A Radius Geocast Routing Protocol *

Dong-Young Lee, Eui Kyung Chung, and Simon S. Lam
{dylee, ekchung, lam}@cs.utexas.edu
Department of Computer Sciences
The University of Texas at Austin

Abstract

We present a protocol, named RadGRPM, which runs on a distributed Delaunay triangulation of a set of nodes in Euclidean space. Given coordinates of the source node and a radius, RadGRPM multicasts a message to all nodes within the given radius from the source. Since the target nodes are all within a spherical region centered at the source, RadGRPM provides a special kind of geocast, which we call radius geocast*. A multicast tree is not explicitly maintained in RadGRPM. Each node determines the next-hop nodes to forward a message solely using local information (the coordinates of its neighbors) together with the radius and coordinates of the center carried in the message. We prove that RadGRPM delivers a message to all nodes within the given radius. RadGRPM is also efficient in the sense that very few nodes within the radius receive duplicate messages, and nodes outside the radius receive no message. Extensive experimental results are presented to investigate the performance and characteristics of RadGRPM. Furthermore, we show that RadGRPM can be combined with unicast greedy routing to provide geocast to any spherical region not centered at the source node.*

Keywords: broadcast, geocast, multicast, distributed virtual environment, Delaunay triangulation, Voronoi diagram, peer-to-peer network, wireless network.

1. Introduction

In a distributed virtual environment, entities interact with other entities in a shared virtual space. A popular example of a distributed virtual environment is *Second Life*. Multiplayer on-line games such as *World of Warcraft* and *Lineage* are also examples of distributed virtual environments.

Supporting a large number of concurrent users in a distributed virtual environment is a great technical challenge, especially in the current server-based architecture. Note that entities (or avatars of players in a game) interact with each other by exchanging events. Thus messages containing the events should be exchanged among users' computers. In a server-based architecture, when an entity generates an event, it is transmitted to the server. Then the server determines which other entities are affected by the event and sends the event to each of the affected entities.

Scalability is an inevitable issue in such a server-based architecture. A straightforward solution is limiting the number of concurrent users per server. Each server is a separate virtual space and users of different servers cannot interact with each other. Another approach is dividing the virtual space into regions with one server per region. In that case it is difficult to handle entities on a boundary.

A peer-to-peer approach is a solution to the above-mentioned scalability problem [2, 7, 14, 16].[1] In a peer-to-peer system, when a node joins, both supply and demand of resources increase. The key to a peer-to-peer system design is whether the role of a server can be distributed among peers. In a distributed virtual environment, a major task of a server is determining which entities should receive an event and transmitting the event to the entities. Note that an entity in a distributed virtual environment interacts mostly with nearby entities. For example, when an entity makes sound, the sound will be heard by entities within a certain range in the virtual space. In general, each entity has an area of interest (AOI), which is typically a circle (in 2D) or sphere (in 3D) centered at the entity. An entity needs to propagate events that occur at itself to other entities within its AOI, and receive events that occur within its AOI. Therefore if an entity can find its nearby entities, the entity can exchange events with the nearby entities.

The problem of finding nearby entities has led to Delaunay triangulation, which we will abbreviate as DT in this paper. A DT-based entity-connection topology for distributed virtual environments has been proposed by various authors including [2, 7, 14]. A DT is constructed for a set of nodes in a Euclidean space, in which each node is a point in the space. An entity in a virtual space can be abstracted as a node. We will use a node (on a DT) and an entity (in a virtual space) interchangeably in this paper. A triangulation in a 2D space means, for a given set of nodes, constructing edges between pairs of nodes such that the edges form a non-overlapping set of triangles that cover the convex hull of the nodes. DT in

*Research sponsored by National Science Foundation grant CNS-0434515.

[1] Though a peer-to-peer approach may address the scalability problem, it raises other technical issues. For example, guaranteeing that every user observes events in a consistent order is a difficult problem in a fully distributed system.

a 2D space is usually defined as a triangulation such that the circumcircle of each triangle does not include any node other than the vertexes of the triangle. DT can be similarly generalized for higher dimensions [6].

We note two interesting properties of DT. First, it connects a node to other nodes that surround the node, which makes it useful for finding nearby entities. Second, greedy routing always succeeds on a DT [1]. In greedy routing, a node forwards a message to one of its neighbors that is closest to a given destination node. Note that greedy routing on an arbitrary graph is prone to the risk of being trapped at a local optimum, i.e., routing stops at a non-destination node that is closer to the destination than any of its neighbors. However, on a DT it is guaranteed that greedy routing always succeeds to find the destination node. Note that greedy routing does not always find the shortest route. However, the quality of the greedy route is often very good, since the length of an optimal route between a pair of nodes on a DT is within a constant time of the direct distance [3, 5, 8]. Definition of a distributed DT and a discussion of its properties are presented in section 3. We present a suite of protocols to construct and maintain a distributed DT for a dynamic set of nodes in d-dimensional space ($d \geq 2$) in [10].

From the two properties of DT, we design a protocol, named RadGRPM (Radius Greedy Reverse-Path Multicast), which runs on a distributed DT of a set of nodes in Euclidean space. Using the RadGRPM protocol, a node can send a message to nodes within a circle (in 2D), sphere (in 3D), or hypersphere (in d-D) of a given radius from itself. Multicast to nodes within a target region is called *geocast* in the wireless networks literature [13]. RadGRPM provides a special kind of geocast, which we call *radius geocast*, because the target nodes are all within a spherical region centered at the source. For distributed virtual environments, since an AOI is usually a circle or sphere centered at an entity, RadGRPM is well suited for propagating events. For other potential applications, such as wireless networks and sensor networks, we will show that RadGRPM can be combined with unicast greedy routing from the source node to the center of a spherical region to provide a general geocast service.

