Integrating Centralized and P2P architectures to Support Interest Management in Distributed Virtual Environments

Emanuele Carlini
IMT, Lucca, Italy and
CNR-ISTI, Pisa, Italy
Email: emanuele.carlini@isti.cnr.it

Laura Ricci
University of Pisa
Pisa, Italy
Email: ricci@di.unipi.it

Massimo Coppola
Institute of Information Science
and Technologies CNR-ISTI, Pisa, Italy
Email: massimo.coppola@isti.cnr.it

***Abstract*—A fundamental problem for the development of P2P Distributed Virtual Environments is the definition of an overlay supporting interest management, i.e. determining all the entities of the virtual world that are relevant for a given peer. To this end, we propose a gossip-based approach that considers the coverage of the Area Of Interest of a peer as the guiding principle for the definition of the overlay and its maintenance. The resulting overlay provides a support for a best-effort resolution of interest management, so that it can be mostly supported through communications on the P2P overlay, with minimal intervention of a centralized entity. The paper presents a set of extensive simulations based on realistic mobility traces. The experimental results show the effectiveness of gossip for the construction of a best-effort Interest Management overlay.**

I. Introduction

Distributed Virtual Environments (DVEs), like massively multiplayer games or distributed training simulations, acquired lots of popularity in the last years from both commercial enterprises and research communities. The reason behind DVEs success is their ability to provide a shared sense of space to their users, whether they are players in a multiplayer game or soldiers in a military simulation. In this work we refer mostly to application like Multiplayer Online Games, as they are one of the leading sectors in the digital entertainment industry.

To enable the engaging experience typical for these applications, information about the entities in the virtual environment has to be replicated in users machines. This issue is referred to as *Interest Management* (IM) [1] and it can be abstracted using a publish-subscribe model [2]. Publishers perform actions (e.g. they move) and subscribers should receive the result of these actions (i.e. their local replica should change accordingly). In this work we consider *avatars* and *objects* as publishers. Avatars are the virtual representations of the users participating to the DVE. Objects are entities that are not directly controlled by human users, but that can possibly move and be interacted with (e.g. a computer controlled enemy). Through this paper we mostly examine movement as the action performed by avatar and objects. However, our reasoning can be extended to other types of actions.

Whenever an avatar (or an object) changes its position it generates an event that should be propagated to all other users in the DVE. In fact, IM poses well recognised scalability challenges, due to the number of participants and the fast-pace nature of the DVEs. A plethora of approaches have been proposed to efficiently support IM (see [3], [4] for a comprehensive list). One of the main strategies to address IM is to replicate only the entities in the visual/interaction area of players, which is called Area of Interest (AOI) and is typically a circular area whose center is the avatar position.

Many DVE architectures [5], [6], [7] strive to resolve IM with Peer-To-Peer (P2P) principles, in order to cope with the limited scalability and flexibility of centralized solutions. Several solutions (like [7]) exploit structured overlay, commonly referred to as Distributed Hash Tables (DHTs), to store the entities in the virtual environment. Peers perform range-queries on the DHT to subscribe to the resources in their AOI in order to receive an update when their state changes. These approaches offer guarantees in terms of availability (i.e. if an entity is stored in the DHT, it will be found) and embedded failure recovery mechanisms. However, range-queries have usually high latency due to the possibly large number of nodes that may contain relevant information. To address this limitation query-caching and pre-fetching mechanisms have been studied [7], [8]. Even if they work in principle, they introduce overhead in terms of complexity of the approach and make strong assumptions on the capability of peers.

Compared to structured approaches, unstructured P2P solutions focus on building the overlay according to the spatial proximity of the peers [9], [10]. In order to perform IM, each peer connects with a subset of its neighbours. In this way, neighbour peers may warn each other both about their movements and about new peers entering their AOI. These approaches naturally adapt to the rapid evolving scenarios of DVEs and allow the system for large scalability. However, when the number of peers in the AOI is very high due to hot spots, i.e. areas of the DVE of particular interest for the peers, a multi-hop communication using peers as relays is required. Even if multi-hop approach successfully bounds the number of connections per peer, it may incur in high latency when the number of hops becomes large. Unstructured networks also make complex the IM of the objects. A natural solution would be to assign objects to peers according spatial proximity. Even if several approaches adopt advanced version of this technique ([11], [12], [13]), this solution still requires a lot of objects migration among peers, which causes a high amount of overhead.

For different reasons, both structured and unstructured P2P overlays present weak points depending on some of their features. Several solutions strive to cope with these drawbacks, but even if they might work in principle, they introduce overhead in the IM mechanisms that may compromise the interactiveness of the application. Note that IM is a core and frequently executed component of a DVE so that its support must be simple as much as possible.

A. Contribution and Paper Structure

To avoid these drawbacks, we propose an IM approach that is lightweight, simple and scalable. To meet these requirements, our solution is based on a combination of a centralized server and a best-effort mechanism providing support for IM. With this combination we are able to reduce consistently the load on the central server. In the best-effort mechanism each peer exploits of local knowledge of its neighbours to discover as much entities as possible belonging to its AOI.

To achieve this task, we use a gossip-based protocol that allows each peer to connect to a relevant subset of its nearby peers. In order to select the relevant peers, our approach defines a spatial ranking function that, based on the local knowledge of each peer, selects the best neighbours according to the portion of the AOI they are able to cover.

