Blue Banana: resilience to avatar mobility in distributed MMOGs

Sergey Legtchenko Sébastien Monnet Gaël Thomas

LIP6/UPMC/CNRS/INRIA
104 av. du Président Kennedy, 75016 Paris - France
Firstname.Name@lip6.fr

Abstract

Massively Multiplayer Online Games (MMOGs) recently emerged as a popular class of applications with millions of users. To offer acceptable gaming experience, such applications need to render the virtual world surrounding the player with a very low latency. However, current state-of-the-art MMOGs based on peer-to-peer overlays fail to satisfy these requirements. This happens because avatar mobility implies many data exchanges through the overlay. As state-of-the-art overlays do not anticipate this mobility, the needed data is not delivered on time, which leads to transient failures at the application level. To solve this problem, we propose Blue Banana, a mechanism that models and predicts avatar movement, allowing the overlay to adapt itself by anticipation to the MMOG needs. Our evaluation is based on large-scale traces derived from Second life. It shows that our anticipation mechanism decreases by 20% the number of transient failures with only a network overhead of 2%.

1. Introduction

The past few years witnessed the emergence of a new class of distributed, highly collaborative applications called Massively Multiplayer Online Games (MMOGs). The main aim of an MMOG application is basically to provide a large virtual universe, or NVE for Networked Virtual Environment. In NVEs, users represented by their *avatars* can freely move and interact with each other [29]. NVE applications involve millions of active participants all over the world and generate substantial financial revenue [2]. Such applications need to be highly scalable to support the colossal number of players and to be reactive with almost real-time constraints to provide a satisfying gaming experience.

Current popular NVEs are based on the client-server paradigm [35, 36]. This necessarily implies poor scalability for NVE applications and expensive financial cost for the NVE provider [2, 16]. To face these limitations, a new generation of decentralized NVEs based on peer-to-peer overlays has emerged [3, 4, 11, 12, 15]. In these NVEs, the load and the applicative data is fairly divided between all the nodes of the overlay. Therefore, each node stores a *local knowledge* of the NVE: a set of data-blocks describing some objects of the virtual world. In order to correctly render the virtual world surrounding its avatar, a node must acquire the set of data-blocks representing the area in the NVE where its avatar is located. We define these set of data-blocks as the *playing area* of the node. To build a playing area, a node must find other nodes that have the required data-blocks in their local knowledge. We define these other nodes as the *elders* of the playing area.

One of the main problems that a distributed NVE must face is the construction and the update of an avatar's playing area when it moves. Indeed, the playing area of a moving avatar changes and its node has to quickly retrieve the data-blocks of the new playing area from new elders. Virtual movement of an avatar thus involves real data exchange through the underlying overlay network. Moreover, the faster the avatar moves, the lesser time its node has to download missing data-blocks. If a node is unable to retrieve the data composing its current playing area in a reasonable time, i.e, in a time that does not degrade the gaming experience, we say that the node *transitory fails*. The threshold delay is typically of a few hundreds of milliseconds [5]. Basically, this notion of failure depends on the quantity of information needed to correctly render a playing area, which is highly application-dependent.

State-of-the-art overlays for NVEs try to deal with the problem by continuously adapting their logical graph in reaction to virtual mobility. When an avatar moves across the NVE, its node changes its neighbor set in order to recover all needed data in a small number of *hops* in the overlay [3, 6, 15]. However, these overlays only *react* to movement: a node changes its neighbor set after the movement of its avatar. This lets only a few hundreds of milliseconds to find the new elders and to retrieve the needed data from them. If the movement is too fast, or if the amount of data to download is too large, the overlay is unable to adapt itself on time, causing transient failures. Moreover, the problem is symmetric for non moving avatars: if a moving avatar enters in the playing area of a non moving avatar too quickly, the non moving avatar will not see the entering one in its playing area.

To solve this problem, we propose a new mechanism called Blue Banana[1] that *anticipates* movement and searches the elders of the forthcoming playing areas when an avatar moves. Concretely, the algorithm tries to predict avatar movement and, if it has a stable movement during a sufficiently long period of time, its node prefetches elder nodes of the playing areas in the direction of its movement with respect to the avatar speed. Our algorithm decreases the number of transient failures of a moving avatar: the loading of the data composing the forthcoming playing areas begins earlier, allowing the prompt construction of a correct playing area image when the avatar effectively enters inside it.

The key challenge to design our algorithm is an accurate understanding of avatar mobility. Indeed, if our algorithm fails to correctly predict avatar movement, it will load useless data. The problem is particularly important if the avatar has erratic movement: our algorithm must not try to prefetch the forthcoming elders and the data of the forthcoming areas at each direction shift. The load of this useless data would overload the node and therefore generate new transient failures.

The contributions of this paper are thus: 1) an analysis and a model of mobility to qualify and detect predictable player movement; 2) the implementation of Blue Banana, our anticipation mechanism in one of the state-of-the-art peer-to-peer overlay networks: Solipsis; 3) a generator of realistic movement traces to evaluate Blue Banana; and 4) a complete evaluation of Blue Banana with our generated movement traces in the PeerSim simulator [14]. We choose to test our algorithm on top of Solipsis because it already selects and updates its overlay neighbors based on avatar virtual positions in the NVE. However, like other current NVE overlays, Solipsis does not anticipate the movement of the avatars: it fetches elders indifferently in all the directions and therefore fails to build playing areas in a reasonable time when the movement increases.

