An Approach to Consistent Displaying of Virtual Reality Moving Objects

Vasily Y. Kharitonov
Department of Computers, Systems and Networks
Moscow power engineering institute (technical university), Russian Federation
E-mail: KharitonovVY@mail.ru

Abstract

Distributed virtual reality systems are a new step in the development of interactive 3d-graphics applications, allowing geographically remote users to interact in a shared virtual environment, as if they situated in one room. The realism of user's experience in such systems depends not only on the quality of graphics, but also on the underlying networking mechanisms. These mechanisms should provide consistent users interaction, eliminating the problems of a particular network. Especially, consistent displaying of virtual reality objects should be achieved.

In this paper the main principles of distributed virtual reality systems design are explored. Special attention is drawn to the reliability issues of such systems in terms of consistent interaction. An approach to consistent displaying of virtual reality moving objects is proposed. It allows to reduce influence of hardware limitations and the overall network workload through a more flexible way of network traffic management taking into account the movement dynamics of the objects.

1. Introduction

The most attractive area for the computer graphics in recent times has been creation of interactive three-dimensional applications because they are very popular in many scientific disciplines and applied problems, as well as in the entertainment industry. While hardware is becoming more complicated, modern computer graphics allow to achieve increasingly high realism of virtual 3d-worlds. A separate class of simulation equipment, named *Virtual Reality* (*VR*) systems, was introduced. Such systems allow user to not only observe the virtual worlds, but easily "immerse" in them, tightly interacting with a computer-simulated environment.

Along with increasing realism the scale of systems is also increased. With the growth of computer networks bandwidth there is a rising interest in creating of *Distributed Virtual Reality* systems (shortly, *DVR* systems). In such systems not one user but multiple users can concurrently immerse in the virtual reality.

In order to ensure interaction of many users in DVR systems computer networks are used. The realism of virtual world depends not only on the quality of graphics, but also on the underlying networking mechanisms. Therefore, one of the key problems when building such systems is the problem of inter-user networking, which generally involves the following issues:

1) choice of user *communication architecture* and *interfacing protocols*;
2) choice of the way data are stored and organized (*data architecture*)
3) network traffic minimization and latency influence compensation;
4) maintaining of consistent virtual environment state for all users;
5) interaction analysis between objects in virtual environment (e.g. collision detection).

Some of these issues, somehow, have been already mentioned in various systems [1, 2, 3], however it is still early to speak about their final decision. This paper addresses issues 1 – 4, which are the most important when organizing a consistent interaction in DVR systems.

978-0-7695-3179-3/08 $25.00 © 2008 IEEE
DOI 10.1109/DepCoS-RELCOMEX.2008.14

2. DVR system definition and its features

The term Virtual Reality is used to describe a computer-generated, highly-realistic artificial world or environment (called a *Virtual Environment*, *VE*), allowing the user to interact with it in real-time by interfacing some of his actions in the real world back into the virtual environment and providing visual, acoustical and, sometimes, haptic feedback.

The soft hardware allowing geographically remote users to interact in the shared virtual environment is referred to as the *Distributed Virtual Reality* system:

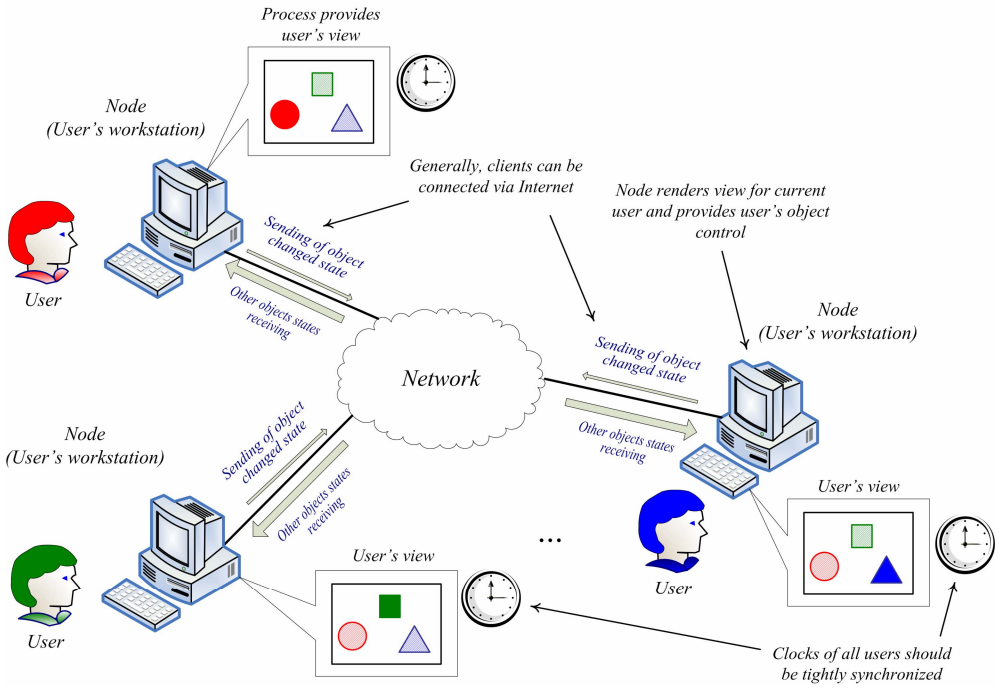

Figure 1. Distributed virtual reality system

Virtual environment represents a collection of virtual objects with certain sets of attributes. These attributes determine the properties and behavior of each object, and together form an *object state*. The *VE state* is a tuple of all states of its objects.

2.1. Representation of DVR systems at different abstraction layers

DVR system can be considered at three different abstraction layers: *user*, *software* and *hardware*.

At the highest, user layer of abstraction DVR system should be indivisible and transparent for the user, hiding from him all of its distributed nature and implementation details. Each user should be given a *view* to the virtual environment and a *logical interface* to interact with it. Interaction of the user with the virtual environment can be implemented by means of an *avatar*. Avatar is a special kind of object associated with a specific user, which state is controlled by the user himself. Thus, the avatar represents the user in the virtual world. The view is rendered image of virtual environment observed from the current position of avatar in the virtual space.

At the software layer, DVR system is a collection of uniform processes (software applications), interacting with each other based on a certain type of *communication architecture* using some kind of a *high-level protocol*. The main types of communication architecture are *client/server* and *peer-to-peer*. There are also mixed architectures, combining these

architectures, such as multi-server architecture. Depending on the chosen architecture, *client*, *server* and *peer* processes can be distinguished. In client/server architecture client processes are focused on the individual user's view visualization and state control of his avatar, while server process provides interaction of multiple users. In peer-to-peer architecture peer processes include both functions. A high-level protocol is based on general network protocols, such as TCP/IP protocols, and makes it possible to transmit data between processes taking into account specific communication architecture.

When considering hardware layer, DVR system consists of computing nodes (in the simplest case, PCs), connected via data network. At this layer a particular type of network is choosen, on which basis the system is to be built, and hardware requirements for the nodes are specified. Also input and display devices for user-to-system interface are determined (*man-machine interface*). Besides, general network protocols and *data transmission techniques* are defined (*unicast*, *multicast* or *broadcast*).

2.2. Data architectures in DVR systems

To ensure that all users of DVR system have a sense of presence in a shared world, they should all have the same data on the current VE state. This is achieved at the software layer by choosing a certain way of data organization and storing, referred to as *data architecture*.

It is convenient to organize data in a hierarchical structure which is shared between users, also called *scene graph*. Scene graph establishes logical and spatial relationships between VE objects, and provides means for implementing various acceleration algorithms, both for networking and rendering (such as various *culling* and *level-of-detail techniques*), and collision detection algorithms.

According to how data are stored there are three main types of data architectures: *centralized*, *replicated* and *distributed*. When using a centralized architecture, all VE state data are stored on a dedicated process. In a replicated architecture each process keeps a copy of the entire VE state. In a distributed architecture VE state is distributed among several processes.

Every time when user changes the state of his avatar or any other object at the user layer, *update messages* are generated at the software layer, which serve to maintain data consistency throughout the system.

2.3. Consistency and responsiveness

The main requirements for DVR systems that determine the qualitative nature of users interaction are *consistency* and *responsiveness* [9]. Consistency requirement means that all users of DVR system should have identical data on the VE state at every moment. At the same time information on state changes (update messages) should be distributed between users in the minimally possible time. When named conditions are satisfied, it is said about *consistent interaction* (in terms of transferring data between nodes). One of the main objectives achieved through consistent interaction is to provide a *consistent displaying* of VE objects, allowing all users to observe *nearly* identical VE states, though, possibly, from different view points. "Nearly" means that in a real system local VE state copies necessarily are to differ, for example, because of data transmission delay (latency). To ensure a high consistency, each user, having performed any action, should wait before undertaking next action, until data safely reach other users. Also, at the software layer processes should be tightly coupled that requires high bandwidth and low latency, as well as imposes restrictions on the number of users.