We have tested our protocol by a set of traces derived from Second Life, one of the most popular DVE. The results we have obtained are encouraging, as, on the average, the 80% of entities in the AOI of a peer may be retrieved by exploiting our technique. Due to the best effort nature of our approach, a further mechanism should be paired with it, to support all the scenarios where it is not able to fully resolve IM. The best-effort nature of our approach pairs seamlessly with the hybrid DVEs architecture [14], [15] which combines centralized and P2P solutions to create effective and economic architectures for DVEs. For example, in the context of on-demand computing, our approach can be used to query servers only when necessary, thus to further reduce the economical cost of maintaining a DVE platform.

The paper is structured as follows. In Section II we summarize the related work in the field of IM for DVEs, and we underline the principal differences with our work. A detailed description of the overall mechanism is presented in Section III, whereas Section IV discuss and evaluate two different ranking functions. Section V discusses a selection of our extensive experimental evaluation that shows the encouraging results obtained by our approach. Finally, Section VI concludes the paper.

II. Related Work

Interest Management (IM) solutions proposed in the literature can be classified according to degree of separation in the roles played by the nodes of the infrastructure. On one hand there are *hierarchical* approaches, in which IM is realized through a set of (super)nodes with enhanced knowledge with respect to regular player's nodes. On the other hand, we have *flat* approaches, where there is no neat distinction between super and regular nodes.

Hierarchical approaches are essentially of two types: server-based and supernodes-based. Server-based approaches exploit full-blown servers with global knowledge of the virtual environment. They periodically inform players' nodes about relevant entities. This approach is naturally used by centralized and distributed server infrastructures [5], [6].

Supernodes-based solutions exploit free user-provided resources to realize IM. Similarly to server-based approaches, the VE is divided into disjoin regions and each region is paired with a supernode that periodically informs the peers about their neighbours. Several supernodes-based approaches have been proposed, as for example [16], [17], [12]. In order to cope with the unreliability that comes with user resources, these approaches have to perform a careful supernode selection, set up failure recovery mechanisms and provide adaptive mechanism for region sizing.

For their intrinsic nature, flat approaches pairs with DVE that are realized in pure P2P fashion. In this kind of solutions, neither supernodes or servers are considered to support IM. One (or more) *overlay* is constructed and maintained in order to deliver the communication for IM. This overlay can be either structured (in case of Distributed Hash Tables) or unstructured. Colyseus [7] uses a DHT to store the state of the entities. IM is realized by periodically performing multi-attribute range queries on the DHT to subscribe to relevant entities. Whenever these entities modify their state, all the subscribers are informed about the changes. Colyseus proposes two approaches in order to guarantee low latency of lookup queries. First, it exploits spatial and temporal locality in object movements in order to predict possible queries. This allows speculative pre-fetching of replicas. Second, it enables soft caching of recent queries, which makes it possible to immediately reply to a query.

The use of DHT yields concrete advantages, as it offers a stable and reliable platform for distributed indexing. However, in spite of optimizations like caching and pre-fetching, latency may still represent an issue. For this reasons we argue that DHT should be used as a backup or a backbone mechanism, and must be queried only when really necessary. Compared with DHTs, unstructured overlays give more emphasis to the dynamic grouping of peers. In fact, unstructured overlay are often built according virtual proximity of the avatars, by choosing the neighbours as a subset of closer peers. Various techniques have been proposed in order performs this choice. The most straightforward and easy-to-implement strategy is to build the overlay considering direct connections from all the closer peers [16], [18]. 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 has been proposed for IM. 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.

Other approaches [9], [10], [21], [22] exploit the position of the players to define a Voronoi partition of the virtual world. The Delaunay Triangulation corresponding to the Voronoi Diagram defines the overlay connections.

pSense, [23] maintains both a list of near nodes and a list of sensor nodes to manage the P2P overlay. The near nodes list contains peers that are within the vision range of the local node and supports IM. The local node attempts to keep its list as accurate as possible. The sensor node list contains nodes that are outside the AOI of a peer, but close to its borders. The sensor nodes of a node notify it of the presence of new nodes entering its AOI so to avoid network partitioning. Sensor nodes must be distributed as evenly as possible around a node to guarantee uniform connections with the rest of the world. A localized multicast is exploited to spread position updates to the peers belonging to the AOI.

III. Best-effort Overlays for IM

This section discusses the first core contribution of the paper, that is the gossip-based mechanism to build the IM overlay. We start by describing the system model and the assumptions in our approach, then we discuss the overlay construction with gossip mechanism,

A. System model

We consider a set of players participating to the same instance of a DVE. Each player has a virtual representation, called *avatar*, that can move around the virtual environment and interact with other avatars or with objects. Each avatar has associated an Area of Interest (AOI) that is enclosed by a circle whose center is the avatar position. Each player runs its own instance of the DVE on a proprietary machine (e.g. a desktop workstation) that we refers to as *node* or *peer*. We assume nodes to have ordinary network capabilities, such as the ability to send and receive messages from the Internet.

In order to resolve IM, each node is connected with two infrastructures. The first infrastructure is a centralized server, which maintains all the position of the entities in the virtual world. Nodes communicate their position to the server, which updates the information in its state. The server periodically communicates to each node the entities in its AOI. The interval between two consecutive message from the server, T_S, is common for all the nodes.

Besides the server, nodes are also connected to a custom overlay. Each node has associated a *profile* that contains the node network address (which is also used as unique identifier) and the position in the DVE. Nodes maintain a set of profiles that represents their partial *view* of the network. When a node n has another node m in its view, a connection between n and m exists in the overlay, but this does not imply that the reverse connection exists. Nodes periodically query the overlay to learn about the entities in their AOI.

