An Improved DR Algorithm Based on Target Extrapolating in ROIA Cloud Platform

Dong Liu

Department of Computer Science, Jinan University, Guangzhou 510632, China

Correspondence should be addressed to Dong Liu; wze2k@163.com

Received 10 October 2013; Accepted 18 October 2013

Copyright © 2013 Dong Liu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Real-time Online Interactive Application (ROIA) is an emerging distributed application recently. ROIA needs a highly robust and efficient architecture to cope with the huge concurrent users. Previous works are almost based on the C/S or P2P mode, and their scalability and resource utilization are relatively low. So we try to take advantage of the cloud computing technologies to achieve higher scalability and resource utilization. However, as ROIA servers focused on several data centers in cloud computing rather than being scattered in many areas, it will increase in part users’ network delays and affect their user experiences in ROIA. To cope with this problem, we propose an improved Dead Reckoning (DR) algorithm. Traditional DR algorithm is mostly based on the classic formula of physics to predict, without taking the influence of the user’s target under the different situations into account, so there are some limitations. This paper proposes an improved DR algorithm based on target-extrapolating in a cloud platform for ROIA, elaborates the basic idea of the improved algorithm and the computational model formula, and then carries out a simulation experiment. The analyses of the simulation results show that the improved algorithm is superior to traditional one.

1. Introduction

Cloud computing is surging in recent years and has attracted the attention of many researches. And in order to obtain better performance and lower costs, more and more traditional applications merge into cloud computing environment. Real-time online interactive application (ROIA), a new Internet-based distributed application model, is also emerging in recent years [1]. Popular and market-relevant representatives of ROIA are massively multiplayer online game (MMOG) and interactive E-Learning systems. The research on ROIA in cloud is just beginning.

ROIA has some challenging features such as: highly intensive user interaction, concurrency at a single application instance, and high demand for quality of service (QoS). These features of ROIA have decided that it needs to have enough network bandwidth and lower network latency to ensure the consistency of the real-time interactions between users. It is impossible in ROIA to maintain consistency by sending update information in a very high frequency, because limited network bandwidth would be exhausted by the huge update information. Usually update information will be sending in the lowest possible frequency in order to save network bandwidth, and in this way it requires some kind of technology to make up for the continuity of state information between the two updates, to ensure consistence and a good user experience. Therefore, how to ensure the consistency and promote the user experience under the conditions of the limited bandwidth and the larger delay has become the hotspot in ROIA research.

In this paper, we firstly review the related works about ROIA and DR algorithm. Secondly, we represent a ROIA cloud platform architecture, and then propose a new target-extrapolating DR algorithm which improve the accuracy of the prediction by taking the user’s state and target into account. Finally, the simulation experiments validate the effectiveness of the new target-based DR algorithm, compared with the traditional DR algorithm.

2. Related Work

Since this work is mainly related to two aspects: the architecture of ROIA and DR algorithm, this section focuses on the related works of these two aspects.
2.1. ROIA and Cloud Computing. Traditional ROIA mostly based on C/S mode architecture; in order to obtain higher scalability and resource utilization, many researchers put forward some new architectures. These new architectures can be classified into the following three aspects.

2.1.1. C/MS Approach. In this approach, the cluster technology has been taken in the ROIA server side. Multiple servers provide different functionalities to meet the performance requirement. And it can achieve better scalability and resource utilization through some techniques, such as zoning, replication, and instancing [2, 3].

This approach has some merits, such as being easy to keep the authority and security [4–6], so it still is the mainstream deployment of ROIA. However, in fact, this approach ROIA also has some disadvantages, such as existing single point of failure and performance bottleneck and poor scalability.

2.1.2. P2P Approach. P2P technologies have greatly changed the situation that the ROIA needs the ROIA providers to continue to invest huge hardware resources to meet the growing number of users. In P2P technology, with the joining of peers, the peer's part hardware resources are also increased, thus reducing the pressure of the ROIA providers and achieving the better scalability.

There are many researchers who have proposed a variety structures and mechanisms of P2P approach ROIA [7–9]. However, implementing the P2P model successfully still needs to address some challenges. For example, literature [10] runs simulations using traces from World of Warcraft and found that pure P2P solutions would result in average latency more than 10x greater than C/S solutions. In addition, the absence of authoritative centralized computing and storage makes the state consistency, security, and many other aspects more challenges. These areas still need much further research.

2.1.3. Cloud Approach. In recent years, cloud computing has been successfully adopted in many applications. And some researchers have begun to adopt in ROIA to address scalability problem and other issues.