To design RadGRPM, we first design a broadcast protocol on a distributed DT, called GRPB (Greedy reverse-path broadcast).[2] To broadcast a message from a source node s, GRPB uses the reverse path of a greedy-routing path from every node u to s. Since greedy routing always succeeds on a DT, there exists a greedy-routing path from every node u to s. Therefore if GRPB forwards a message from s to all the reverse-paths of greedy-routing paths to s, the message will be delivered to every node u. In section 4, we prove that GRPB delivers a message from a source node to all other nodes in the system. Note that GRPB may be considered as a special case of RadGRPM, in which the radius is infinite.

GRPB is similar to HyperCast [12], which is also a broadcast protocol using reverse paths on a distributed DT. The difference between the two protocols is that GRPB is based on greedy routing in d-dimensional space while HyperCast is based on compass routing in 2D space. The major advantage of both approaches is that a broadcast tree does not need to be explicitly maintained. A node can determine next-hop nodes based on the coordinates of its neighbors, itself, and the source node.

We observe that the distance from a source node to each hop in GRPB monotonically increases, since the distance to a destination node decreases in greedy routing. We utilize this observation in designing the RadGRPM protocol. RadGRPM is basically the same as GRPB, except that it additionally checks whether the next-hop nodes are within the radius from the source node. In section 4, we prove that RadGRPM delivers a message from a source node to all nodes within a given radius from the source node. RadGRPM also keeps the advantage of GRPB in that a multicast tree is not explicitly maintained.

We evaluate our protocols in terms of correctness and efficiency. We say that a protocol is *correct* if it delivers a message to every target node. As was stated earlier, we have proved correctness for both GRPB and RadGRPM. We define *efficiency* of a protocol as the ratio of the number of target nodes to the number of message transmissions. For example, if a protocol uses 100 messages to reach 50 target nodes, its efficiency is 50%. Ideally each target node needs to receive exactly one message and non-target nodes should not receive any message, in which case efficiency is 100%. Flooding a message to all neighbors may achieve correctness, but its efficiency will be very low. Both GRPB and RadGRPM achieve high efficiency. In GRPB, very few nodes receive duplicate messages. In RadGRPM, very few nodes within the radius receive duplicate messages, and nodes outside the radius receive no message. A small number of duplicate messages are sent due to limitation of local knowledge at some nodes. By extensive simulation experiments, we investigate correctness and efficiency, as well as other characteristics of our protocols such as node outdegree and hop count.

The organization of this paper is as follows. In section 2, we present our system model and assumptions. In section 3, we introduce concepts and definitions of DT and distributed DT. In section 4, we present our broadcast and geocast protocols and prove their correctness. We present extensive experimental results in section 5. Finally we conclude in section 6.

2. System model

A distributed virtual environment is a virtual space, in which entities interact with one another. Each entity processes and generates events, changing its state. An entity interacts with other entities by generating events that will be processed

[2] GRPB and RadGRPM were first introduced in our prior work [11] as examples of applications of a distributed DT. In this paper we provide proof of correctness as well as an extensive evaluation of their performance characteristics.

at other entities and processing events that were generated by other entities. We assume that the virtual space is a Euclidean space, usually 2D or 3D (actually our protocols in this paper are proved correct for a d-dimentional space). Thus each entity has its coordinates that represent its location in the virtual space. Each entity is assigned to a networked computer, which processes events for the entity and maintains the entity state. The computers are connected with one another via a network, so that an event generated at a computer x for an entity on another computer y can be transmitted from x to y.

We assume that the entities form a distributed DT. That is, each entity is a node on the distributed DT. For convenience's sake, we use an entity and a node interchangeably. In a distributed DT, a node maintains a set of its "neighbor" nodes. We define a DT, a neighbor, and a distributed DT in the next section.

Recall that nodes (entities) are associated with their coordinates. When a node "knows" other nodes, it also knows their coordinates. That is, a node knows its own coordinates, coordinates of its neighbors, and the coordinates of any other nodes that it knows such as the destination node in routing and the source node in broadcasting. The distance between any two nodes can be calculated from their coordinates.

Lastly, for simplicity of our protocol description, we assume reliable delivery of protocol messages. In a real implementation, additional mechanisms such as ARQ can be used to ensure reliable message delivery.

3. Concepts and definitions of Distributed DT

Consider a set of nodes. Conceptually, nodes are points in a Euclidean space.

We first define Voronoi diagram of a set of given nodes and then define DT as the dual of the Voronoi diagram. Note that there is another way of directly defining DT using circumcircles of triangles (or circum-hyperspheres of simplexes in higher dimensions). Since the DT properties of interest to us come from Voronoi diagrams, we believe that our approach is appropriate in the context of this paper.

Definition 1. *Consider a set S of nodes in a Euclidean space. The* **Voronoi diagram** *of S is a partitioning of the space into cells such that a node $u \in S$ is the closest node to all points within its Voronoi cell $VC_S(u)$.*

That is, $VC_S(u) = \{p \mid D(p,u) \leq D(p,w), \text{for any } w \in S\}$ where $D(x,y)$ denotes the distance between x and y. Note that a Voronoi cell in a d-dimensional space is a convex d-dimensional polytope enclosed by $(d-1)$-dimensional facets.