By exploiting the overlay, it is possible to increase T_S and, as a consequence, reduce the load imposed to the server. The next section describes in details the construction and the effectiveness of the overlay.

B. The Interest Management Overlay

IM is a core DVE operation, since it must guarantee to a peer the complete knowledge of the entities it may interact with. These may include further peers, objects and computer-controlled entities.

In this paper, the construction of a best-effort overlay for Interest Management has been inspired by T-Man [24]. Similarly, the effectiveness of our approach depends on the definition of a set of proper ranking functions. In our case, these should favour neighbours which may offer a larger number of entities in the interest set of a peer. To this end, we will consider the spatial coverage of the AOI of a peer offered by its neighbours. However, unlike most existing T-Man-like approaches, our goal is to build a continuing evolving overlay rather than a predefined one. The view of a peer changes continuously in order to reflect the position updates of the peers in the virtual space. In our case, instead of evolving toward a predefined target topology, peer continuously gossip to each other to support the retrieval of new avatars and objects in their AOI.

Our technique to build an overlay supporting IM is based on the following reasoning. Let us consider a given peer P. At an arbitrary point in time it has in its local representation of the environment the replicas of the entities that belong to its AOI. When P moves, its AOI changes accordingly. Hence, to maintain its local representation up-to-date P must discover the new entities belonging to the new AOI. In order to dynamically acquire this information, P builds an overlay by considering a set relevant neighbours. The creation of the overlay poses two issues. First, P needs to know the identifier of its *candidate* neighbours; second, P needs a mechanism to discriminate among peers, in order to choose the more promising neighbours from the set of candidates.

The first issue is resolved continuously refreshing candidate knowledge by means of two gossip protocols. The first gossip protocol is a random peer sampling, enabling each peer to maintain a set of long range links that guarantee the connectivity of the overlay. These links are exploited in situations where a peer have few knowledge about its nearby candidate neighbours and must incrementally acquire new information. These situations include the bootstrap phase and when an avatar is ”teleported” form one place to another of the DVE. In the second gossip protocol each peer chooses its neighbour configuration by exploiting a ranking function based on spatial AOI coverage. Since the selection of the neighbours is done according the proximity, entities are progressively discarded by a peer when they disappear from its AOI. Note that the two protocols work at the same rate. For each gossip iteration, a instance of random peer sampling and one of coverage sampling are executed.

The second problem considers the AOI coverage offered by the neighbours of a peer. To this end, each peer should choose the best *configuration* of its neighbours in order to optimize the number of entities which may be retrieved from them. At each iteration the peer adapts its overlay neighbours set by providing a *partial order* from multiple configurations of neighbour sets. To maximize the area coverage, considering each peer one at time is simply not feasible. The choice of the best neighbour set is realized by means of multiple ranking functions, which are discussed in detail in the next section.

Since avatars are continuously moving, a large part of the IM performances depends on the freshness of peers knowledge. In order to maintain the selection of the neighbours as fresh as possible, each entry in the view of the peers is marked with a time-stamp. The time-stamp gives an estimation on the freshness of the entry. Our mechanism considers the age of the entries in two situation during the execution. First, before to rank the neighbour candidates, all the candidates whose age is greater than a certain threshold are not considered. Second, during the ranking, fresh configurations are favoured with respects to the stale ones. In principle, the internal clock of the peers can be used as the source for the time stamp. However, for simulation purposes, we model the time as a discrete successions of iterations. The simulation starts at iteration zero for all the nodes, and for each gossip-cycle iteration count is increased by one. When an entry is created, the iteration count is used as time-stamp for such entry.

From an application point of view, our approach builds a set of neighbours to query in order to possibly retrieve the most entities in a peer's AOI. The application level should query these peers to actually perform the retrieval. For example, the application level can periodically ask to the neighbour peers to pull fresh data. We do not provide further details on the this mechanism since is is very dependant on the application level requirements and it has little impact on the underlying gossip protocols.

C. Server Bandwidth Reduction

The main goal of the proposed mechanism is to reduce the bandwidth consumption of the server. To measure the reduction of outgoing bandwidth at the server, we have performed several tests by varying T_S, which is the distance in time between two consecutive communications from the server. We have considered networks with 200, 500, and 1000 peers. Results are presented in Figure 5.

As expected, the amount of outgoing data transfer is greatly reduced by increasing the time. With this reduction, the DVE operator is able to evaluate alternative choices regarding the deployment of the IM server. For instance, let us consider an operator willing to deploy the IM server on a on-demand platform. With 1000 nodes, and $T_s = 0.25$ the bandwidth

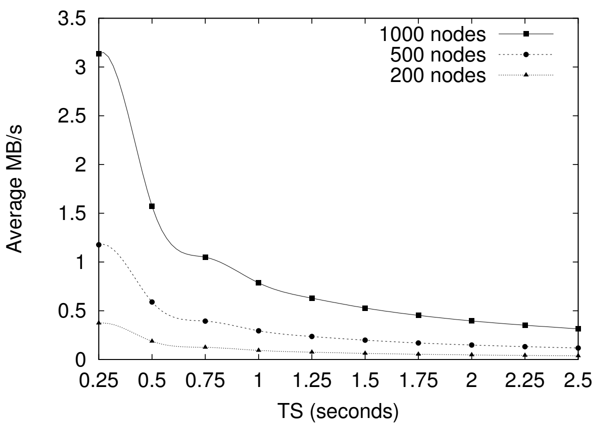


Fig. 1: Server outgoing bandwidth

requirements are 3MB/s. Using the prices of a current commercial on-demand platform [1], the deployment would cost 30$ per day only considering bandwidth. With $T_s = 1$ the cost would be reduced to 10$ per day.