The responsiveness of the system is the time required user action to make the result in the virtual world. To ensure a high responsiveness each user's process should not wait for other

remote users to be notified about user's actions. Instead, it should change the local VE state copy so that the user immediately would become aware of his actions result. Therefore, the users' processes should be loosely coupled, making a large amount of local calculations. The decision on how action will affect the virtual environment should make a user's process itself.

However, the user may not always determine the result of his actions alone. For instance, to perform collision detection of multiple users' objects in a right way, it is necessary to take a collective decision, which is contrary to the requirement of high responsiveness, because such a decision requires some time on the data exchange between users. If each user attempts to detect collision independently of the others, all users can come to different results, and the consistency of the system may be disrupted. Thus, the responsiveness of the system can be in opposition to the requirement of consistency. In most cases it is impossible to achieve both these requirements at the same time and trade-off have to be found.

3. Reliability issues in DVR systems

The concept of consistency in DVR systems is closely linked to reliability issues. The reliability of such systems can be considered as the ability to maintain *consistent* users interaction under stated conditions for a specified period of time. Then, among the critical parameters, affecting the reliability of the DVR system, can be identified:

- average data delivery time from one node to another, t_D;
- the frequency of local VE state copies synchronization between processes, f_S;
- average time that process spends on VE state visualization, t_V.

In order to provide consistent users interaction the following conditions must be met:

$$f_{Vmin} \le f_V \le f_S \le f_D \le f_{Dmax}, \quad (1)$$

where f_{Vmin} — the minimum user-defined VE state visualization rate (referred to as *frame rate*, $f_V = 1/t_V$), f_{Dmax} — the maximum possible (physically) data transmission rate ($f_D = 1/t_D$). If *dead reckoning* is applied (see below) expression $f_V > f_S$ also possible.

Consistent interaction requires both hardware and software support. One of the major obstacles taking place when organizing a consistent interaction, in particular, is hardware limitations imposed by the communication lines and nodes hardware. The main limitations are *network bandwidth*, *latency* and *node processing power*. They are closely related to the critical parameters described above. The first two define parameter t_D and affect parameter f_S. The third limitation specifies parameter t_V, but it may indirectly influence parameters t_D and f_S.

It is impossible to solve consistency problem in DVR systems by hardware only, because hardware resources are always limited. Therefore, dedicated software-based approaches are applied. Among them are the following:

1) protocol optimization;
2) using of *interest management* technique;
3) applying of dead reckoning algorithms;
4) dynamic VE state distribution among several servers.

These approaches minimize network traffic transmitted between nodes and some of them considered in [4 – 9]. Moreover, the approach 3 reduces the impact of latency, and the approach 4 allows to decrease the amount of calculations performed by individual nodes.

4. Proposed Approach to Consistent Displaying of VR Moving Objects

The main aim of creating the proposed approach is the study of DVR systems design principles in order to create a method reducing the influence of hardware limitations. The

proposed approach makes it possible to achieve consistent interaction between users (as well as a consistent displaying of VE objects) and not only reduces the impact of hardware limitations but decreases the overall network workload through a more flexible way of network traffic management. This approach includes:

- selection of user communication architecture and data architecture;
- use of specialized high-level protocol;
- applying of adaptive dead reckoning algorithm with acceleration based threshold (*ATADR*) [4, 5], taking into account the movement dynamics of the objects;
- use of the clock synchronization mechanisms [12].

As communication architecture used in the approach, a client/server architecture is chosen, due to its better scalability and greater flexibility in comparison with peer-to-peer architecture. In this architecture, each client represents single user and server provides user-to-user interaction. The data architecture is replicated, because it is simpler and better suits for small-size world used in our example application (see below). High-level protocol is based on two transport level TCP/IP protocols: TCP and UDP. Service data requiring reliable delivery are transmitted using TCP protocol (e.g. data on connecting/disconnecting of remote users). The general data, mainly represented by state update messages, are transferred using UDP protocol.

