A RIO-Like Technique for Interactivity Loss-Avoidance in Fast-Paced Multiplayer Online Games

CLAUDIO E. PALAZZI, Università di Bologna AND University of California at Los Angeles
AND
STEFANO FERRETTI, STEFANO CACCIAGUERRA, AND MARCO ROCCETTI
Università di Bologna

The astonishing increase in the spread of the Internet has given rise to a globally connected community proficient at deploying online games for a large number of participants geographically located very far from each other. However, online games are characterized by more stringent requirements than traditional distributed applications deployed over the Internet can fulfill Indeed, one of the key factors in determining the success of an online game is its ability to rapidly deliver events to the various game servers that maintain the state of the game over the network. We have already demonstrated [Palazzi et al. 2004] that in this context adapting RED (random early detection) techniques, borrowed from queuing management, can improve the global responsiveness of a game. However, this solution may not be sufficient for a specific class of online games. We deem that fast-paced multiplayer online games (such as shoot 'em ups, for example) in which participants have to behave frenetically, must guarantee a very high degree of interactivity, even at the cost of partially sacrificing the consistency of the game state. In this case having only a partially consistent view of the game state will not affect a player's amusement as much as delaying action-processing activity will. Hence we explore the possibility of applying a RIO-based (RED with in and out) algorithm to manage game delivery to the various game servers, in order to improve the degree of interactivity for fast-paced online games. Preliminary experimental results confirm the viability of our approach.

Categories and Subject Descriptors: K.8.0 [**Personal Computing**]: General—*Games*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Multiplayer computer games, online entertainment, event delivery service, interactivity, consistency, evaluation, synchronization

1. INTRODUCTION

Nowadays, two main reasons above others attract an increasing number of researchers and developers to online electronic entertainment. The first is the very high level of revenue generated every year, which even surpasses the movie business. The second reason is the correlation between the 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.

This work was funded in part by the Italian M.I.U.R., Interlink project and 60% (IT)..
Authors' addresses: C.E. Palazzi is with the Università di Bologna and the University of California at Los Angeles; S. Ferretti, S. Cacciaguerra, and M. Roccetti are with the Università di Bologna, Bologna, Italy; emails: {cpalazzi, sferrett, scacciag, roccetti}@cs.unibo.it}

Under this second aspect, it is particularly interesting to analyze one of the most innovative challenges in networked electronic entertainment: the deployment of an efficient architecture to support massive multiuser online games (MMOGs) over a best-effort network [Griwodz 2002; Mine et al. 2003; El Rhalibi 2004].

Creating an enjoyable online game requires the convergence of solutions belonging to different technical areas. In particular, our focus here is on networking and the computational load on the game servers' architecture. For example, the mirrored game server is an efficient way to provide support for MMOGs; it deploys a constellation of communicating game state servers (GSS) over the network [Cronin et al. 2004; Ferretti and Roccetti 2004]. Each GSS maintains part or the whole game state, takes charge of event deliveries to/from other GSS, 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 GSS, an efficient *synchronization scheme* needs to be employed.

In a previous study [Palazzi et al. 2004], we proposed an innovative game synchronization scheme, named ILA (interactivity loss-avoidance), specifically designed to provide efficient event delivery synchronization in multiplayer online games. The ILA mechanism is able to increase the degree of playability of online multiplayer games by keeping delays in event delivery under a human-perceptivity threshold, while preserving game-state consistency and fluency in favor of the player. Simply stated, this result was obtained by discarding events considered obsolete and employing a dropping probability that depends on the perceived responsiveness at the GSS.

However, our experiences with online games over the Internet leads us to claim that there exist cases where even discarding all the obsolete events in a game is not enough to ensure interactivity. This is particularly true for a class of games that requires extremely fast, 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. Typical examples are the shoot and beat 'em up games. Simply put, we propose to enhance the ILA scheme by adding a further probability function that will even discard some non-obsolete events, when throwing away all the obsolete ones is not enough to ensure an adequate degree of interactivity.

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

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

The remainder of this article is organized as follows. In Section 2 we briefly survey the theoretical background, which is the basis of our 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 evaluation. Finally, Section 6 concludes the article.

2. BACKGROUND

Interactivity vs Consistency

Distributed interactive games are characterized by two main requirements that cannot be considered independent of each other: interactivity and consistency [Palazzi et al. 2004; Ferretti and Roccetti 2004]. 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 [Palazzi et al. 2004; El Rhalibi 2004; Cronin et al. 2004]. This means that the time elapsed from the generation of the event at a sending GSS to its processing time at each receiving GSS results in values below a specific average. For example, scientific literature declares that a delay of 50 ms is not perceived at all by players, while at 150 ms the players' performance may be 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 car racing, but can be relaxed for more strategic games.

A low variance of these values should be guaranteed, and not only to obtain a factual and smooth progression in visualizing the game on the player's screen,

There must be a consistent, uniform, and simultaneous view of the game state at all the nodes in the system [Ferretti and Roccetti 2004]. The easiest way to guarantee consistency is to make the game proceed in discrete *locksteps* [Steinman 1995]. However, allowing a single move for each player and synchronizing all the agents before moving toward the next round, really impairs the interactivity of the system.