Now the main cloud approach is a technology called cloud gaming. The typical representatives of cloud gaming are OnLive [11], Gaikai [12, 13].

In this approach, all the game logics and graphics are rendered in the cloud rather than by the client. The cloud transfers the results by the way of video streaming. The user just needs the streaming like YouTube and need a very small thin client to send user’s game command to the cloud.

A major attraction of this approach is that it frees players from the need to frequently upgrade their computers as they can now play games that host on remote servers. And the scalability and consistency is easy to achieve as all the game logics rendered in the cloud.

But because which is transferred to client is video streaming rather than update information in this approach, the network bandwidth becomes more important. When a player makes a move, he cannot see the results until the cloud receives the command, processes the move, renders a new screen, and delivers the screen to the client. Moreover, the traditional compensation techniques [14], such as dead reckoning [15], cannot be used in cloud gaming, because those techniques require game state information which is not available in cloud gaming clients.

Therefore, this approach is more susceptible to the response latency than other approach. Lee et al. [16] also obtained the conclusion that not all games are suitable for cloud gaming by electromyographic experiments. The same degree of latency may have very different impact on a game's QoE (quality of experience).

2.2. DR Algorithm. DR algorithm [17, 18] was originally used in distributed virtual simulation system to release the real-time issues and reduce network traffic and now is also widely used in various areas of transportation, games, and so on. Many scholars made a number of improvements on DR technology. Cai et al. [18] proposed a DR that adaptively adjusts the threshold value according to the relative position of avatars. Lian-Yue [19] proposed a multithreshold DR technology, in which the data are classified, different types of data corresponding to different distances from avatar.

In update mechanism of DR algorithm, there are inconsistencies between actual position and extrapolated position. The most direct way to measure this inconsistency is to measure the distance between actual position and extrapolated position. But Zhou et al. [20] proved that the sustained time length of inconsistent state also makes greatly impact, so it proposed a named TSI (time-space inconsistency) metrics, considering both the special distance and time. In order to reduce the perceptible in consistency state, Roberts et al. [21] also took into account the impact of inconsistent sustained length of time. In addition, accumulated DR errors during update interval were used as a measure of the inconsistent degree in [22, 23]. Li et al. [24] proposed a new DR update strategy to improve the degree of consistency in the case of message loss.

In the DR algorithm, improving predict accuracy will greatly enhance the overall performance of the DR algorithm. McCoy et al. [25] used neural network technology to predict the speed change of entity. Yahyavi et al. [26] draw on the idea of ant colony algorithm to improve the accuracy of prediction in DR algorithm.

3. ROIA Cloud Platform

Traditional ROIA usually adopts the classic client/server model and through servers cluster or other technologies to support large-scale concurrent users operating. Although this architecture is simple to implement and easy to ensure consistency and security, its scalability is poor.

So we propose a new approach to scalable ROIA in cloud, we call it MRCP (Multi-ROIA cloud platform).
3.1. Motivation of MRCP. The MRCP is proposed mainly based on the following two points to consider.

(1) Fluctuations in the Number of Concurrent ROIA Users around the World at Different Times. The number of concurrent users is not fixed, and it will fluctuate with the time every day. We can use this point; an overload data center can migrate some load to the low-load data center located in other time zone. Thereby this approach can obtain well scalability and high resource utilization.

(2) Differences in Requirements for Network Latency in Different Types ROIA. ROIA can be divided into different types of ROIA. And the different type ROIA has different requirement for network latency in actual operation. For example, MMOG can be divided into different types, such as first person shooting (FPS), role playing game (RPG) and real-time strategy game (RTS). These different MMOG have different requirements for network latency in order to ensure good user experience. According to the results of literature [27], the latency tolerance threshold is about 100 ms in FPS, the threshold is about 500 ms in RPG, and the threshold for RTS is about 1000 ms.

So based on these characters, we can deploy a various types of ROIA on one data center, so that the load balancing has more reasonable choice. Because if only one type of ROIA, the following situation may is happen. The data center A is overloading, and the data center B located in other place (maybe on the other side of the earth) has very low load. However, B is so far from A that the latency is hard to meet the requirements of users on A. So it cannot migrate some load of A to B. In this case, if there are various types of ROIA on A, we can migrate the low latency requirement ROIA to B. This not only reduces the load of A but also does not affect the good user experience.

