A Delaunay Triangulation Architecture Supporting Churn and User Mobility in MMVEs

Mohsen Ghaffari[1], Behnoosh Hariri[1,2], Shervin Shirmohammadi[1,2]
[1]Advanced Communications Research Institute(ACRI), Sharif University of Technology, Tehran, Iran.
[2]Distributed Collaborative Virtual Environment Research Lab., University of Ottawa, Ottawa, Canada.
{Ghaffari | Hariri}@ee.sharif.edu, {BHariri | Shervin}@discover.uottawa.ca

ABSTRACT

This article proposes a new distributed architecture for update message exchange in massively multi-user virtual environments (MMVE). MMVE applications require delivery of updates among various locations in the virtual environment. The proposed architecture here exploits the location addressing of geometrical routing in order to alleviate the need for IP-specific queries. However, the use of geometrical routing requires careful choice of overlay to achieve high performance in terms of minimizing the delay. At the same time, the MMVE is dynamic, in sense that users are constantly moving in the 3D virtual space. As such, our architecture uses a distributed topology control scheme that aims at maintaining the requires QoS to best support the greedy geometrical routing, despite user mobility or churn. We will further prove the functionality and performance of the proposed scheme through both theory and simulations.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed Networks

General Terms

Algorithms, Design

Keywords

Voronoi Diagram, Delaunay Triangulation, Red-Black, Peer to Peer, MMVE, Churn, Mobility.

1. INTRODUCTION

Massively multi-user virtual environments (MMVE) are virtual worlds where a large number of users distributed all over the Internet can interact with each other in real time. Among many problems, MMVEs must deal with the real-time update message exchange among this large group of users to provide them with a common sense of time and place, when interacting in the virtual environment. Therefore, MMVEs must support message distribution over the Internet while considering QoS issues arising mainly from strict end-to-end delay thresholds in such applications. The large number of users highly motivates the use of distributed architectures where there is no central switching point for the updates. In fact, many distributed and Peer to Peer algorithms have already been suggested for MMVEs. One of these methods is Geometric Routing, which can send an update message to a specific coordinate in space, as opposed to sending it to a specific IP address. This can be quite suitable for MMVEs since these applications mainly require a node to send updates to a specific area in the environment known as that node's area of effect. The use of IP routing in this scenario whould require a node to make a location query to know who is located in its *area of effect*, so that it can send them its update messages. Conceptually, it would make more sense to avoid the need for location services and instead, directly send the message to the area of effect. In other words, geometrical routing provides the nodes with a routing service where updates can be sent to locations instead of IP addresses. Therefore, the nodes won't need to know about entities in their area of effect, before sending updates. However, the use of geometrical routing requires the definition of an overlay that is considered as a sub-graph of the Internet underlay network, and the efficiency and success rate of geometrical routing is closely related to the structure of this overlay network. Therefore, the proper choice and maintenance of the overlay topology will be the main focus of this article. To add yet another challenge, the topology control must be a highly dynamic routine as users frequently join, leave or change their locations in the network. We aim at using distributed routines where the whole network should participate in the topology update without having any central decision point.

As mentioned before, one of the important requirements of MMVE applications is the guaranteed delivery of update messages whithin acceptable delay thresholds, i.e, through the shortest possible routes. The use of greedy scheme over geometrical architecture is a good online routing choice regarding simplicity and convergence rate. However, greedy routing does not provide guaranteed message delivery over generic overlays. We will in fact prove that greedy routing is only supported over a specific family of overly graphs whose properties will be discussed in detail in the next section. Therefore, our approach to topology control is the use of a distributed scheme to maintain a greedy- supporting overlay in the presence of churn and mobility in the network. Since network dynamics constantly change the topology graph characteristics, the graph should be constantly checked and updated for greedy support feature. However, the distributed architecture of the network leaves us with no fixed landmarks that can be used as resumption references for the topology. Our proposed solution to this problem is to partition the nodes into two groups referred to as *red* and *black* groups. Each of these groups composes a graph that can be updated using the other graph

NOSSDAV'09, June 3–5, 2009, Williamsburg, Virginia, USA.

as a reference. We will also show how the overall topology graph will be constructed based on the two sub-graphs. Section 3 of this paper explains the general problem model and required preliminaries for the proposed solution. Section 4 discusses the general concept of the proposed architecture while section 5 presents the theory and further details of the architecture. Section 6 addresses the consistency of the proposed architecture and we will provide the simulation results in section 7. Final section will be summary and conclusion, but let's look at the related works, first.