The main lessons learned from our work are:

- Our model of mobility provides the ability to predict avatar movement. By adding an anticipation mechanism in Solipsis, the number of transient failures decreases by 20% while the network bandwidth is only increased by 2%. Moreover, our mechanism does not decrease the robustness of the original protocol: when movement is erratic, transient failures do not increase. Blue Banana also increases the robustness of the NVE because in average, a node knows 7.5 times more elders in the direction of the avatar's movement, allowing a node to prefetch 20 times more data on time.
- The traces of movement generated from our model are realistic and an evaluation shows that they clearly coincide with the real traces collected in Second Life [31]. They permit the construction of larger traces and therefore the evaluation of our protocol.

The rest of this paper is organized as follows. First Section 2 studies the mobility in real existing NVEs and presents our mobility model that allows the overlay to predict movement. Section 3 presents the implementation of Blue Banana on top of Solipsis. Section 4 describes our trace generator, then Section 5 presents the evaluation environment and the evaluation results. Section 6 describes related works before Section 7 concludes.

2. Mobility pattern and movement prediction

Avatars connected to NVEs usually have a total freedom of movement. Resulting NVEs are then very dynamic: data representing objects and avatars may not be uniformly distributed all over the universe. Recent studies of existing popular NVEs like Second Life [35] and World of Warcraft [36] have shown that the distribution of avatars was extremely disparate [17, 25]: most of the avatars are gathered around a few hotspots of interest, while large parts of the NVE are almost desert. In addition to that, the mobility pattern of the avatars has been shown to be highly non-uniform: avatars move slowly and chaotically within the hotspots, whereas the movement between the hotspots is straight and fast [19].

2.1. The state machine

These observations have a consequence on the design of our anticipation algorithm: the anticipation mechanism must discriminate chaotic from straight avatar movement. Therefore, in order to ensure reasonable prediction accuracy, each node of the overlay handles a state machine that describes its avatar mobility. According to the observed mobility pattern, an avatar has two states: ($\tilde{T}$)ravelling, the avatar is rapidly moving on the map and its trajectory is straight, ($\tilde{E}$)xploring, the avatar is exploring an area, its trajectory is chaotic and its speed is low.

As the user is interacting with the NVE, its node locally analyzes the state of the avatar. If it detects a behavioral modification, it switches the state machine to the appropriate state. The behavior of an avatar is defined by its *speed*. If the speed of an avatar reaches a threshold, the state machine is switched to the state $\tilde{T}$, otherwise, it is switched to

[1] The Blue Banana is the pattern of one of the highest concentration of population of the world and the relationship with our work is discussed in Section 4.

the state $\tilde{E}$. This simple model is a first attempt to describe avatar movement and can be refined: it could take into account the acceleration of the avatar, or try to predict player behavior by analyzing its movement history. However, this simple model already provides a sufficient prediction accuracy to decrease the number of transient failures.

If the state machine is in the state $\tilde{T}$ (the avatar is traveling), its trajectory is highly predictable. Therefore, our algorithm tries to prefetch the forthcoming elders, i.e, the nodes that have data of the forthcoming playing area in their local knowledge.

If the avatar is exploring a zone (state $\tilde{E}$) its trajectory is chaotic and its speed is low. In this case, its path is difficult to predict, therefore the Blue Banana module does not anticipate the loading of the forthcoming elders. Notice that because of the slow speed, the native algorithm of an NVE is likely to adapt itself on time anyway.

2.2. Movement anticipation

To maximize the prediction accuracy, we make two assumptions: (i) only short term prediction is accurate, (ii) the faster an avatar is moving, the more it is likely to continue on its current trajectory. The first assumption implies that future probable positions calculated from the avatar's present location and movement vector form a *cone*. Indeed, the more a position prediction is far in the future, the more it is likely to diverge from the real path. The second metric implies that the prediction accuracy increases with the avatar speed: the sharpness of the cone is proportional to the speed.

If all elders of the playing area located inside the cone are prefetched on time and if the avatar stays in the cone, the node of the moving avatar will then *instantly* adapt to the mobility.

3. Implementation of Blue Banana on top of Solipsis

We have implemented our Blue Banana prefetching algorithm over Solipsis [15]. We chose Solipsis because it already takes into account avatar proximity to build the overlay.

3.1. Solipsis overview

Solipsis is an overlay designed to sustain a distributed NVE. Each node of the Solipsis overlay is responsible for one avatar. In Solipsis, the knowledge of a playing area is distributed on the nodes that manage the avatars of this playing area: the elders of a playing area are exactly the nodes which avatars are in this playing area. Solipsis maintains a set of direct neighbors for each node. Nodes communicate by message passing through the overlay: the more the distance in the overlay in number of hops increases, the more the latency increases. To enhance the responsiveness, Solipsis tries to maintain the elders of the current playing area of a node in its neighborhood to communicate efficiently. If two avatars A and B are neighbors in the NVE, the Solipsis overlay adapts itself so that B will *eventually* be in A's neighborhood and vice versa. In order to ensure that behavior, Solipsis is based on two fundamental rules:

1. *Local awareness rule.* An avatar a has a circular playing area ω_a centered on the avatar. If another avatar b is inside ω_a, the nodes of a and b must be neighbors in the overlay. The size of ω_a is adjusted to ensure that a has a number of neighbors contained between a minimum and a maximum bound.

2. *Global connectivity rule.* Let N_e be the neighbor set of a node e in the overlay. The avatar of e must be located inside the convex hull of the set formed by avatars of N_e. This property aims that an avatar will not "turn its back" to a portion of the NVE, causing inconsistent views or possibly partitioning the Solipsis overlay graph.

To ensure these rules, Solipsis implements a mechanism called *spontaneous collaboration*. At each moment, thanks to periodic updates, a node is aware of the coordinates and the awareness area sizes of all nodes in its neighbor set. As it locally detects that one of its neighbors enters the awareness area of another of its neighbors, it sends a message to both entities to warn them that the local awareness rule is about to be broken. As they receive that message, the two entities become neighbors. Our simulations showed that this technique is very efficient: most of the time, a node receives a warning message and does not have to initiate a costly new-neighbor query. The global connectivity rule ensures that a node is always surrounded by its neighbor set, making spontaneous collaboration more efficient.