IV. AOI Coverage Ranking Functions

In this section we explore in details the aspect of AOI coverage, which is the main principle behind our ranking functions. This aspect poses two distinct challenges: (i) to measure the amount of area covered by neighbours peers and (ii) to determine the best subset of neighbour peers that maximize the area coverage. In the rest of this section we formalize these two problems and we provide a description of the adopted solutions.

A. Measuring Coverage

Definition 1 (AOI coverage). *Given a set of AOIs $\mathscr{N} = \{N_1...N_n\}$ and an AOI P such that $P \notin \mathscr{N}$ we define as the coverage of P given $\mathscr{N}$, $c(P, \mathscr{N})$, as the area of P that is overlapped by the AOIs contained in $\mathscr{N}$.*

Computing $c(P, \mathscr{N})$ requires to compute all the unique intersections of AOIs in $\mathscr{N}$ with P's AOI and to evaluate their area. In trivial situations this is easy to compute, as it is just the sum of the intersection of the AOIs. However, in real situation, computing the AOI coverage is far from a trivial problem. For instance, in the case depicted by Figure 2 we show that $c(P, \mathscr{N}) = (P \cap B - P \cap B \cap A) + (P \cap B \cap A) + (P \cap A - P \cap B \cap A)$.

In complex situations, when many peers are close to each other, to compute the effective coverage may be prohibitively expensive in terms of computational effort. Practically, this happens for two reasons. First, the number of the intersections grows quadratically with the number of peers. Second, it might be computationally costly to evaluate the area of an intersection resulting from many AOIs. For this reasons, we approach this issue considering an approximation. The idea is to approximate the continuous surface of the AOI as a grid

[1]0.12$ per GB, Amazon EC2 prices, July 2012

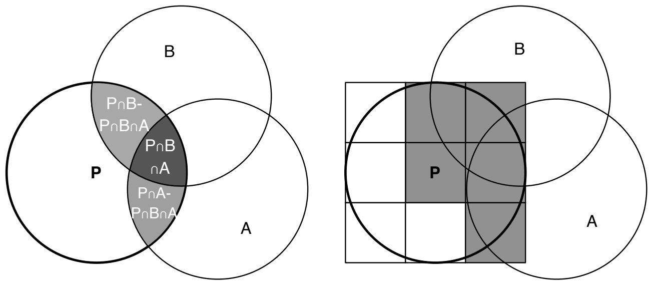


Fig. 2: Continuous and approximate coverage with $\mathcal{N} = \{A, B\}$

of disjoint tiles. In this way, instead of dealing with custom-shaped areas, we consider the tiles as the units to compute the coverage. This approximation reduces the complexity of the problem, since it makes easy to compute the area of each tile. Moreover, the amount of tiles is a parametric value and does not depends on the number of peers. Figure 2 shows an example on how to compute the coverage of a given AOI (P in the figure) considering a 3x3 approximation. In this case, the coverage of P is 5/9. The number of tiles varies proportionally with the degree of the approximation. A high number of tiles leads to higher precision, which in principle increment the performance of our mechanism. Besides, since the AOI to approximate is a circle, tiles at the corners of the approximation square might be out of the actual AOI area. In this case we do not consider such tiles for the coverage area estimation.

In order to compute the coverage, we consider the following steps. For each AOI $\in \mathcal{N}$ and for each tile, we check whether the AOI intersects with the tile. If it is, we check the counter associated to the tile. If the tile counter is zero, it means the tile is overlapped for the first time so we increase the number of covered tiles. If the tile counter is greater than zero, we just increment it. Besides the number of the tiles covered, this function also counts the number of AOIs that cover each tile. It is easy to show that the complexity of the function is $O(n \times t)$, where n is the cardinality of $\mathcal{N}$ and t is the number of tiles. Since t is a parameter that is fixed during the execution, we consider the complexity of $coverage()$ as $O(n)$.

B. Maximizing AOI Coverage

The aim of the network is to discover the highest amount of objects around the peer (possibly all of them). The straight-forward solution is to to keep links with the neighbours that maximizes the coverage. This indeed requires peers to make a choice, due to the bound imposed by the gossip view size. Hence, very often a peer needs to choose what is the best subset of peers to keep in its cache. This subset is defined as follows:

Definition 2 (Maximum AOI coverage)**.** *Given a set of AOIs $\mathcal{N} = \{N_1 ... N_n\}$ and an AOI P such that $P \notin \mathcal{N}$ and a dimension d such that $d \leq n$, we define $b(P, \mathcal{N}, d)$ as the subset of $\mathcal{N}$ with cardinality $|\mathcal{N}| = d$ that maximizes $c(P, \mathcal{N})$.*

This problem is NP-complete. To prove that, in the following we show how it corresponds to an instance of the *set cover* problem. The set cover problem has been proved to be NP-complete by Karp in 1972 [25] and it is defined as follows.

Definition 3 (Set cover problem)**.** *Given a set U of elements (called the universe) and n sets of elements whose union comprises the universe, identify the smallest number of sets whose union contains all elements in U.*

The correspondence with the Maximum AOI coverage problem is resolved by considering: (i) n as $\mathcal{N}$, (ii) elements as the tiles, and (iii) U as the tiles covered by the AOIs in the optimal solution.