2. RELATED WORKS

P. Bose and P. Morin [1] demonstrated that Delaunay Triangulation (DT) is one of the structures that supports the greedy routing, i.e, greedy routing on DT provides guaranteed message delivery. Liebeherr and Nahas [2] proposed the first protocol to build a distributed DT . The protocol exploits the locally equiangular property of DT in the 2D space. It is assumed that every node has a preassigned logical coordinate in the plane. Each node 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.

Steiner and Biersack [3] suggested another distributed approach to construct a distributed DT, but in a 3D space. The tetrahedron which includes a joining node is determined and is split. Then the new tetrahedra are checked whether they include any nodes in their circumspheres and flipped if necessary.

Lee et al. [4] defined a distributed DT to be correct if and only if it has a specific condition and using this condition, presented two join and leave protocols as well as correctness proofs for serial joins and leaves. In these protocols, it has been assumed that just one node joins or leaves, every time. Nonetheless, for handling concurrent joins and leaves, as well as node failures, they presented a maintenance protocol and argued that, in all of their experiments, this protocol converged to a correct distributed DT some time after churn and failure have stopped. The same authors suggested some newer protocols for join, leave and maintenance in [5]. However, despite the fast and frequent motion of the nodes in MMVE, these works do not provide any protocol for maintaining the structure against motion of the nodes.

Hu et al. [6], similarly proposed a set of procedures including join, leave and move procedures for retaining the distributed DT. The join and the leave procedures are similar to [4]. Nonetheless, in this work, they have investigated motions with a separate procedure. For this purpose, they have introduced new definitions, boundary neighbor and enclosing neighbors. With movement of a node, position updates are sent to all currently connected neighbors. Then, the boundary neighbors will help the moving node to find new probable visible nodes and connect to them. With the motion of one node, previous links to the boundary nodes would be checked and deleted if boundary nodes are outside of moving nodes Area of Interest (AOI). In this procedure, it is assumed that all the other relevant nodes do not change their locations and they have reliable links between themselves. However, as we have discussed in previous section, the major difficulty of handling dynamic structures in the MMVEs is due to the frequent motion of potentially all nodes at the same time. As such, the assumption that one node moves and the rest are stationary is not practical.

In this paper, we have an approach comparable with [4]-[6]. However, unlike these works, we have not separated mentioned procedures. We suggest a new architecture that dynamically modifies itself to support desired geometrical routings taking into account high churn characteristics and fast and frequent motion of the nodes in MMVEs.

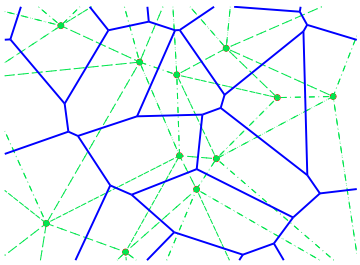

Figure 1: Voronoi Diagram and Delaunay Triangulation.

3. PROBLEM MODEL AND THEORETICAL FOUNDATION

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

3.1 Defintions

3.1.1 Voronoi Cell

Suppose a set of points V, in the Euclidian plane where each point in this set represents position of one 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 Euclidian plane.

3.1.2 Delaunay Triangulation

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

3.1.3 Vertex Region

Consider a graph $G = (V, E)$ on the set of points V, where for each $v_i \in V$, $N(v_i)$ stands for the neighbor set of v_i. For each $v_i \in V$, we define the 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 $N(v_i)$.

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

3.2 Problem Definition

As mentioned before in the section 1, MMVE applications mainly deal with the multicast of update messages to certain areas in the virtual world, i.e, a peer sends state change updates of itself to others in its area of effect. It is almost impossible for the MMVE nodes to gather information of all possible locations and available links in every node due the highly dynamic characteristics of these environments. Therefore, the use of an online routing method is highly motivated where topology data centralization is not required [1]. Greedy routing is among the simplest and the most common online routing algorithms that can be used in geometrical routing context.

