A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games: A Preliminary Study

Claudio E. Palazzi(1,2), Stefano Ferretti(1), Stefano Cacciaguerra(1), Marco Roccetti(1)

1 *Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italia*

2 *Computer Science Department, University of California Los Angeles, Boelter Hall, Los Angeles CA, 90095, USA*
{cpalazzi, sferrett, scacciag, roccetti}@cs.unibo.it

***Abstract*-The astonishing increase of the Internet diffusion has provided global connectivity proficient at deploying online games for a large number of participants located even very far from each other. However, online games are characterized by more stringent requirements than those accomplishable by traditional distributed applications deployed over best-effort networks. Indeed, one of the key factors in determining the success of an online game is represented by the ability to rapidly deliver events among the various game servers that maintain the state of the game over the network. We already demonstrated that adapting in this context RED (Random Early Detection) techniques borrowed from queuing management can improve the global responsiveness of the game [1]. However, this solution may be not sufficient for a specific class of on-line games. We deem that, in case of fast-paced multiplayer online games (such as shoot 'em up, for example) requiring a frenetic behavior of the participants, a highly elevate interactivity degree must be guaranteed even at the cost of partially sacrificing the consistency of the game state. In this case, in fact, having only a partial consistency view of the game state is not so player's amusement affecting as, instead, a delayed action processing activity may be. We have hence explored the possibility to apply a RIO (RED with In and Out) based algorithm to manage the game state delivery among the various game servers, in order to further improve the aptitude of our scheme in maintaining a highly playable interactivity degree for fast-paced online games. Preliminary experimental results confirm the viability of our approach.**

***Keywords- multiplayer computer games; online entertainment; event delivery service; interactivity; consistency*.**

I. INTRODUCTION

Nowadays, two main reasons above the others attract an increasing number of researchers and developers toward online electronic amusements. The first one is the very high level of revenues generated every year, which surpasses even the movie business. The second reason is represented by the correlation between problems that emerge in developing innovative game experiences and those typical of other conventional research fields such as, for example, distributed multimedia applications, virtual environments and simulation. Under this aspect, it is of particular interest to analyze one of the most innovative challenges in networked electronic amusements: the deployment of an efficient architecture to support Massive Multiuser Online Games (MMOGs) over a best effort network [2, 3, 4].

Creating an enjoyable online game entertainment requires the convergence of solutions belonging to different technical areas. In particular, our focus here is centered on networking and computational load at the servers of the game platform architecture. To this aim, Mirrored Game Server represents an efficient solution to support MMOGs, which deploys over the network a constellation of communicating Game State Servers (GSSs) [5, 6]. Each GSS maintains part or the whole game state, takes charge of event deliveries to/from other GSSs or to/from clients connected with it, and can implement policies aimed at increasing the global performance of the system. However, in order to guarantee a uniform view of the game state among all GSSs, an efficient *synchronization scheme* needs to be employed.

In a previous study [1], we proposed an innovative game synchronization scheme, named ILA (Interactivity-Loss Avoidance), specifically designed for providing efficient event delivery synchronization in multiplayer online games. ILA mechanism is able to uplift the playability degree of online multiplayer games by maintaining the event delivery delays under a human-perceptivity threshold whilst preserving the game state consistency and the game evolution fluency at the player's side. Simply stated, this result was obtained discarding events that can be considered obsolete and employing a dropping probability which depends on the perceived responsiveness at GSSs.

However, our experiences with online games over a best effort network lead us to claim that there exist cases where

* This research has been partially funded by the Italian M.I.U.R., Interlink project and 60% (IT).

even dropping all the obsolete events in a game is not enough to ensure interactivity. This is particularly true for a class of games that requires frenetic, and often redundant, actions by the players.

This class of games is widely recognized in the gaming community as fast-paced (or even *fast and furious*) games. A typical example in this class amounts to shoot/beat 'em up games. Simply put, for this class of games we propose to enhance ILA scheme adding a further dropping probability function that discards even some non-obsolete events when throwing away all the obsolete ones is not enough to ensure an adequate interactivity degree.

Obviously, this may generate some sporadic inconsistencies in the game state. However, we deem that partial consistency becomes acceptable with *fast and furious* online games where the lack of consistency lasts only for a small amount of time. In this scenario, in fact, the necessity of a very high interactivity degree emerges as overwhelming even on the full-consistency requirement.

