorithm, which we discuss next.

Liebeherr et al. propose a protocol to build a distributed DT substrate [14] by exploiting the locally equiangular property of DT in the 2D space. The basic idea is that each node has a pre-assigned logical coordinate in the plane and checks whether the equiangular property holds among itself and its neighbor nodes. Whenever a violation is detected, the node flips triangles to maintain a correct DT. Flipping triangles is a basic step that is used in a number of Delaunay Triangulation algorithms, and works as follows: consider a triangle with nodes at a, b, and c. Now suppose that the circumcircle of the triangle abc contains a node p. Note that this should not happen in a Delaunay Triangulation. To fix this issue, the flipping step replaces triangles abc and bcp with triangles abp and apc.

As DT is restrictive in the context of dynamic virtual environments, an alternative overlay structure, named Relaxed Triangulation (RT), is proposed by Eliya et al. compromising DT's equiangular property constraint [15]. The same authors also present a message routing protocol over their RT using local knowledge of peers [16] by forming a connectivity architecture and providing guaranteed message delivery over the overlay structure. While the results are generally good, experiments show higher latency in some particular cases.

Steiner et al. suggest another distributed approach to construct a distributed DT in the 3D space [17]. Upon joining of a new node, the tetrahedron that contains the joining node will be split in two. The new tetrahedra will then be checked and flipped if they include any nodes in their circum-spheres.

Lee et al. prove that a distributed DT algorithm is functional if and only if it meets a specific requirement [18]. They propose two join and leave algorithms and provide the functionality proof for sequential joins and leaves using the preceding criterion. But the proposed algorithm assumes that only one node joins or leaves at a time while a maintenance scheme has been used to handle concurrent joins and leaves, as well as node failures. The same authors have suggested a newer protocol for join, leave and maintenance in [19]. However, considering the fast and frequent motion of nodes in an MMOG, these algorithms fail to maintain the structure in the presence of multiple nodes' motions or departures. It is well-known that unless an algorithm provides a great degree of redundancy (and acts quickly) in dealing with failures, disconnections, and churn, it will not work well in open distributed systems such as MMOGs [7].

Similar to the previous works, Hu et al. propose VON: a set of procedures in the case of join, leave and move in order to retain the distributed DT [20]. The join and leave procedures are similar to [18], although VON handles the motion with a separate procedure. VON introduces the new concepts of boundary and enclosing neighbors. Upon the movement of a node, position updates are sent to all of its currently connected neighbors. Then, the boundary neighbors will help the moving node to find new neighbors to connect to. With the motion of one node, previous links to the boundary

nodes would be checked and deleted if the boundary nodes are outside of the moving node's area of effect. In this procedure, it is assumed that all of the other relevant nodes do not change their locations and they have reliable links between themselves. However, as discussed before, the major difficulty of handling dynamic structures in MMOGs is due to the frequent and concurrent motion of nodes. As such, the assumption of single node movement at a time is not realistic.

Our proposed approach is comparable to [18]–[20]. However, unlike these works, we do not need separate procedures for join, leave, and move procedures. Instead, we propose a new idea where the architecture periodically updates itself to take such changes into account.

It should also be mentioned that, in all of the above approaches including ours, MMOG peers collaborate in the message exchange process. Therefore, the problem of cheating and denial of collaboration cannot be avoided. While this problem is beyond the scope of this paper, we refer the readers to Baughman et al. who propose an asynchronous synchronization (AS) protocol providing cheat-proof and fair play-out of both centralized and distributed network games [21]. The AS allows optimistic execution of events without the possibility of conflicting states due to packet loss or cheating, and without the need for the well-known rollback technique. Nowadays avatars, equipment, and virtual currency are traded for real money. So, there might be ill motivation and criminal energy for cheating. Cheating is always a danger for all MMORPGs. Sebastian presents a cheat handling concept which states that never update the state of a node based on the states of other nodes but each node updates its state based on its own old state and the actions of players [22]. It is also shown that event-based simulation instead of the standard update loop solves both cheating and time synchronization problems.

With the above information in mind, let us now describe the details of our proposed system. We start this by explaining the theory and problem model in the next section.

III. PROBLEM MODEL AND THEORETICAL BACKGROUND

In this section, we present a detailed explanation of the problem and all the requirements and objectives. We will then describe the foundation of our proposed dynamic architecture as a solution to this problem in the next section. Before mentioning the problem specification, we start by presenting definitions that will later be used in the problem description.

A. DEFINITIONS

Voronoi Cell – Consider a set of points V, in the Euclidean plane where each point in this set represents the position of an avatar in the virtual environment. For each point $v_i \in V$, *Voronoi Cell* of v_i, $VC(v_i)$, is defined as the set of all points in the plane that are closer to v_i than any other point in V. It should be noted that according to this definition, *Voronoi Cells* of the members of V partition the Euclidean plane.

Delaunay Triangulation – Presuming a set of points V, Delaunay Triangulation of V is a graph $G = (V, E)$ with the edge set E that satisfies the following property $e = (v_i, v_j) \in E$, iff $VC(v_i)$ and $VC(v_j)$ have a side (or at least a point) in common. If $VC\ (v_i)$ and $VC(v_j)$ share a single point, the edge $e = (v_i, v_j)$ is called *unusual edge*. An example of Voronoi Diagram and Delaunay Triangulation is presented in Figure 1.

Vertex Region – Let $G = (V, E)$ be a graph on a set of points V. For each $v_i \in V$, let $N(v_i)$ denote the set of neighbors v_i. We also define *Vertex Region* of v_i in graph G, $VR_G(v_i)$, as the set of all points in the plane that are closer to v_i than to any other point in $N(v_i)$. Figure 2 illustrates $VR_G(v_i)$ with a red triangle, where the nodes in the red triangle are closer to v_i than to any of its neighbors.

Remark 1: Since for each $v_i \in V$, we have $N(v_i) \subset V$, thus, $VC(v_i) \subseteq VR_G\ (v_i)$.

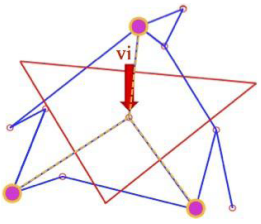

FIGURE 2. **Vertex region of v_i, indicated by the red triangle.**

As Voronoi Diagram partitions the plane according to proximity constraints (the closest node to each point is the node that lies in its pertinent Voronoi Cell), there has recently been a tendency towards the use of Voronoi Diagram (or Delaunay Triangulation as its dual graph) as the base for state management. Tumbde et al. have proposed a Voronoi based partitioning approach to support state management in MMOGs where the nodes in each Voronoi partition communicate through the coordinator of that partition [23]. Therefore, the graph of connections acts as a cluster of star graphs where neighboring stars are connected via their center nodes. However, the authors state that due to the complexity of updating the Voronoi diagram due to node movements, an efficient distributed solution is required.

Chien et al. present Delaunay State Management (DSM) where Delaunay Triangulation is used to partition the virtual world into regions and each region is managed by three super peers [24]. These super peers are handled by servers and therefore are not subject to high mobility. Hu et al. propose Voronoi State Management (VSM) that partitions the Virtual Environment by considering user nodes as points of the Voronoi Diagram [25]. VSM uses the previously discussed VON as its overlay network. However, as discussed already, VON does not support concurrent movement of nodes.

As mentioned in [26], the main difficulty with these schemes is that they are not dynamic enough to maintain a consistent topology in the presence of simultaneous movement of nodes. As our first contribution, we solve this

problem using our proposed Red-Black architecture. But first, we need to define the theoretical aspects of the problem, shown next.