Clock synchronization implies a reference clock, stored on the server. Every newly connected client synchronizes its clock with the server clock using a special procedure, similar to Cristian clock synchronization algorithm [12].

The main component of the proposed approach is adaptive dead reckoning algorithm *ATADR* (*Adaptive Dead Reckoning with Acceleration based Threshold*) that allows to predict VE objects states. It has the following features:

- generating update messages only when the difference between predicted and reference object states exceeds the predefined threshold of maximum deviation (in other words, sending update messages with variable rate);
- using acceleration based threshold;
- compensation of network latency (*direct latency elimination*);
- using the second-order derivative polynomial for object state prediction;
- correction of prediction errors using cubic splines;
- objects rotation prediction.

5. Experimental results

To evaluate functional capabilities of the proposed approach a *distributed system of formation flying over locality* was developed (see fig. 2) [10].

Figure 2. The distributed system of formation flying over locality

All experiments were run within the 100Mbit/s Ethernet network using computers with 3d-hardware support. The functional capabilities include both qualitative and quantitative indicators. Qualitative analysis was made on the basis of the object state prediction accuracy, referred to as *dead reckoning accuracy estimation*. The following estimations have been introduced: *average error in distance calculation between two objects* (ξ) and *average error in acceleration measuring* (γ).

Estimation ξ is calculated by the equation:

$$\xi = \frac{1}{n}\sum_{i=1}^{n}\left|\Delta s_{(A)}(t_i) - \Delta s_{(B)}(t_i)\right|, \tag{2}$$

where $\Delta s_{(A)}(t_i)$ — the distance between two *moving* objects at user *A*'s side at time t_i, $\Delta s_{(B)}(t_i)$ — the distance between these objects at user *B*'s side at time t_i, $|\Delta s_{(A)}(t_i) - \Delta s_{(B)}(t_i)|$ — error in distance calculation between two objects at time t_i, n — total measures for given trajectories of two objects. It is convenient to measure the distance between objects in object bodies, since such unit is invariant with respect to the object size (but compared objects should be nearly the same size).

Estimation γ is defined as:

$$\gamma = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|a_A(t_i) - a_{A(B)}(t_i)\right|}{a_A(t_i)}, \tag{3}$$

where $a_A(t_i)$ — true acceleration value of user *A*'s object at time t_i, $a_{A(B)}(t_i)$ — acceleration value of user *A*'s object measured at user *B*'s side at the same time, n — total measures. Estimation γ is dimensionless.

Based on estimations ξ and γ the method for accuracy comparison of different dead reckoning algorithms is offered. It includes following steps:

1) two arbitrary objects are selected, O_A and O_B, controlled by users *A* and *B* respectively;
2) for objects O_A and O_B fixed motion paths are specified (referred to as *reference paths*);
3) for each analyzed dead reckoning algorithm an array of different input parameter sets is chosen (for example, *ATADR* assumes selecting of *maximum deviation threshold* [4]);
4) for each algorithm, objects O_A and O_B are iteratively launched along reference paths under various input parameters; when objects are in motion, user *A*'s process tracks and records path of object O_B and user *B*'s process records path of object O_A, while server measures and records average *input traffic* from both users;
5) on the basis of reference and recorded paths of objects O_A and O_B estimations ξ and γ are calculated for each input parameter set;
6) according to recorded and calculated parameters, graphs are plotted, showing dependence of estimations ξ and γ on input traffic;
7) steps 4 – 6 are repeated for every evaluated dead reckoning algorithm;
8) relying on graphs for different dead reckoning algorithms one can conclude about which of them provides greater prediction accuracy for given network bandwidth.

The presented method is used to compare the new adaptive algorithm *ATADR* with the traditional dead reckoning algorithm which generates update messages with fixed sending rate (see figures 3 and 4). As shown on the graphs, with the increasing rate of updates messages (incoming traffic on the server) prediction error is reduced for both dead reckoning algorithms. However, at the same traffic adaptive algorithm in most cases provides greater accuracy.

Estimations ξ and γ can be calculated both for the case when there are no other objects except objects of users *A* and *B* and for the case when there are another objects. In the latter case errors will be more significant because of extra network traffic and increased latency.

