Approximation Algorithm for Estimating Distances in Distributed Virtual Environments

Olivier Beaumont[1,2], Tobias Castanet[1], Nicolas Hanusse[1,3], and Corentin Travers[1,4]

[1] LaBRI, 351, cours de la Libération F-33405 Talence cedex France
[2] Inria Bordeaux - Sud-Ouest, 200, avenue de la Vieille Tour 33405 Talence cedex, France
[3] CNRS
[4] Bordeaux INP

Abstract. This article deals with the issue of guaranteeing properties in Distributed Virtual Environments (DVEs) without a server. This issue is particularly relevant in the case of online games, that operate in a fully distributed framework and for which network resources such as bandwidth are the critical resources. Players typically need to know the distance between their character and other characters, at least approximately. They all share the same position estimation algorithm but, in general, do not know the current positions of others. We provide a synchronized distributed algorithm $\mathcal{A}_{lc}$ to guarantee, at any time, that the estimated distance d_{est} between any pair of characters A and B is always a $1+\varepsilon$ approximation of the current distance d_{act}, regardless of movement pattern, and then prove that if characters move randomly on a d-dimensional grid, or follow a random continuous movement on up to three dimensions, the number of messages of $\mathcal{A}_{lc}$ is optimal up to a constant factor. In a more practical setting, we also show that the number of messages of $\mathcal{A}_{lc}$ for actual game traces is much less than the standard algorithm sending actual positions at a given frequency.

Keywords: Distributed Virtual Environments · Online games · Random walks · Distributed approximation algorithms · Peer-to-peer algorithms

1 Introduction

1.1 Context

The term *Distributed Virtual Environment* (DVE) refers to systems where geographically distant users, or players, participate in a highly interactive virtual world. The main examples of DVEs are online games, where players control characters that interact with each other, and may modify the shared environment. Usually, interactions between characters and/or objects of the environment are enabled when they are sufficiently close in the virtual world. For simplicity, in the rest of the paper, we will use *player* to denote both the player and the character.

The main difference between a DVE and a classical distributed system like a database, is that the states of objects in the virtual environment evolve even without changes issued by the users [14] since non-player characters go about their programmed activities, and objects must respect the physics of the game. Moreover, the amount of inputs per time unit is generally high, as players interact a lot with the environment.

DVE participants need to know the state of the virtual world, in order to display it correctly and to be able to interact with it. The two central aspects that need to be optimized in a DVE are consistency and responsiveness. Inconsistencies arise when two users see different versions of the virtual world. On the other hand, responsiveness, the time interval between when a user executes an action and when the effects of this action is perceived by the player is unsatisfactory when this time delay is noticeable.

One difficulty is related to the number of exchanged messages. In general, increasing the number of communications between players contributes both to responsiveness (changes are transmitted earlier) and consistency (more messages allow a more accurate knowledge of the game's state). On the other hand, it has been shown in [13] that too many messages degrade network performance, leading to inconsistencies.

In practice, many games rely on a simple strategy, where players send updates at a regular rate to other players. The main flaw of this technique is a poor scalability in terms of bandwidth, as the number of messages increases quadratically with the number of players. Scalability is a concern for DVEs: some games are intended to be played by a large number of participants at the same time (e.g. MMORPGs). In addition, many online games are based on a client-server architecture. This has many disadvantages, as maintaining a server is often expensive, and exposes a single point of failure [16]. This leads to the incentive to study peer-to-peer solutions, where players share the role of the server among themselves, but in this context, bandwidth becomes crucial, as the network capacities of peers are usually lower than those of powerful servers. This article focuses on reducing bandwidth usage by limiting the number of exchanged messages. Several versatile techniques have been proposed to achieve this goal.

Data compression regroups techniques that can reduce bandwidth usage, but that are dependent on the application. For example Delta encoding [16], is an implementation trick where only differences between states are sent.

Dead-reckoning is a widely used tool, standardized in the Appendix E of [3]. Each player predicts the positions of the other players, extrapolating their movements after each update, typically based on their speed and acceleration.

Error induced by dead-reckoning can be measured by different means [4, 17], but Dead-reckoning aims at bounding *the additive error* on the players positions. The players know their own actual positions at any time, and for the other players, they only know estimated positions. Since all the players share the same estimation algorithm, each player is able to detect if the error on his/her own position as seen by another player is above a given threshold. When this happens, *the player sends a message to this player to correct the outdated estimated position*. Research on dead-reckoning improved bandwidth usage mainly in two ways : get the best prediction possible [10], or improve the update policies (a survey on different update policies is given in [15]).

Interest Management consists in filtering updates in order to send them only to players who might be interested. Different types of interest management are identified in [7,12]. Some application-specific approaches may also use the fact that human attention is limited, as in [6], where a set of five interesting players is defined at any given time, in order to send frequent updates to those players, but much less to other players.

Combinations of all these techniques can be used. In [8], an area of interest, similar to aura interest management, is used to modify the Dead-reckoning threshold.

In the context of interest management, estimating distances between players is very useful, as a player is rarely interested in knowing the exact state of far away objects. In addition, in some application-specific cases, distance may be important, for example when implementing a spell that heals all allies within a certain range. To the best of our knowledge, no distributed algorithm has been proposed to solve the problem of estimating the distance between users of a DVE. The objective of this paper is *to provide a solution allowing players to estimate the distances between them, with a condition on the relative error, while guaranteeing that the use of bandwidth is as small as possible*. In particular, it has to be bounded against an ideal algorithm that would send a minimum number of messages, based on a perfect knowledge of the game's state.

We identify two main articles related to this objective:

Timewarp. In [14], two techniques are proposed. First, *local-lag* reduces short-term inconsistencies, at the cost of less responsiveness: a delay between the time an operation is issued and the time when the operation becomes effective is added. Secondly, `timewarp` is proposed, an algorithm to ensure consistency. In this algorithm, each player remembers all previous operations and the time at which they were issued. If an operation is received by a player too late, the player rewinds the state of the world, immediately recomputing the current state, using all needed operations.

Compensatory Dead-Reckoning. In [11], Dead-Reckoning is used to compensate for latencies and message losses on the network. TATSI, the average spatial error on players' positions over a time interval, is estimated with no latency or loss of message. Then, under the assumption of a constant acceleration, latencies and message losses are added to the model, and it is shown that the same TATSI can be obtained by lowering the dead-reckoning threshold (thus making DVE nodes send more messages than without latency and message losses).

To summarize, solutions from the literature are very consuming in term of messages and target an *additive bound* on the error. By contrast, this paper focuses on bounding the *relative error* on distances and keeping the number of message exchanges low.