3.2. Architecture of MRCP. Based on the above two considerations, we propose a multi-ROIA cloud platform (MRCP). MRCP will have several data centers around the world, and various types of ROIA will be deployed on these data centers. According to the load of data centers, we can make load balancing between the various data centers, so as to achieve high scalability and high resource utilization.

Figure 1 depicts the architecture of MRCP. MRCP consists of multiple data centers distributed around the world. Each data center has many ROIA servers (RS) and one MRCP local controller (MLC).

MLC is responsible for user’s login, storing user information, and the internal load balancing of data center. Moreover, MLC communicates with other MLCs periodically in order to get the load information of other data centers. If necessary, MLC will execute load balancing between data centers.

RS is responsible for the functions of ROIA. Various types of ROIA are deployed on different RS in one data center.

4. The Target-Extrapolating DR Algorithm

4.1. Background. We will discuss the load forecast and balance problem on the ROIA cloud platform; here we focus on maintaining consistency and enhancing user experience. For example, in an MMOG, users through client software manipulate their own avatar to producing, competing and fighting in one virtual game world. Client periodically sends the state and location information of its own avatar to the server. After receiving the information, the server will compute and update the global game world by using the information. And then server will send update information to clients. Clients use the information to update their local game world, for maintaining consistency with the global game world.

Before these updates have not yet reached, if the activities of the local game world remain static until client receives the updates to continue its activities, it will greatly reduce the user experience. Using forecast techniques will alleviate this problem. The various avatars in local game world may move on follow by the prediction before the updates are received. And it needs using smooth processing technology to smoothly pull the avatar back to correct position when receiving updates to find deviation of the actual position and the predicted position. It does not allow the user to feel noticeable pauses and jumping.

Most ROIA use the DR algorithm to predict the movement. The traditional DR algorithm used the three classical physics formula, as follows:

\[ P_{t+\Delta t} = P_t, \]  
\[ P_{t+\Delta t} = P_t + v_t \cdot \Delta t, \]  
\[ P_{t+\Delta t} = P_t + v_t \cdot \Delta t + \frac{1}{2} a_t \cdot (\Delta t)^2. \]

Formula (1) describes the situation that the position of avatar remains static from moment \( t \) to \( t + \Delta t \). Formula (2) describes the position after the avatar moves by uniform speed \( v_t \) during \( \Delta t \). Formula (3) describes the position after the avatar moves by the initial velocity \( v_t \) and the acceleration \( a_t \) during \( \Delta t \). We can see from the above formula that the traditional DR algorithm predict the avatar’s trajectories only associated with the information of location and move statement, but it does not consider other factors, such as the target of interest, the avatar’s own state (healthy or wounded).

However, in many cases, users change their direction of moving because of the influence of other factors. For example, the treasures, equipment, and other avatars appeared in the user domain of interest (DOI) may affect the movement of avatar. So it is not in strict accordance with the iner-tia movement just the traditional DR algorithm predicted. Moreover, avatar’s own state will have a significant impact on the direction of its movement. For example, in the same scene, a healthy avatar is likely to move for treasures and equipment; however, a severely wounded avatar may desert toward medical supplies instead of treasures or equipment. Therefore, we propose a new target-extrapolating DR algorithm which takes into account other factors to improve the precision of the prediction.

4.2. Basic Idea of Target-Extrapolating DR Algorithm. The basic idea of the target-extrapolating DR Algorithm is as
follows. Firstly, algorithm calculates the attractiveness of other avatar and entities (treasures, equipment, or medical supplies) on avatar $E_1$ in $E_1$’s DOI. Then algorithm selects the max attractiveness as the target of $E_1$ and adds the attractiveness to predictive math model as an external force. Finally, algorithm calculates the predicted path through math model.

As shown in Figure 2, there are some avatars ($E_1$, $E_2$ and $E_3$) and some entities ($E_4$, $E_5$, $E_6$ and $E_7$). At time $t$, for only $E_2$, $E_3$, and $E_7$ in the DOI of $E_1$, it just needs to calculate the attractiveness of $E_2$, $E_3$, and $E_7$. Moreover, due to the angle between $E_7$’s attractive direction and $E_1$’s current move direction greater than 90 degrees, it shows that $E_7$ is not in forward direction and $E_7$ does not first appear in DOI of $E_1$. In other words, it indicates that user very likely does not have interest in $E_7$. Therefore, our algorithm is no longer considering the attractiveness of $E_7$. In Figure 2, the attractiveness of $E_2$ is greater than the attractiveness of $E_3$, so $E_2$ is forecasted to be the target. The solid line arrow indicates the moving direction which traditional DR algorithm predicted; the long dashed line arrow indicates the moving direction which our algorithm predicted.