For a graph $G = (V, E)$ on the node set V, greedy routing tries to select the closest available node to the destination as the next hop for the packets. Graph $G = (V, E)$ supports greedy routing *iff* greedy routing on G delivers any 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.
In the following sub-section, we will investigate the problem of greedy routing support over a graph, and present our approach to the dynamic update of network graph with respect to the location of the users in order to maintain the support of greedy routing.

3.3 Theoretical Foundation of the Proposed Approach

THEOREM 1. *Graph $G = (V, E)$ on the nodes V in the Euclidian plane supports greedy routing iff for every node $v_i \in V$, $VR_G(v_i) = VC(v_i)$.*

PROOF. We first show the sufficiency condition and then, we go to the necessity condition.
($\Leftarrow$) Suppose that in the graph $G = (V, E)$, we have $VR_G(v_i) = VC(v_i)$ for each node $v_i \in V$. It can be proven by contradiction that G supports greedy routing. If G does not supports greedy routing, there is some point $v_i \in V$ that the packet will get stuck in v_i, and v_i is not the nearest point in V to the packet's destination. From the assumptions, it can be concluded that the packet's destination is located outside of the $VC(v_i)$ as v_i is not the nearest point in V to the packet's destination. Since $VR_G(v_i) = VC(v_i)$, the destination is also located outside of $VR_G(v_i)$. Therefore, there should be a point $v_j \in N(v_i)$ that is closer to the destination than v_i and v_i can pass the packet to v_j after v_i.
($\Rightarrow$) For the proof of necessity condition, assume that G supports greedy routing. We demonstrate that $VR_G(v_i) = VC(v_i)$ for every $v_i \in V$. It should be noted that according to *Remark 1*, for each $v_i \in V$ $VR_G(v_i) \subset VC(v_i)$. Therefore, it suffices to show that $VC(v_i) \subset VR(v_i)$ as well. This can be proven by contradiction. Suppose that there is a point $x \in VR_G(v_i)$ that does not belong to $VC(v_i)$. Since $x \notin VC(v_i)$ and Voronoi Cells partition the plane, x is in the Voronoi Cell of another node, e.g., v_j. Therefore, v_i is not the closest member of V to x. But $x \in VR_G(v_i)$, and hence, v_i has no closer neighbor to x than itself. Thus, if a packet destined to x reaches v_i or starts in v_i, it can not be delivered to v_j. Therefore G does not support greedy routing. □

THEOREM 2. *Graph $G = (V, E)$ over the set V in the Euclidian plane supports greedy routing iff for all $v_i, v_j \in V$, $VR_G(v_i) \cap VR_G(v_j) = \phi$.*

PROOF. ($\Rightarrow$) If G supports greedy routing, it can be concluded from using theorem 1 that $VR_G(v_i) = VC(v_i)$ for every $v_i \in V$. Moreover, as Voronoi Cells of V partition the plane, $VR_G(v_i)$s are separate sets.
($\Leftarrow$) Suppose that for every $v_i, v_j \in V$, $VR_G(v_i) \cap VR_G(v_j) = \phi$, we show that $VR_G(v_i) = VC(v_i)$ and thus, G supports greedy routing. Since VR_G s are separate sets, every point in the plane can only belong to one of the VR_G. On the other hand, each point in the plane is exactly in Voronoi Cell of one node and according of Remark 1., this point should be located in the VR_G of that node. Therefore, each point in the plane is exactly in one VR_G, i.e., VR_Gs partition the plane. Therefore both VR_Gs and VCs partition the plane. Moreover, according to Remark 1., each VC is a subset of its relevant VR_G, therefore for every $v_i \in V$, $VR(v_i) = VC(v_i)$. □

Theorem 2 provides an interesting insight to the gradual update of random graphs towards the structures that support greedy schemes. Using theorem 2, we can modify any arbitrary graph to support greedy routing. It should be noted that, any random graph can be transformed to the complete graph on the same set of nodes in order to support greedy routing. Therefore, the transformation would be useful if there is a constraint on the nodes degrees during the transformation. Therefore, in order to transform a graph to a graph that supports greedy routing it suffices to connect each pair of nodes that their VR_G's have a nonempty intersection and delete the redundant edges afterwards. In the following sections, we will discuss this solution in detail and discuss its correctness.