Definition 2. *Consider a set S of nodes in a Euclidean space. $VC_S(u)$ and $VC_S(v)$ are neighboring Voronoi cells, or* **neighbors** *of each other, if and only if $VC_S(u)$ and $VC_S(v)$ share a facet, which is denoted by $VF_S(u,v)$.*

Definition 3. *Consider a set S of nodes in a Euclidean space. The* **Delaunay triangulation** *of S is a graph on S where two nodes u and v in S have an edge between them if and only if $VC_S(u)$ and $VC_S(v)$ are neighbors of each other.*

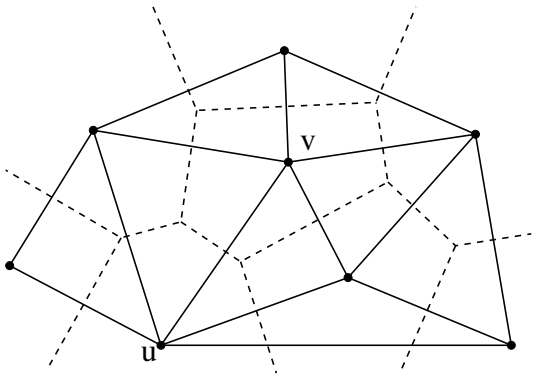


Figure 1. A Voronoi diagram (dashed lines) and the corresponding DT (solid lines) in a 2-dimensional space.

Figure 1 shows a Voronoi diagram (dashed lines) for a set of nodes in a 2D space and a DT (solid lines) for the same set of nodes. Note that $VC_S(u)$ and $VC_S(v)$ are neighbors of each other. We also say that u and v are neighbors of each other when $VC_S(u)$ and $VC_S(v)$ are neighbors of each other. Also note that facets of a Voronoi cell perpendicularly bisect edges of a DT. Therefore, a DT is the dual of a Voronoi diagram.[3] Let us denote the DT of S as $DT(S)$.

By a distributed DT, we mean that each node $u \in S$ maintains a set N_u of its neighbor nodes.

Definition 4. *A* **distributed Delaunay triangulation** *of a set S of nodes is specified by $\{< u, N_u > \mid u \in S\}$, where N_u represents the set of u's neighbor nodes, which is locally determined by u.*

Definition 5. *A distributed Delaunay triangulation of a set S of nodes is* **correct** *if and only if both of the following conditions hold for every pair of nodes $u, v \in S$:*

- *if there exists an edge between u and v on the global DT of S, then $v \in N_u$ and $u \in N_v$,*
- *if there does not exist an edge between u and v on the global DT of S, then $v \notin N_u$ and $u \notin N_v$.*

That is, a distributed DT is correct when for every node u, N_u is the same as the neighbors of u on $DT(S)$ [11].

4. Broadcast and geocast protocols

4.1. Broadcast using reverse greedy paths

As was discussed earlier, the greedy routing algorithm finds a path from a source node to a destination. Consider

[3] In geometry, polyhedra are associated into pairs called duals, where the vertices of one correspond to the faces of the other.

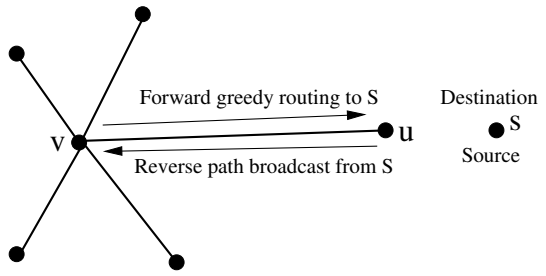


Figure 2. Forward path and reverse path.

such paths from all nodes in S to a node $s \in S$. The union of the paths is a tree rooted at s. Therefore by reversing the direction of each path, we get a broadcast tree from a source node s to every other node in S. Figure 2 illustrates an example of a reverse path. In forward greedy routing, v selects u as the next hop, since u is its closest neighbor to the destination s. Thus in a reverse-path broadcast from the source node s, u should forward a message to v, if u knows that u is the next hop of v in the forward route. Note that s is the destination in the forward greedy routing and the source in the reverse-path broadcast. We introduce a simple broadcast protocol which utilizes the reverse-path tree. Note that our protocol does not require knowledge of the global triangulation over S. Each node u is assumed only to know its set of neighbor nodes, and determines to which node(s) it should forward a message based on its local knowledge. Specifically, node u in the previous example may not know all the neighbors of v. u only knows the neighbors of u, but still has to determine whether u is the closest node to s among v's neighbor nodes.

The idea of using reverse path for broadcast goes back to as early as 1978 [4]. In the context of DT, HyperCast [12] is the first system to introduce the idea. Our protocol is different in that it is based on greedy routing in an arbitrary dimension while HyperCast is based on compass routing in a 2D space. The major advantage of both approaches is that a broadcast tree does not need to be explicitly maintained. A node can determine next-hop nodes based on the coordinates of its neighbors, itself, and the source node.

We name our broadcast protocol as GRPB (greedy reverse-path broadcast). In GRPB, a node u maintains *a local DT* of u and u's neighbors. For each neighbor v, u forwards a message from a source node s to v if both of the following two conditions hold:

C1 u is closer to s than v is.

C2 In the local DT of u and u's neighbor nodes, there does not exist a node $w \neq u$ such that: **C2.1** w is closer to s than u is, and **C2.2** u, v and w are included in the same triangle (or simplex in a d-dimensional space).

Condition C1 is easy to understand. Suppose C1 is true. Then u does not forward to v if u is sure that another node, say w, is the next hop of v in the forward greedy routing. The necessary and sufficient conditions for such w are: **C2.1** w is closer to s than u, and **C2.3** w is a neighbor of v on the global DT. However, u does not have global information and cannot

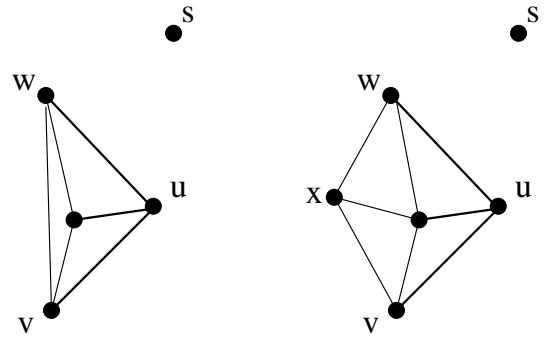


Figure 3. An ambiguous situation due to limited knowledge in GRPB.

check C2.3. Hence we specify condition C2.2 which includes C2.3. C2.1 and C2.2 are necessary but not sufficient.