4.3. The Quantitative Calculation of Attractiveness. To compare the attractiveness of avatars and entities, it needs to quantitatively calculate the attractiveness. Let $A_t(E_2, E_1)$ denote the attractiveness of $E_2$ to $E_1$ at time $t$. It is calculated by the following formula:

$$A_t(E_2, E_1) = \frac{\lambda \cdot \text{CAttr}(E_2, E_1) + (1 - \lambda) \cdot \text{AAttr}_t(E_2)}{\text{Dis}(E_2, E_1)^\alpha}$$  \hspace{1cm} (4)

where $\lambda$ is a parameter to adjust the influence of CAttr($E_2, E_1$) and AAttr(_t($E_2$) on $A_t(E_2, E_1)$. $\alpha$ is a parameter to adjust the influence of Dis($E_2, E_1$) on $A_t(E_2, E_1)$.

CAttr($E_2, E_1$) represents the current attractiveness of $E_2$ to $E_1$ at time $t$. According to the comparison result between life value of $E_1$ and threshold (signed as LifeThreshold), there are two types of processing of quantitative calculation: (1) the life value of $E_1$ is greater than LifeThreshold. It means that the avatar $E_1$ is healthy. In this case, if $E_2$ is an avatar, the lower the level of $E_2$ the greater CAttr($E_2, E_1$); if $E_2$ is a entity, the greater the level of $E_2$ the greater CAttr($E_2, E_1$), (2) the life value of $E_1$ is lower than LifeThreshold. It means that the avatar $E_1$ is seriously wounded. In this case, if $E_2$ is a medical supply, CAttr($E_2, E_1$) is given a big fixed value which is much bigger than the value produced in processing (1); if $E_2$ is not a medical supply, the processing is the same as processing (1).

AAttr(_t($E_2$) denotes the accumulated attractiveness of $E_2$ at time $t$. It accumulates in a recent period of time when other avatars choose $E_2$ as a target. It reflects the popularity of $E_2$ and use it to influence the value of $A_t(E_2, E_1)$. We can learn from the pheromone increasing and decreasing in the ant colony algorithm to regulate the increasing and decreasing of AAttr(_t($E_2$). As an attribute value of entity, AAttr(_t($E_2$) is set between 0 and 100. The initial value of AAttr(_t($E_2$) is 0; AAttr(_t($E_2$) will add $k$ units after $E_2$ is selected as target.

### Table 1: Description of important parameters in the algorithm.

| Parameter | Description | Range | Provisional value |
|-----------|-------------|-------|-------------------|
| $\lambda$ | Adjust the current attractiveness{s} influence on the result | $0 \leq \lambda \leq 1$ | 0.8 |
| $\alpha$ | Adjust the effect that distance impacts on attractiveness | $1 \leq \alpha$ | 2 |
| $k$ | The increase of accumulated attractiveness after the entity being selected as a target | $0 \leq k \leq 100$ | 5 |
| $\epsilon$ | Adjust the decay speed of the accumulated attractiveness | $0 \leq \epsilon \leq 1$ | 0.8 |

Meanwhile, AAttr$_t(E_2)$ decays by time as shown in formula (5). $\epsilon$ is a parameter to adjust the speed of the decay. Consider

$$A_{t+\Delta t} = P_t + \nu_t \cdot \Delta t + \frac{1}{2} \left(\omega \cdot A_t(E_2, E_1) + (1 - \omega) a_t\right) \Delta t^2$$  \hspace{1cm} (5)

where $\omega$ is the parameter to adjust the influence of attractiveness. $\nu_t$ and $a_t$ also can be optimized as follows:

$$\nu_t = \sigma \cdot \frac{P_t - P_{t-\Delta t}}{\Delta t} + (1 - \sigma) \nu_{t-\Delta t}$$  \hspace{1cm} (6)

$$a_t = \tau \cdot \frac{V_t - V_{t-\Delta t}}{\Delta t} + (1 - \tau) a_{t-\Delta t}$$  \hspace{1cm} (7)

Previous researchers [13] had proven the optimization can effectively improve the overall smoothness, because it considers the historical information. Table 1 summarizes the important parameters in the new target-extrapolating DR algorithm.

5. Simulation and Results Analysis

We implemented the traditional DR algorithm and the target-extrapolating DR algorithm by Python 2.5 and designed
a simulation environment. Simulation program interface is shown in Figure 3.