We have developed a preliminary study whose experimental results confirm the viability of our idea.

The remainder of this paper is organized as follows. In Section 2 we briefly survey the theoretical background at the basis of our basic synchronization scheme. Section 3 presents some details of the newly proposed interactivity maintenance technique. Section 4 describes the simulative environment adopted as our test bed. In Section 5, we report preliminary results obtained from the conducted evaluation. Finally, Section 6 concludes the paper.

II. Background

A. *Interactivity vs consistency*

Distributed interactive games are characterized by two main requirements which cannot be considered independent one from the other: interactivity and consistency [1, 6]. It is widely accepted that in order to provide interactivity in a distributed gaming environment, it must be guaranteed that the external stimuli, generated by players, are processed by other participants under a human-perceptivity threshold [1, 4, 5]. This means that the time elapsed from the event generation at a sending GSS and its processing time at each receiving GSS results below a specific average value. For example, scientific literature declares that a delay of 50ms is not perceived at all by players while at 150 ms players' performances may result jeopardized by the lag and finally 225 ms of delay could represent the maximal limit for playable interruptions. These values are valid for games like first person shooter and vehicle racing by can be relaxed in the case of more strategic games.

Not only, in order to obtain a factual smooth progression in the game visualization on the player's screen, also a low variance of these values should be guaranteed.

Consistency regards the contemporary uniformity of the game state view in all the nodes belonging to the system [6]. The easiest way to guarantee consistency is to make the game proceed through discrete *locksteps* [7]. However, having a single move allowed for each player and synchronizing all the agents before moving toward the next round, strongly impairs the interactivity of the system.

In a previous work [1], we demonstrated that a good level of interactivity, coupled with full-consistency accomplishment, may be obtained exploiting the notions of obsolescence and correlation surveyed in the next Subsection.

However, as mentioned in Section 1, full-consistency may be sacrificed in favor of a higher interactivity degree when considering a particular class of online games. In case of frenetic shoot 'em up games, for example, loosing a shoot action among hundreds of them, all comprised in a very tight period of time, may be accepted even if the final outcome of the game evolution results slightly altered.

Conversely, having consistent lags between event generation and action visualization on screens irremediably compromise the velocity of the action which is, in other words, the funniest component of this kind of games.

B. *Obsolescence and correlation*

It is well known that full consistency can be obtained through the use of a completely reliable, totally ordered event delivery scheme [6]. On the other hand, it is also common knowledge that totally ordered delivery approaches imply an increment of the complexity and, most of all, are at the basis of the total delays experienced by the system [8].

Recent studies demonstrated that exploiting the semantics of the application can be put in good use to relax reliability and total order delivery requirements, thus augmenting interactivity [1, 9]. Some events, in fact, can lose their significance as time passes: new actions could make irrelevant the previous ones.

For example, in case of rapid succession of movements of a single agent in a virtual word, the event representing its last destination makes obsolete the older ones. *Obsolescence* can thus be defined as the relation between two received events e_1 and e_2, generated at different times $t(e_1) < t(e_2)$, by which the existence of e_2 diminishes the importance of processing e_1. Dropping obsolete events before processing them clearly reduces the computational cost at GSSs and speeds up the execution of fresher events.

To define as obsolete a game event, we have to be sure that consistency would not be weakened. To this aim, we have also to introduce the notion of *correlation*. Two events, say e_1 and e_2, are correlated if the final game state depends on their execution order. Hence, correlation is to be taken into consideration to determine the obsolescence of a chain of subsequent events.

In fact, an event e_3 makes obsolete a previous event e_1 only if there is not another event e_2 (correlated to e_1), temporary interleaved between e_1 and e_3, that breaks the relationship of obsolescence between e_1 and e_3. In a scenario where we wish to maintain full consistency, correlated events are the only ones that really need to be delivered to the destined GSSs in the same order as generated and that cannot be subject to any dropping action.

Since we are here discussing design issues about *fast and furious* class of multiplayer online games, this requirement can be sporadically relaxed in order to boost the interactivity.

C. *Interactivity Maintenance with RED*

Ferretti and Roccetti demonstrated the interactivity benefits attainable exploiting the semantics of a game during its evolution to relax the total order delivery requirement [6].