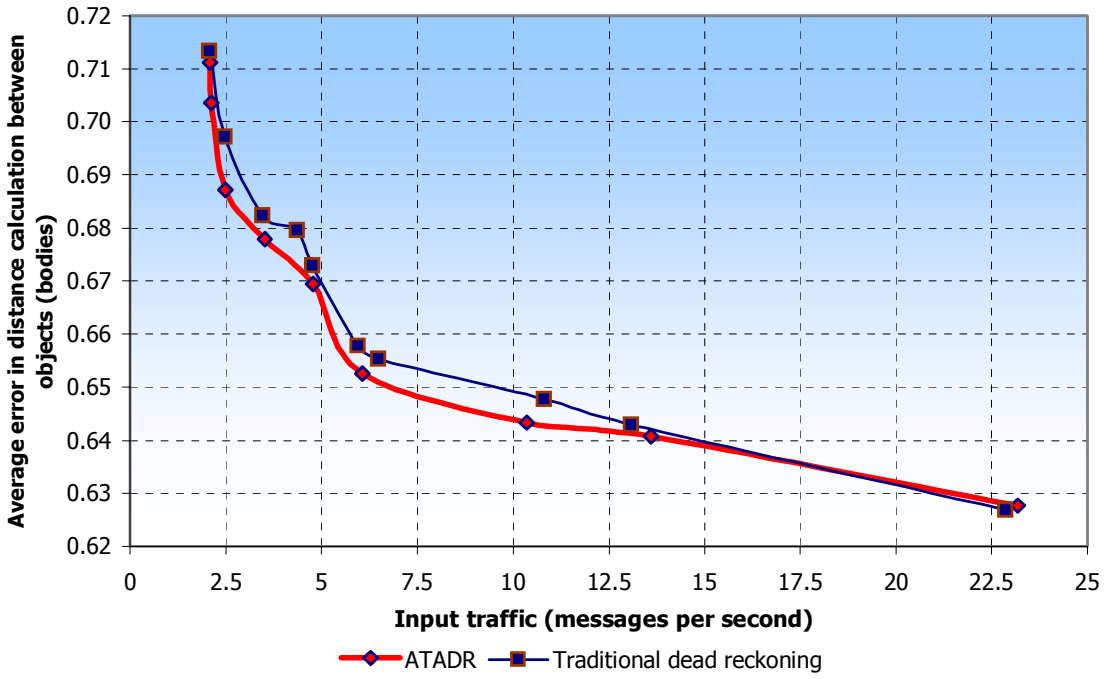

Figure 3. Average error in distance calculation between two objects (ξ)

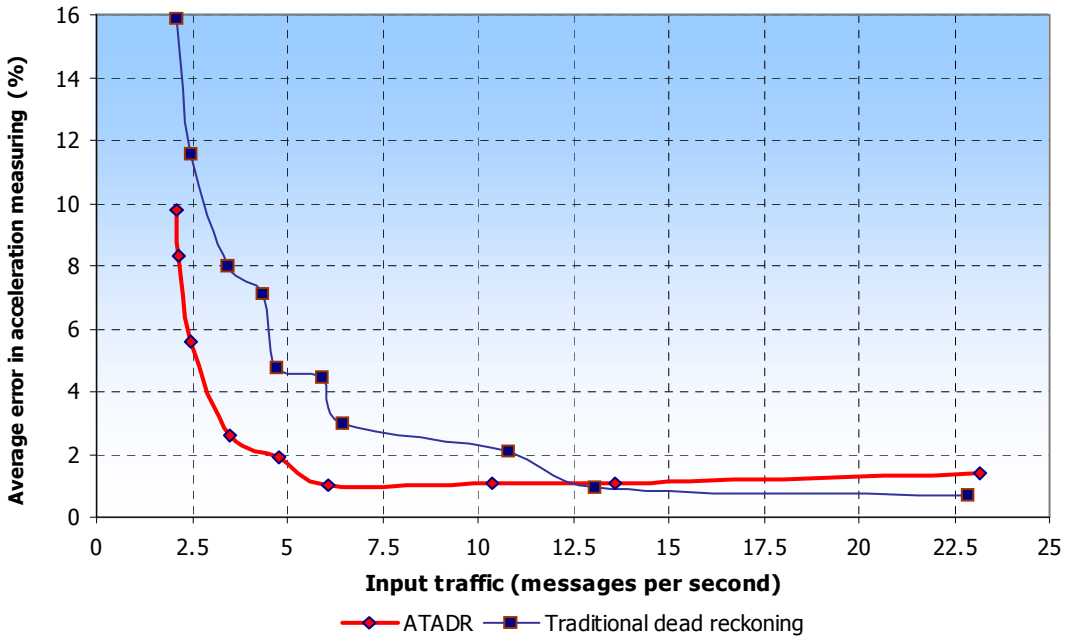

Figure 4. Average error in acceleration measuring (γ)

