AOI-Cast in Distributed Virtual Environments: an Approach Based on Delay Tolerant Reverse Compass Routing

Laura Ricci\textsuperscript{1,3*}, Luca Genovali\textsuperscript{2}, Emanuele Carlini\textsuperscript{1,2}, Massimo Coppola\textsuperscript{1},

\textsuperscript{1}Istituto di Scienza e Tecnologie dell’Informazione “A. Faedo”, ISTI-CNR, Pisa
\textsuperscript{2}IMT, Institutions, Markets, Technology, Institute for Advanced Studies, Lucca
\textsuperscript{3}Department of Computer Science, University of Pisa

SUMMARY

This paper presents a novel AOI-cast algorithm for Distributed Virtual Environments targeted to Delaunay-based P2P overlays. The algorithm exploits the mathematical properties of the Delaunay Triangulations to build a spanning tree supporting the notification of the events generated by a peer to the other ones located in its Area of Interest (AOI). The spanning tree is computed by reversing compass routing, a routing algorithm proposed for geometric networks. Our approach presents a set of novel features. First of all, it requires only the knowledge of the neighbours of a peer, so that the amount of traffic load on the P2P overlay is minimized. Furthermore, we prove that, for circular shaped AOI, the algorithm builds a spanning tree covering all and only the peers of the AOI. Finally, our approach takes into account the possible inconsistencies among the local views of the peers due to the network latency by introducing a tolerance threshold in the reverse compass routing. We present a set of simulations considering both synthetic data and real data traces taken from the HoN multiplayer game which show the effectiveness of our proposal.

Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: Peer to Peer, Overlays, Distributed Virtual Environments, Delaunay, Voronoi, Multiplayer, Consistency

1. INTRODUCTION

Distributed Virtual Environments (DVE)\textsuperscript{[1]} such as military distributed simulations and massively multiplayer online games (MMOG), for instance World of Warcraft or Second Life, are currently gaining increasingly attention in the software market.

In a DVE, users located at \textit{geographically distributed} hosts interact within a \textit{virtual world} which is populated by user controlled \textit{avatars}. Each avatar moves within the DVE and may interact with other avatars and with the passive objects of the virtual world.

Current DVE are generally developed according to a \textit{client server architecture} where a single server is responsible both of the notifications of the avatar positional updates and of the management of the state which is modified due to clients interactions. According to this model, each client notifies any event to the central server, which, in turn, updates the state of the DVE and notifies the event to the interested clients. Furthermore, the server computes a meaningful ordering of the events. The main disadvantage of this solution lies into the low level of robustness and of scalability due to the presence of a single point of aggregation, which may easily saturate the bandwidth of the server.

*Correspondence to: Laura Ricci, Dept of Computer Science, University of Pisa, e-mail: ricci@di.unipi.it
†Please ensure that you use the most up to date class file, available from the CPE Home Page at http://www3.interscience.wiley.com/journal/117946197/group/home/home.html

Copyright © 2010 John Wiley & Sons, Ltd.

Prepared using \texttt{cpeauth.cls} [Version: 2010/05/13 v3.00]
Even if the client-server model is still widely adopted to support DVE applications, the definition of a distributed computational model is mandatory to overcome the low scalability of client server architectures. In this direction, several approaches based on the P2P model [2, 3, 4, 5, 6, 7, 8, 9] have recently been proposed. However, the definition of a fully distributed architecture for DVE is still a research challenge, due to the need of integrating networks, graphics and artificial intelligence.

In distributed architectures, the definition of a scalable communication support is an essential element. The concept of Area of Interest (AOI), a surrounding area centred at the peer position that contains all the relevant entities for the peer, has been introduced to reduce network traffic on P2P-based overlays for DVEs. In a typical DVE, each peer periodically sends a heartbeat, i.e. a message notifying its position to any other peer in its AOI. The concept of AOI allows to decrease the number of out-of-scope message in the network since each event is notified only to the interested peers.

Implementing the AOI requires the definition of a highly dynamic P2P overlay that implies peers to dynamically define a set of connections with all (or a subset of) the peers located in its AOI. The overlay must adapt to the movement of the peers, which can enter and leave the AOI over time. Furthermore, a set of mechanisms should be defined to guarantee the connectivity of the overlay when the DVE is scarcely populated, i.e. the AOI of the peers are empty.

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 of them, for instance the closer ones. The former scenario may present problems in practice because of the huge number of connections which may be required due to hotspots, i.e. regions of the DVE where peers usually crowd. Crowding often occurs in a DVE, for instance in player vs. player (PVP) MMOGs, where a large amount of avatars may meet in a DVE region to fight a virtual battle. In this situation the speed of the avatars is generally high, hence their interaction pattern changes very frequently.

To face the problem of the high number of connections due to crowding, an alternative solution is generally adopted where a peer sends an event to its neighbours and these forward the event to the remaining peers. The simplest forwarding approach is based on flooding, i.e. each peer receiving a notification should propagate it to all its neighbors. This approach generates a large amount of redundant messages and presents evident scalability problems. A more refined solution is based on the definition of an AOI-cast[10, 3, 11, 7] mechanism, i.e. an application level multicast constrained within the boundaries of the AOI that supports the forwarding phase.