To sum up, if the local awareness rule is violated for a node n, it means that an avatar has arrived into the playing area of n and is not yet included to the local knowledge of n, causing a transient failure. If the global connectivity rule is violated for a node n, it means that n is not surrounded by its neighbor set. It will then not receive spontaneous data updates for a part of its playing area, which will mandatorily lead to transient failures.

An avatar keeps breaking fundamental rules as long as it moves because the spontaneous collaboration mechanism is not always able to react on time. For that reason, a more efficient anticipation mechanism is required.

3.2. Implementation of the anticipation mechanism

Blue Banana, our anticipation mechanism, is built on top of Solipsis. However, it could be implemented on top of any overlay that adapts itself in reaction to avatar movement. Blue Banana's main aim is to provide each node with a prefetched node set (the size of the set is user defined). For this purpose, it finds nodes in the direction of the avatar's movement. Once the moving avatar approaches a prefetched node, the prefetched node is added in the regular neighbor set managed by Solipsis. Hence, Blue Banana substantially helps Solipsis native algorithms to restore the fundamental rules, minimizing resulting transient failures.

Important properties of the algorithm. The first important quality of the algorithm is the consideration of avatar movement during message transfer time. Indeed, during a message transfer, the NVE changes, and so do interesting prefetched neighbors. For example, if A and B are 2 meters apart and if A runs toward B, B is probably an interesting prefetched neighbor. But if the network latency is around $200ms$ and if A runs at $36km/h$ $(10m/s)$, the time to transfer a message from A to B is exactly the time to reach B for A in the NVE: the communication time between A and B makes B an uninteresting prefetched neighbor. To take into account message transfer time, each node estimates a low and a high bound of the network latency by using the last observed round trip times with its neighbors. When a prefetching message arrives, the algorithm uses these latency bounds to roughly estimate the new avatar-position of the node that emitted the message. Even if this estimation is clearly rough, it permits to send more accurate responses.

The second important quality of the algorithm is the number of messages generated to prefetch the neighbors: a node receiving a prefetching request answers for all its neighbors whose avatars are in the probability cone (see Section 2). As a consequence, each candidate does not have to answer to the request.

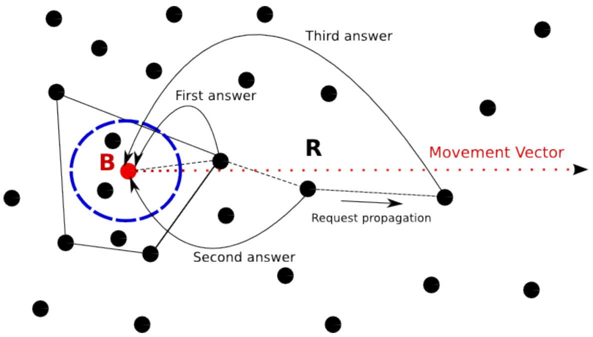

Figure 1. Propagation algorithm: the request is transmitted to nodes ahead of the movement.

Algorithm description. Technically, if the algorithm observes that the avatar of a node B (for Blue Banana) is in the state $\tilde{T}$ (i.e, it reaches the speed threshold) and if the prefetched neighbor set is not full, B starts searching for new prefetched neighbors: it sends a message to its neighbor which is closest to its *movement vector* as illustrated by Figure 1. The message contains the number of prefetched neighbors that B is willing to retrieve (called the TTL) and the description of the probability cone (the apex of the cone, the direction of the movement and the speed).

Algorithm 1: Upon reception of a prefetching request

```
   Result: gathering of prefetching candidates and prefetching request
           propagation.
1  emitterPosition = estimateCurrentEmitterPosition (msg);
2  ttl = msg.getTTL ();
3  if (emitterPosition, myPosition) $\geq$ minDist then
4      trajectoryClosestNodes = chooseClosest (neighborSet, msg);
5      size = trajectoryClosestNodes.getSize ()$-1$;
6      if size $>$ ttl then
7          size = ttl;
8          trajectoryClosestNodes = trajectoryClosestNodes [1 .. size ];
9      end
10     if size $>$ $0$ then
11         ttl = ttl $-$ size $+$ 1;
12         response.addSet (trajectoryClosestNodes);
13         send (response, msg.emitter ());
14     end
15 end
16 if ttl $>$ 0 then
17     msg.setTTL (ttl $-1$);
18     send (msg, findNextNodeInTrajectory (msg));
19 end
```

Upon the reception of a prefetching request on a node R (for Receptor), R first estimates the current position of B by using the estimated network latency, the initial position and the speed of the avatar (line 1 of Algorithm 1). Then, Algorithm 1 checks if B is not too close from R (line 3): if B overpasses R during the message exchange, R is located behind of B when the response is received by B, making the prefetched information useless. Then, if R is located far enough (lines 3 to 13), R analyzes its neighbor set and selects nodes located inside the new estimated prefetching cone of B (line 4) to send them to B (lines 11 to 13). If the size of this set of candidates exceeds the TTL, only the first TTLs are selected (line 5 to 9) and if R does not have interesting neighbors, R does not send its response to B (line 10). While the TTL has not expired, R forwards the request to its neighbor that is closest to the *movement vector* of B (lines 16 to 19). At the end, if no message have been lost and if messages arrive on time, B retrieves TTL prefetched neighbors located inside its probability cone.

Network overhead. Blue Banana does not interfere with the maintenance protocol of Solipsis: the prefetched neighbors are not placed in the regular Solipsis neighbor set, but in a separated one. Therefore, Solipsis does not use network resources to maintain links with prefetched neighbors.