1.2 Contribution

In terms of optimality in number of messages, Dead-reckoning is optimal for position estimation. Indeed, when using Dead-reckoning, players know where other players see them. Thus, a player sends updates if and only if the tolerated error between his/her actual position and his/her estimated position is exceeded, making it an optimal bandwidth strategy. On the other hand, since no two players know the actual distance between them, none of them can determine the exact error over the estimated distance, making distance estimation a much harder problem.

We consider deterministic algorithms that allow each player to estimate, at any time, the distances between him/her and the other players, while having a guarantee on the error. Initially, each player knows the exact position of every other player. The metric we use is the relative error given in Equation 1, where, at each instant t, $d_{act}(t)$ denotes the actual distance between two players, and $d_{est}(t)$ denotes their estimated distance,

$$\text{error measurement } = |d_{act}(t) - d_{est}(t)|/d_{est}(t). \tag{1}$$

We make sure this error measurement never exceeds ε, the maximum tolerated relative error for any pair of players, while minimizing the number of exchanged messages.

That is, Equation 2 must always hold, for every pair of players,

$$(1-\varepsilon)d_{est}(t) < d_{act}(t) < (1+\varepsilon)d_{est}(t). \tag{2}$$

We propose an algorithm, called *local change* and denoted by $\mathcal{A}_{lc}$. It relies on the same underlying principle as Dead-reckoning, where position estimations are deterministic and each player computes his/her own position as seen by other players, using the same deterministic algorithm. In $\mathcal{A}_{lc}$, player Bob sends his actual position p_{actB} to another player Alice as soon as the estimate p_{estB} of the position of Bob as seen by Alice deviates too much from his actual position, *more precisely as soon as Equation 3 is violated*, where d denotes the distance between two points. In addition, Alice will *immediately respond to Bob by also sending her actual position*.

$$d\left(p_{actB}(t),\ p_{estB}(t)\right) < d_{est}(t) \times \varepsilon/2. \tag{3}$$

When there is no latency, with $\mathcal{A}_{lc}$, without any assumption on how players move, the maximal error is never overcome: Equation 2 is always satisfied (Theorem 1).

To quantify the performance of our algorithm, we compare it against an oracle with a full knowledge of the current state of the game, called *ideal algorithm* and denoted by $\mathcal{A}_{id}$. In $\mathcal{A}_{id}$, an exchange of messages happens only when, and as soon Equation 2 is violated.

We focus in this paper on the case where movement is limited to the random part based on players' actions, which cannot be anticipated by the deterministic prediction algorithm. Thus, we first conduct theoretical analyses in which players move randomly. The best possible estimation of the position of other players is to assume they remain still, so that a player will estimate that the other players are at their last known position.

Our analyses are for two types of movement patterns. With both, players will take turns (with a simple round-robin policy) to move on discrete time: at each instant $t \in \mathbb{N}$, the player which it is the turn chooses randomly a direction to move.

Random Walk is a discrete movement taking place on a d-dimensional grid. Thus, positions can be represented as values from $\mathbb{Z}^d$. If at instant t, a player following such movement is at position $p = (p_1, p_2, \ldots, p_d)$ he/she has $2d$ neighbors: $(p_1-1, p_2, \ldots, p_d)$, $(p_1+1, p_2, \ldots, p_d)$, $(p_1, p_2-1, \ldots, p_d)$, etc. The movement consists, at integer instants, to chose one of the neighbors, each one having probability $\frac{1}{2d}$ to be chosen.

Continuous Movement consists at each turn to select a value smaller than one, and to add a vector of norm equal to this value, and with a direction randomly chosen. **In 1D**, a moving player adds at his turn a random number following a uniform distribution on $[-1, 1]$ to their position. **In 2D**, at each instant t, the moving player X chooses ρ_t and θ_t following uniform distributions respectively on $[0, 1]$ and $[0, 2\pi]$, so that $p_{actX}(t+1) = p_{actX}(t) + (\rho_t, \theta_t)$, where (ρ_t, θ_t) is the vector with polar coordinates ρ_t and θ_t. **In 3D**, at each instant t, the moving player chooses ρ_t, θ_t, and φ_t following uniform distributions respectively on $[0, 1]$, $[0, 2\pi]$ and $[0, \pi]$.

For these movement patterns, we prove that $\mathcal{A}_{lc}$ is optimal in terms of number of message exchanges up to a constant factor depending only on ε (and not in particular on the initial distance between the players).

This theoretical analysis is then complemented by experiments. We first performed experiments on synthetic traces in which players follow random walks or continuous movements in 1D, 2D or 3D, and then performed experiments on actual traces from Heroes of Newerth [1], where we compare $\mathcal{A}_{lc}$ with a *fixed frequency algorithm*, denoted by $\mathcal{A}_{ff}$. $\mathcal{A}_{ff}$ is commonly used in practice in online games, and sends updates periodically, by waiting w time units between updates. We show that overall, $\mathcal{A}_{lc}$ behaves better while never exceeding the maximal tolerated error.

In summary, the performance (without latency) of $\mathcal{A}_{id}$, $\mathcal{A}_{lc}$, $\mathcal{A}_{ff}$ and `timewarp` [14] are shown in the following table:

	number of messages	maximal error	number of violations
$\mathcal{A}_{id}$	$m_{id} \leq Tn(n-1)$	$\leq \varepsilon$	0
$\mathcal{A}_{lc}$	$O(m_{id})$	$\leq \varepsilon$	0
$\mathcal{A}_{ff}$	$\frac{T}{w}n(n-1)$	0 if $w = 1$ unbounded otherwise	$\Theta(Tn^2)$
`timewarp`	$O(Tn^2)$	0	0

T denotes the duration of the experiment, and n the number of participants in the DVE. We consider as a reference m_{id}, the (perfect knowledge based) number of messages sent by $\mathcal{A}_{id}$. In the worst case, $\mathcal{A}_{id}$ would make players send one message each instant (when movement is large compared to the distance), thus $m_{id} \leq Tn(n-1)$. Also, `timewarp` functions slightly differently than the others: it is intended to ensure strict consistency, messages are sent when users initiate a change in the game's state; thus, the number of messages is proportional to the number of players, and to the length of time. The *number of violations* counts, over T time units, the number of distance pairs for which the error is above ε.

Organisation: in Section 2, we describe $\mathcal{A}_{lc}$, and the model used for the theoretical analysis. In Section 3 and Section 4, we prove that with the two movement patterns, $\mathcal{A}_{lc}$ is optimal up to a constant factor (due to space limitations, 1D and 2D-cases are left for the interested reader in [5]). Experimental results are presented in Section 5, and conclusions and perspectives are given in Section 6.