The definition of a proper AOI-cast mechanism depends to a large extent on the topology of the P2P overlay. Some recent proposals [12, 5, 7] have discussed the benefits of defining an overlay where the P2P connections correspond to the links of a Delaunay Triangulation generated by considering the locations of avatars of the DVE. According to this proposal, each peer is paired with a site of a Voronoi diagram [13] defined on the virtual space and the position of the peer is exploited to define the space partition. In this way, the area corresponding to a peer P includes all the points of the DVE which are closer to P with respect to any other. The Delaunay Triangulation corresponding to the Voronoi tessellation defines the P2P overlay connecting the peers. The adoption of this solution presents relevant advantages:

- **Bandwidth Saving**: since each site of a Voronoi tessellation has on the average 6 neighbor [13], each peer manages a bounded number of connections with other peers, i.e. those corresponding to the Delaunay links.
- **Overlay Connectivity**: the connections corresponding to the Delaunay links guarantee that the overlay is connected. Even if a peer is located in a uninhabited region of the virtual world, it remains connected with the rest of the DVE through the Delaunay connections.
- **Mapping of Passive Objects to the Peers**: since each point of the DVE is mapped to a single Voronoi region, a straightforward mapping of passive objects to the peers may be defined. This mapping assigns each passive object to the peer which manages the Voronoi region where the object is located.
- **Existence of Routing Algorithms for Delaunay Networks**: compass routing is based on a fast-to-compute angle argument which exploits the mathematical properties of geometric networks and has been proved to be cycle free for Delaunay networks. The algorithm can be exploited to define efficient AOI-cast mechanisms.
Even if some algorithms for the distributed maintenance of the Delaunay network have been recently proposed [12, 7], only a few proposals (like [14]) investigate the possibility of exploiting compass routing to define an AOI-cast mechanism. However, these algorithms generate large traffic load, because they require non local information, i.e. the knowledge of the neighbours of the neighbour of a node.

By comparison, this paper proposes a \textit{AOI cast algorithm} which exploits the properties of the Delaunay Triangulation to build a spanning tree that includes all the peers of the AOI. AOI-cast is defined by reversing compass routing [15] and requires that each peer knows only its neighbours, so that the amount of the traffic on the P2P overlay due to topology maintenance is minimized.

We first present the basic algorithm that 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. Several additional inconsistencies may rise due to network delays, messages loss, or abrupt peer failure that often occurs in typical P2P environments. A certain amount of replication is required to avoid irrecoverable situations due to these inconsistencies. To this end, a certain degree of replication is introduced by \textit{AOI-cast} to guarantee a high event delivery probability and a low probability of a partitioning of the overlay. \textit{Tolerance Based AOI-cast} introduces a tunable degree of replication in the delivery of messages in order to guarantee an acceptable level of consistency.

Tolerance based AOI-cast may be exploited to define a flexible strategy for spreading events within the AOI of a peer. In our approach, each peer may define direct links with a subset of the peers in its AOI and exploit AOI-cast to propagate the events to the other ones. The amount of direct links defined by a peer may be tuned according to the bandwidth of the peer/the density of peers in its AOI.

The paper is structured as follows. Sect. 2 describes and comments the proposals recently presented that are more related to our approach. The basic mathematical concepts of Voronoi Tessellations and Delaunay Networks are introduced in Sect. 3. The proposed AOI-cast algorithm is defined in Sect. 4 and refined in Sect. 5 that discusses the problem of inconsistencies arising in a DVE due to its highly dynamic nature. Experimental results are presented in Sect. 6. Finally Sect. 7 concludes the paper.

2. RELATED WORK

Voronoi diagrams and Delauney-based overlays are a well know solution to maintain network topology for DVEs. Several works exploit these structures because of their ability to fit with DVE requirements.

One of the first works along this line is VON [7]. VON exploits a Voronoi division of the DVE in order to manage event dissemination in a scalable manner. VON defines an overlay such that each peer maintains a direct connection with all the peers within its AOI. In order to maintain overlay connectivity, each peer also has a direct link with peers that may also be outside of the AOI. To reduce bandwidth consumption, VON has been further upgraded with an enhanced event dissemination system [16] and state management [17].

A recent approach, VoroGame [18], proposes an hybrid architecture for the management of passive objects. Their architecture is composed by the combination of a Voronoi-based network and a Distributed Hash Table (DHT). Two different peers, one for each overlay, are responsible for each passive object in the DVE. Voronoi nodes are responsible and maintain a copy for any of the objects that are in their Voronoi area. They also maintain, for each of these objects, a list of peers that have to be notified for a state change of the given object. This list is periodically sent to the corresponding DHT node, whose task is to broadcast state updates.

The work in [5] proposes a solution to deal with cluster of players in Delauney-based topologies. They employ a flooding messaging strategy to spread notification inside the AOI. However, when a peer detects the message rate to exceed its maximum capacity, it triggers a procedure for cluster management. This procedure logically collapses the cluster to a single point, allowing communications to temporarily skip many small neighbouring Voronoi regions, which helps reduce
the communication overhead. The approach has been proved effective with realistic movement traces from Second Life.

2.1. Event Delivery in Constrained Areas

Event delivery in constrained areas (which we refer to as AOI-cast) can be considered as a particular form of application-level multicast [16]. In the context of DVEs, AOI-cast is used to notify events from a source to the interested set of peers.

Essentially, there are two differences between typical application-level multicast and AOI-cast. First, AOI-cast has its scope in a well defined portion of the virtual environment, such as an AOI or a sub-region of the world. Second, in a typical application-level multicast, group membership is static or semi-static. Conversely, AOI-cast membership is highly dynamic since the multicast groups are based on the spatial position of the avatars, which changes frequently.