B. PROBLEM DEFINITION

As mentioned before in Section I, the use of geometric routing alleviates the need for IP-queries and lets nodes directly multicast the updates to their area of effect where the interested entities are located. Greedy routing, which is among the simplest and most common online geometrical routing algorithms, tries to bring a message closer to the destination in each step using only local information. Thus, each node forwards the message to the neighbor that is most suitable from a local point of view. The most suitable neighbor can be the one who minimizes the distance to the destination in each step. Graph $G = (V, E)$ supports greedy routing *iff* greedy routing on G delivers each and every packet to the closest node in V to the packet's destination. It should be noted that if the destination of a packet exists in V, greedy routing should deliver the packet to the exact destination. Therefore, in order to assure guaranteed delivery of updates while using greedy routing, the overlay topology should support greedy routing. The challenge would then turn into the dynamic construction of an overlay topology that supports greedy routing. We shall therefore propose a dynamic topology control scheme that can handle high churn and node mobility in MMOGs.

In the following sub-section, we will show the analytical description of greedy routing support over a graph. We will then present our approach for the generation of greedy supporting graph in section IV.

C. THEORETICAL FOUNDATION OF THE PROPOSED APPROACH

The proposed solution towards the support of a Delaunay graph relies on the following two theorems:

Theorem 1: *Graph $G = (V, E)$ on the node set V in the Euclidean plane supports greedy routing iff for every node $v_i \in V$, $VR_G(v_i) = VC(v_i)$.*

Theorem 2: *Graph $G = (V, E)$ over the set V in the Euclidean plane supports greedy routing iff for all v_i, $v_j \in V$, $VR_G(v_i) \bigcap VR_G(v_j) = \emptyset$.*

The proof of these theorems can be found in [4]. *Theorem* 2 provides an interesting insight to the gradual update of a random graph towards a greedy supporting structure. Based on *theorem* 2, we can modify any arbitrary graph to transform it into a greedy supporting graph. In order to efficiently (resulting in minimum number of connection) transform a graph to a greedy supporting one, it suffices to connect each pair of nodes whose VR_Gs have a nonempty intersection and delete the redundant edges afterwards. In the following sections, we will discuss this solution in detail.

IV. RED-BLACK ARCHITECTURE

Our proposed architecture is generic and composed of a greedy supporting graph covering all MMOG nodes. This overlay will be used for any kind of message exchange, except for positional update and topology control messages.

We mentioned that nodes are usually in movement and they use location update messages for announcing their new locations to the other nodes. A general method for synchronizing nodes is to send their location update messages periodically with a specific rate. Therefore, every T seconds, nodes are supposed to generate location update messages and send it to their areas of effect. T could be anywhere from 10 to 280 msec, depending on the game genre [27]. The challenge of sending these updates in a distributed topology is the lack of a structure that can be used for transferring the update messages to the related areas. In order to deal with this problem, we have devised a dual phase topology update mechanism where the nodes are partitioned into two non-overlapping sets referred to as *Red* and *Black* sets, where each is modified in one of the update phases. The whole topology graph is composed from red and black sub-graphs and these two sub-graphs are kept up-to-date to support the greedy routing; i.e., they are DT. As a result, the main graph will also support greedy routing. An example of the simultaneous red and black graphs is shown in Figure 3. Each red and black set is able to use the other set as its landmark in order to update itself. As such, we color our nodes with *Red* and *Black*. Each group generates and distributes its update messages with a $T/2$ time shift with respect to the other group.

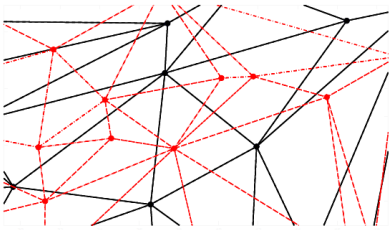

FIGURE 3. The red-black architecture.

Assume that red nodes send their location updates in times $0, T, 2T, \ldots$ and black nodes send their updates in times $T/2$, $3T/2, 5T/2, \ldots$. At the time of updates for red nodes, the red graph has probably changed due to the movement of red nodes and it might not support greedy routing anymore, causing our general graph to fail to support greedy routing. However, the good news is that the black structure, which has just been updated ($T/2$ seconds ago), still supports greedy routing. Of course, during this $T/2$ seconds, some of the black nodes might have also moved from their previous locations; however, the change of their positions has not yet been announced. Considering the positions in the MMOG as virtual concepts, the position change of the black nodes will take effect upon their own update phase and can be ignored for the update of the red graph. The same method will be used during the update of black nodes, except that this time the red structure is used as the reliable structure.

These two sets can be selected arbitrarily, although uniform distribution of red and black nodes in the plane can help improve the convergence rate. A simple and practical method can be the use of *IP-addresses* in the grouping process where even and odd *IP addresses* are assigned to red and black groups, respectively. The other issue that affects the performance is that it requires the nodes to stay synchronized such that their update procedure can run in a common time interval. For MMOGs, we can use two synchronization schemes: distributed synchronization via message exchange, and the continuation of a global time. The former can introduce overhead in terms of message complexity and time delay, thus affecting players' interactivity and game quality. Developers can use the mechanism of a global time like Network Time Protocol (NTP) which fits well in this environment, as it is a protocol for synchronizing the clocks of computer systems over variable-latency data networks. In the following section, we will discuss the updating mechanism of our algorithm, and mathematically prove its functionality.

V. RED-BLACK TOPOLOGY UPDATE

Through the rest of this section, without loss of generality, we will explain the red group update phase, knowing that the update phase for black nodes is similar. The algorithm used in this update phase will allow the red nodes to exchange their virtual location updates and then change the related red subgraph so that it resumes supporting greedy routing. We refer to this algorithm as the *resumption algorithm*. Next, we describe this algorithm.

A. UPDATING RED STRUCTURE USING THE BLACK GRAPH

At the beginning of the update phase, the red nodes announce their new locations to their previous red neighbors. Therefore, each node gets to know the new location of its old neighbors in the red graph and uses this information to calculate its *Vertex Region*. It should be noted that there is no single point where all of the required information for checking the vertex regions intersections is gathered. Instead, a distributed method is exploited where the nodes in the black group collaborate in order to calculate the intersections between the red vertex regions. The specification of each VR_R is sent to the black nodes whose VR_B have at least a point in common with the VR_R. Assume the case when a black node named v_j^b finds a common point such as $x$between the two red VR_Rs where $x \in VR_B(v_j^b)$. The v_j^b will then notify one of the involved red nodes to make a link with the other one. An example of this scenario is presented in Figure 4.

Later we will refer to the desired set of black nodes for every red node v_i^r, the *Black Parents* (BP) of v_i^r, where $BP(v_i^r) = \{v_j^b | VR_B(v_j^b) \cap VR_R(v_i^r) \neq \emptyset\}$. The problem is how to deliver the $VR_R(v_i^r)$ specifications to the members of $BP(v_i^r)$. This can be solved by first delivering the message to one of nodes in $BP(v_i^r)$ and then propagating it to the other nodes of this set. We will discuss each of these two steps separately.

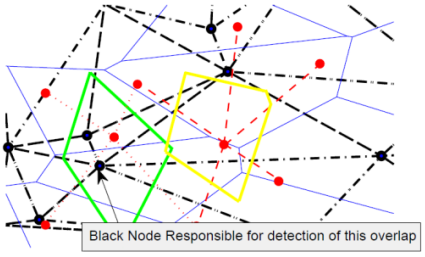

FIGURE 4. The distributed method to checking intersections.