We prefer not to spend network resources to maintain a link with a node which is useless *in the present* since it is not yet in the playing area.

As a consequence, once inserted in the prefetched neighbor set, the position of a node's avatar is not updated, while it can move outside the probability cone. Blue Banana automatically removes useless prefetched neighbors (i) when they have been overtaken by the moving avatar, (ii) when the avatar changes its direction or (iii) when it changes its state. It is possible to consider another policy by periodically updating the state of the prefetched neighbors. However, the risk is to spend network resources to update possibly useless nodes. The comparison of these two policies is part of a future work.

In order to compensate the small network overhead, Blue Banana nodes take advantage of the high predictability of the avatar movement in desert zones. In Solipsis, a node periodically propagates the coordinates of its avatar to all the members of its neighbor set, so the neighbor-nodes can update their view of the NVE. Blue Banana doubles the period of such updates for nodes when the state machine is in state $\tilde{T}$. The neighbors of that node simply predict the position of the avatar between two updates by using its initial position and its speed. This technique is a simple form of *dead reckoning*[2], but it could easily be enhanced with more sophisticated mechanisms widely used in online gaming [7,23,24].

4. Realistic movement trace generation

The evaluation of Blue Banana requires realistic traces of avatar movements. However, all existing commercial NVE projects are based on a client-server architecture which usually implies poor scalability: constraints are generally added to artificially limit the scale, hiding this defect [16]. For instance, the Second Life world [35] is partitioned into separate regions called “islands”, each of them being limited in number of simultaneous users and the World of Warcraft game [36] is split into separated realms. Therefore, because of these scaling limitations, each trace simultaneously involves at most a few hundreds of avatars. Moreover, the number of available real traces is small because they are difficult to obtain [17]. Therefore current real traces are not sufficiently numerous and not sufficiently large-scaled to measure the efficiency and scalability of Blue Banana.

To evaluate Blue Banana, we therefore need to accurately model avatar movements in order to generate realistic *large-scale* traces. This section presents our model of mobility.

Because all current existing popular NVEs are centralized and thus limited in scale, we believe that our trace generator can also be reused to evaluate other NVEs.

As presented in section 2, most of the avatars are gathered inside a few density hotspots. Most of the time, hotspots are towns or interesting locations of the NVE. This kind of distribution with hotspots also corresponds to real density distribution of human populations such as the European *blue banana* [8] that covers one of the world highest concentrations of population around the cities of London, Brussels, Amsterdam, Cologne, Frankfurt and Milan with approximately 20% of the European population.

Moreover, as presented in Section 2, movements of avatars are chaotic in hotspots and straight between hotspots. Regarding the player mobility in NVEs, studies have shown that it is quite similar to human mobility in the real world [17,26]. This mobility pattern is most of the time modeled with Lévy flights [17, 26], however, we propose our own model because Lévy flights do not take into account the specific density of hotspots. Indeed, Lévy flights are particular sort of random walks in which the increments are distributed according to a "heavy-tailed" probability distribution [9] with short and chaotic movements and sometimes long and straight ones. Therefore, Lévy flights naturally differentiate the two observed behaviors of avatars: periods of travel and periods of exploration with chaotic movements. But Lévy flights do no help to model hotspots because they do not ensure that avatars stay grouped around hotspots and that density around hotspots remains the same despite avatar mobility.

Instead, we choose to model the density and the movements of avatars in NVEs with a model based on an automaton to discriminate the periods of exploration from the periods of travel. We define hotspots, i.e, high density zones, and by opposition, the desert. The trace generation is decomposed in two phases. During the first one, all the avatars are placed on their initial positions on the map. The generator ensures that most of the avatars are grouped in the hotspots. During the second phase, step by step, the generator computes new maps from the previous ones by moving avatars. The model of movements ensures that avatars remain principally grouped in hotspots during time.

To generate movements, at each step, each avatar is in one of the following states: (H)alted, the avatar does not move at this step; (E)xploring, the avatar is exploring the map; (T)raveling, the avatar is moving to a new location on the map. Each state has its own maximal speed value S_{max}. Once S_{max} is reached, the acceleration drops. Otherwise, we consider that an avatar has a constant application-defined acceleration during the movement (states E and T). The acceleration value is a parameter of the model generator. As an avatar moves from a position to another, at each step, its speed increases. When an avatar reaches its final position, it suddenly stops and its state machine enters the H state.

The trace generator is configurable and takes the follow-

[2] Dead reckoning is the process of estimating one's current position based upon a previously determined position.

ing parameters: 1) the number of avatars; 2) the size of the map; 3) the number of hotspots and the radius of each hotspot; 4) the proportion of avatars inside hotspots; 5) the maximal speeds for states E and T; 6) the acceleration; 7) the probabilities associated to transitions between the different states.

Generation of the initial map. During the first phase, the trace generator randomly chooses the positions of the hotspots. Then, for each avatar, the generator decides if it should be placed in a hotspot accordingly to the proportion of avatars inside hotspots. If this is not the case, the avatar is randomly placed on the map using a uniform probability law (it can therefore be placed in a hotspot or in the desert). Otherwise, the generator randomly chooses one of the hotspots using a uniform law and computes the polar coordinates of the avatar from the center of the hotspot: the angle is chosen using a uniform law and the distance to the hotspot center with a Zipf's law [34]. The Zipf's law ensures a very high density in the center of the hotspot, comparable to the ones observed in both NVEs and real life. Initially, all avatars are in the state H.

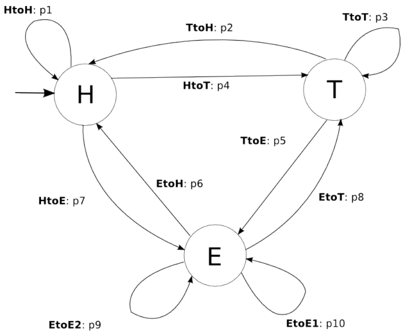

Figure 2. State machine and transition probabilities.