2 Model and Algorithms

Model: Let us first assume that $\varepsilon \in]0;1[$. Indeed, $\varepsilon = 0$ means that no error is tolerated, while $\varepsilon = 1$ would accept any estimate on the distance, provided it is larger than half the actual distance, which is not very informative. Since $\mathcal{A}_{lc}$ must enforce that Equation 3 holds true for any pair of players, we focus on two players Alice and Bob. We assume that the communication channel connecting them is without message loss nor latency, that local computations do not take time and that all players share a synchronized clock. At any instant t ($t \in \mathbb{N}$), let us denote the positions of both players as $p_{actA}(t)$ and $p_{actB}(t)$. A position is a vector whose dimension depends on the virtual world (for example, for a 3D world, a position is described by a vector in $\mathbb{N}^3$, or $\mathbb{R}^3$ in the case of continuous moves). Each player knows his/her own actual position, but may not know exactly where the other player is. These positions can change unpredictably, through the actions of users.

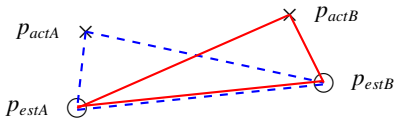


Fig. 1: Knowledge of Alice (dashed blue lines) and Bob (continuous red lines)

Algorithm 1 Local change ($\mathcal{A}_{lc}$), from the point of view of Alice

1: $\boldsymbol{p_{actA}} \leftarrow$ Alice's initial position ▷ Actual position of Alice. This is a read-only input to the algorithm
2: $p_{estA} \leftarrow$ Alice's initial position ▷ Position of Alice, as estimated by Bob, the other player
3: $p_{estB} \leftarrow$ Bob's initial position ▷ Estimated position of Bob
4: $d_{est} \leftarrow d(p_{estA}, p_{estB})$ ▷ Estimated distance. Will always be equal to $d(p_{estA}, p_{estB})$
5: **procedure** CHECK_FOR_UPDATE ▷ to be called at each $t \in \mathbb{N}$, after movement
6: **if** $d(\boldsymbol{p_{actA}}, p_{estA}) \geq \frac{\varepsilon}{2} d_{est}$ **then**
7: $p_{estA} \leftarrow \boldsymbol{p_{actA}}$
8: $d_{est} \leftarrow d(p_{estA}, p_{estB})$
9: send message ($\boldsymbol{p_{actA}}$, begin_update) to Bob
10: **end if**
11: **procedure** RECEIVE_MESSAGE(position, type) from Bob ▷ to be called when receiving a message
12: $p_{estB} \leftarrow$ position
13: $d_{est} \leftarrow d(p_{estA}, p_{estB})$
14: **if** type = begin_update **then** ▷ type distinction is to avoid infinite messages
15: send message ($\boldsymbol{p_{actA}}$, update_reply) to Bob
16: **end if**

Algorithm: As explained in Section 1.2, players will estimate their distance to each other. To do this, each player will compute a deterministic estimation of the other player's position, in order to get $d_{est}(t)$, i.e. Bob computes $p_{estA}(t)$, the estimate of the position of Alice, and Alice computes $p_{estB}(t)$. As they use the same deterministic algorithm, these computations can be replicated, and $p_{estA}(t)$ and $p_{estB}(t)$ become a shared knowledge, as seen on Figure 1 (even without communication). Thus, we will use the distance between those two (estimated but shared) positions as distance estimate, $d_{est}(t)$. In practice, $p_{estA}(t)$ is generally based on an extrapolation of Alice's position, speed and acceleration, from the time of the last message exchanged between Alice and Bob.

In Theorem 1, we prove that $\mathcal{A}_{lc}$ satisfies Equation 2, provided that $\mathcal{A}_{lc}$ sends an update of the actual position as soon as Equation 3 is not satisfied, as depicted in Algorithm 1. Thus, the correctness of $\mathcal{A}_{lc}$ is established.

Theorem 1. *Using $\mathcal{A}_{lc}$, Equation 2 holds true at any instant (regardless of movement).*

Proof. The following inequalities hold true:

$$d_{act}(t) - d_{est}(t) \leq d(p_{actA}(t), p_{estA}(t)) + d(p_{actB}(t), p_{estB}(t)) \text{ (triangle inequality)}$$
$$d_{est}(t) - d_{act}(t) \leq d(p_{actA}(t), p_{estA}(t)) + d(p_{actB}(t), p_{estB}(t)) \text{ (triangle inequality)}$$

$$d(p_{actB}(t), p_{estB}(t)) < \frac{\varepsilon}{2} d_{est}(t) \text{ (by construction)}$$
$$d(p_{actA}(t), p_{estA}(t)) < \frac{\varepsilon}{2} d_{est}(t) \text{ (by construction)}$$

so that $|d_{act}(t) - d_{est}(t)| < \varepsilon d_{est}(t)$, which is equivalent to Equation 2. □

3 Random Walk

The performance of $\mathcal{A}_{lc}$ is measured by M, the number of message exchanges (a message and its response counting as one) between two players using $\mathcal{A}_{lc}$, before the first message sent with $\mathcal{A}_{id}$. We prove in this section that with random walks, M is upper bounded by a constant value depending only on ε. This is formally stated in Theorem 2:

Theorem 2. *Let* $\Delta_r = \left\lceil \frac{\log(1+\varepsilon)-\log(1-\varepsilon)}{\log(1+\frac{\varepsilon}{2})} \right\rceil$*, with* $\varepsilon \in]0;1[$*. For any two players following a random walk on* $\mathbb{Z}^d$ *(with* $d \leq 3$*),* $\mathbb{E}[M] \leq \Delta_r \times \left(2^{d+1}\right)^{\Delta_r}$*. Moreover, if only one of the players moves on* $\mathbb{Z}$*, with* $\Delta_l = \left\lceil \frac{\log(1-\varepsilon)-\log(1+\varepsilon)}{\log(1-\frac{\varepsilon}{2})} \right\rceil$*, then* $\mathbb{E}[M] \leq \min\left(\Delta_l \times 2^{\Delta_l} \; ; \; \left\lceil \frac{4}{\pi} {\Delta_l}^2 \right\rceil \times 8\right)$.

Due to space limitations, only the 2D-case with two players moving will be proved in this Section. 1D and 3D cases can be found in [5].