Note that in case of a tie between w and u in C2.1, u must forward to v at the cost of possible duplication, since v may or may not choose u as the next hop in the forward greedy routing. Note also that even if node w appears to be v's neighbor in u's local DT, w may not actually be v's neighbor in the global DT. Figure 3 illustrates an example in a 2D space. The left graph shows u's local DT, in which v and w are neighbors. However, as shown in the right graph, there may exist a node x outside u's local knowledge and thus w may not actually be a neighbor of v. Without including C2.2 in C2, u might erroneously conclude that it does not need to forward to v, since w appears to be the closest node to s among v's neighbors. C2.2 detects such ambiguous situations and requires that u forwards to v at the cost of possible duplication. We performed experiments to broadcast a message using GRPB on a distributed DT of 200 randomly-placed nodes in various dimensions. In the experiments, the number of duplicate messages was from 3% to 10% of the number of nodes. For further experiments on message duplication, refer to Section 5.2. The protocol pseudocode is presented in Figure 4.

The following theorem guarantees the correctness of

Start_broadcast(msg) of node u
; u is a source node, loc is location of u
for all $v \in N_u$ **do**
  send BROADCAST(msg, loc) **to** v
end for

On u**'s receiving BROADCAST(**msg**,** loc**)**
; u is a recipient of a BROADCAST message
Deliver(msg)
for all $v \in N_u$ **do**
  if v satisfies conditions C1 and C2 from loc **then**
    send BROADCAST(msg, loc) **to** v
  end if
end for

Figure 4. GRPB protocol at a node u.

GRPB, namely it delivers a message to all nodes in the system.

Theorem 1. *Let a set S of nodes form a correct distributed DT. The GRPB protocol delivers a message from a source node $s \in S$ to all the other nodes in S.*

Proof. We prove the theorem by showing that if there exists an edge from u to v in the global reverse-path tree, the GRPB protocol also forwards a message from u to v.

Our proof is by contradiction.

(1) Assume that the theorem is not true. Suppose a node u fails to forward to its neighbor v, while there exists an edge from u to v in the global reverse-path tree.
(2) v is a neighbor of u on the local DT of u, since the distributed DT is correct. [From Definition 5.]
(3) u is closer to s than v is, since there is an edge from u to v in the global reverse-path tree.
(4) On the local DT of u, there exists a node w that is a mutual neighbor of u and v, and the distance between w and s is shorter than the distance between u and s. [From (1), (2), (3), and conditions C1 and C2.]
(5) w is not a neighbor of v on the global DT. [If w were a neighbor of v, the next hop of v in the forward path should not be u since, from (4) , w is closer than u to s.]
(6) On the local DT of u, there exists a simplex that includes u, v and w. Let the simplex be denoted by p. [From condition C2.2.]
(7) p does not exist on the global DT, since w is not a neighbor of v. [From (5).]
(8) On the global DT, the space of p is occupied by other simplexes.
(9) Let x be one such simplex that includes u and v. Let $x_1, ..., x_k$ be the other nodes of x other than u or v.
(10) $x_1, ..., x_k$ are neighbors of u on the global DT.
(11) $x_1, ..., x_k$ are neighbors of u on the local DT of u, since the distributed DT is correct. [From Definition 5.]
(12) There exists the same simplex x on the local DT of u, since v and $x_1, ..., x_k$ are neighbors of u. [From (2) and (11).]
(13) It is impossible that x and p co-exist on the local DT of u, since they overlap. □

4.2. Radius geocast

Geocast is a special case of multicast in which a message is delivered to all nodes in a given region. Our radius geocast protocol, RadGRPM, is designed to deliver a message to all nodes within a given radius from a source node.

We observe that in the GRPB protocol the distance of next hop from the source monotonically increases, since the distance to the destination monotonically decreases in forward greedy routing. We utilize this observation in designing RadGRPM.

In RadGRPM from a source node s to all the other nodes within a given radius r, s first sends the message to all its neighbors within r. Then, for each neighbor node v, a node u forwards a message to v if the following condition holds as well as C1 and C2 in GRPB:

C3 The distance from s to v does not exceed the radius r.

Essentially the protocol is the same as the original GRPB protocol, except that forwarding stops when the distance from the source exceeds the given radius (condition C3). Note that no node outside the given radius receives any message, which is one reason that RadGRPM is efficient. Pseudocode of the protocol is presented in Figure 5. Theorem 2 guarantees that RadGRPM delivers the message to all nodes within a given radius.

```
Start_radius_geocast(msg, rad) of node u
; u is a source node, loc is location of u
for all v ∈ N_u within rad from loc do
  send GEOCAST(msg, rad, loc) to v
end for

On u's receiving GEOCAST(msg, rad, loc)
; u is a recipient of a GEOCAST message
Deliver(msg)
for all v ∈ N_u do
  if v satisfies conditions C1, C2, and C3 from loc
  then
    send GEOCAST(msg, rad, loc) to v
  end if
end for
```

Figure 5. RadGRPM protocol at a node u.

Theorem 2. *Let a set S of nodes form a correct distributed DT. The RadGRPM protocol delivers a message from a source node $s \in S$ to all nodes within a radius r from s.*

Proof. By Theorem 1, the GRPB protocol delivers a message to all other nodes in S. Since the distance from s monotonically increases whenever a message is forwarded and the forwarding stops when the distance from s exceeds r, all nodes along the reverse greedy paths after stopping have distances from s longer than r. Therefore the RadGRPM protocol delivers the message to all nodes within the radius r. □

4.3. General geocast

Note that RadGRPM by itself is a special kind of geocast in the sense that a source node is at the center of a spherical target region. RadGRPM can be combined with unicast greedy routing to provide general geocast. That is, in case a source node is not at the center of a spherical target region, the source sends a unicast message to the center location using greedy routing; the message is then propagated within the target region using RadGRPM, as described below.