Different icons in the simulation program simulate the different types of entities in the ROIA. The data below each icon indicate the coordinates (the upper left corner is (0,0), in units of pixel value) and accumulated attractiveness.

The point in the central of figure represents the avatar which carries out DR algorithm, and the square box is its domain of interest (DOI). Only the points in DOI will be calculated. Furthermore, in order to reduce the difficulty of implementation, we have not introduced obstacles in the figure. But because $\text{Dis}(E_2, E_1)$ is the length of the shortest accessible path between $E_2$ and $E_1$, the obstacles have been taken into account. Therefore, the result of the simulation is still not losing its universality.
Through repeated experiments, we found that target-extrapolating DR algorithm has higher accuracy than the traditional DR algorithm.

Figure 4 represents the deviations of traditional DR algorithm and the target-extrapolating DR algorithm. From Figure 4, we can see that the deviation of the target-extrapolating DR algorithm is less than the traditional DR algorithm’s in most of the time (about 70% experimental time).

In Figure 5, the curve represents the differences between the deviating values extrapolated by traditional DR algorithm and the values extrapolated by the target-extrapolating DR algorithm. The curve is greater than zero in most of experimental time, so it means that the deviation of the improved DR algorithm is smaller and the improved DR algorithm makes more accurate extrapolations.

Figure 6 represents the time comparison between the target-extrapolating DR algorithm and the traditional DR algorithm. From Figure 6, we can see that the target-extrapolating DR algorithm is more accurate than the traditional DR algorithm in the 71% of the total experimental time. And, only in 29% of the total experimental time, the traditional DR algorithm performs better than the target-extrapolating DR algorithm.

Figure 6 shows that the performance of the target-extrapolating DR algorithm is superior to the old one from the aspect of time. Meanwhile, Figure 7 shows that the target-extrapolating algorithm is superior to the old one from the aspect of deviation values. From Figure 7, we can see that the average deviation value of the target-extrapolating DR algorithm is 11.58 pixels. However, the average deviation value of the traditional algorithm is 16.46 pixels.

6. Conclusions

More and more traditional applications merge into cloud computing environment, in order to obtain better performance and lower costs. The research on ROIA in cloud environment is also just beginning. Previous works are almost based on the C/S or P2P architectures. We propose
a new approach to ROIA by taking advantage of the cloud computing. However, the requirement for network latency in the new ROIA under cloud platform is different as the requirement of the traditional ones. Because ROIA servers focused on several data centers in cloud computing rather than scattered in many areas, it will increase part users’ network delays and affect their user experiences in ROIA. So, we propose an improved DR algorithm to cope with this problem.

DR algorithm is a widely used algorithm in ROIA. Usually ROIA uses DR algorithm to reduce network traffic and relieve the consistency problem caused by network delay. However, traditional DR algorithms are mostly based on the physics formula to predict. In fact, the state and target of avatar may have great impact on the trajectory of avatar. The traditional DR algorithms have not taken these impacts into account, so that the algorithm predicted accuracy decreases. This paper proposes an improved DR algorithm based on target-extrapolating. The results of simulation experiment show that the new target-extrapolating DR algorithm is superior to traditional DR algorithm. This work will be helpful for further research on ROIA in cloud environment.

Acknowledgments

This work is supported by Guangzhou Science and Technology Project (2010Y0-C681), Guangdong Science and Technology Project (2010B060100056), and the Natural Science Foundation of Guangdong (S201201008831). Great thanks goes to these projects and foundations.