1) MESSAGE DELIVERY TO A MEMBER OF BLACK PARENT SET

Stojmenovic et al. have investigated the solution to a problem similar to this step [28]. Their problem was how to deliver a message to a point whose location is not precisely specified; instead it only needs to be located within a specified polygon. Consider the first phase of our problem; i.e, the delivery of $VR_R(v_i^r)$ to a member in $BP(v_i^r)$. If $v_i^r \in VR_B(v_j^b)$, it will be clear that $v_j^b \in BP(v_i^r)$. Therefore, the first step would be the delivery of this message to v_j^b. In order to do this, it is enough for each red node to know the nearest black node to itself (so called its black owner). During the red update period, each red node encapsulates the specifications of VR_R in a message and sends this message to its previous black owner and asks the black owner to return the message to it using the black graph. As the black structure supports greedy routing, the packet will reach the new black owner of the red node that is probably the same as the current black owner.

2) BROADCAST OF THE MESSAGE AMONG THE MEMBERS OF THE BLACK PARENT SET

When the message reaches the desired parent set, it should be broadcasted to the other members of $BP(v_i^r)$. It should be noted that each black node only knows about its own VR_B and the location of its black neighbors. Upon the reception of every new VR_R specification by a black node, that black node propagates the received VR_R specification to every black neighbor in the black graph whose VR_B's common side (or point) has a nonempty intersection with the VR_R encapsulated in the message (except the node that the message has just been received from).

Theorem 3: *Using the above method, the message encapsulating $VR_R(v_i^r)$ will be received by a black node v_j^b, iff $v_j^b \in BP(v_i^r)$.*

Proof: The necessity condition is obvious referring to the algorithm description. The sufficiency part can be proved using contradiction. Suppose that there exists a black node $v_j^b \in BP(v_i^r)$ that does not receive the message. Figure 5 highlights the message in red and the black node in yellow. Assume that S is the biggest connected subset of $BP(v_i^r)$ that have not received the message and $v_j^b \in S$. Obviously, $1 \leq |S|$. Suppose that v_k^b is the black owner of the v_i^r. As each VR is convex, if we draw a line from a point in $VR_B(v_j^b) \cap$

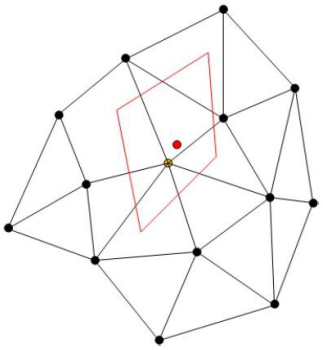

FIGURE 5. Proof of Theorem 3: the message highlighted in red will be received by the black node highlighted in yellow because it satisfies the condition specified in theorem 3.

$VR_R\left(v_i^r\right)$ to a point $VR_B(v_k^b) \cap VR_R(v_i^r)$, all of points belonging to this line should belong to $VR_R(v_i^r)$, as well. If S has at least one side in common with VR_Bs of the nodes that have received the message, that causes a conflict with the assumption of S being the biggest possible subset, because we can add another black node to S: the node from the side of v_j^b such that the line passes through its black VR_B to S. ■

It should be noted that that the locations of nodes in a virtual environment is a virtual concept. As such, we keep using the older red-black graph while one of the two subgraph (red, in the case discussed in this section) is being updated. After the update, the new virtual locations of the red nodes will be considered in the new red-black graph. Therefore, at all times, even when the red nodes were using the old locations, the red-black graph is a Delaunay Triangulation and thus provides guaranteed delivery of the routing messages.

B. TOPOLOGY REFINEMENT

Previously, we described our approach to delivering the specification of $VR_R(v_i^r)$s to the members of $BP(v_i^r)$ in order for them to check the probable nonempty intersections between $VR_R s$. We have also mentioned that each black node is responsible for finding VR_R intersections happening inside its VR_B, and if it finds an overlap of two nodes between VR_Rs, it will send a message to one of these pair requesting it to consider the other red node as a neighborhood candidate. After receiving this message, the red node will discover the new set of red nodes and send a "Neighborhood Request" message to all those future neighbors.

Upon the reception of a new neighborhood request message at a red node v_i^r, it draws the orthogonal bisector of line passing through it and the candidate. The red node will then add the candidate to its neighbor set, *iff* this orthogonal bisector has an internal intersection with $VR_R(v_i^r)$; i.e., there is a subset of this $VR_R(v_i^r)$ that is closer to the candidate than to v_i^r. If this condition is satisfied, the neighborhood request is granted and $VR_R(v_i^r)$ will change to add the new $VR(v_i^r)$ member to the previous subset. Therefore, some neighbors might be considered as redundant in the new $VR_R(v_i^r)$. The redundant neighbors are recognized according to the following criteria: the orthogonal bisector of edges between the main node and the redundant neighbors are located fully outside $VR_R(v_i^r)$ of the main node. Therefore, deleting the edges between the main node and the redundant neighbors will not affect $VR_R(v_i^r)$ of the main node.

The good point is that it is not necessary to inform the other end of an edge when the edge has been deleted. In order to prove this claim, we refer to the definition of neighborhood in the *Delaunay Triangulation* (and also any other undirected graph) that is a mutual concept. Therefore, the neighborhood possibility is enough to be checked from the viewpoint of only one of the ends of an edge. Similarly, a redundant edge is enough to be redundant from the viewpoint of both of its ends.

The proposed algorithm needs to check the old members of the neighbor set for possible redundancy upon the joining of each new neighbor. Therefore, the complexity of this algorithm is of $O(n^2)$. As the degree of the nodes in a structured graph is usually small, the algorithm execution won't consume a considerable amount of processing power. However, if the number of neighbors increases beyond a certain threshold, it will be probably more rational to take advantage of Fortune's algorithm [29], designed for finding the *Voronoi Diagram* for a set of points. The complexity of Fortune's algorithm is of $O(nlog(n))$, where n is the cardinal of set of points. In this case, it is preferred to maintain all of the candidates in a set and run the update algorithm only once after the reception of all of these messages. The algorithm will determine the *Voronoi Cell* of the main node, and the neighbors set for the main node will then be defined as in the Delaunay Triangulation case.

C. TOPOLOGY GRAPH CONSTRUCTION BASED ON RED AND BLACK SUB-GRAPHS

We previously discussed the solution to the problem of updating the red graph using the greedy supporting black graph as the reference. In this section, we will adapt this method for use in the update procedure of the overall topology graph. This can be achieved by a slight modification as described in the following paragraph.

After receiving a new $VR_R(v_i^r)$, each black node v_j^b checks whether the orthogonal bisector of the line drawn from v_j^b to v_i^r has an internal intersection with the specified $VR_R(v_i^r)$. If so, v_j^b makes itself candidate as a neighbor of that red node in the overall topology graph. Therefore, it sends neighborhood request message to v_i^r in which it includes its own location. v_j^b re-computes its overall graph in the presence of v_i^r as a new neighbor. All the other details of the algorithm are similar to the previously explained algorithm for red sub-graph update.

It should be noted that the entry of a new node in the network or departure of an existing node may temporarily make a small part of the graph around the entering/exiting nodes unable to support greedy routing. However, these new

entries or exits will be taken into account in the next update cycle and from there it resumes supporting greedy routing.