If no node exists at the center of a spherical target region, greedy routing towards the center (specified by its coordinates) will succeed to forward the unicast message to a node

closest to the center (see Theorem 1 and its proof in [9]). For clarity of explanation, let us assume for now that there is only one node that is closest to the center point. The case where there are two or more closest nodes is addressed below. Let c denote the center point and c' the closest node to c. As soon as c' receives a unicast message by greedy routing towards c, c' determines that it is a closest node to c among its neighbors and starts RadGRPM. More specifically, it executes the Start_radius_geocast() function in Figure 5 using the center's location for parameter loc instead of its own location. Starting RadGRPM from c' does not affect correct execution of RadGRPM. Correctness of GRPB and RadGRPM is based on the fact that greedy routing to a node always succeeds on a DT. Since greedy routing towards c always succeeds to reach the closest node c', c' is the root of the reverse-path tree of greedy routing from every node towards c. Note that the same greedy routing towards c is used, whether a node exists at c or not. Therefore the same reverse-path conditions (C1 and C2) can be used even if RadGRPM is started from c'. Also, the distance from c monotonically increases in the reverse greedy paths, allowing use of the same stopping condition (C3).

It is possible that there are two or more nodes closest to the center. (These nodes are equidistant from the center.) Greedy routing towards the center will forward the unicast message to one of the closest nodes. Let c'_1 denote the closest node that receives the unicast message. Let $c'_2, ..., c'_k$ denote the other closest nodes, where $k > 1$. Note that the greedy paths from all nodes form a forest of trees, the root nodes of which are the closest nodes, $c'_i, 1 \leq i \leq k$. When the unicast message arrives at c'_1 by greedy routing, c'_1 determines that none of its neighbors is closer to the center than itself, which means c'_1 is one of the closest nodes, and it executes the Start_radius_geocast() function in Figure 5 using the center's location for paramter loc instead of its own. In the function, c'_1 sends a geocast message to each of its neighbors within the radius of the center. Therefore if another closest node is a neighbor of c'_1, it will receive the geocast message; it also determines that it is a closest node to the center and it executes the Start_radius_geocast() function in Figure 5 using the center's location for parameter loc instead of its own. Thus, if the set of closest nodes to the center and DT edges between them form a connected graph, then all of the other closest nodes are guaranteed to receive the geocast message. Subsequently, the geocast message will be propagated in all reverse-path trees of the forest and it will be delivered to all nodes within the radius of the center location.

In the following Lemma, we prove that the set of nodes closest to the center and DT edges between them form a connected graph, which is sufficient to prove correctness of our general geocast protocol described above.

Lemma 1. *Let a set S of nodes form a correct distributed DT. Let p denote a point in the space. Let $c'_1, c'_2, ..., c'_k$ denote the closest nodes to p in S, , $k > 1$. Then the subgraph of DT that includes $c'_1, c'_2, ..., c'_k$ and edges between them is a connected graph.*

Proof. Our proof is by contradiction.

(1) Suppose the subgraph of DT is not a connected graph. Without loss of generality, suppose that $c'_1, c'_2, ..., c'_h$ are connected, $h < k$, but they are not connected to c'_k.
(2) Let $N_c = N_{c'_1} \cup N_{c'_2} \cup ... \cup N_{c'_h} - \{c'_1, c'_2, ..., c'_h\}$. Let n_c be the closest node in N_c to p. Let $\Delta = D(n_c, p) - D(c'_1, p)$.
(3) Let p' be a point that is $\frac{\Delta}{4}$ away from p towards c'_k.
(4) $D(c'_k, p') = D(c'_k, p) - \frac{\Delta}{4}$.
(5) $D(c'_i, p') > D(c'_i, p) - \frac{\Delta}{4} = D(c'_k, p'), i \neq k$. [$c'_i, i \neq k$ cannot be in the same direction as c'_k from p.]
(6) c'_k is the only closest node to p'.
(7) $D(c'_i, p') \leq D(c'_1, p) + \frac{\Delta}{4}, 1 \leq i \leq h$. [From (3) and the assumption that $D(c'_1, p) = D(c'_i, p), 1 \leq i \leq k$.]
(8) $D(c'_1, p) + \frac{\Delta}{4} \leq D(n'_c, p) - \Delta + \frac{\Delta}{4}, n'_c \in N_c$. [From (2).]
(9) $D(n'_c, p) - \Delta + \frac{\Delta}{4} < D(n'_c, p) - \frac{\Delta}{4}, n'_c \in N_c$.
(10) $D(n'_c, p) - \frac{\Delta}{4} \leq D(n'_c, p'), n'_c \in N_c$. [From (3).]
(11) $D(c'_i, p') < D(n'_c, p'), 1 \leq i \leq h, n'_c \in N_c$. [From (7) – (10).]
(12) Greedy routing from c'_1 towards p' will be stuck at one of the nodes $c'_1, ..., c'_h$, and cannot reach c'_k.
(13) Greedy routing from any node in S towards p' always succeeds to reach the closest node to p', which is c'_k. [From (6) and Theorem 1 in [9].]
(14) (12) and (13) are contradictory to each other. □

5. Experimental results

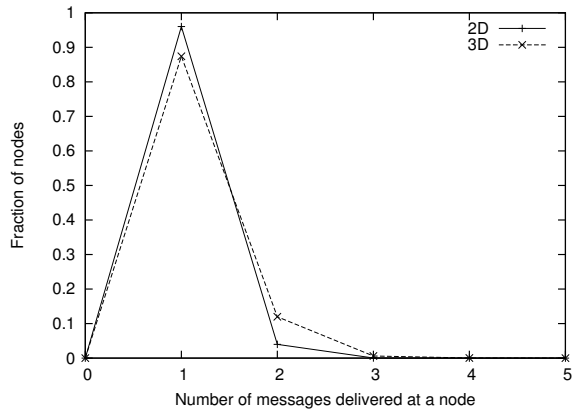
Using simulation experiments, we first evaluate our protocols in terms of correctness and efficiency. Then we investigate characteristics of our protocols in terms of node outdegree and hop count. In our experiments, 1000 nodes are randomly placed in a 2D (or 3D) space, each axis of which has a range of 0 to 9999. We run our protocols from each of the 1000 nodes and the average of the 1000 experiments is shown.