A classification of AOI-cast mechanisms (originally presented in [16]) can be done according to the delivery strategies used for event dissemination. The most straightforward and easy-to-implement strategy is to exploit direct connections from the source of the event to the recipient. Several approaches like Solipsis [1] and VON [7] and Colyseus [19] use direct connections to notify events. This strategy assures low values for messages latency, since each recipient is always one hop away from the source. However, as the number of recipient nodes grows, this method may oversaturate the bandwidth capability of the source.

To overcome these limitations, various forwarding approaches have been proposed for AOI-cast implementation. Forwarding-based solutions use subset of the peers as relay, whose task is to forward events to other possible interested recipient. These solutions have usually high scalability, since the necessary bandwidth to deliver events is split among a number of nodes. On the other hand, forwarding-based solutions may increase latency since event source and recipient may be separated by multiple hops.

Forwarding approaches can be classified according to the degree of structuredness of the connections. Structured approaches normally make use of tree-like structures to build the AOI-cast paths. One of the most relevant examples of structured multicast applied to DVE is SimMud [20]. SimMud divides the world in regions, and each region has assigned a coordinator that acts as the root of a multicast tree for the region. To create the multicast paths, SimMud exploits Scribe [21], a well know approach to build multicast infrastructures over structured P2P networks. Peers generate events and notify them to their region coordinator, which in turn forward them along the multicast tree. Even if from a structural point of view Scribe is able to manage dynamic membership and large groups, a potential problem is the latency of messages. In fact, the number of hops and the length of the paths may dramatically increase the latency. Also, it has been pointed out in [22] that application-level multicast may saturate the capability of nodes in presence of heterogeneous bandwidth capability, which is the case in wide distributed DVE applications.

Structure-less approaches differ from structured ones since they do not maintain any long-term structure to store multicast paths. Instead, they often employ local decisions in order to forward events. Compared with structured approach, these mechanisms yield two relevant advantages. First, they have no overhead for peer churn, since they work without any long term and synchronized structure. Second, only local peer information are exploited to forward messages.

One of the first solutions based on structure-less AOI-cast approach is APOLO [23]. Each peer divides its space of interest into quadrants and maintains a link to the closest neighbour in each quadrant. In order to notify an event, a peer sends the message to these four neighbours, which in turn recursively forward the message until it has reached all the possible interested peers. This solution strictly bounds the number of outgoing connections per peer, nevertheless, it may dramatically increase the number of hop and the bandwidth consumption in case of crowded situations.

A later approach, VON-forwarding [24], divides the space according to a Voronoi diagram. Each peer broadcasts a message to all its Voronoi neighbours in order to notify the peer in its AOI. This solution exploits that, on average, a peer in a Voronoi diagram has six neighbours. Compared with the direct link approach, VON-forwarding helps reduce the number of messages per event.
sent by peers. Compared with APOLO, the number of hops decreases due to the wider degree of the AOI-cast tree. In spite of that, the bandwidth use is not efficient due to the elevated number of messages replication in the network. This model has been subsequently refined with VoroCast and FiboCast [16]. VoroCast builds a multicast spanning tree using the underlying Delauney network and sends the notifications of events along the edges of this tree. FiboCast is a further optimization of VoroCast. It models messages frequency rates using the Fibonacci sequence, in a way that farthest nodes from the source receive updates less frequently than nearby nodes. The main disadvantage of these systems is the fact that they require non-local information to correctly forward messages. In particular they need to know the neighbours of the neighbour of a node, and since it depends on the position of the peers, this information has to be updated frequently. This may cause an increasing of bandwidth consumption, especially in crowded situation, where the Voronoi diagram change rapidly. Another aspect to consider is that VoroCast and FiboCast do not take into account the effects of the latency when considering the position of the peers (i.e. the positional drift). Due to this reasons, delivered messages may be duplicated and travel along path that are longer than necessary.

In our solution we propose a Delauney-based AOI-cast that copes with these two drawbacks. First, we employ a forwarding schema based on compass routing that exploits only information local to peers, i.e. theirs one-hop neighbours. This avoids the extra-usage of bandwidth for maintaining the n-hop neighbours, which is typical in approaches like VoroCast. Second, our solution takes into account the latency in information diffusion, by considering the possible positional drift occurred to the peer when computing AOI-cast paths. This allows to reduce messages redundancy and decrease the probability of message losses.

3. DELAUNAY TRIANGULATIONS: BASIC CONCEPTS

A Voronoi diagram, [13] also referred as Voronoi tessellation, is a special kind of decomposition of a metric space determined by the distances of the points of the space to a specified discrete set of objects in the space, i.e. the sites. Let us denote the Euclidean distance between two points \( p \) and \( q \) by \( \text{dist}(p,q) \).

**Definition 1**

Let \( S = \{s_1, s_2, ..., s_n\} \) be a set of \( n \) distinct points in the plane, i.e. the sites. The **VoronoiDiagram** of \( S \) is a partition of the plane into \( n \) cells, one for each site in \( S \), such that the point \( q \) belongs to the cell corresponding to a site \( s_i \) if and only if \( \text{dist}(q, s_i) < \text{dist}(q, s_j) \) \( \forall s_j \in S, i \neq j \).

Two sites are **Voronoi neighbours** iff their cells share a common edge.