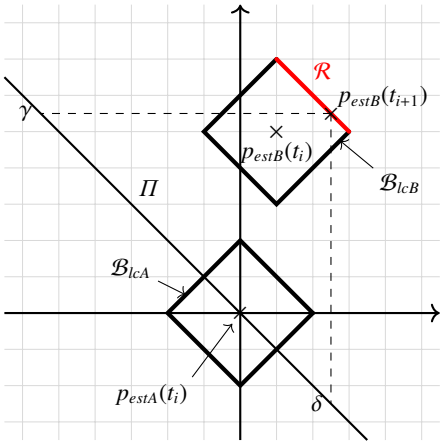
As seen in Section 1.2, a 2D random walk consists, at each instant $t \in \mathbb{N}$, for a moving player to add one of the following vector to their position: $(-1, 0)$, $(1, 0)$, $(0, -1)$, or $(0, 1)$. With two players, one does this for t even, while the other when t is odd. For our analysis, we will use the L^1 distance (Manhattan distance).

Let us denote by d_{est} and p_{est} the estimates for $\mathcal{A}_{lc}$. Algorithm $\mathcal{A}_{lc}$ generates a message exchange as soon as for a player X, p_{actX} leaves $\mathcal{B}_{lcX}$, where $\mathcal{B}_{lcX}$ is the L^1-ball of radius $d_{est}\left(1 - \frac{\varepsilon}{2}\right)$, and of center p_{estX} (see Figure 2a). As seen in Section 1.2 Algorithm $\mathcal{A}_{id}$ generates a message exchange as soon as Equation 2 becomes false (this is the definition of $\mathcal{A}_{id}$). Equivalently, $\mathcal{A}_{id}$ generates a message exchange as soon as d_{act} leaves $\mathcal{I}_{id}$, where $\mathcal{I}_{id}$ is defined by $\mathcal{I}_{id} =]d_0\,(1-\varepsilon) \; ; \; d_0\,(1+\varepsilon)[$, with $d_0 = d_{act}(0)$.

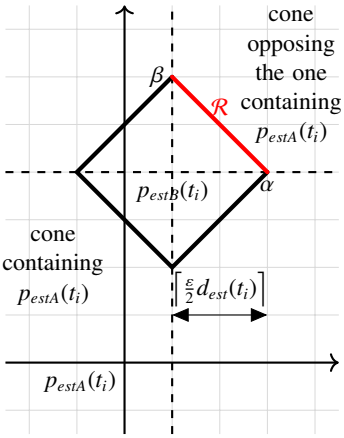
Let us consider t_i (with $i \geq 1$), defined as the instant at which the i-th round trip of the messages is sent with $\mathcal{A}_{lc}$. With $t_{opt} = \min\{t : d_{act}(t) \notin \mathcal{I}_{id}\}$, the time of the first message sent by $\mathcal{A}_{id}$, we have $M = \max\{i, \; t_i \leq t_{opt}\}$. We may then define the auxiliary random variable $M' : \min\{i, \; d_{est}(t_i) \notin \mathcal{I}_{id}\}$. M' represents the index of the first message of $\mathcal{A}_{lc}$ so that d_{act} is outside $\mathcal{I}_{id}$. At this instant, by construction, $\mathcal{A}_{id}$ already sent a message, thus $M' \geq M$.

As players take turns in moving, they cannot simultaneously go out of their $\mathcal{B}_{lcX}$. Thus, let us assume without loss of generality, that Bob is the player that triggers the $(i+1)$-th message, by getting out the first of $\mathcal{B}_{lcB}$ at instant t_{i+1}. Similarly, as we are interested only in the positions of Alice and Bob relatively to each other, we can always put the center of the coordinates on p_{estA}; thus, at each instant, $p_{estA}(t) = 0$.

Each time t_i Bob gets out of $\mathcal{B}_{lcB}$, we will have a new estimated distance $d_{est}(t_i)$. Before being able to identify the effect a message has on the estimated distance (Lemma 4), we have to look, in Lemma 1 at the change in positions.



(a) When Bob gets on $\mathcal{R}$, half of the possible positions of Alice are further away.



(b) One of the outer bounds of $\mathcal{B}^+_{lc}$ is always sufficiently far away from $p_{estA}(t_i)$.

Fig. 2: Random walk, two-dimensional situation

Lemma 1. *With $\mathcal{A}_{lc}$, the $(i+1)$-th message is sent when Bob is on the border of the L^1-ball of center $p_{estB}(t_i)$, and of radius $\left\lceil \frac{\varepsilon}{2} d_{est}(t_i) \right\rceil$.*

Proof. With $\mathcal{A}_{lc}$, the $(i+1)$-th message is sent when Bob gets out of $\mathcal{B}_{lcB}$, which has a radius of $\frac{\varepsilon}{2} d_{est}(t_i)$, but as movement is on integer positions, the first positions outside of $\mathcal{B}_{lcB}$ are all on the L^1-ball of center $p_{estB}(t_i)$, and of radius $\left\lceil \frac{\varepsilon}{2} d_{est}(t_i) \right\rceil$. □

This ball has 4 faces. We may draw cones over each of these faces, with $p_{estB}(t_i)$ as the apex: all points of the space will be in only one of the cones, except for points on the borders (see Figure 2b, where the borders of the cones are the dashed lines). Let us call $\mathcal{R}$ the face that is included in the cone opposing the cone (or one of the cones) containing $p_{estA}(t_i)$.

Lemma 2. *If $p_{estB}(t_i) \neq p_{estA}(t_i)$, $\mathbb{P}\left(d(p_{estA}(t_i), p_{estB}(t_{i+1})) \geq \left\lceil d_{est}(t_i)\left(1+\frac{\varepsilon}{2}\right)\right\rceil\right) \geq \frac{1}{4}$*

Proof. All points of $\mathcal{R}$ are at distance $\left\lceil d_{est}(t_i)\left(1+\frac{\varepsilon}{2}\right)\right\rceil$ of $p_{estA}(t_i)$. As the random walk is symmetric, and by Lemma 1, we have a probability of at least $\frac{1}{4}$ that Bob sends the $(i+1)$-th message by going on face $\mathcal{R}$. □

In Lemma 2, the movement of Alice is not taken into account. Let us call Π the line parallel to $\mathcal{R}$ and containing $p_{estA}(t_i)$ (see Figure 2a).

Remark 1. As Π contains $p_{estA}(t_i)$, the center of $\mathcal{B}_{lcA}$, Π divides $\mathcal{B}_{lcA}$ into two halves of same area.