5.1. Correctness

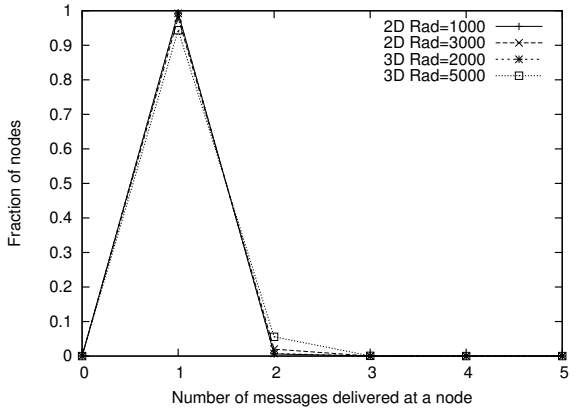
We say that a protocol is *correct* if it delivers a message to every target node. That is, GRPB should deliver a message to all nodes in the system and RadGRPM should deliver a message to all nodes within the given radius. Recall that both GRPB and RadGRPM are proved to be correct by Theorem 1 and Theorem 2, respectively. In every one of many thousands of experiments we conducted, GRPB and RadGRPM always worked correctly, namely, delivered a message to every target node.

5.2. Efficiency

We define *efficiency* of a protocol as the ratio of the number of target nodes to the number of message transmissions. For

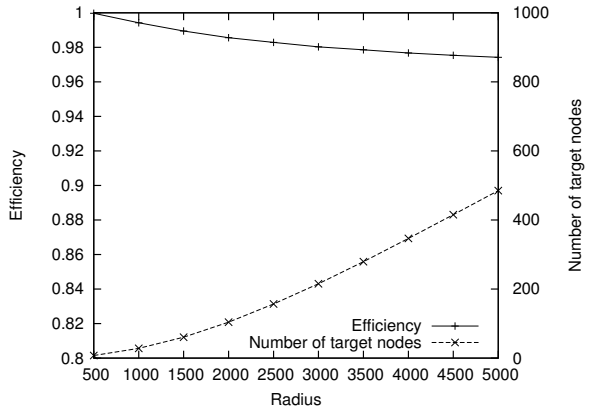


(a) GRPB

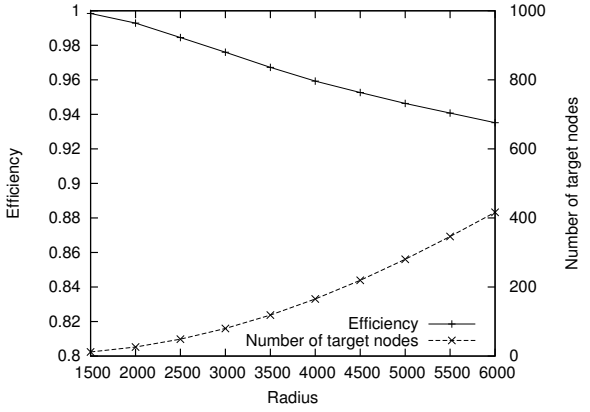


(b) RadGRPM

Figure 6. Distribution of number of messages delivered at a node.



(a) 2D



(b) 3D

Figure 7. Efficiency and number of target nodes.

example, if a protocol uses 100 messages to deliver to 50 target nodes, its efficiency is 50%. Ideally each target node needs to receive exactly one message and non-target nodes should not receive any messages, in which case efficiency is 100%. There are two sources of inefficiency: (a) a non-target node receives a message and (b) a target node receives a message more than once. Our protocols are carefully designed such that non-target nodes do not receive any message and very few target nodes receive duplicate messages. A small number of duplicate messages are sent due to limitation of local knowledge at some nodes.

Figure 6(a) shows the distribution of number of messages delivered at a node in GRPB. The solid line represents the result in 2D and the dashed line represents the result in 3D. In both results, most of nodes receive the broadcast message exactly once. Most of the other nodes receive the message twice. The efficiency is 96.1% in 2D and 88.3% in 3D.

Figure 6(b) shows the distribution of number of messages delivered at a node in RadGRPM. The results of radius 1000 in 2D, radius 3000 in 2D, radius 2000 in 3D, and radius 5000 in 3D are shown. The average number of target nodes in each case is 28.2, 215.4, 26.3, and 280.2, respectively. The efficiency in each case is 99.4%, 98.0%, 99.3%, and 94.6%, respectively.

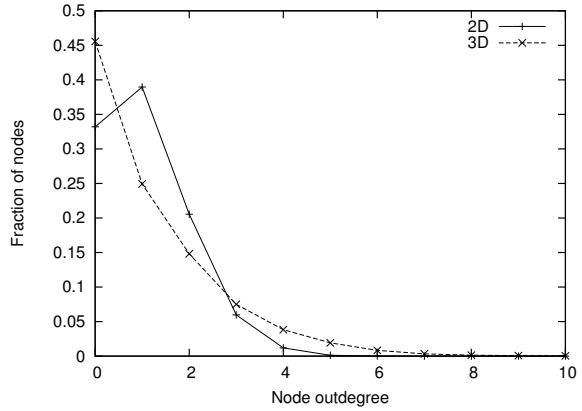
Figure 7 shows a trend that efficiency decreases as the number of target nodes increases. However, the efficiency is still very high for hundreds of target nodes.