**Definition 2**

A **Delaunay triangulation** \( Dt(S) \) for a set \( S \) of sites in the plane is a triangulation, i.e. a partition of the plane into a set of triangles, such that the circumcircle of each triangle \( T \) in \( Dt(S) \) is empty, i.e. it does not include any other point in \( S \) besides the vertexes of \( T \).

Given a set of \( n \) sites \( S = \{s_1, s_2, ..., s_n\} \) of the plane, the **Delaunay triangulation** is the dual structure of the Voronoi diagram, where the sites correspond to the vertexes of the triangles, and an edge of a triangle connects two vertexes \( s_1, s_2 \) if and only if their Voronoi cells share a common edge, i.e. \( s_1 \) and \( s_2 \) are Voronoi neighbours.

Fig.1 shows a **Delaunay Triangulation** on the top of a **Voronoi diagram**, where the borders of the Voronoi regions are shown by dotted lines and the corresponding **Delaunay Triangulation** links are shown by continuous lines.

The properties of **Delaunay Triangulations** has been exploited to define **compass routing** [15, 25], an efficient routing algorithm originally defined for geometric networks which minimizes the information required at each routing step. Compass routing is based upon the following observation. Consider a connected graph \( G \) and assume of being located at a node \( n \) of \( G \) with the goal to reach a destination node \( d \). [15] shows that the best strategy looks at the edges incident in \( n \) and chooses the
edge whose slope is minimal with respect to the segment connecting \( n \) and \( d \). Consider, for instance, Fig. 2 and suppose to be located at \( A \) with target \( R \). The best way to reach the target \( R \) is to pass through \( B \), because the angle \( \angle RAB \) is smaller than the \( \angle RAC \). [15] shows that while compass routing is not cycle free for general graphs, it can always find a finite path between two nodes of a Delaunay Triangulation.

The original formulation of Compass Routing makes it possible to discover a path from a node \( n \) toward a root node \( r \). [25] suggests to exploit compass routing to implement multicast on a Delaunay Overlay. The basic idea is to define a Spanning Tree by reversing the path from any node to the root of the spanning tree computed by compass routing. It is worth noticing that although [25] suggests the basic idea for the definition of compass-based multicast, it does not present any algorithmic solution.

A naive algorithm implementing compass-based AOI-cast, requires a node \( n \) to know its Delaunay neighbors as well those that are distant two hops in the Delaunay graph. As a matter of fact, \( n \) is the parent of a node \( v \) in the tree iff \( v \) chooses \( n \) as its next-hop neighbour toward the root of the AOI-cast. This implies that \( n \) must be aware of the positions of all the neighbors of \( v \) to compute the reverse path.

In the next section, we propose an optimized algorithm requiring at each node the knowledge of its Delaunay neighbours only. In this way the information required at each node is reduced to its local view of the overlay. The algorithm is suitable for a fully distributed P2P environment where a node cannot make any assumption about the structure of the entire overlay, since the information available at each node is limited. The algorithm computes a multicast spanning tree whose root is the node which generates the event, and the tree includes all other peers belonging to its AOI. The spanning tree is exploited by AOI-cast to propagate events within the AOI.

### 4. AOI-CAST BY REVERSE COMPASS ROUTING

In the previous sections we have presented the mathematical properties of Delaunay networks and we have shown the advantages of defining a Delaunay based P2P overlay. We have also discussed that the feasibility of this approach depends on the definition of a proper AOI-cast algorithm.

This section first describes an algorithm to build a spanning tree on a generic Delaunay network, then we discuss the behaviour of the algorithm when a constrained area including a subset of the sites is considered.

The example shown in Fig. 3 is exploited to describe the algorithm. \( \text{Root} \), the peer which is the root of the spanning tree, generates an event, for instance an heartbeat, which is propagated through the Delaunay links. Suppose that peer \( A \) receives that event from one of its Delaunay neighbours, \( A \) must choose its children in the spanning tree to propagate it. The algorithm executed at \( A \) checks, for instance, if \( A_5 \) is a child of \( A \) by evaluating whether \( A_5 \) would have chosen \( A \) in its path toward \( \text{Root} \). This would happen if \( \angle \text{Root}A_5A \) is smaller than \( \angle \text{Root}A_5A_1 \) and \( \angle \text{Root}A_5A_4 \). Note that to check this condition \( A \) only needs to know the further vertexes of the triangles sharing the edge \( AA_5 \), i.e. \( A_4 \) and \( A_1 \), which are Delaunay neighbours of \( A \) as well. Let us discuss this point in...
Figure 3. Spanning Tree Construction

more details, since this is the key point of our algorithm because it allows to reduce the information required to the neighbours of a node. Let us consider the line $l$ connecting $A_5$ and the Root. The point is that if compass routing executed at $A_5$ would have chosen a further neighbour, different from $A$, $A_1$, $A_4$, then the slope of the line connecting $A$ to $A_5$ with respect to $l$ would have been larger than the slope of the lines connecting $A_5$ to $A_1$ with respect to $l$. The same argument holds for the other common neighbour $A_4$. This implies that, if this happens, $A$ does not choose $A_5$ as its child. A formal definition is reported in [26].

In the general case, when a peer $p$ checks if $v$, one of its neighbours, is its child in the spanning tree, $p$ must detect the Delaunay triangles sharing the edge $pv$. The vertexes of these triangles are then exploited by the angle evaluation procedure. A straightforward procedure to detect the Delaunay triangles defined by the neighbours of $n$ is to sort its neighbours, for instance in counter-clockwise order. When considering internal links of the Delaunay triangulation, two successive neighbours of the counter-clockwise ordering define, together with $n$, a Delaunay triangle. This is not true when considering the borders of the region including the triangulation, i.e. when the links belong to the convex hull of the nodes.