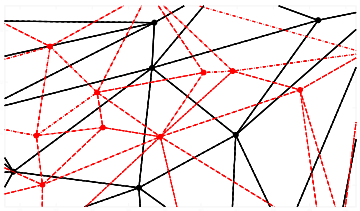

Figure 2: The Red-Black Structure.

4. RED-BLACK ARCHITECTURE

In the previous sections, we have provided the background on our proposed dynamic structure. This section includes the outline of the solution that has been defined based on the previous foundations. We mentioned that nodes are usually in movement and they use location update messages for announcing their new locations to the other nodes. Since nodes can not proclaim their locations ceaselessly, a general method is to synchronize the nodes 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.
We also mentioned that the routing of update messages to specified locations is best to be done over a geometrical routing architecture that alleviates the use of location services. However, in order to assure the guaranteed delivery of updates while using greedy routing, the overlay topology should support greedy routing. The challenge would then turn in to the dynamic construction of an overlay topology that supports greedy routing. We would therefore propose a dynamic topology control scheme that can handle the high churn and nodes mobility in MMVE applications.
The challenge for the fully distributed topology update is the lack of any fixed points that can be used as landmarks to help the nodes in topology reconstruction after any movement. In order to deal with this problem, we have used the dual phase topology update approach where the nodes are assigned to two non-overlapping sets called *Red* and *Black* and each is updated in one of the update phases. Therefore each set is able to use the other set as its landmark in order to update itself. Therefore, we color our nodes with two colors *Red* and *Black*. Then, red nodes generate their location update messages and send them, with a $T/2$ time shift, compared to the updating time of black nodes. Our structure is composed of a general graph on all of the nodes that would support the greedy routing and will be used for any kind of message transmission, except for positional update and topology control messages. However, we have two red and black graphs in the background. These two graphs are updated in a way that supports greedy routing. An Example of these simultaneous red and black graphs is represented in Fig. 2.
Assume that red nodes send their location updates in times $0, T, 2T, ...$ and black nodes send their updates in times $T/2, 3T/2, 5T/2, ...$ At the location update time of red nodes, the red graph is probably changed due to the movement of red nodes and it might not support greedy routing anymore. Moreover, our general graph might also fail to support greedy routing. However, the good news is that the black structure has remained untouched and still supports

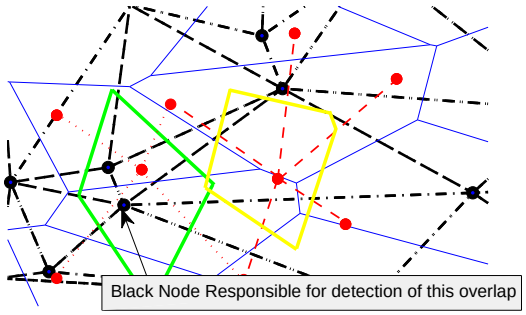

Figure 3: The distributed method of checking intersections.

the greedy routing. It should be noted that the black nodes might have also moved from their previous locations during this $T/2$ interval; however the change in their positions has not yet been announced. Considering the positions in the Virtual Environments as virtual concepts, the position change of the black nodes can be kept for their own update phase and ignored upon the update of red graph. The same method will be used during the update time of black nodes, except that this time the red structure is used as the reliable structure.

These two sets can be selected arbitrarily. However, a more uniform distribution of these two sets in the plane would help the convergence rate of the proposed method. Therefore, a simple and practical method would be to divide nodes by evenness of their IP-addresses. In the following section we will discuss the details of the structure and the algorithm exploited for it and theoretically prove its functionality.

5. RED-BLACK TOPOLOGY UPDATE

After describing the outline of our dynamic structure in the previous section, we will further discuss the details of the update method and provide the proof on how the update of connections graph can result in support for greedy routing. In the rest of the section, without loss of generality, we will explain the algorithm for the update time of the red nodes, knowing that the update procedure for black nodes is similar.

5.1 Updating Red Structure Using the Black Graph

At the start of this update time, 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. In the next step of the algorithm, the vertex regions are checked against their probable overlaps and the pairs of red nodes with overlapping vertex regions are nominated to be new neighbors. It should be noted that there is no single point where all of information required for checking the vertex regions intersections is located. Thus, we will exploit a distributed method, using nodes of black structure in order to check the intersections between the vertex regions. The specifications 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_i^b finds a common point such as x between the two red VR_R's where $x \in VR_B(v_i^b)$. v_i^b will then notify one of the involved red nodes to make a link with the other one. An example of this scenario is resented in Fig. 3.