5.3. Node outdegree and hop count

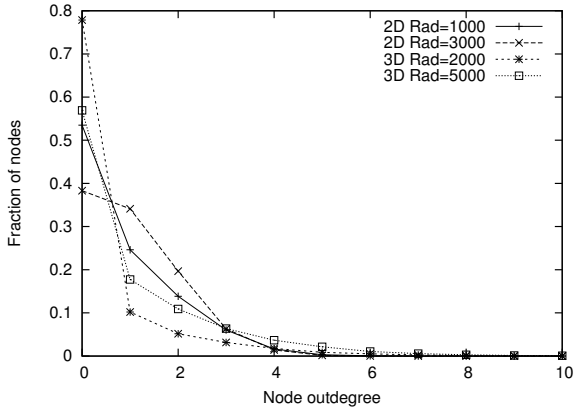
Node outdegree and hop count are important characteristics of a broadcast/multicast tree.[4] The outdegree of a node is the number of other nodes to which the node sends a message in a broadcast/multicast. A low node outdegree is preferred since a node has limited resources, especially in a peer-to-peer environment. A low hop count is also preferred to reduce delay. There is a trade-off between node outdegree and hop count. That is, a tree cannot have a low node outdegree and a low hop count at the same time.

Figure 8(a) shows the distribution of node outdegree in GRPB. The solid line represents the result in 2D and the

[4]Even though our protocols do not explicitly maintain a broadcast/multicast tree, the graph consisting of all message-forwarding paths is called a tree. Note that, to be strict, the graph may not be a tree due to duplicate messages delivered to a node.

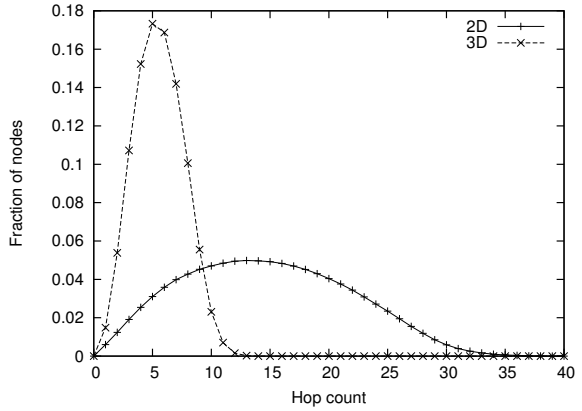


(a) GRPB

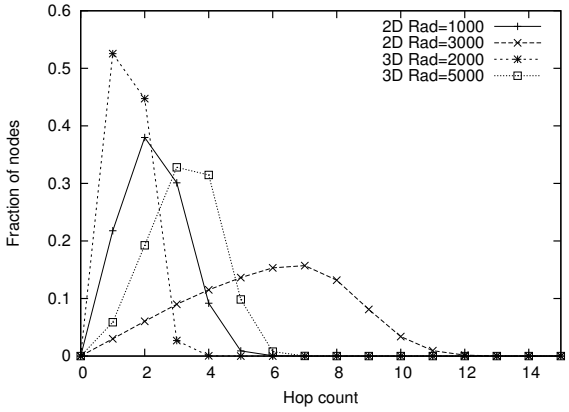


(b) RadGRPM

Figure 8. Distribution of node outdegree.



(a) GRPB



(b) RadGRPM

Figure 9. Distribution of hop count.

dashed line represents the result in 3D. The average node outdegree is higher in 3D than 2D, which is expected since a node has more neighbors in a higher-dimension DT. Both in 2D and 3D, very few nodes have outdegree of four or higher.

Figure 8(b) shows the distribution of node outdegree in RadGRPM. The results of radius 1000 in 2D, radius 3000 in 2D, radius 2000 in 3D, and radius 5000 in 3D are shown. In all cases, very few nodes have outdegree of four or higher.

Figure 9(a) shows the distribution of hop count in GRPB. The solid line represents the result in 2D and the dashed line represents the result in 3D. The average hop count is 14.8 in 2D and 5.6 in 3D, which is smaller in 3D since the average node outdegree is higher in 3D.

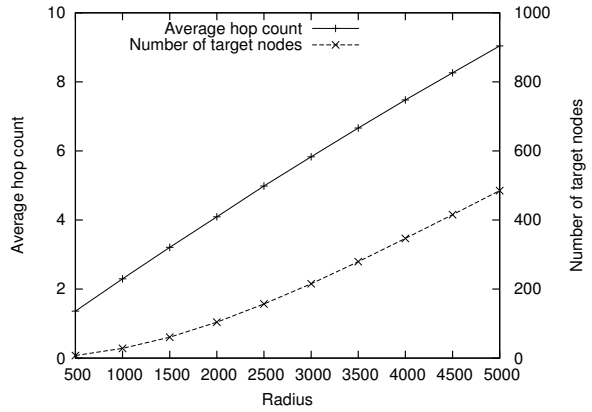
Figure 9(b) shows the distribution of hop count in RadGRPM. The results of radius 1000 in 2D, radius 3000 in 2D, radius 2000 in 3D, and radius 5000 in 3D are shown. The average hop count in each case is 2.3, 5.8, 1.5, and 3.2, respectively.