Generation of movements. During the second phase, the trace generator moves the avatars step by step. The figure 2 presents the automaton used for state transition with the associated state transition probabilities. At each step, the generator reevaluates the state of all the avatars thanks to the state transition probabilities:

State H: If an avatar enters or stays in the state H (transitions $TtoH$, $EtoH$ and $HtoH$), the avatar does not move at this step.

State E: If the avatar takes one of the transitions $HtoE$, $TtoE$ or $EtoE2$, the avatar picks a new position on the map. To ensure that the density remains globally the same during the trace, if the avatar is in a hotspot its new position is chosen inside the same hotspot with the Zipf's law, otherwise, its position is chosen randomly on the map. If the avatar is in the state E and takes the transition $EtoE1$ it continues its movement to its new position. We differentiate the two transitions $EtoE1$ and $EtoE2$ to ensure that an avatar regularly changes its direction and therefore has a chaotic movement.

State T: If the avatar enters in state T ($HtoT$ or $EtoT$), the avatar picks a new position on the map by using the initial placement function to ensure that the density remains roughly the same. It also begins its movement to its new position. If it takes the transition $TtoT$, it continues its movement to its new position.

Evaluation of generated traces. Due to lack of space, the evaluation of the trace generator is presented in [18]. It compares generated traces to real traces collected by La and Michiardi [17] from Second Life. These real traces have been collected by crawling two Second Life islands called "Dance" and "Isle Of View" which were chosen to be representative of the Second Life players' behavior. The evaluation uses metrics that do not depend neither on the size of the map nor of the avatars' number.

The results show that the traces generated from this model are similar to the real ones.

5. Evaluation

This section presents a detailed evaluation of Blue Banana. The evaluation compares Solipsis *with* and *without* Blue Banana to measure the performance of the anticipation mechanism. Both Solipsis and Blue Banana are implemented on top of the PeerSim discrete event simulator [14].

5.1. Description of the simulations

The PeerSim simulator is a widespread platform for testing distributed applications [1, 10, 13]. It has been designed for scalability and is simple to use. It is composed of two simulation engines, a simplified (cycle-based) one and an event driven one. The simulation is realized with the event driven engine which performs more accurate simulations.

At the beginning of the simulation, the initial map of the trace (described in Section 4) is injected in the simulator. The simulator, based on this map, initializes the Solipsis overlay and then waits until every node respects the two Solipsis rules (see Section 3.1). After the convergence, mobility of avatars is simulated by injecting the rest of the trace. Evaluating Blue Banana with the real traces is irrelevant because the benefits are not significant with small-scale NVEs.

The parameters of the simulations are: 1) 1000 avatars, 2) A surface equivalent to 9 Second Life maps, 3) 3 high density hotspots, 4) Hotspot density: 9549 avatars per square kilometer (24720 avatars per sq. mile), which is, for example, just below the density of New York City, 5) The constant acceleration of avatars during movement is 5 $m.s^{-2}$, 6) Nodes have an ADSL connection with a 10Mbit download and 1Mbit upload bandwidth, 7) The network latency between nodes is randomly set between 80 and 120 ms with an uniform distribution. Notice that with the constant acceleration, the *maximum* speed of avatars *between* hotspots can reach the speed of a helicopter (100 $m.s^{-1}$). This speed may seem exaggerated, but MMOG participants need to be provided with a fast mean of transportation[3]. Moreover, the constant acceleration is 5 $m.s^{-2}$, so the avatars do not instantly reach the maximal speed. In fact this speed is only reached in the worst case: when an avatar moves from a hotspot in a corner of the map to a hotspot in the opposite corner. The actual speed of most avatars is much lower: Figure 3.a shows that 99% of overall movements have a length inferior to 40m, which means that the speed of 99% of the avatars does not exceed 20 $m.s^{-1}$.

5.2. Evaluation metrics

To highlight the qualities and the drawbacks of Blue Banana, each experiment depends on the *mobility rate* of the NVE: the proportion of avatars that have a straight and high speed movement, i.e, that are in the state (T)raveling of the mobility state machine (see Section 4). Indeed, these avatars are the ones that need to quickly download data to maintain their continuously and rapidly changing playing areas. The higher the NVE mobility rate is, the faster the underlying overlay has to adapt, which means that high mobility rates are likely to cause a lot of transient failures. The mobility model, tweaked to be close to Second Life traces (see Section 4), has a mobility rate of approximatively 55‰ (which means that the average number of avatars simultaneously in the state T is around 55 per thousand at each moment of the simulation). Therefore, we vary the mobility rate between 5‰ and 110‰ during the evaluation. To achieve that, we vary the probabilities of the transitions that lead to the T state of the trace generator.

The following metrics are used to evaluate mobility resilience of Blue Banana:

- *Violation of Solipsis fundamental rules.* The failure of the global connectivity rule or the local awareness rule leads to transient failures (see the description of Solipsis in Section 3.1).
- *Knowledge of nodes ahead of the movement.* This metric measures, for fast-moving avatars (in state $\tilde{T}$ of the Blue Banana state machine), the average knowledge time of elders: for how long time, in average, a node knows another node ahead of its movement. The *number* of nodes known ahead of the movement is also measured. These measures are important because the NVE application constantly needs to download new information about the playing area of a moving avatar. These measures therefore give an indication of the quantity of information an avatar can retrieve about its future playing area before reaching it.
- *Exchanged messages count.* This metric measures the impact of Blue Banana on the network. The measures only count the number of messages because Blue Banana messages and Solipsis maintainance messages are small: they only contain the coordinates of the prefetched/maintained nodes (a Solipsis identifier, geographic coordinates of its avatar and the IP address of the node).

