AOI-cast by Compass Routing in Delaunay Based DVE Overlays

Laura Ricci  
Dep. of Computer Sc., University of Pisa  
ricci@di.unipi.it

Luca Genovali  
IMT, Institutions, Markets, Technology, Lucca  
genovali@imtlucca.it

Emanuele Carlini  
IMT, Institutions, Markets, Technology, Lucca  
Istituto “A. Faedo”, ISTI-CNR, Pisa  
emanuele.carlini@isti.cnr.it

Massimo Coppola  
Istituto “A. Faedo”, ISTI-CNR, Pisa  
massimo.coppola@isti.cnr.it

ABSTRACT

This paper presents a AOI cast strategy for P2P Distributed Environments which is exploited to notify the position updates of a peer P, i.e. its heartbeats, to all the peers located in its Area of Interest. An algorithm for the construction of a spanning tree covering all the peers is presented. The algorithm exploits the properties of Delaunay Triangulations to reduce the traffic load on the P2P overlay. The paper presents a set of formal results which hold when the AOI is a circular area and the root of the tree is at the center of the area. The algorithm is then refined to take into account possible inconsistencies among the local views of the peers due to the latency of the underlying network. A set of experimental results are presented.

KEYWORDS:  Peer to Peer Architectures and Networks, Routing Synchronization and Consistency, Cooperative Information Systems and Applications, Message Passing

1. INTRODUCTION

Distributed Virtual Environments (DVE)[15] such as military or civil protection distributed simulations and massively multiplayer online games (MMOG) are currently gaining increasingly attention in the software market. Even if the client server model is still widely adopted to support DVE applications, several approaches based on the P2P model [5, 19, 23, 11, 12, 3, 6] have recently been proposed. Since in these architectures the definition of a scalable communication support is mandatory, the concept of Area of Interest (AOI) has been introduced to reduce the traffic on the P2P overlay. The implementation of the AOI requires that a peer dynamically defines a set of connections either with all the peers located in its AOI, or with a subset S of them, for instance the nearest ones. In the first case, each peer directly sends its heartbeats, i.e. its position updates, to the peers located in its AOI while in the latter case it sends them to the peers belonging to S and these forward the heartbeats to the remaining peers. An AOI cast mechanism [19, 20, 12], i.e. an application level multicast constrained within the boundaries of the AOI, is exploited to support the forwarding phase. In [20], we have proposed a P2P overlay for DVE based on a Delaunay Triangulation [9] of the virtual world. Our approach is based on the definition of an AOI cast mechanism thus requiring a proper mechanism to dynamically build a tree spanning all the peers located in the AOI of a peer. This paper proposes a tolerance based compass routing, an AOI cast algorithm which exploits the properties of the Delaunay Triangulation to reduce the number of messages required in the spanning tree construction. We first present the basic algorithm which is then refined to take into account the problems arising in the definition of a DVE support, such as the inconsistencies introduced by the unavoidable latency of the messages. The paper is structured as follows. Sect. 2 describes the proposals recently presented which are more related to our approach. The algorithm is defined in Sect.3, while Sect.4 discusses some properties of the algorithm which hold when the AOI is circular. Sect. 5 refines the basic algorithm by taking into account the inconsistencies arising in a DVE. The experimental results are reported in Sec.6. Finally Sect.7 reports some conclusions and outlines the future work.