A naive solution to this problem would be to enumerate the possible combinations of peers and for each of them compute the coverage. Unfortunately, this is highly impracticable since the combinatorial nature of the problem. Hence we propose two heuristics with different characteristics. The second, is a greedy heuristic that we prove to be near-optimal. The first is a score-based heuristic that performs better than the greedy in terms of computational requests but at the cost of an higher approximation.

1) Score-based Heuristic: The rationale behind this heuristic is to assign a score to each tile. The tiles that are intersected by few peers have a higher score than tiles intersected by a larger amount of peers. The idea is then to favour such peers that overlap high score tiles. The heuristic works as in the pseudo code in Algorithm 2.

First, it computes the coverage of the AOI by considering all the peers in $\mathcal{N}$. Each tile has a score that is the reciprocal of the number of intersected AOIs. Second, it computes the score for each AOI as the sum of the scores of each intersected tiles. Finally, it sorts the AOIs in descending order according to their score, and it chooses the first d entries.

Algorithm 1: Score-based Heuristic

```
Input  : P, the considered peer, $\mathcal{N}$, the set of neighbours
         AOIs, d, the size of the returned set
Output: a subset of $\mathcal{N}$ with cardinality $d$

1  coverage(P, $\mathcal{N}$);
2  foreach AOI $\in \mathcal{N}$ do
3      foreach tile $\in$ getTiles(P) do
4          if intersect(AOI, tile) then
5              AOI.score $\leftarrow$ AOI.score + $\frac{1}{tile.count}$;
6          end
7      end
8  end
9  sort AOIs in descending order according to score;
10 return the first $d$ AOIs;
```

The time complexity of the score based heuristics is the sum of the following: (i) the coverage procedure, which we have already seen to be $O(n)$, (ii) the computation of the score, that can be considered as $O(n)$ (similarly to the coverage, we consider the number of tiles as a constant parameter), and (iii) the sorting, which is $O(nlogn)$. Hence we can consider the complexity as $O(n)$.

2) Greedy Heuristic: The idea behind the greedy heuristic is simple: at each step to choose the peer that yields the higher increment on the number of unique tiles covered. The pseudocode of the greedy heuristic is represented at Algorithm 3. For each position of the cache, it is selected the AOI that maximizes the number of further covered tiles considering the already chosen AOIs. Note that: (i) an AOI can be selected only once as, upon selection, it is removed from the list of candidates, and (ii) to evaluate the number of tiles covered we use the function $coverage()$ described and evaluated in the previous section.

Algorithm 2: Greedy Heuristic

```
Input  : P, the considered peer, $\mathscr{N}$, the set of neighbour
         peers,d, the size of the returned set
Output: a subset of $\mathscr{N}$ with cardinality $d$
Data: C $\leftarrow \emptyset$
1  while $|C| < d$ do
2      chosen $\leftarrow \emptyset$;
3      max_score $\leftarrow$ 0;
4      foreach $AOI \in \mathscr{N}$ do
5          score $\leftarrow$ coverage(P, C $\cup$ AOI);
6          if $score > max\_score$ then
7              max_score $\leftarrow$ score;
8              chosen $\leftarrow$ AOI;
9          end
10     end
11     remove chosen from $\mathscr{N}$;
12     add chosen to C;
13 end
14 return C;
```

The complexity in time of the greedy heuristic can be studied as following. The outer cycle (line 1) is repeated d times. The inner cycle (line 4) is repeated at maximum $|\mathscr{N}| = n$ times. The biggest set of AOI on which the function $coverage()$ is executed has dimension d (at the first iteration). Hence, the total complexity in time is $O(nd^2)$.

To prove approximation guarantees of the greedy heuristics, first we have to introduce the concept of *submodular* function [26]. Consider Ω to be a finite set and an arbitrary function $f : 2^{\Omega} \to \mathbb{R}$, we can say f is submodular if it satisfies the following property: the marginal gain of adding an element to a set S is at least as high as the marginal gain from adding the same element to a superset of S. More formally a submodular function must satisfy

$$f(X \cup x) - f(X) \geq f(Y \cup x) - f(Y) \qquad (1)$$

for all elements $x \in \Omega$ and for all pairs $X \subseteq Y$. Now, suppose f to be submodular, *non-negative* (i.e. takes only positive values) and *monotone* (i.e. adding an element to a set cannot cause f to decrease. Let also suppose that our aim is to find a set S of cardinality k such that $f(S)$ is maximized. It has been proved in [26] that a greedy algorithm resolves this problem with a worst-case approximation of $(1 - 1/e)$, where e is the base of the natural logarithm. In other words, if the optimum value is 100, the greedy algorithm is guaranteed to find a solution with a value of *at least* 63.