We will later refer to the desired set of black nodes for every red node v_i^r, the Black Parents (BP) of v_i^r, where $BP(v_i^r) = \{v_j^b | VR_B(v_j^b) \cap VR_R(v_i^r) \neq \phi\}$. The problem is how to deliver the $VR_R(v_i^r)$ specifications to the members of $BP(v_i^r)$. This can be achieved 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.

5.1.1 *Message Delivery to a member of Black Parent Set*

Stojmenovic et al. have investigated the solution to a similar problem as the first step [7]. Their problem was how to deliver a message to a point whose location is not precisely specified; instead it only needs to be located inside a specific polygon.

Consider the first phase of our problem that is the delivery of $VR(v_i^r)$ 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, each red node reminds the nearest black node of itself (named its black owner). During the red update period, each red node encapsulates specifications of its VR_R in a message and sends this message to its previous black owner and asks the black owner to return the message to itself using the black graph. As the black structure supports greedy routing, the packet will get to the red node's new black owner that might probably be the same as the old black owner.

5.1.2 *broadcasting the message among members of Black Parent Set*

When the message reaches the desired parent set, it should be broadcasted to the other members of the $BP(v_i^r)$. 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. *With 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 "only if" condition is obvious referring to the algorithm description. The "if" part can be proven by contradiction. Suppose that there exists a black node $v_j^b \in BP(v_i^r)$ that does not receive the message. 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 $VR_B(v_j^b) \cap VR_R(v_i^r)$ to a point $VR_B(v_k^b) \cap VR_R(v_i^r)$, all of points of this line should belong to $VR_R(v_i^r)$. If S has at least one side in common with VR_B's of the nodes that have received the message, that causes a conflict with the definition of the method. Otherwise, we can add the next black node (from the side of v_j^b) that the line passes through its black VR_B to the set S. This also conflicts with the definition of S as the biggest possible set. □

5.1.3 *Topology refinement*

We have previously discussed how to deliver the specification of $VR_R(v_i^r)$'s to the members of $BP(v_i^r)$ to check the probable nonempty intersections between VR_R's. We have also mentioned that each black node is responsible for finding the intersections in its VR_B and if it finds an overlapping between VR_R's of two nodes, it will send a message to one of these pair and asks it to consider the other red node as a candidate for the neighborhood. Then, that node runs a specific algorithm and finds out what new red nodes are really needed for neighborhood and sends a "Hello Neighbor" message to all new neighbors.

Upon the reception of a new neighborhood candidate message at a red node v_i^r, it draws the orthogonal bisector of line passing through itself 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 candidate's neighborhood request is granted. $VR_R(v_i^r)$ will then change where new $VR_(v_i^r)$ would be a subset of previous $VR_R(v_i^r)$. Therefore, some neighbors might be considered as redundant in the new $VR_R(v_i^r)$. The redundant neighbors are defined by the following criteria: the orthogonal bisectors of edges between the main node and the redundant neighbors would be located fully outside the $VR_R(v_i^r)$ of main node. Therefore, deleting the edges between the main node and the redundant neighbors will not affect the $VR_R(v_i^r)$ of the main node. The good point is that it is not necessary to inform the other end of an edge when the edge is been deleted. This will be explained in the following paragraph.

For providing the proof to the above 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 redundant from the viewpoint of both of its ends.

The proposed algorithm needs to check the old members of the neighbor set for probable redundancy upon the joining of each new neighbor. Therefore, the algorithm complexity is of $O(n^2)$. As the degree of the nodes in a structured graph is usually small, the algorithm won't require a considerable amount of processing power. However, if number of neighbors increases beyond a certain threshold, it will be probably more rational to take advantage of Fortune's algorithm [8], designed for finding the Voronoi Diagram of 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.

5.2 Topology graph Construction Based on Red and Black Sub-graphs

We have 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.