References

[1] F. Glinka, A. Raed, S. Gorlatch, and A. Ploss, “A Service-oriented interface for highly interactive distributed applications,” in Euro-Par 2009—Parallel Processing Workshops, vol. 6043 of Lecture Notes in Computer Science, pp. 266–277, 2010.
[2] V. Nae, A. Iosup, and R. Prodan, “Dynamic resource provisioning in massively multiplayer online games,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 3, pp. 380–395, 2011.
[3] C. E. B. Bezerra, J. L. D. Comba, and C. F. R. Geyer, “Adaptive load-balancing for MMOG servers using KD-trees,” ACM Computers in Entertainment, vol. 10, no. 3, article 5, 2012.
[4] L. Fan, Solving key design issues for massively multiplayer online games on peer-to-peer architectures [Ph.D. dissertation], Heriot-Watt University, Edinburgh, UK, 2009.
[5] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, “New algorithms for secure outsourcing of modular exponentiation,” To appear in IEEE Transactions on Parallel and Distributed Systems.
[6] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, “Fuzzy keyword search over encrypted data in cloud computing,” in Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM ’10), pp. 441–445, IEEE, March 2010.
[7] J. Luo, H. Chang, and Y. Yi, “Resource assignment algorithm under multi-agent for P2P MMOG,” Journal of Computer Research and Development, vol. 47, no. 12, pp. 2067–2074, 2010.
[8] L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan, “Hydra: a massively-multiplayer peer-to-peer architecture for the game developer,” in Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames ’07), pp. 37–42, September 2007.
[9] L. Fan, H. Taylor, and P. Trinder, “Mediator: a design framework for P2P MMOGs,” in Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames ’07), pp. 43–48, September 2007.
[10] J. L. Miller and J. Crowcroft, “The near-term feasibility of P2P MMOGs,” in Proceedings of the 9th Annual Workshop on Network and System Support for Games (NetGames ’10), November 2010.
[11] http://www.onlive.com.
[12] http://www.gaikai.com.
[13] R. Brown, Smoothing, Forecasting and Prediction of Discrete Time Series, Dover, New York, NY, USA.
[14] Y. W. Bernier, “Latency compensating methods in client/server in-game protocol design and optimization,” in Proceedings of the 15th Game Developers Conference (GDC ’01), March 2001.
[15] L. Pantel and L. C. Wolf, “On the suitability of dead reckoning schemes for games,” in Proceedings of the 1st Workshop on Network and System Support for Games (NetGames ’02), pp. 79–84, 2002.
[16] Y.-T. Lee, K.-T. Chen, H.-I. Su et al., “Are all games equally cloud-gaming-friendly? An electromyographic approach,” in Proceedings of the 11th Annual Workshop on Network and Systems Support for Games (NetGames ’12), ACM, November 2012.
[17] K. C. Lin, “Dead reckoning and distributed interactive simulation,” in International Symposium on Aerospace/Defense Sensing and Control and Dual-Use Photonics (AeroSense ’95), Proceedings of SPIE, pp. 16–36, Orlando, Fl, USA, April 1995.
[18] W. Cai, F. B. S. Lee, and L. Chen, “Auto-adaptive dead reckoning algorithm for distributed interactive simulation,” in *Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS ’99)*, pp. 82–89, IEEE CS, Atlanta, Ga, USA, May 1999.

[19] H. E. Lian-Yue, “Research on technique of dead reckoning with multi-threshold,” *Journal of Computer Research & Development*, vol. 37, no. 8, pp. 990–993, 2000.

[20] S. Zhou, W. Cai, B.-S. Lee, and S. J. Turner, “Time-space consistency in large-scale distributed virtual environments,” *ACM Transactions on Modeling and Computer Simulation*, vol. 14, no. 1, pp. 31–47, 2004.

[21] D. Roberts, R. Aspin, D. Marshall, S. McLoone, D. Delaney, and T. Ward, “Bounding inconsistency using a novel threshold metric for dead reckoning update packet generation,” *Simulation*, vol. 84, no. 5, pp. 239–256, 2008.

[22] S. Aggarwal, H. Banavar, S. Mukherjee, and S. Rangarajan, “Fairness in dead-reckoning based distributed multi-player games,” in *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames ’05)*, pp. 1–10, ACM, 2005.

[23] H. Dai and T. Yonekura, “A proposal of dead reckoning protocol in distributed virtual environment based on the taylor expansion,” in *Proceedings of the International Conference on Cyberworlds (CW ’06)*, pp. 107–114, IEEE CS, Lausanne, Switzerland, November 2006.

[24] Z. Li, W. Cai, X. Tang, and S. Zhou, “Dead reckoning-based update scheduling against message loss for improving consistency in DVEs,” in *Proceedings of the 25th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS ’11)*, pp. 1–9, IEEE CS, Nice, France, June 2011.

[25] A. McCoy, T. Ward, S. McLoone, and D. Delaney, “Multistep-ahead neural-network predictors for network traffic reduction in distributed interactive applications,” *ACM Transactions on Modeling and Computer Simulation*, vol. 17, no. 4, Article ID 1276929, 2007.

[26] A. Yahyavi, K. Huguenin, and B. Kemme, “AntReckoning: dead reckoning using interest modeling by pheromones,” in *Proceedings of the 10th Annual Workshop on Network and Systems Support for Games (NetGames ’11)*, pp. 1–6, ACM, Ottawa, Canada, October 2011.

[27] M. Claypool and K. Claypool, “Latency and player actions in online games,” *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, 2006.