In order to apply this result to our greedy algorithm, $c(P, \mathscr{N})$ must be submodular, non-negative and monotone. Non-negativity is immediate: since we measure an (approximation of) area, it can not be negative. Monotonicity is also immediate, since adding an AOI to a set cannot change the number of tiles already counted. To prove that the function in our greedy algorithm is submodular, we show how it satisfies (1). Let us consider what happens when we add an arbitrary AOI x to a set X whose Y is a superset of: (i) x neither intersects with AOIs in X or AOIs in Y. In this case the equality holds since the marginal gain for both sides of the equation is zero; (ii) x intersects only with AOI's in X. In this case we possibly have an increment on the left side, so the equality holds; (iii) x intersects only with AOI's in Y. In this case the left part of the equation is greater, since it considers all the area covered by x, whereas the right part is incremented only of the part that is non overlapping, so the equation holds; (iv) x intersects with both X and Y. The equation holds since for the left side it counts also the intersection of the AOI's with the elements in Y, that it would not count for the right side. Finally, since we have proved that our greedy algorithm is submodular, non-negative and monotone we can assert that in the worst case we obtain an approximation of $(1 - 1/e)$.

V. Experimental Evaluation

This section presents: (i) the description of the metrics used to evaluate our mechanism, (ii) the workload definition, including the description of the mobility model we used to simulate avatars' movement, and (iii) a selection of experimental results evaluating the key performances of the approach.

A. Metrics

To evaluate our approach we consider two different metrics. The first metric evaluates the coverage of peers AOI. We refer to this metric as *AC*. *AC* is a value in the interval $(0, 1)$ and, given a peer at an arbitrary iteration, is defined as the ratio between the AOI coverage obtained by the P's view and the best AOI coverage defined by considering all the peers in the DVE. The second metrics measures the difference between the local replica of the peer' state against the server state. To measure this difference, we exploit a slightly modified version of the *Jaccard similarity coefficient* [**?**]. Let us consider C as the local replica of a peer and S as the remote replica of the server. The original Jaccard coefficient is computed as $S \cap C / S \cup C$. However, this formulation either does not take in account the difference of the positions of the entities, or considers entities with a different position as distinct. In order to take into account at the same time the difference in position and the presence of the entities we exploit the following formula to compute the Jaccard coefficient (in short *JC*):

$$JC = \frac{\sum_{x_S, x_C \in S \cap C} 1 - \frac{dist(x_S, x_C)}{d_{MAX}}}{S \cup C} \qquad (2)$$

where d_{MAX} is the diameter of the peer's AOI. A peer with $JC = 1$ has its local replica perfectly synchronized with the state of the server while $JC = 0$ implies that the replica is completely out-of-sync with that of the server. Any value in between 0 and 1 gives a quantitative evaluation on the quality of the synchronization.

While AC measures how good the heuristics performs in a dynamic environment, JC measures the quality of the approach in terms of the quality of the application. A direct correlation between AC and JC would be desirable. In the following section we show how the simulation support the existence of this correlation.

B. Workload and Environment

The simulation code is written in Java. In a single simulation run, each node (executed in a private thread) executes a number of gossip iterations, and two consecutive iterations are separated by a fixed amount of time. For all the simulations we consider a VE defined as a squared region of 5000 x 5000 points. The map has 5 circular fixed hotspot, whose radius is tuned so that the 20% of the total area of the DVE is a hotspot. The remaining 80% is considered as *outland*. The 65% of the total number of the objects of the DVE lie in hotspot areas where their concentration follows a power law distribution, with a peak in the hotspot center. The total amount of objects in the simulation is 1000, and players AOI have a radius of 100 points. The remaining 35% of the objects stay in the outland area and it is distributed according to a uniform random distribution.

Avatars move on the map according to realistic mobility traces that have been computed according the mobility model presented by Legtchenko et al. [27], which simulates avatars movement in a commercial DVE, Second Life [28]. The model works according to the hotspots defined in the DVE. When an avatar reaches a hotspot, it explores the hotspot for a span of time and eventually it moves to another hotspot. This behaviour is defined by a state finite automata characterized by three states: halted, exploring and travelling. When in halted state avatars stay still, whereas in the exploring state avatars explore a portion of the DVE close to their current position. Finally when in travelling state avatars move from one hotspot to another. This mobility model exposes a fair balance between the time spent by avatars in hotspots and outland. Furthermore, the path followed by avatars when moving between hotspots is not fixed, i.e. no predefined path connects two hotspots.

C. Behaviour over T_s

In this section we discuss the result of several simulation runs by varying the interval of time (T_s) between two consecutive communications to the central server. Requests to the overlay are done every 0.25 seconds. For instance, with $T_s = 1$ there is a server communication followed by three requests to overlay in row and then another server communication. Where it is not indicated differently, the simulations consider: 500 nodes with a cache of 10 elements each, 1000 objects, and an AOI approximation of 32x32 tiles. Figure 7 shows the comparison of the JC between the greedy heuristic and the score heuristics. In general, we can observe how the reduction in the JC is limited even with high values of T_s. For instance, with $T_s = 1$ the average JC value for both the heuristics is around 0.9. Form an application point of view, this means that the mechanism is able to fully support IM. As expected, further increments of T_s imply a JC reduction. Note However, that even with the $T_s = 2.5$, the JC is still around 0.8.

The effectiveness of the mechanism is further supported by the values of the JC when using only the server. In other words, increasing T_s would be problematic if not supported by the overlay. For example, with $T_s = 1.5$, the JC with the support of the overlay is around 0.9, whereas is 0.65 using only the server.

As regards the comparison between heuristics, the greedy outperforms the score heuristics. With these simulation parameters, the AC, which is independent from T_s, is 0.8 and 0.85 respectively for the greedy and the score heuristics. This would suggest a correlation between AC and the JC.