The evaluation of the second metric only takes in account the subset of avatars in state $\tilde{T}$ of the Blue Banana state machine. This specificity is due to the fact that Blue Banana sends prefetching requests only when an avatar is in this state. Yet, the proportion of that subset of avatars is extremely small: at maximal mobility rate, there are simultaneously only about 110 avatars in the state $\tilde{T}$ for 1000 avatars. The avatars that are not moving do not often change their playing area, thus knowing their elders for a long time. If they were all considered for that metric, the mean values would have been skewed, and the benefits of Blue Banana would have been difficult to evaluate.

5.3. Result analysis

Figure 3 presents the evaluation for the three metrics for Blue Banana (solid lines) compared with Solipsis (dashed lines). The most interesting results for a realistic mobility rate of 55‰ shows that Blue Banana (i) decreases the number of transient failures by 20%, (ii) increases the average knowledge time of forthcoming elders by 270% and (iii) generates a network overhead of only 2%. This positive results are analyzed in detail in the rest of this section.

Violation of Solipsis rules. The first metric evaluation presented in Figure 3.a shows that the Blue Banana prefetching technique helps the Solipsis overlay to adapt itself on time, significantly reducing the number of violations of the Solipsis fundamental rules. With a mobility rate lower than 80‰, Blue Banana decreases the number of transient failures. For the mobility rate observed in real traces (55‰), the Blue Banana algorithm decreases the number of transient failures by 20%. For low mobility rates, Blue Banana helps avoiding approximatively half of the rule-violations. As the mobility rate increases, the efficiency of Blue Banana decreases. This is due to the fact that when the mobility rate increases, the avatars of the prefetched nodes are more likely to move fast and thus to become useless when the avatar reaches their supposed position. For very dynamic NVEs (mobility rate greater than 80‰), Blue Banana stops helping the overlay. Most of the prefetched neighbors are also moving and when they are injected in the regular neighbor set of Solipsis they are useless, forcing Solipsis to find new interesting neighbors. However, this kind of dynamicity is far above the mobility rates observed in real Second Life traces (mobility rate equals 55‰). To

[3] For example, Second Life players are able to fly.

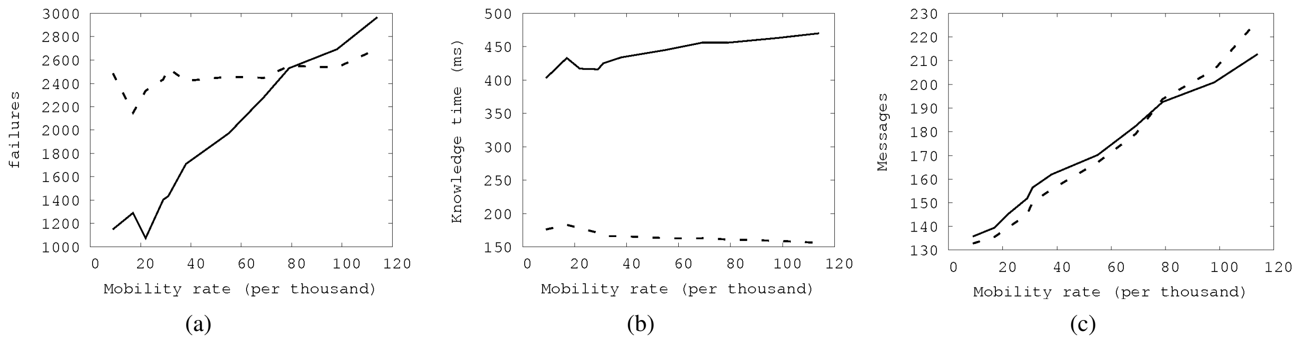

Figure 3. Dashed lines: Solipsis, Solid lines: Blue Banana. (a) Average number of overlay transient failures per second (lower is better), (b) Average knowledge time of nodes ahead of movement (higher is better), (c) Average number of messages sent per node per second (lower is better).

summarize, this first experiment shows that Blue Banana decreases the number of transient failures and suggests that the mobility of prefetched nodes should be taken into account when responding to a prefetching request.

Knowledge time of forthcoming elders. The second metric evaluation presented in Figure 3.b shows that, for fast moving avatars, the knowledge of nodes ahead of the movement is far greater with Blue Banana than with the basic Solipsis overlay: a node knows every neighbor ahead of its movement between 2 and 3 times longer than with Solipsis (2.7 times longer for a real-trace-like mobility rate of 55‰). Moreover, subsidiary measures show that with Blue Banana, the node of an avatar in state *T* is in average aware of 7.5 nodes located ahead of its movement. On the other hand, a basic Solipsis node is in average only aware of one node ahead of its movement. These two results permit the evaluation of the average quantity of information that a fast-moving avatar can download ahead of its movement. By using Blue Banana with the Second-Life-like mobility rate of 55‰, a node has time to download up to about 430 KBytes of information (with a 10down/1up ADSL connection) about its playing area, versus only 20 KBytes without prefetching. This means that the NVE application can display substantially more information (about 20 times more) about the playing area *on time*, thus clearly limiting applicative transient failures.