In their proposed scheme, the player's actions are collected by the closer GSS, transformed into events and finally forwarded to the other GSSs in order to maintain a global identical view of the game state. Events are marked at their creation with a generation timestamp and then sent to destination: they are hence orderable.

Obviously, a global conception of time is to be maintained by all the GSSs, for example exploiting a variety of different solutions proposed in literature that enable the GSSs' physical clocks synchronization [10, 11, 12], or employing new technological synchronization devices, such as, for example, GPS.

Each receiving GSS considers the arrival time of the event and measures the difference elapsed since its generation; the resulting value is named *Game Time Difference* (GTD). The GTD of the event is then compared with a predefined constant *Game Interaction Threshold* (GIT) and normal delivery operations are performed until the former value is lower than the latter. When the GTD value exceeds the GIT, the GSS turns on a stabilization mechanism which exploits the obsolescence notion to drop useless events so as to bring the GTD back within the GIT.

Taking inspiration from the RED (Random Early Detection) approach in case of incipient congestion in best effort networks [13], we have recently enhanced the aforementioned Interactivity Restoring mechanism with the Interactivity-Loss Avoidance (ILA) approach [1].

The main innovation presented by this latter scheme is the capability of avoiding interactivity loss before it may happen, discarding some packets when the level of interaction among GSSs descends significantly.

In practice, ILA substitutes the basic binary dropping mechanism for obsolete events (OFF when interactivity is present and ON when interactivity is lost) with a continuously-working proactive mechanism that drops obsolete events with a probability depending on the level of interactivity.

Even if, similarly to RED, ILA utilizes a uniformly distributed dropping function, however, the parameter taken under control is the average GTD instead of the average queue size. Upon each packet arrival, in fact, each GSS determines the GTD of the arrived event, namely *sample_GTD*, and feeds a low pass filter to compute the updated average GTD, namely *avg_GTD*. When *avg_GTD* exceeds a certain threshold, the GSS drops obsolete events with a certain probability *p*, without processing them. If *avg_GTD* exceeds a subsequent limit, *p* is set equal to *1*, and all obsolete events waiting for being processed are discarded.

III. A RIO-LIKE TECHNIQUE FOR INTERACTIVITY LOSS AVOIDANCE

Our previous RED based ILA scheme can be usefully applied in those multiuser online games that pursue a good interactivity degree whilst maintaining full consistency in game state views.

However, as already mentioned, there exist particular classes of games where it might be desirable to guarantee a very high interactivity degree even at the cost of sporadically renounce to the full-consistency requirement. This is the case when the core attractiveness for players emerges from a feverish, sometimes even chaotic, action sequence, namely, *fast and furious* multiuser online games.

To this aim, our intention here is to add the possibility to discard even non-obsolete game events when dropping all the obsolete ones is not sufficient to maintain an adequate level of interactivity. In particular, we want to create two discarding functions, respectively for obsolete and for non-obsolete events, featured with specific boundaries and slopes, that work independently one from the other and that take action in sequence with the increasing of the game event GTDs at the GSSs.

Obviously, dropping non-obsolete events can be done without consequences only for a category of games where little inconsistencies are not highly deleterious for the aim of the game and for player's fun (e.g., fast-paced games).

Even in this case, if the number of dropped non-obsolete events becomes significant, a consistency restoring mechanism may be required to re-establish a coherent game state view among all the GSSs [14].

We hence propose to enhance our ILA scheme with new features deriving from the integration of a RIO-like algorithm in place of the RED-like one. RIO (RED with In and Out) scheme is an enhanced version of RED mechanism able to discriminate between two different classes of traffic, non-prioritized (Out) and prioritized (In), and calculates two distinct dropping probabilities.

As illustrated in Fig. 1, three parameters (and three phases) characterize each of the twin algorithms: $\min_o$, $\max_o$ and $P\max_o$, for obsolete events, and $\min_v$, $\max_v$ and $P\max_v$ for valid (i.e., non-obsolete) ones.

In the graph, the y-axis represents the dropping probability corresponding to the *avg_GTD* indicated by the x-axis. Focusing on obsolete events, for values of *avg_GTD* in $[0, \min_o)$ the mechanism performs normal operations, with no packet drops, while in $[\min_o, \max_o)$ obsolete packets are discarded with a computed probability, and finally in $[\max_o, \infty)$ all obsolete packets are thrown away.