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

5.3 Further Improvements of the Algorithm

We have previously proposed a method to check the intersections of nodes in both red and overall graphs and adapt these graphs (connect new links with probable deletion of some of old links) in order to support greedy routing. We explained that a message containing the specification regarding the 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: Upon the reception of a $VR_R(v_i^r)$ in a black node v_j^b, v_j^b checks for the red VR_R-s overlaps 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 $VR_R(v_i^r)$ and then uses the same broadcast scheme using this new $VR_R(v_i^r)$. The update in the $VR_R(v_i^r)$ is based on the assumption that all new red candidates (found in v_j^b), will finally become neighbors of v_i^r.

5.3.1 Functionality Proof

Earlier in this section, we have proposed a method for construction of a greedy supporting red graph. According to theorem 1, $VR_R(v_i^r) = VC(v_i^r)$ for every v_i^r in the resulting structure. Using the new method, in each step the new $VR_R(v_i^r)$ will be a subset of the previous $VR_R(v_i^r)$ that still contains $VC(v_i^r)$ that is supposed to be the final $VR_R(v_i^r)$. Therefore, the new method checks for any probable nonempty overlap of regions that are more similar to final VR_R's. At the final stage of the algorithm, VR_R's of the constructed red graph will be separate sets. Therefore, the red graph will support greedy routing according to theorem 2.

6. CONSISTENCY ISSUES

In this section we will discuss the consistency issues of the proposed algorithm. In order to provide a criterion for consistency measurement, we introduce a new term "location age". At each arbitrary time instance, "location information age" of a node is defined as the average time that passed since that node has received the last location update message from every other node. According to this definition, an updating method is considered to be more consistent *iff* it achieves a lower value of location age. It is sufficient to evaluate this parameter only for a period of T seconds for both of the previously mentioned methods as it will be periodically repeated in T intervals. For the simple update method used in general, the age location parameter is $\int_0^T (N-1)t \cdot dt = (N-1)T^2/2$. As for the red-black architecture, the location information age can be found according to the following relations. This parameter for the red nodes would be

$$\int_0^{T/2} ((m-1)t + (N-m)(t+T/2)) \cdot dt + \int_{T/2}^{T} ((m-1)t + (N-m)(t-T/2)) \cdot dt = (N-1)T^2/2 \quad (1)$$

and for the black nodes, it would be

$$\int_0^{T/2} ((m)(t+T/2) + (N-m-1)t) \cdot dt + \int_{T/2}^{T} ((m)(t-T/2) + (N-m-1)t) \cdot dt = (N-1)T^2/2 \quad (2)$$

Therefore, the expected age of location information for this method will still be $(N-1)T^2/2$ that is the same as the simple update procedure and hence, our change in the method of sending location update messages does not affect the consistency of our architecture.

7. SIMULATION RESULTS

In this section, we present experimental results obtained from simulations of the proposed architecture. Simulations have been executed for various number of nodes spread randomly in a specific virtual area of $[50 \times 50]$ meters in the virtual environment. In the course of this article, we have argued and proved that after completion of the resumption algorithm, the resumed structure will be

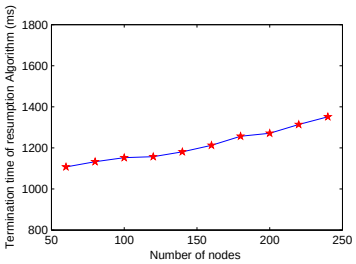

Figure 4: **Termination time of resumption algorithm versus number of nodes.**

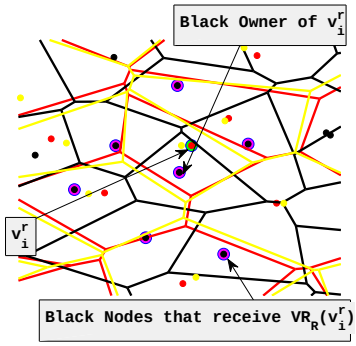

Figure 5: **Black nodes that receive $VR_R(v_i^r)$ specification message. The yellow points are previous position of red nodes, and red and black points represent red and black nodes, respectively.**

absolutely accurate and the connection graph will support greedy routing. But, the question is that how fast this method resumes the structure. To investigate this problem, RTT times are selected to be uniform in [100, 200] ms and we assumed that in each period, every node moves at most one meter in an arbitrary direction. We have plotted the average run time of our algorithm versus the number of nodes in the virtual environment in Fig. 4.