Network overhead. The last important result of the experimental evaluation is the low network overhead induced by Blue Banana. Figure 3.c shows that this overhead is around five messages per node per second, which is almost negligible compared to the number of messages generated by the Solipsis overlay. Indeed, a basic Solipsis node sends, depending on the mobility of its neighbors, between 130 and 230 maintenance messages per second, thus the network overhead of Blue Banana is approximatively between 1 and 3%. Moreover, these are maintenance messages, with a small, constant size (see the *Exchanged messages count* metric description). This overhead is low thanks to the fact that the prediction technique of Blue Banana is sufficiently accurate. In most of the cases, the prefetching requests provide information about nodes that will be requested in the near future: as these nodes are actually needed, the overlay simply takes them in the prefetched set, without emitting additional messages (see Section 3.2). The little overhead comes then from the wrongly prefetched nodes that are not reused by the overlay. In addition to that, the update interval for rapidly moving avatars is doubled (see also Section 3.2), which also lowers the overhead. This optimization explains that beyond a mobility rate of 80‰, the basic Solipsis overlay generates more messages than Blue Banana: as the mobility rate grows, the proportion of rapidly moving avatars increases. Therefore, the number of economized messages due to the relaxed updating proportionally grows.

6. Related work

Un-adaptable overlays. Considerable research effort has been conducted in the last decade in the field of peer-to-peer overlay networks. However, most of existing overlays do not take specific application needs into account at all [20, 22, 27, 28]. Therefore, building a distributed NVE on top of such an overlay is a hard task: the nodes sharing a same playing area (or a part of a playing area) have no reason to be close in the overlay, and yet they need to communicate a lot because they share a consequent common knowledge. The main reason for this is historical: these overlays have been designed for *one* specific target application: large-scale read-only file-sharing. Therefore, they are supposed to build a graph that connects all the nodes together and permits efficient search operations. Blue Banana anticipation algorithm is not compatible with these overlays.

Overlays reacting to application needs. Recent works have focused on dynamically adapting the overlay to better satisfy the application needs. For instance, semantic overlays [32] build links between semantically close peers. This allows semantically close peers to be close in the overlay, which is a good point because they are likely to interact, for instance to exchange data. Few recent overlays are able to gracefully adapt themselves to the applications [21, 33]: they react to the application evolution, generally by detecting communication between nodes. Solipsis [15] is part of this class of overlays. However, if the application is too dynamic, the reaction of the overlay may come too late which may lead to *inconsistencies* or at least inefficiencies at the application level. These works differ from ours because the overlay adapts itself by reacting to the detected application needs, while we propose to predict and anticipate those needs by adapting the overlay in advance. Nevertheless, our work can easily be implemented on top of any of these overlays.

Overlays for distributed MMOGs. Last years, some research efforts have focused on building overlays tailored for NVEs, but without anticipating application needs. The constraints imposed by the NVE applications are extremely hard to sustain. In particular, the overlay has to be very responsive in order to ensure mobility resilience. Varvello et al. implemented an NVE over a distributed hash table [31]. The authors show that the responsiveness of the DHT is acceptable with light virtual mobility but not if virtual mobility increases. In this case, implementing a reverse binary trie on top of a DHT could help to lower the latency [30]. Colyseus [6], a decentralized architecture to support MMOGs with tight latency constraints (First Person Shooters) is also based on a DHT for virtual object discovery. At storage level, Colyseus prefetches objects. However, this prefetching mechanism is built on top of the DHT's overlay. The overlay itself does not adapt to bring closer the elders from which the object prefetching is done. Blue Banana's main goal is precisely to help an adaptive overlay to support such prefetching mechanisms. Donnybrook, the sequel of Colyseus, takes advantage of elaborated approximations and dead reckoning techniques to decrease the network load [7].

Another approach uses flexible peer-to-peer overlays. In such systems, the logical neighborhood of a node in the overlay is determined by the virtual neighborhood of its avatar in the NVE: for each node, the overlay tries to keep the elders of the playing area in the node's neighbor set. As that avatar moves in the virtual environment, the logical neighborhood of its node evolves: the overlay adapts itself in reaction to the application. Thanks to that, the logical neighborhood of every node in the overlay will eventually be adapted to the virtual neighborhood of its avatar: each node will know the elders of its playing area. Several overlays of that kind have been designed in the past few years: this is the case for Solipsis [15] on which we have experimented Blue Banana, and of Voronoi tessellations-based overlays like VoroNet/RayNet or VON [3, 4, 12].

However, to our knowledge, none of these overlays anticipate the application needs. There again, our algorithm can be implemented on top of any of these overlays to allow them to anticipate application needs and adapt in advance.

7. Conclusions and perspectives

This paper presents a study of avatar mobility in existing NVEs and proposes a model that provides the ability to generate arbitrary-scale traces. We then show that even if the overlay tries to remain adapted to the application by reacting to avatar movement, the NVE suffers many transient failures due to the lateness of the overlay adaptation. Thus, we propose Blue Banana: a mechanism that predicts avatar movement and anticipates it by adapting the overlay in advance. We show that our anticipation mechanism cuts down by more than 20% the number of transient failures affecting the state-of-the-art Solipsis overlay without degrading its network performance. Moreover, we show that our anticipation mechanism permits to load 20 times more data about playing areas in case of mobility. We believe that our study can be used in the design of future MMOG overlays.

As a perspective, we plan to study more accurate anticipation mechanisms, and particularly to explore the possibility to anticipate the relative movement between avatars, independently from their position. This study should lower the number of transient failures, even in case of high mobility rates.

References

[1] M. Agosti, F. Zanichelli, M. Amoretti, and G. Conte. P2pam: a framework for peer-to-peer architectural modeling based on peersim. In S. Molnár, J. Heath, O. Dalle, and G. A. Wainer, editors, *SimuTools*, page 22. ICST, 2008.

[2] R. T. Alves and L. Roque. Because players pay: The business model influence on mmog design. In B. Akira, editor, *Situated Play: Proc. of the 2007 Digital Games Research Association Conference*, pages 658–663, Tokyo, September 2007. The University of Tokyo.

[3] O. Beaumont, A.-M. Kermarrec, L. Marchal, and E. Riviere. Voronet: A scalable object network based on voronoi tessellations. In *21th International Parallel and Distributed Proc. Symposium (IPDPS 2007), Long Beach, USA*, pages 26–30. IEEE, March 2007.