Figure 10 shows that the average hop count increases as the number of target nodes increases. Since the average node outdegree is higher in 3D than 2D, the average hop count increases faster in 2D. This is due to the planar nature of a DT. That is, a DT does not have a shortcut that connects a node to a faraway node. If necessary, the average hop count may be reduced by introducing additional shortcut edges to a DT. Correctness GRPB and RadGRPM is not affected by forwarding additional messages over shortcut edges. Efficiency decreases in exchange for a decreased average hop count, since the additional messages are redundant. Tsuboi *et al.* [17] recently proposed Skip Delaunay Network (SDN), which is a hierarchy of DTs and enables a unicast and a geocast protocol with $log(N)$ hop counts. Their geocast protocol (GeoMulticast) delivers a message to a rectangular region. A hierarchy of DTs is also mentioned in [12].

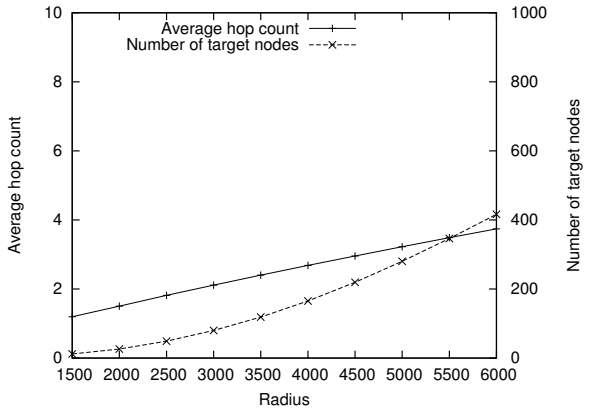
6. Conclusions

Distributed virtual environments are gaining popularity in recent years. In a distributed virtual environment, an entity communicates with its nearby entities. To support that operation, we design and investigate a radius geocast protocol, RadGRPM, which is derived from our broadcast protocol, GRPB. Our protocols run on a distributed DT of a set of nodes in Euclidean space. A suite of protocols to construct and maintain a distributed DT for a dynamic set of nodes in d-dimensional space ($d \geq 2$) is presented in our prior work [10].

GRPB and RadGRPM take advantage of two properties of DT: (i) DT connects an entity with its nearby entities and (ii)



(a) 2D



(b) 3D

Figure 10. Average hop count and number of target nodes.

greedy routing always succeeds on a DT. Neither of our protocols requires an explicit broadcast/multicast tree.

Our protocols can be used for a network of nodes with known coordinates in a Euclidean space. Other potential applications of our protocols are sensor networks and wireless *ad hoc* networks, where nodes are equipped with GPS devices or nodes use signal strengths, message delays, and/or signal directions to determine locations. Typical applications of geocast include querying nodes within a region of a sensor network and sending an emergency warning to users within a region of a wireless *ad hoc* network [13, 15].

We evaluate our protocols in terms of correctness and efficiency. We prove that both of our protocols are correct, which means that no target node fails to receive a message. In all simulation experiments we conducted, GRPB and RadGRPM worked correctly. Simulation experiments also show that our protocols are highly efficient in the sense that the number of messages used for a broadcast/multicast is not much more than the number of target nodes, which is the lower bound.

References

[1] P. Bose and P. Morin. Online routing in triangulations. *SIAM journal on computing*, 33(4):937–951, 2004.

[2] E. Buyukkaya and M. Abdallah. Data management in Voronoi-based P2P gaming. In *Proc. IEEE International Workshop on Digital Entertainment, Networked Virtual Environments, and Creative Technology (CCNC 2008)*, January 2008.

[3] L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[4] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040–1048, 1978.

[5] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete and Computational Geometry*, 5(1):399–407, 1990.

[6] P. M. Gruber and J. M. Wills. *Handbook of convex geometry*. North-Holland, 1993.

[7] S. Y. Hu, J. F. Chen, and T. H. Chen. VON: A scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, Jul/Aug 2006.

[8] J. M. Keil and C. A. Gutwin. The Delauney triangulation closely approximates the complete Euclidean graph. In *Proc. of the Workshop on Algorithms and Data Structures (LNCS 382)*, pages 47–56. Springer-Verlag London, UK, 1989.

[9] D.-Y. Lee and S. S. Lam. Protocol design for dynamic Delaunay triangulation. Technical Report TR-06-48, The Univ. of Texas at Austin, Dept. of Computer Sciences, October 2006.

[10] D.-Y. Lee and S. S. Lam. Efficient and Accurate Delaunay Triangulation Protocols under Churn. Technical Report TR-07-59, The Univ. of Texas at Austin, Dept. of Computer Sciences, November 2007. Revised May 2008.

[11] D.-Y. Lee and S. S. Lam. Protocol design for dynamic Delaunay triangulation. In *Proc. of 27th International Conference on Distributed Computing Systems*, Toronto, Canada, June 2007.

[12] J. Liebeherr and M. Nahas. Application-layer multicast with Delaunay triangulations. *Proceedings of Global Internet Symposium, IEEE Globecom 2001*, 3:1651–1655, November 2001.

[13] J. C. Navas and T. Imielinski. Geocast - geographic addressing and routing. In *Proceedings of the ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom '97)*, Budapest, Hungary, September 1997.

[14] L. Ricci and A. Salvadori. Nomad: Virtual environments on P2P Voronoi overlays. In *Proceedings of First International Workshop on Peer to Peer Networks (LNCS Vol. 4806)*, November 2007.

[15] K. Seada and A. Helmy. Efficient geocasting with perfect delivery in wireless networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC 2004)*, volume 4, pages 2551–2556, Atlanta, Georgia, USA, March 2004.

[16] A. Steed and C. Angus. Supporting scalable peer to peer virtual environments using frontier sets. In *Proc. of the IEEE Virtual Reality Conference 2005*, pages 27–34, Bonn, Germany, March 2005.

[17] S. Tsuboi, T. Oku, M. Ohnishi, and S. Ueshima. Generating skip Delaunay network for P2P Geocasting. In *Proc. of the Sixth International Conference on Creating, Connecting and Collaborating through Computing*, Poitiers, France, January 2008.