Consider, for instance, Fig.3 where the neighbors of node $A$ are numbered according to the counter clockwise ordering, starting from the smallest one which is $A_1$. All the links connecting $A$ to its neighbours are internal to the Delaunay graph. Each pair of consecutive nodes in the counter clockwise ordering of the neighbours of $A$ define, together with $A$ a triangle of the Delaunay triangulation. Consider now node $B$ in Fig.3, even if $B_3$ and $B_1$ are consecutive in the counter-clockwise ordering of the neighbours of $B$, they do not define a triangle belonging to the Delaunay triangulation. This situation occurs because the links connecting $B$ to $B_3$, rs. $B$ to $B_1$ belongs to the convex hull of the nodes of the triangulation. This situation should be properly detected because $B$ should consider only the triangle $BB_2B_3$ when checking if $B_3$ is its child in the spanning tree.

The empty circle property [13] can be exploited to detect if a triangle belongs to the Delaunay triangulation. To reduce the computational overhead for the evaluation of this condition, we have defined two simpler conditions. A triangle defined by a pair of consecutive neighbours $n_i$ and $n_{i+1}$ of a node $n$ belongs to the Delaunay triangulation iff the following conditions are verified:

1. The link connecting $n_i$ and $n_{i+1}$ does not intersect one of the links connecting $n$ and a node between $n_i$ and $n_{i+1}$.
2. the triangle defined by vertexes $n$, $n_i$ and $n_{i+1}$ does not include a further node between $n_i$ and $n_{i+1}$.

For instance, in Fig. 4 the line connecting $B$ and $C$, which are consecutive in the counter clockwise ordering of $A$, intersects the link from $A$ to $D$. Note that $B$ and $C$ are consecutive in the counter-clockwise ordering, but they are not connected by a Delaunay link.
Fig. 5 refers to the second condition: $B$ and $C$ are consecutive in the counter-clockwise ordering of $A$, but the triangle $BAC$ includes the node $E$, hence the link connecting them is not a Delaunay link.

Note also that a simple test that checks if a pair of neighbors of a node are connected by a Delaunay link cannot replace previous conditions. Consider for instance Fig. 3. Even if node $B_1$ and $B_3$ are connected by a Delaunay link, $B$ should consider only node $B_2$ when evaluating whether node $B_1$ is its child in the tree. Note also that this implies that two neighbors of a node $n$ which are contiguous in the counter-clockwise ordering and connected by a Delaunay link, does not necessarily define a correct Delaunay triangle together with $n$.

Let us now describe the algorithm executed at each node in more details. The algorithm requires the sequential execution of the following steps:

- **Neighbors Sorting**
- **Children Detection**

In the **Neighbors Sorting** phase, $n$ sorts its Delaunay neighbors according to a counter-clockwise ordering which is then exploited in the **Children Detection** phase. **Neighbors Sorting** computes the neighbors of a node by exploiting a 2D coordinate system whose origin is at the node $n$ executing the algorithm and with unit vectors $\hat{i}$ and $\hat{j}$. The counter-clockwise ordering of the neighbors of $n$ is defined by considering the angles between $\hat{j}$ and the vector connecting $n$ to each of its Delaunay neighbors $v$.

The function $\text{compareneighbors}(n,a,b)$, shown in Fig. 6, takes as input the coordinates of a pair of neighbors $a$ and $b$ of node $n$ with respect to the defined coordinate system and returns the smallest one with respect to the ordering. The algorithm first checks if the neighbors have opposite x-coordinates and, in this case, it marks the neighbor with the negative x-coordinate as the smallest one. Otherwise, the convex angles $\alpha$, r.s. $\beta$ between $\hat{j}$ and $\vec{na}$, r.s. $\vec{nb}$ are computed. If the x-coordinate of both $a$ and $b$ are negative the smallest neighbor is the one corresponding to the smallest angle, the other way round if both $a$ and $b$ have a positive x-coordinate.

The **Children Detection** phase determines the children of a node $n$ in the spanning tree. This procedure implements reverse compass routing. The function $\text{SpanningTreeChildren}(r,n,i)$ described in Fig. 7 implements the Children Detection phase. The function takes as input the coordinates of the root of the spanning tree, those of $n$ and the index $i$, of the $i$-th Delaunay neighbor of $n$ according to the counter-clockwise ordering. The function returns $true$ iff the $i$-th Delaunay Neighbor of $n$ is a child of $n$ in the spanning tree rooted at $r$. Note that the indexing of the neighbours of $n$ has to be considered modulo the number of the neighbours.

[15] shows that the distance from the target decreases at each step of compass routing, 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 farthest 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
compareneighbours\((n, a, b)\):
\[
\begin{align*}
&\text{if } (a_x \times b_x) < 0 \\
&\quad \text{if } a_x < 0 \text{ return } a \text{ else return } b \\
&\text{else} \\
&\quad \alpha = \arccos(a_y) / \sqrt{a_x^2 + a_y^2} \\
&\quad \beta = \arccos(b_y) / \sqrt{b_x^2 + b_y^2} \\
&\quad \text{if } a_x < 0 \text{ and } b_x < 0 \\
&\quad \quad \text{if } \alpha < \beta \text{ return } a \text{ else return } b \\
&\quad \text{else} \\
&\quad \quad \text{if } \alpha < \beta \text{ return } b \text{ else return } a
\end{align*}
\]

Figure 6. Neighbors Comparison

\text{DelaunayTriangle}(n, v_i, v_j)$, where $v_j$ is the predecessor or the successor of $v_i$ in the counterclockwise ordering, checks if the triangle defined by the three nodes belongs to the Delaunay triangulation. To this end, the two condition introduced previously are exploited. For each valid triangle, the function executes the angle evaluation phase implementing reverse compass routing. The construction of the spanning tree stops when a node cannot find a child in the tree among its neighbors.