In this figure, it can be seen that delightfully, the number of nodes in the area has a slight effect on the termination time of the proposed method. This problem could be described reasonably.

In section 5, it was explained that in the proposed resumption method, every red node sends specifications of its VR_R to its black owner and then, this message will be 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 the other neighbor black parents. Therefore, with finding nonempty overlaps 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 the $VR_R(v_i^r)$ specification (Fig. 5). 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 $VR_R(v_i^r) = VC(v_i^r)$. If red and black nodes are uniformly distributed in the Euclidian plane, $VC(v_i^r)$ will overlap with around 4 to 8 black VR_B-s and thus, the $VR_R(v_i^r)$ specification will be delivered to nearly 4 to 8 black nodes. It can be argued that average degree of vertexes in a Delaunay triangulation graph is usually in the range of 3 to 7, and with increasing the number of nodes in a specific area, this parameter does not change. However, with this change, average distance of neighbor nodes, i.e, length of edges will decrease. But, length of edges in graphs of connections in VE-s is just a virtual concept and is not relevant to real time of message transfer between two ends of that edge.

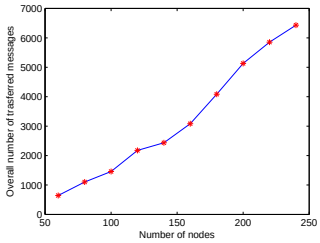

Figure 6: **Overall number of transferred messages versus number of nodes.**

We have also depicted the overall number of messages sent for the resumption algorithm versus number of nodes in Fig. 6. As it can be seen from this figure, number of messages sent in the resumption algorithm has an approximately linear relation with the number of nodes and thus. Average number of messages transmitted by every node is almost constant. This problem is similar to the previous problem about the methods termination time.

Hence, it can be concluded that the proposed method is nicely distributed and increasing the number of nodes in the plane does not affect its complexity and therefore, it could be an alternative substitute for the centralized client-server architectures.

8. SUMMARY AND CONCLUSION

An important issue in MMVEs is user mobility and churn, which becomes a challenging issue if a P2P overlay is used for networking support. In this paper we have proposed a distributed geometrical architecture for MMVE applications. The use of a geometrical routing scheme requires the constant update of the overlay topology as an important factor on the routing performance and success rate. The overlay topology to supports greedy routing as a simple and efficient online routing scheme in the geometrical routing context because we use a distributed algorithm for maintaining a greedy support graph among MMVE nodes. The basic idea behind our approach is the partitioning of the topology in two different parts where each part is updated in a separate phase using the other part as a reference. Simulation results are promising and demonstrate the functionality and performance of the proposed algorithm.

9. REFERENCES

[1] P. Bose and P. Morin, “Online routing in triangulations", *Proc. of Int. Symp. on Algorithms and Comp.*, 1999, pp. 113 - 122.

[2] J. Liebeherr and M. Nahas, “Application-Layer Multicasting With Delaunay triangulation Overlays", *IEEE Journal on Sel. Areas. in Comm.*, vol. 20, no. 8, October 2002.

[3] M. Steiner and E. Biersack, “A fully distributed peer to peer structure based on 3D Delaunay Triangulation",*in Proc. Algotel*, 2005, pp. 93 - 96.

[4] Dong-Young Lee and Simon. S. Lam, “Protocol Design for Dynamic Delaunay Triangulation", *Inter. Conf. on Distributed Systems*, 2007.

[5] Dong-Young Lee and Simon. S. Lam, “Efficient and Accurate Protocols for Distributed Delaunay Triangulation under Churn", *in Int. Conf. on Net. Prot., ICNP*, 2008.

[6] Shun-Yun Hu and Jui-Fa Chen and Tsu-Han Chen, “VON: a scalable peer-to-peer network for virtual environments", Network, IEEE , vol.20, no.4, pp.22-31, July-Aug. 2006.

[7] I. Stojmenovic and A. P. Ruhil and D. K. Lobiyal,“Voronoi diag- ram and convex hull based geocasting and routing in wireless networks", *in Proc.Wirel. Commun. Mob. Comput*, Sep.2006 ,vol.6, pp.247-258.

[8] S. Fortune, “A sweepline algorithm for Voronoi diagrams", *in Proc. Symp. on Comp. Geometry*, 1986, pp.313-322.