The intervals $[0, \min_v)$, $[\min_v, \max_v)$ and $[\max_v, \infty)$ define the corresponding phases for valid events. The dropping probabilities are computed as a function of *avg_GTD* and of, respectively, $P\max_o$ or $P\max_v$.

Persistent situations of low interactivity result in high *avg_GTD* and hence in high discarding probabilities. High dropping probability values (for $P\max_o$ or $P\max_v$) will make the GSS discarding events without processing or forwarding them, thus helping in restoring an adequate level of time interaction between servers.

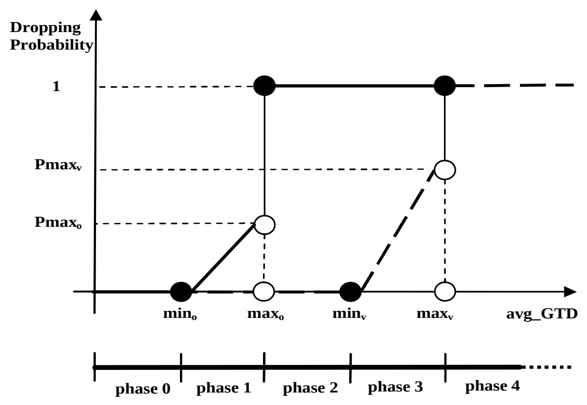

Fig. 1: Discarding probability functions.

Since valid (i.e., non-obsolete) events are strictly linked to consistency, the possibility to discard them should be taken into account only as last resort, in case of heavy disruption of interactivity. For this reason, our scheme starts dropping obsolete packets much earlier than valid packets. Not only, we have set the parameters such that the algorithm throws away all the obsolete packets before considering any dropping probability on valid events; this is done by choosing $\max_o$ smaller than $\min_v$.

Moreover, diverse aggressiveness in dropping packets, depending on their belonging class, can be decided by adjusting the values of $Pmax_o$ and $Pmax_v$.

The new ILA-RIO algorithm, implementing the behavior of our scheme in all its phases, is given in Fig. 2 and is obtained endowing the RED algorithm presented in [13] with the RIO features.

In essence, the algorithm repeats a block of operations each time a new event arrives at the considered GSS.

In particular, the GTD of the packet is calculated (*sample_GTD*, line 1) as the time difference elapsing between the generation of the associated game control event at the sender GSS and its delivery to the considered GSS.

```
0]  for each event_packet arrival {
1]   determine the sample_GTD
2]   calculate the new average delay avg_GTD
3]   if ($min_o$ = $avg\_GTD$ < $max_o$) then
4]     calculate the probability $P_o$ of dropping an obsolete event
5]     determine if ONE obsolete event has to be discarded
6]   else if ($max_o$ = $avg\_GTD$) then
7]     drop ALL obsolete events
8]     if ($min_v$ = $avg\_GTD$ < $max_v$) then
9]       calculate the probability $P_v$ of dropping a valid event
10]      determine if ONE valid event has to be discarded
11]    else if ($max_v$ = $avg\_GTD$) then
12]      drop ALL valid events
13]    endif
14]  endif
15] endfor
```

Fig. 2: ILA-RIO algorithm.

Table. 1: Configuration of the GSSs.

GSS ID	1	2	3	4	5	6	7
Latency Avg (ms)	15	40	75	90	80	30	100
Latency Std Dev (ms)	15	15	30	10	20	15	25

The scheme feeds a low pass filter with the just calculated *sample_GTD* in order to update the average of the GTDs (*avg_GTD*, line 2).

In particular, the filter is implemented by resorting to the following formula:

$$avg\_GTD = avg\_GTD + w*(sample\_GTD - avg\_GTD) \quad (1)$$

where w is a sensitivity coefficient, with values comprised in (0, 1], that determines how closely the trajectory of the average follows the movements of the samples.

While *avg_GTD* lies below $\min_o$, the process stays in phase 0 and no particular operations are performed. Conversely, when *avg_GTD* is comprised between $\min_o$ and $\max_o$, then the scheme is in phase 1 and lines 4-5 are executed.

Basically, a dropping probability is computed in order to establish if an obsolete event must be discarded. Such a probability increases until an event is discarded. This is done exploiting a *counter* variable in order to have a uniform distribution of the drops, following the method well explained in [13].