Lemma 3. *At least half of the $p \in \mathcal{B}_{lcA}$ satisfy $d(p, p_{estB}(t_{i+1})) \geq d(p_{estA}(t_i), p_{estB}(t_{i+1}))$.*

Proof. By definition of the L^1-norm, and because Π is parallel to $\mathcal{R}$, if we draw, on Π, the points γ and δ that are the projections of $p_{estB}(t_{i+1})$ parallel to the two axes (see Figure 2a), then all points on the line segment $[\gamma\delta]$ are at the same distance to $p_{estB}(t_{i+1})$. Also, by definition of $\mathcal{R}$, $p_{estA}(t_i) \in [\gamma\delta]$. Thus, all points of $[\gamma\delta]$ are at a distance to $p_{estB}(t_{i+1})$ equal to $d(p_{estA}(t_i), p_{estB}(t_{i+1}))$.

If we draw the L^1-ball of center $p_{estB}(t_{i+1})$ and of radius $d(p_{estA}(t_i), p_{estB}(t_{i+1}))$, then $[\gamma\delta]$ is one of the faces of the ball. By Remark 1, we have that at least half of the points from $\mathcal{B}_{lcA}$ are outside this ball, with a distance to $p_{estB}(t_{i+1})$ higher than the radius. □

Lemma 4. *As long as* $p_{estB}(t_i) \neq p_{estA}(t_i)$, $\mathbb{P}\left(d_{est}(t_{i+1}) \geq \left\lceil d_{est}(t_i)\left(1+\frac{\varepsilon}{2}\right)\right\rceil\right) \geq \frac{1}{8}$

Proof. As Alice does not get out of $\mathcal{B}_{lcA}$, we know that $p_{estA}(t_{i+1}) \in \mathcal{B}_{lcA}$. By Lemma 3, and by symmetry of the random walk, $d(p_{estA}(t_{i+1}), p_{estB}(t_{i+1})) \geq d(p_{estA}(t_i), p_{estB}(t_{i+1}))$ with probability $\frac{1}{2}$. Combined with Lemma 2, we get the result. □

Let $r_{rw} : x \mapsto \left\lceil x\left(1+\frac{\varepsilon}{2}\right)\right\rceil$. We can now prove Lemma 5 which states that, if there are enough successive messages so that $d_{est}(t_{i+1}) \geq r_{rw}(d_{est}(t_i))$, then Bob will get out of $\mathcal{I}_{id}$, whatever his initial position in the interval $\mathcal{I}_{id}$.

Lemma 5. *For all* $x \in \mathcal{I}_{id}$, ${r_{rw}}^{\Delta_r}(x) \geq d_0(1+\varepsilon)$.

Proof. $x \in \mathcal{I}_{id} \Rightarrow x \geq d_0(1-\varepsilon) \Rightarrow {r_{rw}}^{\Delta_r}(x) \geq {r_{rw}}^{\Delta_r}(d_0(1-\varepsilon))$ since r_{rw} is increasing, implying that ${r_{rw}}^{\Delta_r}(x) \geq d_0(1-\varepsilon)\left(1+\frac{\varepsilon}{2}\right)^{\Delta_r}$ since $\forall x$, $r_{rw}(x) \geq x\left(1+\frac{\varepsilon}{2}\right)$. Moreover, since, $\Delta_r \geq \frac{\log(1+\varepsilon)-\log(1-\varepsilon)}{\log\left(1+\frac{\varepsilon}{2}\right)}$, then $(1-\varepsilon)\left(1+\frac{\varepsilon}{2}\right)^{\Delta_l} \geq (1+\varepsilon)$ and $x \in \mathcal{I}_{id} \Rightarrow {r_{rw}}^{\Delta_r}(x) \geq d_0(1+\varepsilon)$ □

Proof. We can now prove the 2D-case of Theorem 2. Let us split the sequence of all the instants t_i into *phases* of length Δ_r and let us denote by j the index of the phase containing instants from $t_{(j-1)\Delta_r}$ to $t_{j\Delta_r-1}$. Let us consider the following possible events (i) $\mathcal{S}_j$: there is at least one $i \in [\![(j-1)\Delta_r; j\Delta_r]\!]$ such that $d_{est}(t_i) \notin \mathcal{I}_{id}$ and (ii) $\mathcal{S}'_j$: for all $i \in [\![(j-1)\Delta_r; j\Delta_r - 1]\!]$, $d_{est}(t_{i+1}) \geq r_{rw}(d_{est}(t_i))$. In turn, these events can be used to define useful random variables: (i) $X_j = 1$ if $\mathcal{S}_j$ is true, 0 otherwise (ii) $X'_j = 1$ if $\mathcal{S}'_j$ is true, 0 otherwise, (iii) $Y = j$ if $X_j = 1$ and $X_k = 0$ for every $k < j$ and (iv) $Y' = j$ if $X'_j = 1$ and $X'_k = 0$ for every $k < j$. Thus, Y denotes the index of the first phase during which $\mathcal{A}_{id}$ sends a message.

If $\mathcal{S}'_j$ is true, then $d_{est}(t_{j\Delta_r}) = l^{\Delta_r}(d_{est}(t_{(j-1)\Delta_r}))$. Thus, by Lemma 5, $\mathcal{S}'_j \Rightarrow \mathcal{S}_j$, so that $X'_j = 1 \Rightarrow X_j = 1$.

$$\text{Therefore } Y' = j \Rightarrow X'_j = 1 \Rightarrow X_j = 1 \Rightarrow Y \leq j \text{ and finally } \mathbb{E}[Y] \leq \mathbb{E}[Y'] \qquad (4)$$

Moreover, we know that Y' follows a geometric distribution with parameter $\mathbb{P}(\mathcal{S}'_j) \geq \frac{1}{8^{\Delta_r}}$ (by Lemma 4, there is at least a probability $\frac{1}{8}$ that $d_{est}(t_{i+1}) \geq r_{rw}(d_{est}(t_i))$), and $\mathbb{E}[Y'] \leq 8^{\Delta_r}$. Thus, by Equation 4, we have $\mathbb{E}[Y] \leq 8^{\Delta_r}$. Since Y denotes the index of the first phase during which d_{act} gets out of $\mathcal{I}_{id}$, $M' \in [\![(Y-1)\Delta_r; Y\Delta_r]\!]$. In particular, $M' \leq Y\Delta_r$ and $\mathbb{E}[M'] \leq \Delta_r \times 8^{\Delta_r}$. Finally, as $M' > M$, we have $\mathbb{E}[M] \leq \Delta_r \times 8^{\Delta_r}$. □

4 Continuous Movement

As in the previous section, we present bounds on M for continuous movements (Theorem 3), but prove only the 2D-case.