2. RELATED WORK

Several proposals [15, 16, 5, 23, 11, 17, 12, 4, 21, 20, 18, 19] of P2P overlays for DVE have recently been presented. Solipsis [15] is one of the first proposals of a DVE architecture where nodes are self-organized in a pure peer to peer network in which the relationships between nodes depend upon virtual proximity. [16] considers an alternative
approach exploiting a Distributed Hash Table, DHT, to assign passive objects to the peers. Closer to our approach are the proposal based on the definition of a Voronoi tessellation [9] of the DVE both for the creation of the P2P overlay and for the management of the passive objects. [5] exploits edge flipping[9], an efficient procedure to build Delaunay triangulations, to define an efficient algorithm for the management of the overlay topology. Edge flipping reduces of the high cost of the overlay updates due to the continuous movement of the peers. The basic idea is to maximize the area where a node is allowed to move without triggering a flip operation. The problem of the cost of maintenance of the Delaunay overlay is also tackled in [23] which proposes a dynamic clustering algorithm where each peer monitors this cost and triggers the creation of a new cluster when it exceeds a predefined threshold. The management of the game state is investigated in [7] which proposes an hybrid overlay topology where two peers are the responsible of each passive object O of the DVE. The Voronoi responsible manages the Voronoi area where O is located while the DHT responsible stores its state. Thus each peer belongs to two distinct overlays, a structured and an unstructured one. The Voronoi Responsible of an object is aware of peers which need to be informed about the updates to the object and notify the list of these peers to the DHT Responsible. The DHT Responsible should notify each update to the object to the peers in this list. Note that the use of the DHT avoids the continuous transfer of objects between peers. [11] proposes a distributed scheme to maintain a greedy-supporting overlay based on the partition of the nodes into two non-overlapping sets, which are updated in two distinct phases. [12] defines a Delaunay overlay defining direct links between a peer and any other one in its AOI. The resulting overlay includes these links besides the Delaunay ones, which have to be maintained to guarantee the connectivity of the overlay. This solution minimizes the latency, but increases the number of connections of each peer. In a crowding scenario where a huge amount of peers are located in the same area of the DVE, a peer should manage a large number of connections. [12] tackles this problem by dynamically enlarging or shrinking the size of the AOI according to the bandwidth of the peers.

3. THE ALGORITHM

We recall that, given k nodes corresponding to the peers, a Voronoi tessellation partitions the DVE into k areas such that the area corresponding to a node n includes all the points of the DVE which are nearer to n with respect to any other site. Two sites are Voronoi neighbours iff the borders of their areas overlap. The connected graph defined by linking neighbour sites is the Delaunay Triangulation associated to the Voronoi tessellation. In the following we assume a proper support for the distributed maintenance of a Delaunay overlay is available. We have exploited VAST[2], a library for the management of Voronoi Diagrams, but different supports like those proposed in [5, 11, 17, 14] can be exploited as well. In the following the terms peer and node will be used in interchangeable way. Compass Routing has been proposed by Kranakis in [8] and is based on the following simple observation. Consider a connected graph and assume of being located at one of its nodes n with the goal to reach a destination node d. It is shown that the best strategy is to look at the edges incident in n and choose the edge whose slope is minimal with respect to the segment connecting n and the destination d. Besides, while compass routing is not cycle free for general graphs, it can always find a finite path between two nodes of a Delaunay Triangulation. Aurenhammer [9] suggests to exploit compass routing to define a Spanning Tree supporting an application level multicast. In our case, this strategy may be adopted to propagate the heartbeat generated by a peer to other peers located in its AOI. In this case the root of the multicast tree is the node which generates the heartbeat and the tree includes all the peers belonging to its AOI. Even if [9] suggests to build the spanning tree by reversing the path which compass routing computes from any node to the root of the spanning tree, no algorithmic solution is presented. Our contribution is the definition of an algorithm for the construction of a spanning tree on Delaunay networks.