To summarize, the spanning tree construction requires the following steps. The root $r$ sends to its Delaunay neighbors a message including its coordinates. Each neighbor receiving the message determines its children in the spanning tree with respect to $r$ and forward them the message received from $r$. The procedure is recursively executed by each node receiving the message until the borders of the area including the sites is reached.

In conclusion, our approach requires that each node knows its own coordinates, a counterclockwise ordering of the coordinates of its Delaunay neighbors and those of the root of the spanning tree to determine its children in the tree. This minimizes the amount of information exchanged on the overlay and makes our approach more scalable with respect to [10].

The general algorithm we have presented so far is able to compute a spanning tree over a generic Delaunay overlay. Let us now take into account the behaviour of the algorithm on a restricted area. In this case our goal is the computation of a spanning tree involving the minimal number of nodes necessary to cover the AOI of a peer.

We first need some preliminary definitions. Consider a 2D area $R$ including a set $S$ of $n$ sites and let $Dt(S)$ be a Delaunay Triangulation of $S$ and $A$ be a subregion of $R$. In our case, $R$ corresponds to the AOI of a peer.

The general algorithm we have presented so far is able to compute a spanning tree over a generic Delaunay overlay. Let us now take into account the behaviour of the algorithm on a restricted area. In this case our goal is the computation of a spanning tree involving the minimal number of nodes necessary to cover the AOI of a peer.

We first need some preliminary definitions. Consider a 2D area $R$ including a set $S$ of $n$ sites and let $Dt(S)$ be a Delaunay Triangulation of $S$ and $A$ be a subregion of $R$. In our case, $R$ corresponds to the AOI of a peer.

\[
\text{SpanningTreeChildren}(r, n, i):
\begin{align*}
&\text{if } \text{dist}(v_i, r) < \text{dist}(n, r) \text{ return false} \\
&\text{else} \\
&\quad \text{if } \text{Delaunay_Triangle}(n, v_i, v_{i+1}) \text{ and } \text{Delaunay_Triangle}(n, v_i, v_{i-1}) \\
&\quad \quad \text{if } \angle v_i r v_{i+1} < \angle v_i r v_{i+1} r \text{ and } \angle v_i r v_{i-1} < \angle v_i r v_{i-1} r \\
&\quad \quad \text{return true} \\
&\quad \text{else return false} \\
&\text{else} \\
&\quad \text{if } \text{Delaunay_Triangle}(n, v_i, v_{i+1}) \\
&\quad \quad \text{if } \angle v_i r v_{i+1} < \angle v_i r v_{i+1} r \text{ return true} \\
&\quad \text{else return false} \\
&\quad \text{else if } \text{Delaunay_Triangle}(n, v_i, v_{i-1}) \\
&\quad \quad \text{if } \angle v_i r v_{i-1} < \angle v_i r v_{i-1} r \text{ return true} \\
&\quad \text{else return false}
\end{align*}
\]

Figure 7. Angle Evaluation Phase
to the entire DVE, and $A$ to the AOI of a peer. Let $G(A)$ be the subgraph of $Dt(S)$ induced by the sites of $S$ belonging to $A$.

We want to exploit compass routing to define a spanning tree whose root is a site in $G(A)$ and covering all the sites belonging to $G(A)$. In the general case, the construction of the restricted spanning tree should consider also nodes not belonging to $G(A)$. This happens in the following cases:

- $G(A)$ is not connected
- $G(A)$ is not a Delaunay triangulation

Consider, for instance, Fig. 8, where $A$ is a rectangular region defined upon the Delaunay Triangulation connecting all the sites of the DVE. For instance, site $a$ corresponds to a node of the DVE and the rectangular region to its Area of Interest. No path between $a$ and $c$ including only sites in $G(A)$ does exist. Therefore, the path from $a$ to $c$ should consider node $b$ which is located outside $G(A)$.

Consider now Fig. 9 which shows a circular region $A$ centered at the site $a$ which represents its AOI. $G(A)$ includes the nodes $a$, $b$, $d$, $e$, $f$, but not node $c$. Hence, the Delaunay edges represented by dashed lines do not belong to $G(A)$, since one of their end points, i.e. $c$, does not belong to the AOI. $G(A)$ is not a Delaunay Triangulation because it does not include the convex hull of its nodes.

The following theorem shows 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 and, as a consequence, 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 $S \in A$.

We have proven the previous theorems in [26]. Our previous results guarantee that AOI-cast does not require to step out the AOI to build the spanning tree when considering a circular region. Furthermore, the last theorem suggests that any peer belonging to the AOI of a node $P$ should consider, during the angle evaluation phase of the spanning tree construction, only its Delaunay neighbors which belong to the the AOI of $P$. 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.
4.1. Flexible AOI-Cast

This section shows that the bandwidth required for the notification of the events generated by a peer and the latency of their delivery can be balanced by a flexible use of the AOI-cast mechanism introduced in the previous section.