If *avg_GTD* grows beyond $\max_o$, the scheme enters in phase 2 or successive and all obsolete packets have to be discarded in the attempt of re-establishing interactivity (line 7).

Moreover, the algorithm has to distinguish between phase 3 (lines 9-10) and phase 4 (line 12). In the former case, valid events are dropped with a certain probability ($Pmax_v$) with a behavior analogous to the already explained phase 1, while in the latter case all events, with no distinction, are discarded.

IV. SIMULATION ASSESSMENT

To evaluate our event processing strategy, we have simulated a general Mirrored Game Server architecture comprising various GSSs dispersed over the Internet. As previously mentioned, the events generated are totally ordered based on a global notion of time. These can be achieved iehter by resorting to the variety of different software solutions proposed in literature for clocks synchronization [10, 11, 12] or by exploiting some technological device useful for synchronization like the GPS.

For the sake of a deeper comprehension, we have focused our attention on the event receiving aspect of a single GSS, while the other GSSs are sending events to it. GSS_0 is the receiving GSS and the others are the sending GSSs.

Based on results obtained by other authors in [15, 16], the values of the network latencies among each sending GSS and the receiving GSS_0 have been obtained based on a lognormal distribution having the average and standard deviation values as shown in Table 1.

Furthermore, the event generation rate i.e., the interval of time between two subsequent event departures at each GSS, was sampled from a lognormal distribution (average equal to 30ms and standard deviation equal to 10ms). These values represent approximately the traffic generated by from 5 to tens players (depending on the semantics of the game) connected to each GSS and are utilized to generate a trace file containing 1000 events for each GSS. Trace files also include the information needed to identify (correlated and) obsolete events.

In our simulations, we considered two different event trace configurations where the probability that an event makes obsolete previous ones was set, respectively, to 50% and 90%.

For each event trace the size of the generated game events was 200 Bytes on average.

The event delivery service was built by exploiting a receiver-initiated communication protocol over the UDP, that utilizes NACKs (Negative ACKnowledgments) to provide reliability to the communication.

In our tests we compared three different synchronization schemes: the proposed ILA-RIO scheme, the ON-OFF mechanism (Interactivity Restoring as reviewed in Section 2, Subsection C), the traditional OFF approach (having no mechanism to restore interactivity).

Focusing on the parameters exploited in the ILA-RIO algorithm, we set *w*=1/8 in (1) in the attempt to make the algorithm able to filter out sporadic high GTDs, while maintaining a prompt responsiveness to a persistent decline of the interactivity degree.

Moreover, the other parameters involved in the algorithm were set as follows: min_o = 50ms, max_o = 100ms, $Pmax_o$ = 0.2, min_v = 150 ms (equivalent to the GIT for the ON-OFF scheme), max_v = 225 ms and $Pmax_v$ = 0.3.

Several considerations can be expressed about the most appropriate values for the above cited parameters. The phase boundaries, in fact, should be chosen in order to activate phase 1 when the delay between the generation of a player's action and its execution on the screens provides the first perceivable symptoms of interactive degradation. Instead, the threshold for the more aggressive phase 2 should be chosen in order to be surpassed when the lag results annoying and low-performance determining for players.

As rationale for our chosen values, scientific literature declares that a delay of 50ms (i.e., our min_o parameter) is not perceived at all by players while at 150ms (i.e., our min_v and GIT parameters) players' performance results disturbed by the lag and 225ms (i.e., our max_v parameter) of delay could represent an upper bound for playable interaction [17, 18, 19, 20]. These limits hold for games like vehicle racing, first person shooters and fast shoot/beat 'em up, but can be relaxed in case of strategic games (e.g. Starcraft, Age of Empire, etc.) [21].

V. RESULTS

We intend to demonstrate the benefits attainable by implementing an event discarding algorithm in case of an increasing trend of the GTDs.

In a previous work we already assessed the efficacy of the single ILA (RED-based) mechanism in a similar scenario [1]. In particular, we experimented: i) an improvement of 27% and 4% on the average GTD w.r.t OFF and ON-OFF respectively, and ii) an improvement of 66% and 41% on the standard deviation w.r.t. OFF and ON-OFF.

As to the approach presented in this paper, in Fig. 3 and Fig. 4, we compare for ILA-RIO, ON-OFF and OFF schemes:

i) the percentage of events arrived at GSS_0 with a GTD value larger than the GIT, and
ii) the amount of events dropped by ILA-RIO and ON-OFF.