[4] O. Beaumont, A.-M. Kermarrec, and E. Riviere. Peer to peer multidimensional overlays: Approximating complex structures. In E. Tovar, P. Tsigas, and H. Fouchal, editors, *OPODIS*, volume 4878 of *LNCS*, pages 315–328. Springer, 2007.

[5] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The effects of loss and latency on user

performance in unreal tournament 2003. In W. chang Feng, editor, *NETGAMES*, pages 144–151. ACM, 2004.

[6] A. Bharambe, J. Pang, and S. Seshan. Colyseus: a distributed architecture for online multiplayer games. In *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*, pages 12–12, Berkeley, CA, USA, 2006. USENIX Association.

[7] A. R. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. Donnybrook: enabling large-scale, high-speed, peer-to-peer games. In V. Bahl, D. Wetherall, S. Savage, and I. Stoica, editors, *SIGCOMM*, pages 389–400. ACM, 2008.

[8] R. Brunet. Lignes de force de l'espace Européen. *Mappemonde*, 66:14–19, 2002.

[9] A. Chechkin, V. Gonchar, J. Klafter, and R. Metzler. Fundamentals of lévy flight processes. *Advances in Chemical Physics*, 133B:439–496, 2006.

[10] C. Comito, S. Patarin, and D. Talia. A semantic overlay network for p2p schema-based data integration. In P. Bellavista, C.-M. Chen, A. Corradi, and M. Daneshmand, editors, *ISCC*, pages 88–94. IEEE Computer Society, 2006.

[11] D. Frey, J. Royan, R. Piegay, A. Kermarrec, E. Anceaume, and F. L. Fessant. Solipsis: A decentralized architecture for virtual environments. In *The Second International Workshop on Massively Multiuser Virtual Environments at IEEE Virtual Reality (MMVE' 09)*, Lafayette, USA, March 2008.

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

[13] C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors. *Artificial Immune Systems: 4th International Conference, ICARIS 2005, Banff, Alberta, Canada, August 14-17, 2005,*, volume 3627 of *LNCS*. Springer, 2005.

[14] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The Peersim simulator. `http://peersim.sourceforge.net/`.

[15] J. Keller and G. Simon. Solipsis: A massively multiparticipant virtual world. In H. R. Arabnia and Y. Mun, editors, *PDPTA*, pages 262–268. CSREA Press, June 2003.

[16] S. Kumar, J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim. Second life and the new generation of virtual worlds. *Computer*, 41(9):46–53, 2008.

[17] C.-A. La and P. Michiardi. Characterizing user mobility in Second Life. In *SIGCOMM 2008, ACM Workshop on Online Social Networks, August 18-22, 2008, Seattle, USA*, August 2008.

[18] S. Legtchenko, S. Monnet, and G. Thomas. Blue Banana: resilience to avatar mobility in distributed MMOGs. Technical Report 7149, INRIA, December 2009.

[19] H. Liang, I. Tay, M. F. Neo, W. T. Ooi, and M. Motani. Avatar mobility in networked virtual environments: Measurements, analysis, and implications. *CoRR*, abs/0807.2328, 2008.

[20] J. Liang, R. Kumar, and K. Ross. The kazaa overlay: A measurement study. In *Proc. of the 19th IEEE Annual Computer Communications Workshop*, 2004.

[21] S. Monnet, R. Morales, G. Antoniu, and I. Gupta. Move: Design of an application-malleable overlay. In *Symposium on Reliable Distributed Systems 2006 (SRDS 2006)*, pages 355–364, Leeds, UK, October 2006.

[22] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter Gnutella, pages 94–122. O'Reilly, May 2001.

[23] J. Pang, F. Uyeda, and J. R. Lorch. Scaling peer-to-peer games in low-bandwidth environments. In *IPTPS '07: Proc. of the 6th International Workshop on Peer-to-Peer Systems*, Feb. 2007.

[24] L. Pantel and L. C. Wolf. On the suitability of dead reckoning schemes for games. In L. C. Wolf, editor, *NETGAMES*, pages 79–84. ACM, 2002.

[25] D. Pittman and C. GauthierDickey. A measurement study of virtual populations in massively multiplayer online games. In *NetGames '07: Proc. of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 25–30, New York, NY, USA, 2007. ACM.

[26] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong. On the levy-walk nature of human mobility. In *INFOCOM*, pages 924–932. IEEE, 2008.

[27] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, volume 2218 of *LNCS*, pages 329–250, Heidelberg, Germany, November 2001. Springer.

[28] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the Symposium on Communications Architectures and Protocols (SIGCOMM '01)*, pages 149–160, San Diego, USA, August 2001.

[29] D. Thalmann, N. Magnenat-Thalmann, and I. S. Pandzic. *Avatars in Networked Virtual Environments*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[30] M. Varvello, C. Diot, and E. W. Biersack. A walkable kademlia network for virtual worlds. In *Infocom 2009, 28th IEEE Conference on Computer Communications, April 19-25, 2009, Rio de Janeiro, Brazil*, 04 2009.

[31] M. Varvello, C. Diot, and E. W. Biersack. P2P Second Life: experimental validation using Kad. In *Infocom 2009, 28th IEEE Conference on Computer Communications*, pages 19–25, Rio de Janeiro, Brazil, April 2009.

[32] S. Voulgaris, A. M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, Suzhou, China, May 2004.

[33] S. Voulgaris, E. Riviere, A.-M. Kermarrec, and M. van Steen. Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collborative networks. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, USA, February 2006.

[34] G. K. Zipf. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge MA, 1949.

[35] Second Life. `http://secondlife.com/`.

[36] World of Warcraft. `http://www.worldofwarcraft.com/`.