D. FURTHER IMPROVEMENTS OF THE ALGORITHM

We proposed a method to check the intersections of nodes in both red and overall graphs, and adapted these graphs (connect new links and maybe delete some old links) to support greedy routing. As explained before, a message containing the specification of VR of node v_i^r, $VR_R(v_i^r)$ should be sent to the black owner of v_i^r where it will be sequentially broadcasted to the other Black Parents of v_i^r.

This algorithm can be further enhanced as follows: after the reception of $VR_R(v_i^r)$, the black node v_j^b checks the overlap of VR_Rs in $VR_B(v_j^b)$. However, before propagating the message to the other black owners of v_i^r, v_j^b updates the received $VR_R(v_i^r)$ and broadcasts this new $VR_R\left(v_i^r\right)$ following the same scheme. As the updated $VR_R(v_i^r)$ is relayed, the black parents can work on the latest *vertex region*. The update in $VR_R(v_i^r)$ assumes that all new red candidates (found in v_j^b) will finally become neighbors of v_i^r.

E. FUNCTIONALITY PROOF

According to theorem 1, $VR_R\left(v_i^r\right) = VC(v_i^r)$ for every v_i^r in the resulting structure. With this new method, in each step, the new $VR_R\left(v_i^r\right)$ will be a subset of the previous $VR_R\left(v_i^r\right)$ that still contains $VC(v_i^r)$ that is supposed to be the final $VR_R\left(v_i^r\right)$. Therefore, the new method checks for any probable nonempty overlap of regions that is similar to final VR_Rs. At the final stage of the algorithm, VR_Rs of the constructed red graph will be separate sets. Therefore, the red graph will support greedy routing according to theorem 2.

VI. CONSISTENCY ISSUES

In order to provide a criterion for consistency measurement, we introduce a new term named location age. At each arbitrary instance of time, location age of a node is defined as the time passed since the node has received the last location update message from the other nodes. According to this definition, an updating method is considered to be more consistent *iff* it maintains a lower value on location age. To evaluate this parameter, let us first consider a simple case where all nodes send their information at times T × i for integer values of i. In each of these intervals [T × (i-1), T × i], at each moment, the age of the information is a parameter t between 0 and T which grows as we move forward in time, until we reach the next update time. Thus, the average is $\int_0^T (N-1)\,t/T \cdot dt = (N-1)T/2$, where N is the total number of nodes.

The average for our proposed red-black algorithm is calculated similarly but taking into account the fact that red and black nodes send their updates with a T/2 time shift. Here, the location age can be found as follows. If the total number of red nodes is m, this parameter for the red nodes would be:

$$\int_0^{\frac{T}{2}} \left((m-1)\,t+(N-m)\left(t+\frac{T}{2}\right)\right)/T \cdot dt + \int_{\frac{T}{2}}^{T} \left((m-1)\,t+(N-m)\left(t-\frac{T}{2}\right)\right)/T \cdot dt = (N-1)\,T/2 \tag{1}$$

and for the black nodes, it would be:

$$\int_0^{\frac{T}{2}} \left((m)\left(t+\frac{T}{2}\right)+(N-m-1)\,t\right)/T \cdot dt + \int_{\frac{T}{2}}^{T} \left((m)\left(t-\frac{T}{2}\right)+(N-m-1)\,t\right)/T \cdot dt = (N-1)\,T/2 \tag{2}$$

Therefore, the expected location age for this method will still be the same as that of the simple case. Hence, our change in the method of sending location update messages does not affect the consistency of the architecture.

VII. INFRASTRUCTURE FOR STATE MANAGEMENT

In the previous sections we described our proposed solution and mathematically proved that it works; i.e., it can guarantee to deliver update message between distributed players in a timely fashion while dynamically updating its substrate to support mobility and churn. However, in addition to players, there are other objects in the scene that require update messages. For example, a table that lies in a room might change position due to being kicked or moved. But unlike a player, a table is not a node in the network. In this section, we discuss how we perform state management for such objects.

A. CONNECTION TO NODES RESPONSIBLE FOR OBJECTS IN AREA OF EFFECT

Non-player objects in the game require an owner to take care of their state management while the owner is not necessarily assigned in a fixed term basis, because the owner could move far from the object due to mobility. In the approaches proposed in [23], [25], and [30], the state of every object is managed by the closest node and in case of a change in this state, this node is responsible for sending the pertinent update message to all nodes interested in the state of this object. For example, [31] presents a two-fold approach for state management in P2P virtual environments where a structured overlay is used for object state management and a triangulation overlay is used for avatar state and group membership management. [32] presents a state management and persistency architecture for P2P virtual environments called Pithos, which is a group-based distance-oriented method that covers both responsive storage system and the security aspects of scalable distributed storage. While the above approaches lead to good results, they are not optimized for our specific case of Delaunay-based P2P overlays. The only exception is [33] which also uses Delaunay-based P2P overlays and forms a spanning tree to support event notification to peers positioned

in the object's area of effect. The spanning tree is constructed based on reverse compass routing originally proposed for geometric networks. But it is hard to predict the behavior of a spanning tree in terms of hops and latency. On the other hand, our approach takes advantage of greedy routing which is a simple and fast routing process. In addition, we introduce new techniques to reduce hop count by using a hybrid structure for our state management.

If the peers are assumed to be the centers of Voronoi regions in the Voronoi diagram, the closest node to each object is the owner of the Voronoi region in which the object is located. Therefore, the object can be managed by the node owning the Voronoi region in which the object lies. In our proposed approach, we build a DT substrate and use it for state greedy routing, as show in the previous sections. Greedy routing over a DT is guaranteed to deliver a message to the closest node to the destination. Therefore, if node v desires to change the state of object O which lies in the Voronoi cell of node u, node v will send its request toward O using greedy routing. Being routed over the DT substrate, the request will eventually reach the closest node to O, i.e., node u. Therefore, the first part of a state change is easily taken care of. Node u will then process this request considering appropriate game logic and this would probably result in a change in the state of object O. If so, node u has to announce this state change to all nodes that are interested in the state of object O; i.e., nodes that are in O's area of effect. Such announcements can be done using greedy routing over the DT substrate. Area of effect of a node is usually considered to be circular [25]. However, the more realistic view is to consider it a general area. Moreover, the effect of probable obstacles in the virtual environment should be taken into account. Generally, we can choose any arbitrarily shape for the area of effect of each node, but we can always assume that area of effect is a connected area.

B. STATE MANAGER RENEWAL

Another problem in state management is the possible change in the state manager of an object due to the movement of nodes. As the nodes move, state managers will also change. A new manager should become aware of the existence of an object and get its complete state. This problem can also be solved using the greedy routing algorithm supported by our proposed structure. At the end of each topology update phase, each node checks to see if all of its previously relevant objects are still in its Voronoi region or not. If an object has moved out of its Voronoi region, it is no longer responsible for state management of this object and the complete state of this object should be sent to its new manager. This can be accomplished by sending a message containing the state towards the location of the object using greedy routing. Then the message would get to the closest node to the object; i.e., the new state manager of this object.