We describe our algorithm through the example shown in Fig.1. Let us suppose that the peer Root generates an heartbeat, i.e. it is the root of the spanning tree, and let us consider the peer A which receives the heartbeat. A should choose among its Voronoi neighbours its children in the spanning tree. For instance, to decide if peer A5 is A’s child, the algorithm evaluates whether A5 would have chosen A in its path toward Root. This would happen if A5 is nearer to A than to Root. Note that if this happens, A5 does not need to compare the slopes of the edges connecting it to further neighbours with respect to the
segment $\overline{\text{Root}A_5}$, since they are surely larger. This argument is reversed in order to detect the children of a peer in the spanning tree. The basic point is that, in order to detect whether a Voronoi neighbour $v$ is its child in the spanning tree, a peer $p$ must determine the vertexes of the Delaunay Triangles sharing the Delaunay edge $\overline{pv}$ before performing angle evaluation. For instance, in Fig. 1 $A$ should consider the triangles $AA_3A_1$ and $AA_5A_4$ to detect whether $A_2$ is its child, while $B$ should consider only the triangle $\overline{BB_2B_3}$ to detect whether $B_3$ is its child in the tree, because of the borders of the $DVE$. The algorithm should distinguish these different scenarios. It is worth noticing that the Delaunay Triangles to consider cannot be detected by simply exploiting the Voronoi neighbourhood relation between the peers. For instance $B_1$ and $B_3$ in Fig. 1 are both Voronoi neighbours of $B$ and Voronoi neighbours themselves, but the triangle $\overline{B_1B_2B_3}$ does not belong to the Delaunay triangulation. Let us now describe the algorithm in more details.

The algorithm takes as input the coordinates of a peer $p$, those of $p$’s Voronoi neighbors and of the root of the spanning tree and returns the children of $p$ in the tree. The algorithm requires a minimal support for the management of Voronoi tessellations. VAST [2] includes a function returning the Voronoi neighbors of a given node, while more complex functionalities, like a function that returns the Delaunay triangles sharing a Delaunay edge, are seldom available. The algorithm requires the execution of two phases, Neighbors Sorting and Children Detection. In the Neighbors Sorting phase, a peer $p$ sorts its Voronoi neighbors according to a circular counter-clockwise ordering which is then exploited to determine, for each Voronoi neighbour $v$, the Delaunay triangles which are involved in the angle evaluation phase. To define the counter clockwise ordering, a coordinate system whose origin is at $p$ with unit vectors $\overrightarrow{j}$ and $\overrightarrow{j}$ is defined. The counter-clockwise ordering of the neighbors of $p$ is defined by considering the angle between $\overrightarrow{j}$ and the vector $\overrightarrow{pv}$. The convex angle $\alpha$ between $\overrightarrow{j}$ and $\overrightarrow{pv}$ is considered if the x-coordinate of $v$ is negative, otherwise the angle obtained by adding up $\pi$ to the supplementary of $\alpha$ is considered. Consider Fig. 1 where the Voronoi neighbours of $A$ are numbered according to their counter clockwise ordering. Note that the predecessor-successor $x$ of each neighbour $v$ in the counter clockwise ordering is the vertex of a Delaunay triangle whose vertexes are $v$, $x$ and $A$. Nevertheless, this simple argument does not work at the border of the $DVE$. Consider for instance peer $B$ in Fig. 1. Even if $B_3$ and $B_1$ are consecutive in the circular counter clockwise ordering, they do not define a valid Delaunay triangle together with $B$. To check if two consecutive neighbours $v_i$ and $v_j$ of $n$ define a Delaunay triangle together with $n$, the algorithm exploits the function $\text{DelaunayTriangle}(n, v_i, v_j)$ which returns whether the three input vertexes define a Delaunay triangle. Such a function exploits the Delaunay condition which states that the circumcircle of any Delaunay triangle must not contain any other vertex of the triangulation. Nevertheless, we define the following simpler conditions. The nodes $n$, $v_i$ and $v_j$ define a Delaunay triangle iff the following conditions hold: (i) the triangle defined by vertexes $v_i$, $v_j$ and $n$ does not include any further neighbour of $n$ and (ii) the straight line which connects $v_i$ and $v_j$ does not intersect any Delaunay edge connecting $n$ to one of its neighbours. Consider the two consecutive neighbours $B_1$ and $B_3$ of $B$. The triangle $\overline{BB_1B_3}$ is not a Delaunay triangle because $B_2$, another neighbour of $B$, belongs to it and the first condition does not hold. Let us consider now $A_3$ in Fig. 1. Its neighbours $A_2$ and $A_4$ are consecutive with respect to the counter clockwise ordering but the $\overline{A_2A_3A_4}$ is not a Delaunay triangle, because the edge $\overline{A_2A_3}$ intersect the edge $\overline{A_3A_4}$ and the first condition does not hold. Note that both these scenarios always occur at the border of the $DVE$.