Figures 3 and 4 respectively refer to a specific event trace configuration, with a different probability of obsolescence among events.

As observable (Fig. 3-a, Fig. 4-a), in both configurations ILA-RIO and ON-OFF schemes outperform the traditional OFF method in terms of GTDs. Moreover, ILA-RIO further reduces the number of events with GTD above the GIT w.r.t. ON-OFF. These results give a preliminary confirmation that ILA-RIO is able to guarantee a higher interactivity degree when contrasted with the two other alternative approaches.

Furthermore, while in the former event trace configuration there is no significant difference among ILA-RIO and ON-OFF when comparing the amount of dropped events (Fig. 3-b), in the second configuration ILA-RIO greatly reduces this value (Fig. 4-b). Therefore, ILA-RIO scheme augments the game evolution fluency.

Finally, we evaluated the amount of valid events dropped by our ILA-RIO approach. Indeed, while the ON-OFF approach discards only obsolete events, ILA-RIO is enabled to drop valid events when the interactivity degree results highly jeopardized.

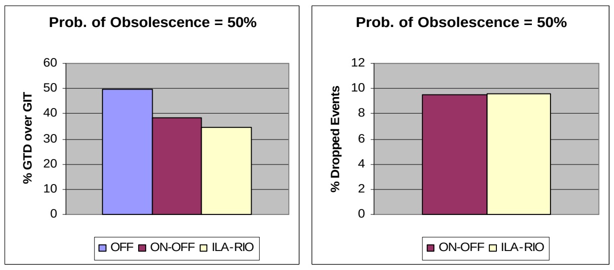

Fig. 3: Probability of obsolescence = 50%; (a) Event percentage with GTD > GIT; (b) Percentage of discarded events.

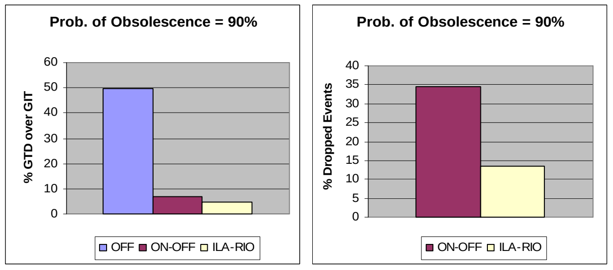

Fig. 4: Probability of obsolescence = 90%; (a) Event percentage with GTD > GIT; (b) Percentage of discarded events.

Table. 2: % of obsolete and valid discarded events in ILA-RIO.

Obsolescence Prob.	**50%**	**90%**
Obsolete	9,46%	13,64%
Valid	0,16%	0%

Table 2 reports the percentage of obsolete and valid events that are discarded, depending on the event trace. As expected, the number of dropped obsolete events increases with the probability of obsolescence. Accordingly, the amount of dropped valid events diminishes as the percentage of obsolete events is greater, since this imply a lower percentage of valid ones.

In particular, while a small amount of valid events has been discarded in correspondence of the first event trace (probability of obsolescence equal to 50%), no valid event has been dropped in the second configuration (probability of obsolescence equal to 90%).

This tendency is due to the fact that if an adequate number of obsolete events is available during the events exchange activity, then our scheme is able to exploit all these (obsolete) events to restore interactivity when enters in phase 2. Simply stated, interactivity is promptly restored by dropping only obsolete events without the need of discarding valid ones.

VI. Conclusions and future work

In this paper we presented a new scheme for a fast event delivery service among mirrored GSSs, aimed at supporting fast-paced networked games. The proposed approach exploits an event dropping mechanism, inspired by the RIO algorithm, devised to maintain a higher interactivity degree among players while preserving only partial consistency in the system.

The novelty of our proposal amounts to the possibility of dropping also non-obsolete events when the interactivity degree results highly jeopardized. We claim that this approach may be utilized in certain games having very elevated interactivity requirements and when small temporary inconsistencies are not highly deleterious for the aim of the game. As last resort, consistency restoring mechanisms among GSSs may be exploited to re-establish a coherent view of the game state. A preliminary experimental study has shown that a good interactivity degree may be obtained by exploiting our mechanism.

Acknowledgment

We wish to thank the anonymous referees of the GDTW Workshop for their helpful comments on an earlier version of this paper.