VIII. DISTANT ROUTING

In environments such as MMOGs, messages are usually desired to be sent to specific coordinates in space. But since greedy geometric routing takes only geographic information into account, it can be subject to significant performance degradation in the case of long distance connectivity in the game. To explain this problem, consider Figure 6 and assume that node s wants to send a message to node d (Figure 6 represents the location of nodes in the virtual environment). Since the virtual distance between two nodes is not related to the real delay of transferring a message between them (the physical delay), it is more reasonable to use hop count as a measure of delivery delay. Using geometric routing, the message needs to go through a route like the exhibited path in Figure 6 which consists of 16 hops. On the other hand, if s knew the IP-address of d or a node in the vicinity of d, the message could be delivered by fewer hops. This is due to the fact that, if IP address is known, the message can be transmitted directly instead of going through the overlay, facing shorter hop count.

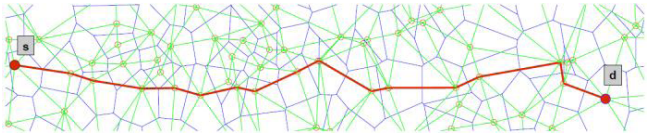

FIGURE 6. **Routing from node s to node d.**

To achieve this, we propose to add some *shortcuts* to different regions in the MMOG through the use of super peers or cloud servers assigned to each of the MMOG regions. In particular, the participation of super peers as virtual nodes that provide a sort of shortcut to the different regions of the MMOG can be effective. Steiner et al. present a Delaunay based overlay by inserting a small and bounded number of additional links called shortcuts [34]. A peer can use its shortcuts among the nodes that are physically close. Like our proposed method, [34] can reduce the number of hops in the overlay by augmenting the overlay with additional links. However, [34] assumes peers are capable of inserting extra links. Furthermore, this algorithm requires information like peers' physical location that could make its implementation costly. For this purpose, we divide the environment into a number of small partitions such as S_1, S_2, etc, and assign a node v_i to every S_i as the representative of nodes in S_i. If a message is to be sent from node s to a distant node d, the routing protocol can send the message to the representative of the area where d lies. Greedy routing will then be used within the partition to guide the message to d, as discussed next.

A. PARTITIONING AND REPRESENTATIVE PLACEMENT

As just mentioned, we aim at portioning the virtual environment into a number of regions where each region has a representative. Generally speaking, area representatives can be potentially chosen either from the peers in each partition, referred to as super peers in P2P terminology, or a pool of MMOG nodes including cloud servers. The actual selection process of the super peer or the cloud server is beyond the

scope of this article, and many techniques can be used for this purpose [35], [36]. For simplicity, in this paper we refer to both super peers and cloud servers as super peers.

To balance the load on super peers, it is desired to partition the environment such that every division contains almost the same number of nodes. This problem is known to be intractable from theoretical point of view [37]. On the other hand, since an exactly balanced partition is not so important in this case, this can be accomplished roughly by partitioning the environment with a small grid pattern, and then assigning a number of adjacent boxes to each super peer such that the number of assigned nodes is approximately balanced. The grid pattern can be changed over time; we just need it to be fixed for the duration in which we try to route some messages via the super-peers in this grid. Since node locations are just a virtual concept, we can do the computations of a new grid without switching to it, and then, after the computation is done, at an agreed upon time, all super-peers switch to the new virtual locations of their representatives.

Along the process, the IP-address of super peers and assigned partitions are shared among the super peers. In this distant routing process, every super peer needs to be assigned a virtual position as the representative of its area. If the super peer is not a cloud sever, this virtual position is different from the virtual location of the avatar pertinent to this peer and therefore, the virtual position of every super peer would not be changed by the movement of its related avatar. In fact, the avatar can even be out the partition assigned to a super peer.

In the fully-distributed architecture, due to the natural movement of nodes in the game, a specific point cannot be selected to be the best position for the representative, as the nodes in the partition may move away from this point and gather around another point far from it. To mitigate this problem, every super peer can use m_i representatives instead of a single one and uniformly distribute these representatives in the assigned partition. This would increase the number of connections to the super peer. However, as the typical node degree in Delaunay Triangulation graph is around 5 to 7, super peers are expected to have enough processing power to handle these connections. Every super peer can adjust the relevant m_i number in its area considering communication capabilities and number of nodes in the area. It is clear that increasing m_i would decrease the routing delay but increase the load on the super peer. The representatives are randomly colored, red or black, and their existence is announced in the next pertinent structure update period.

B. MODIFIED ROUTING PROTOCOL

In order to explain the modified routing protocol for long distance messages, we describe the case where node s wants to send a message to position d. Node s refers to a *greedy radius threshold* in order to decide about the next routing hop. This greedy radius can either be selected by every node according to its range of interest, or it can be selected and advertised by the super peers of each partition. The proper selection of greedy radius depends on the application. If the distance from s to d is less than the greedy radius of node s, the message will be routed through simple greedy routing. Otherwise, node s forwards the message to its own area representative, which in turn forwards the message to the super peer belonging to the partition in which d lies. Upon reception of this message, the super peer uses normal greedy routing to deliver the message to d. It can be observed that increasing greedy radius causes the routing algorithm to converge to simple greedy routing while decreasing this parameter will force the algorithm to route most packets through super peers and would lead to a larger load on the super peers. It should also be noted that the proposed method does not have any effect on the communication or computation load of the nodes as they only need to know their own super peer. The super peers would then deal with other partitions and their representatives. In order to avoid the search query for finding the super peer, another convention is to assign the super peer of a region to be the node that is closest to the center of that region. When a super peer moves away from the center or leaves the virtual environment, the super peer task is handed over to the next most eligible candidate. More details regarding super peer selection can be found in [38].

IX. SIMULATIONS AND RESULTS

In this section, we present the experimental results obtained from MATLAB simulations of the proposed architecture. The simulations are presented in two different subsections. The first subsection presents the result of the experiments with dynamics of the Red-Black Architecture, and the second subsection demonstrates the results of the simulations of the proposed distant routing structure presented in Section VIII.

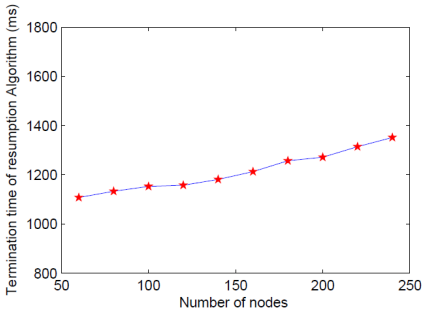

FIGURE 7. Termination time of resumption algorithm versus number of nodes.

A. RED-BLACK ARCHITECTURE

In the previous sections, we have argued and proved that after completion of the resumption algorithm, the resulting graph will support greedy routing. But, the question is how fast does it come to completion? We have performed simulations for various numbers of nodes that spread randomly in a specific virtual area of [50 × 50] units in the game. Round Trip Times (RTT) are selected to be uniformly distributed in [100, 200] ms interval. We also assumed that each node can move at most one unit in an arbitrary direction in each simulation step. We have plotted the average termination time of our

algorithm versus the number of nodes in the game in Figure 7. Termination time is defined as the time that the protocol takes until all the messages in one update phase, say that of the red nodes, have reached their respective destinations. That is, after this time, there is no more communication happening for the resumption algorithm of this phase. Naturally, this time directly depends on the number of hops a message needs to take before reaching its destination. As can be seen in the figure, the number of nodes in the area has a slight effect on the termination time of the proposed method. This could be explained as follows.

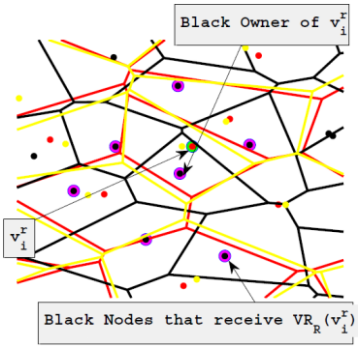

FIGURE 8. Black nodes that receive $VR_R\left(v_i^r\right)$ – the specification message, the yellow points are previous position of red nodes, and red and black points represent red and black nodes, respectively.