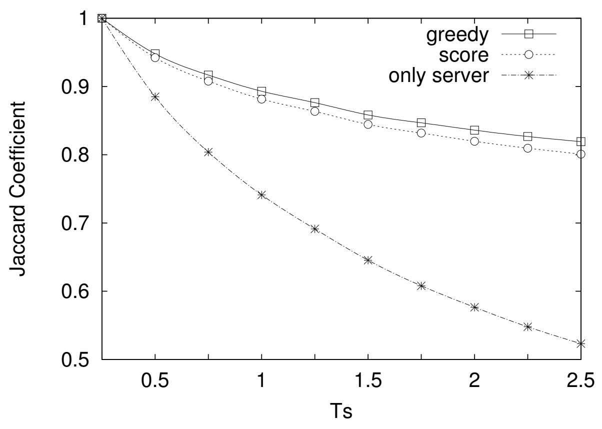
Figure 6 shows the JC when selecting the more fresh entries during a gossip iteration. The ranking algorithm considered is the score, but similar results have been obtained with the the greedy heuristics. The results are evident and not surprising: to prefer fresh entries gives a neat increment on the performance. As the previous, even this result indicates a possible correlation between AC and JC as the score's $AC = 0.80$ with stale control and $AC = 0.70$ without.

D. Tiles Variation

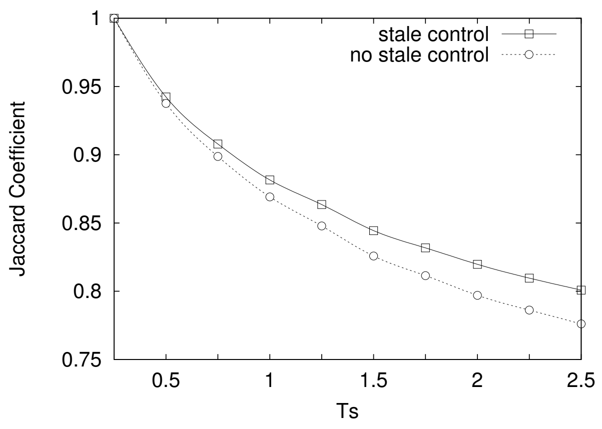
Figure 8 shows the JC and the AC of the greedy and score heuristics with various degree of AOI approximation (from 16 to 1024 tiles). The points for the plot have been obtained by averaging the outcome of 20 simulation runs with a $T_s = 1$. The results show that the score-based heuristics is basically agnostic to the approximation whereas the increment in the number of tiles implies an increment of the performance of the greedy-based heuristics. From the graph it is clear how with approximations larger than 400 tiles for AC and 600 for JC, the greedy-based heuristics outperforms the score-based one. The reason why it happens lies on the order used by the greedy heuristics for choosing the areas. Indeed, with higher approximation, the greedy would have greater chance to choose a worse area. When the approximation is reduced, the greedy heuristics performance increases.

E. Number of Peers

Figure 10 shows how the number of peers affects the JC of greedy and score heuristics. Each point in the plot is the average of the outcome of 20 independent runs. The simulations have been run with a 32x32 AOI approximation, a fixed cache size of 10 elements, and a $T_S = 1$.

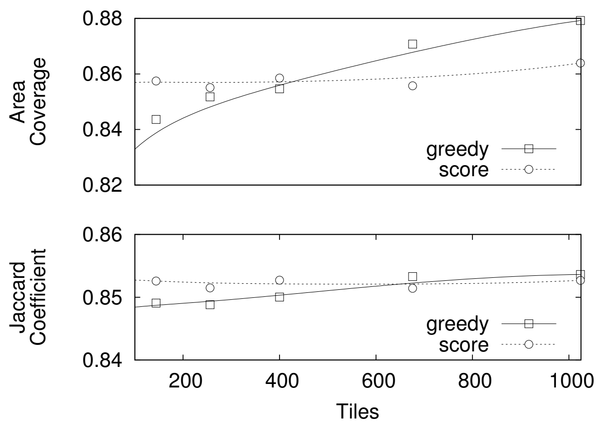


(a) Comparison between score-based heuristics and greedy-based heuristics

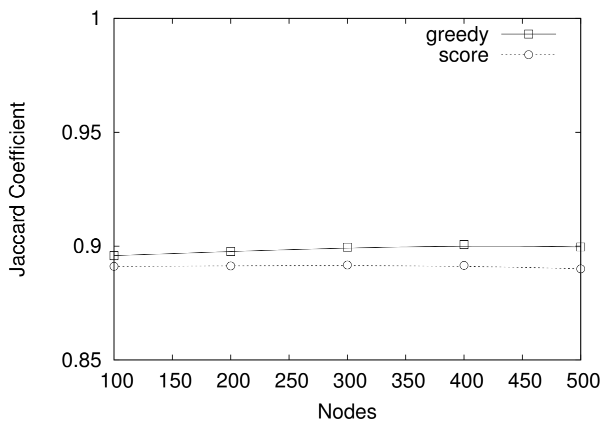


(b) Score heuristics, comparison between considering or not freshness of entries

Fig. 3: Evaluation of the heuristics over T_s



(a) AC and JC with different number of tiles



(b) JC with different network sizes

Fig. 4: Evaluation of the heuristics with different AOI approximation and network sizes

As expected, the greedy overcomes the score, but both the heuristics show a similar behaviour. Their performance are essentially independent from the number of nodes, even with a fixed-size cache. In fact, there is a slight increment on the JC as the number of peers increases. With more nodes there is a higher chance of crowded zones. This situation allow the node to exploit the knowledge of the neighbours more often.