References

[1] Palazzi C. E., Ferretti S., Cacciaguerra S., Roccetti M., “On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures”, to appear in *1st IEEE International Workshop on Networking Issues in Multimedia Entertainment* (NIME'04), GLOBECOM 2004, Dallas, TX, 2004.

[2] Griwodz C., “State Replication for Multiplayer Games”, in *Proc. of NetGames2002*, Braunschweig, Germany, 2002, pp.29-35.

[3] Mine R. M., Shochet J., Hughston R., “Building a Massively Multiplayer Game for the Million: Disney's Toontown Online”, *ACM Computers in Entertainment* (CIE), vol.1, no.1, pp.15-15, 2003.

[4] El Rhalibi A., “Peer-to-Peer Architecture and Protocol for a Massively Multiplayer Online Game”, ”, to appear in *1st IEEE International Workshop on Networking Issues in Multimedia Entertainment* (NIME'04), GLOBECOM 2004, Dallas, TX, 2004.

[5] Cronin E., Kurc A. R., Filstrup B., Jamin S., “An Efficient Synchronization Mechanism for Mirrored Game Architectures”, *Multimedia Tools and Applications*, vol.23, no.1, pp.7-30, 2004.

[6] Ferretti S. and Roccetti M., “A Novel Obsolescence-based approach to Event Delivery Synchronization in Multiplayer Games”, *International Journal of Intelligent Games and Simulation*, vol.3, no.1, pp.7-19, 2004.

[7] Steinman J. S., “Scalable Parallel and Distributed Military Simulations Using the SPEEDES Framework”, in *Proc. of 2nd Electronic Simulation Conference* (ELECSIM95), Internet, 1995.

[8] Cheriton D.R. and Skeen D., “Understanding the Limitations of Causal and Totally Ordered Multicast”, in *Proc. of the 14th Symposium on Operating System Principles* (SOSP '93), Asheville, NC, 1993, pp.44-57.

[9] Ferretti S., Roccetti M. and Cacciaguerra S., “On Distributing Interactive Storytelling: Issues of Event Synchronization and a Solution”, in *Proc. of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment* (TIDSE 2004), LNCS 3105, Darmstadt, Germany, 2004, pp.219-231.

[10] Drummond R. and Babaoglu O., “Low-Cost Clock Synchronization”, *Distributed Computing*, vol.6, no.3, pp.193-203, 1993.

[11] Cristian F. 1989. “Probabilistic clock synchronization”, *Distributed Computing*, vol.3, no.3, pp.146-158.

[12] Mills D. L. 1991. “Internet Time Synchronization: the Network Time Protocol”, *IEEE Transactions on Communications*, vol.39, no.10 pp.1482-1493.

[13] Floyd S. and Jacobson V., “Random Early Detection Gateways for Congestion Avoidance”, *IEEE/ACM Transactions on Networking*, vol.1, no.4, pp.397-413, 1993.

[14] M. Mauve, “Distributed Interactive Media”, PhD thesis, University of Mannheim, ISBN 3-89838-471-3. infix. Berlin, 2000.

[15] Park K. and Willinger W., *Self-Similar Network Traffic and Performance Evaluation*, Wiley-Interscience, 1st Edition, 2000.

[16] Farber J., “Network Game Traffic Modelling” in *Proc. of NetGames2002*, Braunschweig, Germany, 2002, pp.53-57.

[17] Armitage G., “An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3”, in *Proc. of ICON*, Sydney, Australia, 2003.

[18] Borella M.S., “Source Models for Network Game Traffic”, *Computer Communications*, vol.23, no.4, pp.403-410, 2000.

[19] Pantel L., Wolf L.C., “On the Impact of Delay on Real-Time Multiplayer Games”, in *Proc of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 12-14, 2002, Miami, FL, USA.

[20] Henderson T., “Latency and User Behaviour on a Multiplayer Game Server”, in *Proc. of the 3rd International Workshop on Networked Group Communication* (NGC01) pp.1-13, London (UK), Nov 2001.

[21] Fitzek F., Schulte G., Reisslein M., “System Architecture for Billing of Multi-Player Games in a Wireless Environment Using GSM/UMTS and WLAN Services”, in *Proc. of NetGames2002*, pp.58-64, Bruanschweig, Germany, Apr 2002.