Theorem 3. *With* $\Delta_l = \left\lceil \frac{\log(1-\varepsilon)-\log(1+\varepsilon)}{\log(1-\frac{\varepsilon}{2})} \right\rceil$, *and with two players following a random continuous movement in 1D, then* $\mathbb{E}[M] \leq \Delta_l \times 4^{\Delta_l}$. *Let* $\Gamma = 2\,\frac{\log(1+\varepsilon)-\log(1-\varepsilon)}{\log\left(1+\frac{\varepsilon}{\sqrt{2}}+\frac{\varepsilon^2}{4}\right)}$. *If two players follow a random continuous movement in 2D, then* $\mathbb{E}[M] \leq \Gamma \times 8^{\Gamma}$. *With moves in 3D, then* $\mathbb{E}[M] \leq \Gamma \times 14^{\Gamma}$.

In two dimensions, at each instant t, a moving player X chooses θ_t and ρ_t following continuous distributions respectively on $[0, 2\pi]$ and $[0, 1]$, so that $p_{actX}(t+1) = p_{actX}(t) + (\rho_t, \theta_t)$, where (ρ_t, θ_t) are polar coordinates.

In 2D, Theorem 3 can be proved following the same general principle as with Theorem 2, but using Lemma 8 and Lemma 9 instead of Lemma 4 and Lemma 5. Instead of r_{rw}, we will use $r_{cm} : x \mapsto x\sqrt{\left(1 + \varepsilon^2/4 + \varepsilon/\sqrt{2}\right)}$.

Once again, Bob is the player who gets out the first of his set of authorized positions with $\mathcal{A}_{lc}$, meaning that Bob is the player to initiate communication at instant t_{i+1}. In this setting, we will use the euclidian distance, thus $\mathcal{B}_{lcB}(t_i)$ takes the form of a disk of center $p_{estB}(t_i)$ and of radius $\frac{\varepsilon}{2}d_{est}$. In order to identify messages that makes a sufficient increase on the estimated distance, we will look at the annulus of inner circle $\mathcal{B}_{lcB}(t_i)$, and with an outer circle of radius $\frac{\varepsilon}{2}d_{est} + 1$. We will call $\mathcal{R}$ the portion of this annulus on the opposite side of $p_{estA}(t_i)$, (represented as a red hatched zone on Figure 3a), that deviates not more than $\frac{\pi}{4}$ from the straight line between $p_{estA}(t_i)$ and $p_{estB}(t_i)$. More formally, with t the intersection between $\mathcal{B}_{lcB}$ and the line $(p_{estA}(t_i)p_{estB}(t_i))$, on the opposite side of $p_{estA}(t_i)$, then $\mathcal{R} = \left\{s,\, s\widehat{p_{estB}(t_i)t} \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \text{ and } d(s, p_{estB}(t_i) \in \left[\frac{\varepsilon}{2}d_{est}(t_i), \frac{\varepsilon}{2}d_{est}(t_i) + 1\right]\right\}$.

Lemma 6. *In two dimensions,* $\mathbb{P}(p_{estB}(t_{i+1}) \in \mathcal{R}) = \frac{1}{4}$.

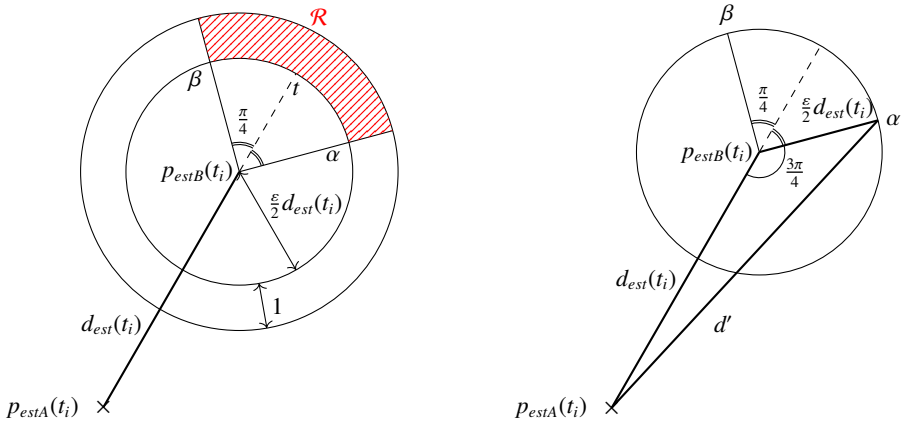
Proof. As Bob does not move more than one distance unit per turn, the first turn where he is outside of $\mathcal{B}_{lcB}(t_i)$, on $p_{estB}(t_{i+1})$, he will be in the annulus. $\mathcal{R}$ represents one fourth of the total area of the annulus, and movement is symmetric with respect to the center of the annulus, thus we have probability one fourth that $p_{estB}(t_{i+1}) \in \mathcal{R}$. □

Lemma 7. *With two players moving,* $\mathbb{P}\Big(d_{est}(t_{i+1}) \geq r_{cm}(d_{est}(t_i)) \mid p_{estB}(t_{i+1}) \in \mathcal{R}\Big) \geq 1/2$.

Proof. Let us assume $p_{estB}(t_{i+1}) \in \mathcal{R}$. The two points of $\mathcal{R}$ that are closest to $p_{estA}(t_i)$ are the rightmost and leftmost points that are both on $\mathcal{R}$ and the border of $\mathcal{B}_{lcB}(t_i)$ (α and β on Figure 3a). Thus, if we call d' the distance between $p_{estA}(t_i)$ and α, we have $d(p_{estA}(t_i), p_{estB}(t_{i+1})) \geq d'$. As can be seen on Figure 3b, the value of d' can be resolved by the law of cosines, relatively to the value of $d_{est}(t_i)$:

$$d' = \sqrt{d_{est}(t_i)^2 + \varepsilon^2/4 d_{est}(t_i)^2 - d_{est}(t_i)^2 \varepsilon \cos(3\pi/4)} = d_{est}(t_i)\sqrt{\left(1 + \varepsilon^2/4 + \varepsilon/\sqrt{2}\right)}$$

This corresponds to r_{cm}, so $\mathbb{P}(d(p_{estA}(t_i), p_{estB}(t_{i+1})) \geq r_{cm}(d_{est}(t_i)) | p_{estB}(t_{i+1}) \in \mathcal{R}) = 1$. We may then notice that, as Alice remains inside $\mathcal{B}_{lcA}(t_i)$, the probability that $p_{estA}(t_{i+1})$ is further away from $p_{estB}(t_{i+1})$ than $p_{estA}(t_i)$ is at least one half. This ends the proof. □



(a) Representation of the points corresponding to a growth in distance

(b) Representation of the different values used to measure a growth in distance

Fig. 3: Continuous movement, two-dimensional situation

Lemma 8. *With two players moving,* $\mathbb{P}\Big(d_{est}(t_{i+1}) \geq r_{cm}(d_{est}(t_i))\Big) \geq \frac{1}{8}$

Proof. The result is immediate with lemmas 6 and 7, and the law of total probability. □

Lemma 9. *With* $\Gamma = \frac{\log(1+\varepsilon)-\log(1-\varepsilon)}{\log\left(\sqrt{1+\frac{\varepsilon^2}{4}}+\frac{\varepsilon}{\sqrt{2}}\right)}$*, for all* $x \in \mathcal{I}_{id}$*,* ${r_{cm}}^{\Gamma}(x) \geq d_0(1+\varepsilon)$.