VI. Conclusion

In this work we proposed a gossip-based mechanism to build overlay for best-effort IM in DVEs. Conversely to the other approach in the field, we trade some precision in the result to keep the mechanism fast, simple and lightweight. However, our proposal can be further extended and studied. To this end, we plan to improve the precision of the result by considering additional information when ranking peers, such as movement forecasts and different neighbours selection functions. As to further validate our solution, we intend to test it with movement traces from different mobility models and to compare it with the non best-effort works presents in literature.

Acknowledgements

The authors acknowledge the support of Project FP7-257438, Contrail: Open Computing Infrastructures for Elastic Services. The authors thank the reviewers for the complete and helpful reviews. The authors would also thank Patrizio Dazzi, Rossano Venturini and Luca Colombini for their substantial help.

References

[1] K. Morse, L. Bic, and M. Dillencourt, "Interest management in large-scale virtual environments," *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 1, pp. 52–68, 2000.

[2] S. Hu, "Spatial publish subscribe," in *Proc. of IEEE Virtual Reality (IEEE VR) workshop, Massively Multiuser Virtual Environment (MMVE09)*, 2009.

[3] D. Ahmed and S. Shirmohammadi, "A dynamic area of interest management and collaboration model for p2p mmogs," in *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 27–34.

[4] J. Boulanger, J. Kienzle, and C. Verbrugge, "Comparing interest management algorithms for massively multiplayer games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, p. 6.

[5] S. Rooney, D. Bauer, and R. Deydier, "A federated peer-to-peer network game architecture," *IEEE Communications Magazine*, vol. 42, no. 5, pp. 114–122, 2004.

[6] L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan, "Hydra: a massively-multiplayer peer-to-peer architecture for the game developer," in *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2007, pp. 37–42.

[7] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *NSDI '06: 3rd Symposium on Networked Systems Design & Implementation*, 2006, pp. 155–168.

[8] A. Bharambe, J. Douceur, J. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: Enabling large-scale, high-speed, peer-to-peer games," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 389–400, 2008.

[9] J. Jiang, Y. Huang, and S. Hu, "Scalable AOI-cast for peer-to-peer networked virtual environments," *Journal of Internet Technology*, vol. 10, no. 2, pp. 119–126, 2009.

[10] L. Ricci, E. Carlini, L. Genovali, and M. Coppola, "AOI-cast by compass routing in Delaunay based DVE overlays," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE, 2011, pp. 135–142.

[11] E. Buyukkaya and M. Abdallah, "Data management in voronoi-based p2p gaming," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*. IEEE, 2008, pp. 1050–1053.

[12] S. Hu, S. Chang, and J. Jiang, "Voronoi state management for peer-to-peer massively multiplayer online games," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*. IEEE, 2008, pp. 1134–1138.

[13] L. Ricci and L. Genovali, "State management in distributed virtual environments: A voronoi base approach," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on*. IEEE, 2010, pp. 881–887.

[14] J. Jardine and D. Zappala, "A hybrid architecture for massively multiplayer online games," in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*. ACM, 2008, pp. 60–65.

[15] E. Carlini, M. Coppola, and L. Ricci, "Integration of p2p and clouds to support massively multiuser virtual environments," in *Network and Systems Support for Games (NetGames), 2010 9th Annual Workshop on*. IEEE, 2010, pp. 1–6.

[16] D. Frey, J. Royan, R. Piegay, A. Kermarrec, E. Anceaume, and F. Le Fessant, "Solipsis: A decentralized architecture for virtual environments," *Proceeding of 1st International Workshop on Massively Multiuser Virtual Environments (MMVE'08)*, pp. 29–33, 2008.

[17] A. Chen and R. Muntz, "Peer clustering: a hybrid approach to distributed virtual environments," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, p. 11.

[18] S. Hu, J. Chen, and T. Chen, "Von: a scalable peer-to-peer network for virtual environments," *Network, IEEE*, vol. 20, no. 4, pp. 22–31, 2006.

[19] H. Backhaus and S. Krause, "QuON - a Quad-Tree Based Overlay Protocol for Distributed Virtual Worlds," in *Proceeding of 2nd International Workshop on Massively Multiuser Virtual Environments (MMVE'09)*, 2009.

[20] J. Lee, H. Lee, S. Ihm, and T. Gim, "Apolo: Ad-hoc peer-to-peer overlay network for massively multi-player online games," *Science And Technology*, 2006.

[21] J. Jiang, Y. Huang, and S. Hu, "Scalable aoi-cast for peer-to-peer networked virtual environments," in *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*. IEEE, 2008, pp. 447–452.

[22] L. Ricci, E. Carlini, L. Genovali, and M. Coppola, "Aoi-cast by compass routing in delaunay based dve overlays," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE, 2011, pp. 135–142.

[23] A. Schmieg, M. Stieler, S. Jeckel, B. Kabus, P. Kemm, and B. A., "psense: maintaining a dynamic localized peer-to-peer structure for position based multicast in games," in *8th International Conference on Peer-to-Peer Computing 2008, P2P 2008*. IEEE, 2008.

[24] M. Jelasity, A. Montresor, and B. O., "T-Man: Gossip-based fast overlay topology construction," *Journal Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 53, no. 13, 2009.

[25] R. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, 1972.

[26] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[27] S. Legtchenko, "Blue Banana: resilience to avatar mobility in distributed MMOGs," *Networks*, pp. 171–180, 2010.

[28] "Second life website," http://secondlife.com/.