The Children Detection phase exploits the function $\text{SpanningTreeChildren}(r, n, i)$ defined in Fig. 2 to detect the children of a node $n$ in the spanning tree rooted at $r$. The function receives in input the coordinates of the root $r$ of the spanning tree, those of $n$ and the index $i$ of the $i$-th Voronoi neighbor of $n$ according to the counter-clockwise ordering and returns true iff the $i$-th Voronoi Neighbor of $n$ is a child of $n$ in the spanning tree rooted at $r$. Since [8] proves that in compass routing the distance from the target decreases at each step, in our case the distance from the root should increase at each step, since compass routing is reversed. For this reason $\text{SpanningTreeChildren}(r, n, i)$ first checks if $v_i$ is farther from the root of the spanning tree with respect to $n$. If this is not true, $v_i$ cannot be a children of $n$ and the function returns a false value, otherwise the function executes the angle evaluation phase. The function $\text{DelaunayTriangle}(n, v_i, v_j)$, where $v_j$ is the predecessor or the successor of $v_i$ in the counter-clockwise or-

```plaintext
SpanningTreeChildren(r, n, i):
if dist(vi, r) < dist(n, r) return false
else
    if DelaunayTriangle(n, vi, vi+1) and DelaunayTriangle(n, vi, vi-1)
        if $\angle nvr < \angle vi+1vr$ and $\angle nvr < \angle vi-1vr$
            return true
        else return false
    else if DelaunayTriangle(n, vi, vi+1)
        if $\angle nvr < \angle vi+1vr$ return true
        else return false
    else if DelaunayTriangle(n, vi, vi-1)
        if $\angle nvr < \angle vi-1vr$ return true
        else return false
```

Figure 2. Spanning Tree Children Detection
dering, checks if the triangle defined by the three nodes belongs to the Delaunay triangulation by exploiting the previous conditions. For each valid triangle, the angle evaluation phase previously described is executed. The construction of the spanning tree stops when a node cannot find among its neighbors a child in the tree.

4. ROUTING IN CONSTRAINED AREAS

In the previous section we have described a compass routing based strategy to compute a spanning tree covering all the nodes belonging to the DVE. On the other hand, since our goal is the definition of an AOI cast mechanism, the spanning tree should include a subset of the Delaunay nodes of the DVE, i.e. the nodes corresponding to the peers located in the AOI of the root peer. This implies that compass routing may require to step out the AOI in order to build a spanning tree covering all its peers. It is interesting to evaluate the number of external peers which should be visited, because each of these links implies performing a routing hop so that a further latency in the delivery of heartbeats is introduced. Let $D$ be a DVE including a set of nodes. If we consider a subregion $A$ of $D$ we can define $G(A)$ as the graph comprising the nodes of the $D$ that belong to $A$ and the subset of Delaunay links whose end points both belong to $A$. Note that, in the general case, $G(A)$ may be not connected or it may not be a Delaunay triangulation, because the convex hull of the nodes may include links not belonging to $G(A)$. For these reason, when compass routing is exploited to define a spanning tree rooted at one of the nodes in $A$ and covering all the nodes in $A$, it should consider, in the general case, a set of nodes and links not belonging to $G(A)$. Consider, for instance, Fig.3, where $A$ is the rectangular region corresponding to the Area of Interest of peer $a$. Note that $G(A)$ is not connected, hence no path between $a$ and $c$ includes only nodes in $G(A)$. Therefore, compass routing should consider node $b$ which is located outside $G(A)$ to compute a path from $a$ to $c$.

The following theorem show that compass routing is able to build a spanning tree including all and only the nodes of the AOI when the AOI has a circular shape. Hence, no hop outside the AOI is required in this case and latency is not increased. The first result shows that the graph $G(A)$ is always connected when $A$ is a circular region.