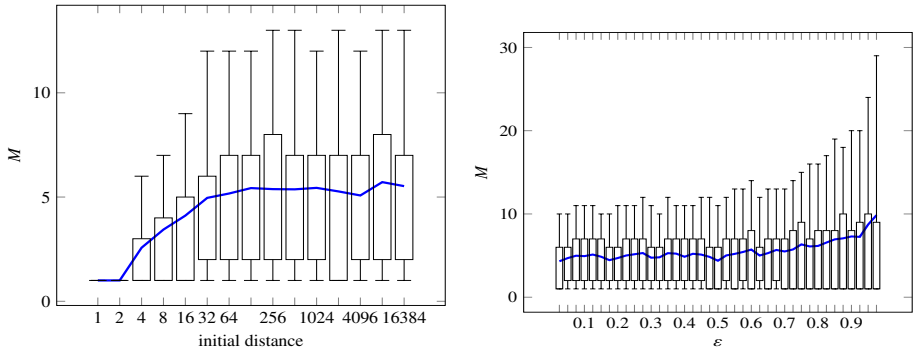
Proof. $x \in \mathcal{I}_{id} \Rightarrow x \geq d_0(1-\varepsilon) \Rightarrow {r_{cm}}^{\Gamma}(x) \geq {r_{cm}}^{\Gamma}(d_0(1-\varepsilon))$ since r_{cm} is increasing, so that ${r_{cm}}^{\Gamma}(x) \geq d_0(1-\varepsilon)\left(1+\varepsilon^2/4+\varepsilon/\sqrt{2}\right)^{\Gamma/2}$. Moreover, by definition of Γ, $(1-\varepsilon)\left(1+\varepsilon^2/4+\varepsilon/\sqrt{2}\right)^{\Gamma/2} \geq (1+\varepsilon)$, so that finally $x \in \mathcal{I}_{id} \Rightarrow {r_{cm}}^{\Gamma}(x) \geq d_0(1+\varepsilon)$. □

5 Experiments

In order to analyze in practice the performance of $\mathcal{A}_{lc}$ with respects to $\mathcal{A}_{id}$ and $\mathcal{A}_{ff}$, we propose simulation results of two types : with synthetic and with real traces.

5.1 Synthetic Traces

The first set of simulations corresponds to random walks and continuous movements. We execute both $\mathcal{A}_{lc}$ and $\mathcal{A}_{id}$ with the same set of random movements (of one or two players) and we display M, the number of message exchanges induced by $\mathcal{A}_{lc}$ at the time the first message is induced by $\mathcal{A}_{id}$. Everywhere, we repeat the experiments 500 times to account for the stochastic nature of the movements, which are represented with boxplots (with first and ninth decile and first ans third quartile). The blue lines indicate the average value of M.



(a) M depending on initial distance, $\varepsilon = 0.5$ (b) M depending on ε, for $d_0 = 400$

Fig. 4: Two players following a random walk in 2D: values of M.

In the case of a two-dimensional random walk, with two players moving, the evolution of M with the initial distance is depicted in Figure 4a. As expected, we observe that M remains bounded and does not depend much on the initial distance (except when the distance is very small with respect to movement amplitudes). We also plot the evolution of M with the given maximal tolerated error, ε in Figure 4b. We observe that M increases when ε gets close to 1, which suggests that dependency on ε in our theoretical bounds is unavoidable. The same set of experiments with random walks and continuous movements in 1D, 2D, and 3D were also performed, and very similar results were obtained (see the companion technical report [5]).

5.2 Actual Traces

Comparison of $\mathcal{A}_{lc}$ with fixed frequency strategies. We compare $\mathcal{A}_{lc}$ to a fixed frequency strategy, denote by $\mathcal{A}_{ff}$, that is used in practice in actual games [2]. This algorithm does not take a maximal error as parameter, but a fixed wait time w between message exchange of any pair of players. In [9], traces containing the time-stamped positions of players in 98 games of Heroes of Newerth [1] are used.[5] There are 10 players, therefore, a wait time of w induces an average of $\frac{9*10}{w}$ messages per time unit. Even if a smaller w makes information more accurate, $\mathcal{A}_{ff}$ comes without guarantee on maximal error violations, contrarily to $\mathcal{A}_{lc}$. To evaluate the performance of $\mathcal{A}_{ff}$ in terms of accuracy, we simulated its behavior for several values of ε and w. We counted the *number of violations per time unit*, that is, the number of distance estimates among the players that violate Equation 2. As there are ten players, and each one has an estimate for all nine others, the number of violations has a maximum of 90 for one time unit.

In order to perform a fair comparison between $\mathcal{A}_{lc}$ and $\mathcal{A}_{ff}$, we used the following protocol. First, we ran $\mathcal{A}_{lc}$ for several values of ε, and measured the resulting average number of messages per time unit. Then, we plugged the obtained value as w in $\mathcal{A}_{ff}$, so as to compare both algorithms in terms of accuracy (to estimate approximated distance) while they use the same average message frequency. The average proportion of

[5] The traces are available at https://doi.org/10.5281/zenodo.583600

violations is shown in bold font in Table 1, along with the optimal number of messages, that is, $\mathcal{A}_{id}$, for different values of ε. We can observe that $\mathcal{A}_{lc}$ is far better than $\mathcal{A}_{ff}$ for satisfying Equation 2. For instance, it sends only 10.44 messages per time unit for $\varepsilon = 0.1$. With $\mathcal{A}_{ff}$, the only way to ensure Equation 2 is by having $w = 1$. This would lead to 90 messages per time unit with $w = 1$, that is, about ten times more than $\mathcal{A}_{lc}$.