As previously mentioned in Section V, every red node sends specifications of VR_R to its black owner. This message will be then broadcasted to other black parents of this node. In the improvement of the proposed method, we mentioned that every black node will update the received VR_R specification before propagating the message to other neighboring black parents. Therefore, when there exists a nonempty overlap between this VR_R and some other VR_R-s, we can roughly say that the VR_R will be cut from this side and thus, the message will not be sent further in this direction. Assume that we are considering the black nodes that receive $VR_R(v_i^r)$ specification (Figure 8). Therefore, it can be said that this VR_R specification will be delivered only to those black nodes that are black parents of v_i^r corresponding to the final version of $VR_R\left(v_i^r\right) = VC(v_i^r)$. If red and black nodes are uniformly distributed in the Euclidean plane, $VC(v_i^r)$ will overlap with around 4 to 8 black VR_Bs and thus the $VR_R(v_i^r)$ specification will be delivered to nearly 4 to 8 black nodes. It can be argued that the average degree of vertices in a DT graph is usually in the range of 3 to 7, and with the increase in the number of nodes in a specific area, this parameter does not change. However, with this change, the average distance of neighbor nodes; i.e., the length of edges will decrease. But, the length of edges in graphs of MMOGs is just a virtual concept and is not relevant to the real time of message transfer.

We have also demonstrated the overall number of messages sent for the resumption algorithm versus the number of nodes, in Figure 9. As can be seen, the number of messages

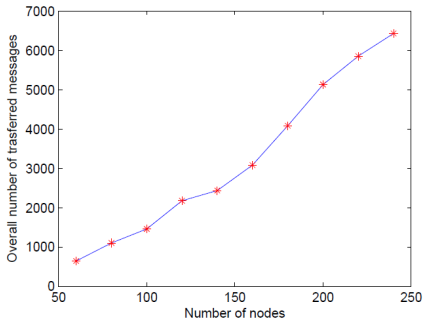

FIGURE 9. The number of transferred messages versus number of nodes.

sent in the resumption algorithm has an approximately linear relation with the number of nodes; i.e., the average number of messages transmitted by every node is almost constant. This issue is similar to the previous one seen in algorithm termination time. Hence, it can be concluded that the proposed method is fully scalable, and increasing the number of nodes does not affect its complexity. Therefore, it can be an alternative to the centralized-only client-server architecture.

B. DISTANT ROUTING

In this subsection, we demonstrate the efficiency of our proposed distant routing structure described in Section VIII. We used 12000 virtual nodes (peers) uniformly spread over a [1000(units) × 1000(units)] area in the virtual space and simulated the proposed routing for various combinations of number of super peers, number of representatives of each super peer and greedy radius thresholds. For the sake of simplicity, we assumed that all super peers use the same number of representatives and all virtual nodes exploit an equal greedy radius threshold. However, it should be noted that all of the previous assumptions including the uniform distribution of nodes do not affect the functionality of the proposed algorithm. In the simulation, the nodes will constantly change their locations by moving one step in a randomly selected direction during each simulation step where the direction is derived according to a uniform distribution.

As argued before, we use average hop count as a reasonable criteria for assessing the performance. Therefore, we ignore the conversion of the hop count to real delivery delay time and hence we do not consider the possible effect of traffic and congestion in the routing protocol delay. Virtual nodes generate packets with an arbitrary rate toward destinations selected randomly in the area with a specific distribution intended to resemble the real situation in virtual environments. As mentioned before, the information exchange and communications of each peer is usually related to the perceptions of its avatar. Therefore, the nodes have more tendency towards connecting with closer neighbors and to send messages to nearer points. To simulate this condition, nodes select destinations of their generated packets with a distance distributed according to gamma distribution, which is a general form of probability distribution that has the flexibility to present a wide range of

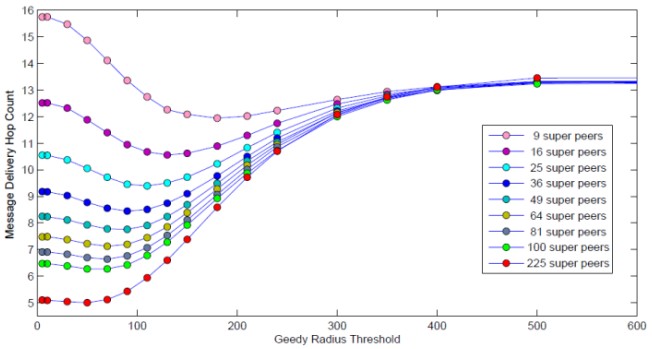

FIGURE 10. The average message delivery hop count versus greedy radius threshold.

behaviors of probability distributions by changing its parameters. The probability density function of this distribution is of the form:

$$f_X(x;k,\theta) = x^{k-1}\frac{e^{-x/\theta}}{\theta^k \Gamma(k)} \quad \text{for } x \geq 0 \text{ and } k,\theta > 0 \quad (3)$$

where k and θ are two parameters named *shape parameter* and *scale parameter*, respectively. We choose $k = 3$ and $\theta = 50$ as a pragmatic choice representing the conditions of the game, since that choice means a given node will send messages to closer nodes with higher probability and to farther nodes with lower probability, which makes sense in a gaming environment. Therefore, in our simulation, nodes choose the destination of their packets with a distance selected according to gamma distribution with $k = 3$ and $\theta = 50$ and a uniformly distributed angle. Using the above probability function, the average hop count of message delivery versus greedy radius threshold was simulated for various numbers of super peers, as shown in Figure 10 and assuming that every super peer uses only one representative. Considering our proposed routing algorithm, it is clear that the delivery delay of simple greedy routing can be obtained by setting the greedy radius threshold to infinity. In this simulation, a radius of 1000 is equivalent to infinity given the total size of the experiment. However, as can be seen from Figure 10, with a radius as small as 500, the hop count value converges already. Therefore, the average message delivery hop count for simple greedy routing in this experiment is about 13.32 hops.

As can be seen in Figure 10, using very small numbers of super peers (9 super peers compared to 12000 nodes) would not decrease the message delivery delay. This is due to the fact that with such a small number of super peers, the size of every partition would be very large and hence, the average distance from even the closest super peer to destination would be larger than the average distance from sender to destination. However, this situation improves with increasing the number of super peers. Using the proposed structure with 25 or more super peers would lead to a far better performance compared to simple greedy routing in terms of delay metric.

Another important point in Figure 10 is the relation between message delivery delay and greedy radius threshold. It can be seen that starting from a small (close to zero) greedy radius threshold and increasing this radius, the average delivery delay decreases until reaching a maximal point. Further increase in the radius (from that maximal point on) slightly increases the average delivery delay towards converging to the average message delivery delay of simple greedy routing protocol.

This behavior can be easily explained. In the lower part of the horizontal axis (small radii), we should increase the greedy radius threshold to decrease the message delivery delay. This is because of the larger distance from the sender to the destination compared to the distance from the closest super peer to the destination, and it means that it is better to use the route through super peers for wider range of destinations.

On the other hand, sending more messages using simple greedy routing becomes more beneficial in the higher part of the horizontal axis where the local tendency of nodes dominates the presented effect of the first part.