**Theorem 1** Let $D(R)$ be a Delaunay triangulation defined on a set of nodes belonging to the 2-dimensional space $R$. If $A$ is a circular shaped subregion of $R$, then $G(A)$ is connected.

Even if the $G(A)$ is connected, compass routing may require to consider some Delaunay links not belonging to $G(A)$ to build the spanning tree. The following theorem shows that this is not necessary if a circular region is considered.

**Theorem 2** Let $D(R)$ be a Delaunay Triangulation defined by a set $S$ of sites belonging to a 2D space $R$. If $A$ is a circular shaped subregion of $R$ centered on the node $s \in S$, compass routing is able to compute a spanning tree rooted at $s$ and including all and only the sites of $S \subseteq A$.

The previous theorems have been proved in [20, 10]. Previous results guarantee that the algorithm introduced in Sect.1 does not require to step out the AOI to build the spanning tree. Furthermore, the last theorem suggests that any peer belonging to the AOI of a peer $P$ should consider, in the angle evaluation phase of the spanning tree construction, its Voronoi neighbors belonging to the AOI of $P$ only. As a matter of fact, peers located outside the AOI cannot belong to the spanning tree and should not be considered. The theorems are not valid for AOIs of different shape. For instance, if rectangular or squared areas are considered, some paths of the spanning tree may zigzag around the borders of the considered region.

5. TOLERANCE BASED ROUTING

The definition of a routing algorithm for DVEs must take into account the inconsistencies which may arise because of the movement of the peers. As a matter of fact two peers may have a different perception of the position of a common neighbor, due to the delay of heartbeat notifications. This implies that these peers may perceive a positional drift with respect to the real position of their common neighbor.

Consider, for instance, the scenario shown in Fig. 4 where the upper triangulation corresponds to the local view of the peer $B$ while the lower one to the local view of $C$. Suppose that $B$ and $C$ receive an heartbeat from their common neighbor $R$, they both neglect the propagation the heartbeat to $A$ because of their different view of the DVE. Due to the positional drift, each peer supposes that the other one should propagate the heartbeat to $R$. As a matter of fact,
compass routing at peer $B$ decides to neglect the propagation of the heartbeat because the slope of the segment $AC$ with respect to the segment $AR$ is smaller than that of the segment $AB$, the other way round for peer $C$. The positional drift may also generate redundant notifications, because $B$ and $C$ may decide to propagate the same heartbeat to their common neighbor. This scenario occurs when the local views in Fig.4 are inverted. Note that these problems are introduced by the highly dynamic nature of the $DVE$. It is worth noticing that the first problem is more serious, since it may lead to the overlay partition. We have modified the algorithm of the previous section to reduce the number of the peers which do not receive an heartbeat. Our strategy is to define a constant network wide tolerance threshold so that a peer states that one of its Voronoi neighbours is its child in the spanning tree whenever the difference between the angles considered by compass routing is lower than the tolerance threshold. The resulting algorithm will be referred in the following as tolerance based compass routing. Note that in this case an heartbeat may be notified to a peer by more than one neighbour. For instance, in Fig.4, both $B$ and $C$ should send the heartbeat to $A$ if the difference of the angles is lower than the threshold. As a consequence, the resulting algorithm introduces a number of redundant messages. Anyway, in DVEs, it is better to send a larger number of messages, instead of having some peers that do not receive the heartbeat at all.

A further mechanism to reduce the probability of partitioning the overlay is applied when the $DVE$ is scarcely populated, i.e. the opposite scenario w.r.t. to crowding. In this case the $AOI$ of a peer $P$ may be empty but $P$ sends its heartbeat to its Voronoi neighbors anyway so that the topology of the Delaunay overlay is dynamically maintained. However, overlay disconnections may still arise because of the positional drifts or of unexpected peers crashes. To reduce the probability of this scenario, a $TTL$ is paired with each heartbeat and the algorithm is modified such that the propagation of the heartbeat goes on as long as the border of the $AOI$ is not reached or the value of the $TTL$ is $\neq 0$. Note that this implies that, if the $AOI$ of a peer is crowded, the heartbeat is propagated only inside it, otherwise if it is scarcely populated, the heartbeat is propagated at least $TTL$ hops away from its source. In the last case, the number of peers which receive the heartbeat is increased. Furthermore, the heartbeat mechanism can be exploited as a 'pass the word' approach in order to notify new neighbours to the source of the heartbeat.