Table 1: Comparison of $\mathcal{A}_{lc}$ and $\mathcal{A}_{ff}$, **without Dead-reckoning** (with Dead-reckoning)

	$\mathcal{A}_{id}$	$\mathcal{A}_{lc}$		$\mathcal{A}_{ff}$		
ε	msg/turn	messages per turn	violations	w	msg/turn	violations
0.1	**3.26** (2.23)	**10.44** (4.71)	0.0	**9** (19)	**10.00** (4.73)	**2.9%** (5.13%)
0.2	**1.49** (1.24)	**5.41** (3.02)	0.0	**17** (30)	**5.30** (3.00)	**2.74%** (4.66%)
0.3	**0.91** (0.84)	**3.60** (2.26)	0.0	**25** (40)	**3.60** (2.25)	**2.6%** (4.26%)
0.4	**0.63** (0.62)	**2.65** (1.81)	0.0	**34** (50)	**2.65** (1.80)	**2.53%** (3.88%)
0.5	**0.46** (0.46)	**2.07** (1.50)	0.0	**43** (60)	**2.09** (1.50)	**2.42%** (3.51%)

Influence of better prediction strategies. As mentioned in Section 1.1, Dead-reckoning is a popular method for reducing the error on positions of elements of an online game. This is why we added Dead-reckoning to our simulations to assess its benefits. To do this, we rely on a speed based position prediction algorithm, where speed is calculated according to the two last known positions, and is used to extrapolate the previous known position. The results of the same experiment as above, with this prediction algorithm, are shown on Table 1, within parenthesis. We can observe that the number of message exchanged in $\mathcal{A}_{lc}$ decreases more significantly than $\mathcal{A}_{id}$. Moreover, Dead-reckoning seems more beneficial to $\mathcal{A}_{lc}$ than to $\mathcal{A}_{ff}$, as the decrease in message number is not compensated for in terms of violations by the improved prediction precision.

6 Conclusion and future work

In this paper, we propose a distributed algorithm $\mathcal{A}_{lc}$, for each player to estimate the distance separating them from each other player, with a relative condition on the error. This type of property is desirable in DVE such as online games. We prove that (in a restricted setting), this algorithm is optimal in terms of number of message exchanges up a to a constant factor. We also show through simulations, based on actual game traces, that $\mathcal{A}_{lc}$ performs significantly less communications than the fixed frequency algorithm which is commonly used in online game, while bounding the error.

A summary of our bounds can be found in the following table:

	random walk	continuous movement
1D case	$\min\left(\Delta_l \times 2^{\Delta_l}\ ;\ \left\lceil \frac{4}{\pi}{\Delta_l}^2 \right\rceil \times 8\right)$	$\Delta_l \times 4^{\Delta_l}$
2D case	$\Delta_r \times 8^{\Delta_r}$	$\Gamma \times 8^{\Gamma}$
3D case	$\Delta_r \times 16^{\Delta_r}$	$\Gamma \times 14^{\Gamma}$

This work opens several perspectives. The first one is to extend the theoretical results proved in this paper, either by improving the constants or by increasing the scope

of the results and to consider more sophisticated prediction algorithms. Another longer term perspective is to extend the set of properties that can be maintained in DVEs at the price of re-computations and a (constant) increase in exchanged messages. It was known in the literature that maintaining the positions was possible with no increase in the number of messages and the present paper shows that a constant increase is enough to maintain relative distances. Extending the class of such properties is highly desirable, both in theory and practice.

References

1. Heroes of Newerth - Home, http://www.heroesofnewerth.com/, accessed 27 August 2019
2. Source Multiplayer Networking - Valve Developer Community, https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking, accessed 27 August 2019
3. IEEE standard for distributed interactive simulation–application protocols. IEEE Std 1278.1-2012 (Revision of IEEE Std 1278.1-1995) pp. 1–747 (Dec 2012). https://doi.org/10.1109/IEEESTD.2012.6387564
4. Aggarwal, S., Banavar, H., Khandelwal, A., Mukherjee, S., Rangarajan, S.: Accuracy in dead-reckoning based distributed multi-player games. In: 3rd ACM SIGCOMM Workshop on Network and System Support for Games. pp. 161–165. NetGames, ACM (2004)
5. Beaumont, O., Castanet, T., Hanusse, N., Travers, C.: Approximation Algorithm for Estimating Distances in Distributed Virtual Environments. Research report (Feb 2020), https://hal.archives-ouvertes.fr/hal-02486218
6. Bharambe, A., Douceur, J., Lorch, J., Moscibroda, T., Pang, J., Seshan, S., Zhuang, X.: Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. In: ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. pp. 389–400. ACM (2008)
7. Boulanger, J.S., Kienzle, J., Verbrugge, C.: Comparing interest management algorithms for massively multiplayer games. In: Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games. NetGames '06, ACM, New York, NY, USA (2006)
8. Cai, W., Lee, F.B.S., Chen, L.: An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In: Workshop on Parallel and Distributed Simulation (1999)
9. Carlini, E., Lulli, A.: A spatial analysis of multiplayer online battle arena mobility traces. In: Euro-Par: Parallel Processing Workshops. pp. 496–506. Springer (2018)
10. Kharitonov, V.Y.: Motion-aware adaptive dead reckoning algorithm for collaborative virtual environments. In: 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry. pp. 255–261. VRCAI '12, ACM (2012)
11. Li, Z., Tang, X., Cai, W., Li, X.: Compensatory dead-reckoning-based update scheduling for distributed virtual environments. SIMULATION **89**(10), 1272–1287 (2013)
12. Liu, E.S., Theodoropoulos, G.K.: Interest management for distributed virtual environments: A survey. ACM Comput. Surv. **46**(4), 51:1–51:42 (Mar 2014)
13. Marshall, D., Mcloone, S., Ward, T., Delaney, D.: Does reducing packet transmission rates help to improve consistency within distributed interactive applications? (01 2006)
14. Mauve, M., Vogel, J., Hilt, V., Effelsberg, W.: Local-lag and timewarp: Providing consistency for replicated continuous applications. Multimedia, IEEE Transactions on **6**, 47–57 (03 2004)
15. Millar, J.R., Hodson, D.D., Peterson, G.L., Ahner, D.K.: Consistency and fairness in real-time distributed virtual environments: Paradigms and relationships. Journal of Simulation **11**(3), 295–302 (Aug 2017)

16. Ricci, L., Carlini, E.: Distributed virtual environments: From client server to cloud and P2P architectures. In: 2012 International Conference on High Performance Computing Simulation (HPCS). pp. 8–17 (July 2012)
17. Zhou, S., Cai, W., Lee, B.S., Turner, S.J.: Time-space consistency in large-scale distributed virtual environments. ACM Trans. Model. Comput. Simul. **14**(1), 31–47 (Jan 2004)