Quantitative analysis of the proposed approach was carried out on various parameters including scalability analysis of the computing environment in the context of a real network. It has been found experimentally that it can be simultaneously connected up to 32 users to the

system within 100Mbit/s LAN while keeping the required dead reckoning accuracy (which is equal to 0.7 object bodies at average per-user traffic equal to 2.5 messages per second).

6. Conclusion and future work

This paper covers a number of issues arising when designing DVR systems. First, generalized representation of such systems at different abstraction layers is suggested. It allows a particular developer to divide system-building process into separate stages. Second, reliability issues of such systems in terms of consistent interaction are explored. The proposed approach to consistent displaying of VR moving objects improves both qualitative and quantitative indicators of DVR systems. Finally, results of the approach experimental study are presented including qualitative and quantitative analysis. In particular, qualitative analysis is based on dead reckoning accuracy estimation. For this purpose special method is offered.

In the near future, we plan further development and expansion of the proposed approach by improving existing mechanisms, as well as by adding new ones, namely:

- adding collision detection between objects;
- enhancement of users interaction model by using multi-server architecture and implementing distributed data architecture;
- use of more precise clock synchronization algorithms [12, 13];
- exploring the question of applying statistical methods for objects state prediction.

Later on, the proposed approach can be used as a basis for programming library with the unified API allowing to create DVR systems for specific application areas.

7. References

[1] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A network software architecture for large scale virtual environment", Presence: Teleoperators and Virtual Environments, vol. 3, no. 4, Aug. 1994, pp. 265-287.

[2] T. Funkhouser, "RING: A Client-Server System for Multi-User Virtual Environments", Symposium on Interactive 3D Graphics, April 1995, pp. 85- 92.

[3] H. Tramberend, "Avocado: A Distributed Virtual Reality Framework", IEEE Virtual Reality Conference 1999, Houston, TX, 1999, pp. 14-21.

[4] V.Y. Kharitonov, "Methods of efficiency enhancement of network interaction in distributed systems of virtual reality", Proceedings of 2nd International Conference on Dependability of Computer Systems DepCoS-RELCOMEX 2007, IEEE Computer Society, Los Alamitos, CA, USA, 2007, pp. 305-308.

[5] V.Y. Kharitonov, "Exploring the principles of consistent interaction in distributed virtual reality systems", Proceedings of Scientific and Technical Conference "Information Tools and Technologies", MPEI, Moscow, Russia, October 16-18, 2007, Vol. 3, pp. 214-217.

[6] Y. B. Bernier. "Latency Compensating Methods in Client/Server In-Game Protocol Design and Optimization", Proceedings of the Game Developer Conference, 2001, http://www.resourcecode.de/stuff/clientsideprediction.pdf.

[7] N. Caldwell, "Defeating Lag With Cubic Splines", http://www.gamedev.net/reference/articles/article914.asp, 2000.

[8] S. Singhal , M. Zyda, Networked virtual environments: design and implementation, ACM Press/Addison-Wesley Publishing Co., New York, NY, 1999.

[9] J. Smed, H. Hakonen, Algorithms And Networking for Computer Games, UK, Chichester: John Wiley & Sons, 2006.

[10] V.Y. Kharitonov, D.A. Orlov, "Project of distributed system of formation flying visualization over locality", Proceedings of the III International Conference "Parallel Computations and Control Problems", V.A. Trapeznikov Institute of Control Sciences, Moscow, Russia, October 2-4, 2006, pp. 990-998.

[11] L. Lamport, "Time, clocks, and the ordering of events in a distributed system". ACM, 1978, pp. 558-565.

[12] A. Tanenbaum, M. van Steen, "Distributed Systems: Principles and Paradigms", 2nd Edition, Prentice Hall, New Jersey, USA, 2006.

[13] E. Cronin, A. R. Kurc, B. Filstrup and S. Jamin, "An Efficient Synchronization Mechanism for Mirrored Game Architectures", Multimedia Tools Appl, 23(1), 2004, pp. 7-30.