6. EXPERIMENTAL RESULTS

The approach proposed in the previous sections has been evaluated using Peersim [13], a simulator that supports large scale simulations of P2P overlays. To highlight the behavior of our algorithm in two quite different situations, we have considered two mobility models, that is two different strategies to decide the movement of each peer at each simulation cycle. We have considered a Random Walk Model, where the movement of the peers is defined fully at random and a Battle Model whose goal is the evaluation of our algorithm in crowding scenarios. In the Battle Model, all the peers are initially positioned randomly in the $DVE$, afterward each peer moves toward a single battle area, which is an area centered at the target, that is a randomly generated point. A peer approaching the target may modify its direction with a low probability in order to simulate a behavior in a battlefield including a set of obstacles. Once the battle area is reached by all the peers of the $DVE$, each peer moves at random and, each time a new direction is generated, the peer randomly decides whether staying in the battlefield or leaving the battle area and returning to the home base. In the latter case, the peer chooses its home base as a new target and moves toward it. Since the peers are partitioned into two different teams, we have defined two different home bases located in different areas of the map. Note that this model is able to generate a high degree crowding scenario when all the peers of the $DVE$ are fighting in the battle area. Whenever all the peers return at the home base, they start moving in a random fashion and after a while, a new battle target is generated and the previous phases are iterated.

The goal of our experiments has been to measure the network consistency, that is the level of consistency computed by comparing the local views of the peers toward a global view of the $DVE$. To enable this comparison, the simulation builds, at each simulation cycle, a global Voronoi diagram including all the peers of the $DVE$. The simulator is able to build the global view of the $DVE$ because it knows the position of all the peers, at each simulation cycle. The consistency is defined as the ratio between the number of links which are present both in the global view and in the local view of the peers and the total number of links in the global view. Even if a typical $DVE$ may include millions of peers, a crowd of peer in a restricted area of the $DVE$ generally does not exceeds 100-200 peers. Since our goal

![Figure 4. Local View Inconsistencies](image-url)
is the analysis of crowding scenario, we have considered 200 peers. The value of the TTL has been fixed to 3 in all the experiments. All graphs shown in this section aggregate data by considering the average values computed for each 10 simulation cycles. The first set of experiments evaluates the network consistency at different simulation cycles, by considering different speeds of the peers. The considered speeds are between 1 and 6. The upper part of Figure 5 shows the network consistency with respect to the Random Walk Model where no tolerance is exploited in the compass routing algorithm, while the lower one reports the same results for the Battle Model. The curves corresponding to the different speeds are drawn with different colours. Both figures show that an increase of the speed of the peers corresponds to a decrease of the network consistency. The difference of consistency between the two models is more substantial when the speed is low. For instance, while the level of consistency for the Random Walk Model is above 80% when the speed of the peers is 2, the level of consistency for the same speed drops at 60% after the 250th simulation cycle for the Battle Model. As a matter of fact, after this cycle the peers tend to crowd while approaching the battle target and this implies a larger number of inconsistencies.

The goal of the second set of experiments is to evaluate the effect of the compass tolerance technique on the consistency of the overlay. The experiments consider 200 peers, TTL = 3 and the speed is fixed to 2. The tolerance angle ranges from 0 to 16 degrees. The upper part of Fig. 6 shows the network consistency when the Random Walk model is considered, while the lower part of the same figure reports the same results for the Battle mobility model. The network consistency is above 0.8, i.e. more than 80% of the links are correct in the Random Walk Model, even with a low tolerance angle. The tests show that the introduction of the tolerance angle in the compass routing is effective, because it improves network consistency in both models. This improvement is larger when the Battle Model scenario is considered, for instance Fig.6 shows that an improvement of about 20% may be obtained.