We have seen in the introduction that a solution entirely based on forwarding has the obvious drawback of the propagation latency, especially in crowding scenarios where several routing hops may be required to notify an event. In this case, the resulting latency may be not tolerable and may compromise the interactivity of the application.

On the other hand, if each event is directly sent from a peer to any other ones located in its AOI the latency is minimized but this solution may imply a large number of connections in crowding scenarios. Therefore, both previous solutions suffer problems in crowding scenarios.

The approach of dynamically enlarging or shrinking the size of the AOI according to the bandwidth of the peers [7] is not fair, because the size of the AOI depends upon the semantics of the application and players with a larger AOI could be favored. Since any proposal must handle crowding in a very scalable way to be eligible to support a real-time DVE, an alternative solution should be proposed.

To this end, we have proposed in [11] a flexible solution which defines direct connections between a peer \( P \) and a subset of the peers in its AOI, and exploits AOI-cast to propagate an heartbeat to any other peer in the AOI. The resulting overlay includes connections between a peer \( P \) and its Delaunay neighbours and a set of further connections between \( P \) and a subset of close peers in its Area of Interest. Note that some Delaunay neighbors of a peer may not belong to its Area of Interest, but these connections are required to guarantee that the overlay is connected. The main advantage of this approach is that the number of connections which must be managed by each peer is reduced with respect to the direct link solution.

The AOI of each peer \( P \) is thus partitioned into two areas, the Internal Area of Interest of \( P \), \( IAOI(P) \) and the Peripheral Area of Interests such that \( PAOI(P) = AOI(P) − IAOI(P) \). The P2P overlay includes, besides the Voronoi links, a link between \( P \) and any peer located in \( IAOI(P) \).

Fig. 10 shows the \( IAOI \) of a peer \( a \) which is delimited by the internal circumference, while its \( PAOI \) overlaps the *annulus* between the circumferences. The corresponding overlay includes direct links between \( a \), and \( b, c, d, e, f, h \), while AOI-cast is exploited to reach \( g \). Notice that since \( a \) and \( e \) are not Voronoi neighbors, the direct link connecting them is not a Voronoi link. Furthermore, it is worth noticing that \( a \) should send its heartbeats to its Delaunay neighbour \( i \) as well, even if it is located outside its area of interest so that \( i \) may act as ‘beacon node’ that informs \( a \) about peers approaching from farthest positions. Network connectivity is guaranteed by these nodes, even when the AOI of a peer is empty.

The peer \( P \) which generates an heartbeat sends it to all its Voronoi neighbors and to any peer in its \( IAOI \). The forwarding of the heartbeat is started by the peers located ‘at the border’ of its \( IAOI \), i.e. the boundary peers, then, any peer located in \( PAOI(P) \) carries on the forwarding by the AOI-cast mechanism, which is described in the next section.

Note that, in order to avoid network partitioning, a proper mechanism to guarantee that each peer correctly keeps its Delaunay neighbors is required. In a crowding scenario, peer belonging to the \( AOI(P) \) may notify \( P \) of other approaching peers that are supposed to become its Delaunay neighbours. In the opposite scenario where \( AOI(P) \) is empty a new peer becoming a Delaunay neighbor of \( P \) should be notified to \( P \) by one of the previous Delaunay neighbors of \( P \). To decrease the probability of event loss, a TTL may be exploited to guarantee that each position update is propagated at least \( k \) hops away from its source.

[11] reports a set of experiments evaluating the scalability of our approach. This result shows the feasibility of a trade off solution exploiting only Delaunay links in the general case and an Internal Area of Interest characterized by a small radius in crowding scenarios. As a matter of fact, while a small radius corresponds to a reasonable number of links, it may constrain the number of routing steps with respect to the solution including Voronoi links only.
5. TOLERANCE BASED AOI-CAST

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. 11 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.11 are inverted.

Note that these issues are introduced by the highly dynamic nature of the DVE. It is worth noticing that the problem of notification loss is more relevant, since it may lead to the overlay partition. We have modified the algorithm of the Sec.4 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 Delaunay 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.11, 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 overlay partitioning 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 heartbeats to its Voronoi neighbours 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 about the source of the heartbeat.

6. EXPERIMENTAL RESULTS

The approach proposed in the previous sections has been evaluated using Peersim [27], a simulator that supports large scale simulations of P2P overlays.

First of all, it is worth noticing that the effectiveness of our solution depends on the availability of an efficient support for the distributed maintenance of a Delaunay overlay. DVE applications are characterized by severe real-time constraints, since state consistency must be maintained as fresh as possible. The definition of a highly efficient support for the management of Delaunay Triangulations is therefore mandatory for our approach. We have exploited VAST[28], a library for the distributed management of Voronoi Diagrams. We have tested the performance of different functions of the VAST library in order to understand if they may be exploited in a DVE such that real time constraints are respected. Our results [26] show that VAST may support a Delaunay based overlay without loss off scalability up to 1000 nodes within the same AOI. Other environments for the distributed support of Delaunay triangulation [28, 2, 6, 29, 25] have been recently proposed and could be exploited as well.

To extensively evaluate the behaviour of our algorithm in different situations, we have considered a wide range of mobility patterns for avatars, that is the strategies to decide the movement of each peer at each simulation cycle.

- **Two fully synthetic models.** 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, afterwards 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 choices 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.