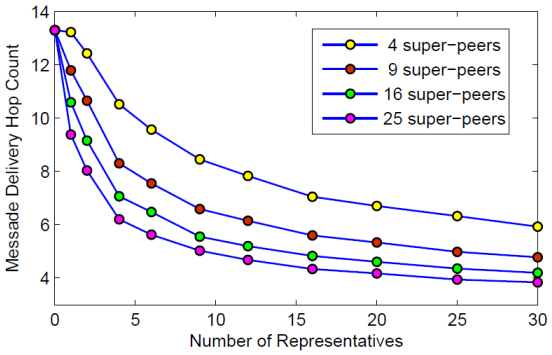

FIGURE 11. The average message delivery hop count versus the number of representatives.

We have also simulated the average message delivery delay versus the number of representatives, used by a given super peer, for different number of super peers, as shown in Figure 11. In the figure, the beginning of each curve exhibits the case with no representative, which is equivalent to the simple greedy routing protocol. Therefore this point is independent of the number of super peers and is common among all curves.

For each simulation (each point on the curves), the greedy radius threshold is set to the optimal value obtained by experimental results, i.e., the value which leads to the lowest possible message delivery delay. As can be seen from the figure, increasing the number of the representatives of super peers will decrease the average message delivery delay in an exponential manner. The convergence point of all curves is observed to be around 3 hop counts. This is due to the fact that with a very large number of representatives, it would be rational to always use the route through the super peers and this route consists of 3 hops, one from the sender to the super peer of the sender's partition, one from there to the super 'peer of the destination's partition, and one from this super peer to its closest neighbor to the destination (note that the mes-

sage should be delivered to the closest ordinary node to the destination). The figure also shows the efficiency of the proposed routing structure compared to the simple greedy routing protocol. As can be seen, even in the presence of a small number of super peers; i.e., 25 super peers (which is about 0.2% of 12000 peers), and a moderate number of representatives for each super peer (9 representatives), the obtained average message delivery delay is about 60% less than the simple greedy routing protocol. This can be computed as follows: the first data point is with 25 super peers and 0 representatives, which means with just greedy routing, and there the hop count is roughly 13 (read from the leftmost point on the lowest curve). The second data point is with 25 super peers and 9 representatives per super peer, and there the hop count is roughly 5. The reduction from 13 to 5 is $(13\text{-}5)/13 = 0.61 \approx 60\%$

X. CONCLUSIONS AND FUTURE WORK

Handling users' mobility and churn is one of the most important issues in the design of resilient distributed or hybrid MMOGs. In this article, we proposed a solution for that which benefits from greedy routing's simple and fast routing process, and we introduced a new distributed algorithm for maintaining a greedy support graph among MMOG nodes, which partitions the topology into two different parts where each part is alternatively updated in one phase using the other part as a reference. We also introduced new techniques to reduce hop count by using a hybrid structure and also proposed an infrastructure for state management. Simulation results demonstrated the efficient performance of our solutions.

For future work, the proposed routing scheme can be further optimized to take load balancing and path cost into account. Moreover, the optimal selection of super peers and partitioning the virtual world for best routing and load balancing in the cloud are issues that warrant further studies. Finally, evaluating our design using real world implementations or actual network traces is an important future work that can help us understand both the positive aspects and the shortcomings of the proposal. That evaluation can take into account many practical issues that are not considered in a simulation environment, such as network congestion and actual link delays.

REFERENCES

[1] S. Shirmohammadi and M. Claypool, "Massively multiplayer online gaming systems and applications," *Multimedia Tools Appl.*, vol. 45, no. 1, pp. 1–5, 2009.

[2] M. Claypool, "The effect of latency on user performance in real-time strategy games," *Int. J. Comput. Telecommun. Netw.*, vol. 49, no. 1, pp. 52–70, Sep. 2005.

[3] D. T. Ahmed, S. Shirmohammadi, and J. C. de Oliveira, "A hybrid P2P communications architecture for zonal MMOGs," *Multimedia Tools Appl.*, vol. 45, nos. 1–3, pp. 313–345, Oct. 2009.

[4] M. Ghaffari, B. Hariri, and S. Shirmohammadi, "On the necessity of using Delaunay triangulation substrate in greedy routing based networks," *IEEE Commun. Lett.*, vol. 14, no. 3, pp. 266–268, Mar. 2010.

[5] B. Hariri, M. R. Pakravan, S. Shirmohammadi, and M. H. Alavi, "Using geometrical routing for overlay networking in MMOGs," *Multimedia Tools Appl.*, vol. 45, no. 1–3, pp. 61–81, 2009.

[6] M. Ghaffari, B. Hariri, and S. Shirmohammadi, "A Delaunay triangulation architecture supporting churn and user mobility in MMVEs," in *Proc. ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video (NOSSDAV)*, Williamsburg, VA, USA, Jun. 2009, pp. 61–67.

[7] E. Buyukkaya, M. Abdallah, and G. Simon, "A survey of peer-to-peer overlay approaches for networked virtual environments," in *Peer-to-Peer Networking and Applications*. New York, NY, USA: Springer-Verlag, Sep. 2013.

[8] A. Yahyavi and B. Kemme, "Peer-to-peer architectures for massively multiplayer online games: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, p. 9, Oct. 2013.

[9] E. Léty, T. Turletti, and F. Baccelli, "SCORE: A scalable communication protocol for large-scale virtual environments," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 247–260, Apr. 2004.

[10] M.-J. Tsai, H.-Y. Yang, B.-H. Liu, and W.-Q. Huang, "Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1228–1241, Aug. 2009.

[11] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *Proc. 11th Can. Conf. Comput. Geometry*, Aug. 1999, pp. 51–54.

[12] F. Kuhn, R. Wattenhofer, and A. Zollinger, "An algorithmic approach to geographic routing in ad hoc and sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 51–62, Feb. 2008.

[13] P. Bose and P. Morin, "Online routing in triangulations," in *Proc. Int. Symp. Algorithms Comput.*, Chennai, India, Dec. 1999, pp. 113–122.

[14] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with delaunay triangulation overlays," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1472–1488, Oct. 2002.

[15] E. Buyukkaya and M. Abdallah, "A flexible connectivity architecture for avatar management in P2P virtual environments," in *Proc. 10th ACM/IEEE Int. Workshop Netw. Syst. Support Games*, Ottawa, ON, Canada, Oct. 2011, pp. 1–2.

[16] E. Buyukkaya and M. Abdallah, "Routing over relaxed triangulation structures for P2P-based virtual environments," in *Proc. Int. Conf. Adv. Inform. Netw. Appl. Workshops*, Barcelona, Spain, Mar. 2013, pp. 1167–1173.

[17] M. Steiner and E. Biersack, "A fully distributed peer to peer structure based on 3D Delaunay triangulation," in *Proc. Rencontres Francophones Aspects Algorithmiques Télécommun.*, May 2005, Presqu'île de Giens, France, pp. 93–96.

[18] D.-Y. Lee and S. S. Lam, "Protocol design for dynamic Delaunay triangulation," in *Proc. Int. Conf. Distrib. Comput. Syst.*, Toronto, ON, Jun. 2007, Canada, p. 26.

[19] D.-Y. Lee and S. S. Lam, "Efficient and accurate protocols for distributed Delaunay triangulation under churn," in *Proc. IEEE Int. Conf. Netw. Protocols*, Orlando, FL, USA, Oct. 2008, pp. 124–136.