The third set of experiments evaluates the trade off between the DVE consistency and the number of messages required to implement AOI cast. The upper part of Fig. 7 reports the average number of messages received by a peer at each simulation cycle. As expected, the number of messages is larger when the tolerance degree of the algorithm increases. However, this increment is negligible in the Random Walk
Model, while it is more considerable in the Battle model when the tolerance angle is larger than 6 degrees. Anyway this tolerance is sufficient to guarantee a reasonable degree of consistency, as shown in Fig.6. In general, however, a compromise must be chosen. For instance, the number of redundant messages can be limited and the loss of heartbeat messages may be recovered by exploiting some dead reckoning strategy as a counter measure.

To evaluate our approach in a further scenario, we have exploited the mobility model [22] defined for Second Life [1]. The model is based in the definition of a set of hotspots corresponding to cities or to interesting locations placed within the DVE. Each avatar moves between the hotspots of the DVE. When the avatar reaches an hotspot it explores it for a span of time, afterwards it moves toward a new hotspot. This behaviour is defined by a set of three states for the avatars and a Markov chains defining the transition between them. In the halted state the avatar stay still. In the exploring state, the avatar explores the portion of the DVE close to its current position, while in the travelling state the avatar moves from one hotspot to another one. We have implemented in [6] the mobility model based on the Markov chain defined in [22] where the probability transition of the automaton are derived from a set of Second Life traces. Our simulation allows to choose both the initial configuration of the avatars on the map, the radius of the hotspots and the percentage of avatars which may be present within the hotspot. In the first experiment we have evaluated the degree of the consistency of the overlay by varying the radius of the hotspots. Note that a smaller radius corresponds to a larger degree of crowding because the same number of peers are distributed in a smaller area of the DVE. The results of the experiment are shown in Fig. 8 top, where 100 peers and 3 hotspots are considered. The experiment confirms that a larger degree of crowding implies a lower degree of overlay consistency. However, notice that the consistency ratio is over 0.9, which is a higher degree of consistency with respect to the complex battle model. This is due to the presence of a larger set of hotspot with respect to the single hotspot created by the complex battle model. This implies that the peers are evenly distributed among the hotspot and a smaller number of peers is present in each hotspot. Finally Fig. 8 bottom shows the consistency of the overlay in the Second Life model for different speeds of the avatars whose state is explore. It is worth noticing that the impact of the speed on the consistency of the overlay remains high also for the Second Life model.
7. CONCLUSIONS

This paper presents a routing algorithm for a P2P DVE which exploits the properties of Delaunay networks to minimize the traffic on the overlay. The basic algorithm has been modified in order to cope with inconsistencies arising in a DVE due to the network latency. Our algorithm minimizes the amount of information needed to implement compass routing and thus the number of messages exchanged through the overlay. For this reason, our approach is more scalable with respect to [12] that requires a larger amount of information. Furthermore our algorithm is, at the best of our knowledge, the first one which faces the problems introduced by highly dynamic nature of the DVE by introducing a tolerance threshold in the evaluation of the angles required by compass routing. A set of experimental results validates the effectiveness of our approach. We are currently evaluating the AOI cast on several realistic mobility models by considering widely diffused commercial DVE. The first one model the behaviour of the avatars in a typical battleground of World of Warcraft. Furthermore, we are exploiting a set of traces of a real multiplayer game to test the effectiveness of our approach. Finally, we are investigating a set of aggregation techniques to further reduce the traffic on the overlay network.

ACKNOWLEDGMENTS

The authors acknowledge the support of Project FP7-257438, Contrail: Open Computing Infrastructures for Elastic Services, and of Project FP6-033576 XtreemOS: Building and Promoting a Linux-based Operating System to Support Virtual Organizations for Next Generation Grids. The authors wish to thank Michele Albano, Antonio Quartulli, Davide Agrifoglio and Luca Medone who conducted part of the experiments.

REFERENCES