- **A model derived from commercial DVEs.** We have exploited the mobility model [30] defined for Second Life [31]. The model is based on 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 [9] the mobility model based on the Markov chain defined in [30] 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.

- **Real traces from a commercial DVE.** We have considered real traces from a Multiplayer Online Battle Arena (MOBA) game called Heroes Of Newerth [32]. Avatars in MOBAs are divided in two factions, that battle against each other to obtain control of several game objectives. Since the avatars movements are free across all the map, the strategies followed by avatars greatly diverge from each other, ranging from surprise ambushes to direct attacks. Usually there are two broadly distinct game phases. In the first, avatars try to obtain game objectives that are spread across all map. This reflects in the creation of numerous small-size hotspots. In the second phase, avatars battle for the final game objective. In this phase, avatars tend to gather in the same points of the map, creating few, but large, hotspots. This results in MOBA to have two different and attracting characteristics: (i) it is a fast paced game, where hotspots rise in different positions of the map, and (ii) the map is relatively small so the creation rate of hotspot is high. Unfortunately, MOBAs typically have a small number of players. To overcome this limitation we set up the simulation in order to run up to 4 traces on the same map simultaneously. This results in up to 40 players on a single and relatively small map, which also increases the size and creation rate of hotspots.

The goal of our experiments has been to measure two key metrics, the network consistency and the network load. We have defined network consistency 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 Delaunay triangulation 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. The network load is measured with the average number of message sent by each during a simulation cycle. This measure has been implemented by simply counting at each cycle the total number of messages that are present in the system, and dividing such amount for the number of peer in the simulation.

Our experimental evaluation focuses mostly on crowded scenarios. Even if a typical DVE may include thousands of peers, a crowd of peer in a restricted area of the DVE generally does not exceeds a few dozens of peers. Where not stated differently, we have used 200 peers for random-way point based models, 100 for the Second Life mobility model and 20 peers for MOBA traces. Please note that, along with a reduction in the peers number, the dimensions of the DVE maps is shrunk according to the different traces used, in order to keep the density of peer comparable in all the experiments.

Finally, the value of the TTL has been fixed to 3 in all the experiments, and all graphs shown in next sections aggregate data by considering the average values computed for 10 simulation cycles.
6.1. Synthetic Models

The first set of experiments aims to evaluate the network consistency at different simulation cycles, by considering different speeds of the peers. The considered speeds are between 1 and 6. Fig. 12 shows the network consistency with respect to the Random Walk Model where no tolerance is exploited in the compass routing algorithm, while Fig. 13 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. Fig. 14 shows the network consistency when the Random Walk model is considered, while Fig. 15 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. 15 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. Fig. 16 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. 14 and in Fig. 15. 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.

6.2. Second Life Model

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. 18, 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. 19 right 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.
6.3. MOBA traces

We first briefly describe the MOBA races used. To this end, the graph in Figure 20 shows, per each cycle, the average number of peers in avatars’ AOIs, which we define as \textit{AOI population}. This measure has been taken for all the four traces analysed, each of them composed by 500 cycles and 10 peers. All the traces broadly present the same characteristics. Each trace contains a minimum of AOI population around the 100th and 200th cycle, ranging from 0.35 of trace 2 to 0.5 of trace 1. This couples with the aforementioned first phase of the MOBAs, where avatars goal is to obtain objectives spread thorough the map. After this first phase, peers gathers themselves and the AOI population becomes stable on about 0.5 in all traces.

Figure 21 shows network overhead for different values of tolerance angles (\(ta\)). With no tolerance angle, the average messages per peer is around 15, whereas with \(ta \geq 0\) the network overhead increases. This happens since with no tolerance angle, the peers, even the closer ones, are not reached by the AOI-cast. However, a small increment in the tolerance angle implies an increment the number of messages. Moreover, further increasing of tolerance angle does not imply any increase of message, due to the small size of the map and the tendency of avatars to gather into clusters. Also, the network overhead follows the trend of the AOI population discussed above. In the first phase peers are more distant from each other, and consequently the number of messages drops down. However, when peers get closer, the number of messages increases and keeps a stable trend during the rest of the simulation.

Figure 22 shows aggregated data about consistency. Each point in the graph represents the average of the consistency in 10 different experimental setups, in which speed is varied from 1 to 10. This average has been plotted for various network sizes. As it is clear from the graph, with 10 and 20 nodes, the consistency always stays above 75%. On the other hand, with a size of 30 and 40, network
consistency drops down evidently, suggesting that up to 20 nodes can be managed concurrently in a MOBA scenario.

7. CONCLUSIONS

This paper presents an AOI-cast 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 [7] that requires a larger amount of information. Furthermore our algorithm is, to the best of our knowledge, the first one which faces the problems introduced by the highly dynamic nature of the DVE by introducing a tolerance threshold in the evaluation of the angles required by compass routing. We have developed a set of simulations to evaluate the effectiveness of our approach and we have considered both synthetic data and real traces generated by a commercial game. The experimental results validate the effectiveness of our approach.

ACKNOWLEDGMENTS

The authors acknowledge the support of Project FP7-257438, Contrail: Open Computing Infrastructures for Elastic Services, and of Project FP6-033576 XtremOS: 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 Mascitti, Davide Agrifoglio and Luca Medone who conducted part of the experiments. The authors wish to thank the reviewers for their accurate and useful suggestions, which helped us to improve the presentation.

REFERENCES