[20] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "VON: A scalable peer-to-peer network for virtual environments," *IEEE Netw.*, vol. 20, no. 4, pp. 22–31, Jul./Aug. 2006.

[21] N. E. Baughman, M. Liberatore, and B. N. Levine, "Cheat-proof playout for centralized and peer-to-peer gaming," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 1–13, Feb. 2007.

[22] S. Schuster, "Cheating prevention in peer-to-peer-based massively multiuser virtual environments," Ph.D. dissertation, Faculty Eng. Sci., Comput. Sci. Appl. Cognit. Sci., Univ. Duisburg-Essen, Essen, Germany, 2013.

[23] A. Tumbde and S. Venugopalan, "A voronoi partitioning approach to support massively multiplayer online games," Dept. Comput. Sci., Univ. Wisconsin, Madison, WI, USA, Tech. Rep. CS 740, 2004.

[24] C.-H. Chien, S.-Y. Hu, and J.-R. Jiang, "Delaunay state management for large-scale networked virtual environments," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst.*, Melbourne, Australia, Dec. 2008, pp. 781–786.

[25] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, "Voronoi state management for peer-to-peer massively multiplayer online games," in *Proc. IEEE Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, Jan. 2008, pp. 1134–1138.

[26] A. P. Yu and S. T. Vuong, "MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," in *Proc. ACM Workshop Netw. Oper. Syst. Support Dig. Audio Video (NOSSDAV)*, Washington, DC, USA, Jun. 2005, pp. 99–104.

[27] S. Ratti, B. Hariri, and S. Shirmohammadi, "A survey of first-person shooter gaming traffic on the internet," *IEEE Internet Comput.*, vol. 14, no. 5, pp. 60–69, Sep./Oct. 2010.

[28] I. Stojmenovic, A. P. Ruhil, and D. K. Lobiyal, "Voronoi diagram and convex hull based geocasting and routing in wireless networks," in *Proc. 8th IEEE Int. Symp. Comput. Commun.*, Jun. 2003, pp. 51–56.

[29] S. Fortune, "A sweepline algorithm for voronoi diagrams," in *Proc. 2nd Annu. Symp. Comput. Geometry*, Yorktown Heights, NY, USA, Jun. 1986, pp. 313–322.
[30] E. Buyukkaya and M. Abdallah, "Data management in voronoi-based P2P gaming," in *Proc. 5th IEEE Int. Workshop Dig. Entertainment, Netw. Virtual Environ., Creative Technol.*, Las Vegas, NV, USA, Jan. 2008, pp. 1050–1053.
[31] E. Buyukkaya and M. Abdallah, "A twofold approach to object and avatar data management in P2P-based virtual environments," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, San Jose, CA, USA, Jul. 2013, pp. 1–7.
[32] J. Gilmore, "State management and persistency architecture for peer-to-peer massively multi-user virtual environments," Ph.D. dissertation, Dept. Electr. Electron. Eng., Stellenbosch Univ., Stellenbosch, South Africa, 2013.
[33] L. Ricci, L. Genovali, E. Carlini, and M. Coppola, "AOI-cast in distributed virtual environments: An approach based on delay tolerant reverse compass routing," in *Concurrency and Computation: Practice and Experience*, New York, NY, USA: Wiley, Dec. 2012.
[34] M. Steiner and E. W. Biersack, "Shortcuts in a virtual world," in *Proc. ACM CoNEXT*, Lisbon, Portugal, Dec. 2006, p. 17.
[35] S.-H. Min, J. Holliday, and D.-S. Cho, "Optimal super-peer selection for large-scale P2P system," in *Proc. Int. Conf. Hybrid Inform. Technol.*, Cheju Island, Korea, Nov. 2006, pp. 588–593.
[36] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey, and J. Li, "Scalable supernode selection in peer-to-peer overlay networks," in *Proc. Int. Workshop Hot Topics Peer-to-Peer Syst.*, Jul. 2005, pp. 18–25.
[37] M. H. Alsuwaiyel, "A parallel algorithm for partitioning a point set to minimize the maximum of diameters," *J. Parallel Distrib. Comput.*, vol. 61, no. 5, pp. 662–666, 2001.
[38] B. Hariri, S. Shirmohammadi, and M. Pakravan, "A distributed interest management scheme for massively multi-user virtual environments," in *Proc. IEEE Conf. Virtual Environ., Human-Comput. Inter., Meas. Syst.*, Istanbul, Turkey, Jul. 2008, pp. 111–115.

MOHSEN GHAFFARI is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He received the M.S. degree from the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, in 2013, and the dual B.Sc. degree in electrical engineering (communications) and computer science from the Sharif University of Technology, Tehran, Iran, in 2010. His current research interest revolves around theoretical problems in networking and distributed computing. During his undergraduate program, he was involved in the research subject of this paper with the Distributed and Collaborative Virtual Environment Research (DISCOVER) Laboratory, University of Ottawa, Ottawa, ON, 1276 Canada. He was a recipient of the 2014 Simons Award for Graduate Students in Theoretical Computer Science and the prestigious MIT Presidential Fellowship.

BEHNOOSH HARIRI received the Ph.D. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2009, and was a MITACS Elevate Post-Doctoral Fellow with the DISCOVER Laboratory, University of Ottawa, Ottawa, ON, Canada, from 2009 to 2011. Her research was focused on gaming systems and networks, in particular, massively multiuser virtual environments, and vision-based multimedia systems. She is currently with Google Inc., New York, NY, USA.

SHERVIN SHIRMOHAMMADI (M'04–SM'04) received the Ph.D. degree in electrical engineering from the University of Ottawa, Ottawa, ON, Canada, where he is currently a Full Professor with the School of Electrical Engineering and Computer Science. He is the Co-Director of the DISCOVER Laboratory and the Multimedia Communications Research Laboratory, University of Ottawa, conducting research in multimedia systems and networking, in particular, gaming systems and virtual environments, video systems, and multimedia-assisted biomedical engineering. The results of his research have led to more than 250 publications, over 20 patents and technology transfers to the private sector, and a number of awards and prizes. He is an Associate Editor-in-Chief of the IEEE Instrumentation and Measurement Magazine, a Senior Associate Editor of *ACM Transactions on Multimedia Computing, Communications, and Applications*, an Associate Editor of the IEEE Transactions on Instrumentation and Measurement, and was an Associate Editor of *Journal of Multimedia Tools and Applications* (Springer) from 2004 to 2012, and chairs or serves on the program committee of a number of conferences in multimedia, virtual environments, and games. He is a University of Ottawa Gold Medalist, a licensed Professional Engineer in Ontario, and a Professional Member of the Association for Computing Machinery.

DEWAN TANVIR AHMED received the Ph.D. degree in computer science from the University of Ottawa, Ottawa, ON, Canada, in 2009, where he was a Post-Doctoral Fellow with the DISCOVER Laboratory from 2009 to 2011. He is currently an Assistant Professor with the Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC, USA, where he is conducting research in distributed systems and algorithms, simulation and modeling, multimedia communication, and computer networks, with applications in real-time systems, surveillance systems, and serious games. He was a recipient of the IEEE ICME Quality Reviewer Award in 2011, the Industrial Research and Development Fellowship from the Natural Sciences and Engineering Research Council of Canada in 2010, the Best Student Researcher Award from the Ottawa Centre for Research and Innovation in 2009, the Best Poster Award in Computer Science from University of Ottawa's Faculty of Engineering in 2008, and the Best Paper Award from the IEEE Collaborative Peer-to-Peer Information Systems Workshop in 2007.