In a previous work [Palazzi et al. 2004], we demonstrated that a good level of interactivity coupled with full-consistency may be obtained by exploiting the notions of obsolescence and correlation, surveyed in the next section.

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

Conversely, consistent lags between the generation of the event and the visualization of action on the screen, irremediably compromise the velocity of the action, which is the most entertaining component of this kind of game**.**

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 [Ferretti and Roccetti 2004]. On the other hand, it is also common knowledge that totally ordered delivery approaches increase complexity, and, most of all, are the basis for all the delays experienced by the system [Cheriton and Skeen 1993].

Recent studies demonstrate that exploiting the semantics of the application can be put to good use to relax the requirements for reliability and the delivery of total order, thus augmenting interactivity [Palazzi et al. 2004; Ferretti et al. 2004]. Some events, in fact, can lose their significance as time passes: new actions may make the previous ones irrelevant.

For example, where there is a rapid succession of movements by a single agent in a virtual world, the event representing the last destination makes the older events obsolete. Thus *obsolescence* can be defined as the relation between two received events e_1 and e_2, generated at different times $t(e_1) < t(e_2)$, thereby diminishing the importance of e_1 and the need to process it. Dropping obsolete events before processing them clearly reduces the computational cost at the GSS and speeds-up the execution of fresher events.

To define a game event as obsolete, we have to be sure that consistency will not be weakened. To this end, we also have to introduce the notion of *correlation*. Two events, say e_1 and e_2, are correlated if the final game state depends on their order of execution.

Hence, to determine the obsolescence of a chain of subsequent events, correlation is to be taken into consideration.

In fact, an event e_3 makes a previous event e_1 obsolete only if there is not another event e_2 (correlated to e_1), temporarily interleaved between e_1 and e_3 that breaks the obsolescence relationship 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 destination GSS in the same order as generated and cannot be subject to any dropping action.

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

Interactivity Maintenance with RED

Ferretti and Roccetti [2004] demonstrated the interactivity benefits attainable by exploiting the semantics of a game during its evolution towards relaxing the requirement for the delivery of total order.

In their scheme Ferretti and Roccetti [2004] propose that the player's actions be collected by the closest GSS, transformed into events, and finally forwarded to the other GSS in order to maintain a globally identical view of the game state. Events are marked at their creation with a generation timestamp and then sent to the destination: hence they are orderable.

Obviously, a global conception of time must be maintained by all the GSS, for example by exploiting a variety of solutions that enable the synchronization of their physical clocks [Drummond and Babaoglu 1993; Cristian 1989; Mills 1991], or by employing new technological synchronization devices such as GPS.

Each receiving GSS considers the arrival time of the event and measures the difference in elapsed time since its generation; the resulting value is called the *game time difference* (GTD). The GTD of the event is then compared to 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 that exploits the obsolescence notion in order 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 the Internet [Floyd and Jacobson 1993] we have recently enhanced the aforementioned interactivity-restoring mechanism with the interactivity loss-avoidance (ILA) approach [Palazzi et al. 2004].

The main innovation of the latter scheme is that it can anticipate the loss of interactivity, discarding packets when the level of interactions among the GSS declines significantly.

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

Even if, similarly to RED, ILA utilizes a uniformly distributed dropping function, the parameter taken under control is the average GTD instead of the average queue size. Upon the arrival of each packet each GSS determines the GTD of the event upon its arrival, namely the *sampleGTD*, 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 probability p, without processing them. If *avg_GTD* exceeds

a subsequent limit, *p* is set equal to *1*, and all obsolete events waiting for processing are discarded.

3. 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 high degree of interactivity while 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 degree of interactivity, even at the cost of sporadically ignoring the full-consistency requirement. This is the case when the core attractiveness for players emerges from feverish, sometimes even chaotic, action sequences, namely, *fast and furious* multiuser online games.

To advance this aim, our intention here is to add the possibility of discarding 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, one for obsolete and another for non-obsolete events, respectively, with specific boundaries and slopes that work independently of each other and take action in sequence with the increase in game event' GTDs at the GSS.

Obviously, dropping non-obsolete events without consequences can be done only for a category of games, where small inconsistencies will not be highly deleterious to the goal of the game and to the players' entertainment (e.g., fast-paced games).

Even in this case, if the number of dropped non-obsolete events becomes significant, a mechanism to restore consistency may be required to re-establish a coherent game state view among all the GSS [Mauve 2000].

Hence we propose enhancing our ILA scheme with new features derived from the integration of a RIO-like algorithm in place of the RED-like one. The RIO (RED with in and out) scheme is an enhanced version of RED mechanism that is able to discriminate between two different classes of traffic, non-prioritized (out) and prioritized (in), and calculates two distinct dropping probabilities.

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

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 $Pmax_o$ or $Pmax_v$, respectively.

Persistent states of low interactivity result in high *avg_GTD*, and hence in a high probability for being discarded. High dropping probability values (for $Pmax_o$ or $Pmax_v$) will cause the GSS to discard events without processing or forwarding them, thus helping to restore timely interactions between servers.

Since valid (i.e., non-obsolete) events are strictly linked to consistency, discarding them should be considered only as a last resort, in case of a heavy disruption of interactivity. Hence our scheme starts dropping obsolete packets much earlier than valid ones. We have set the parameters such that the algorithm throws away all the obsolete

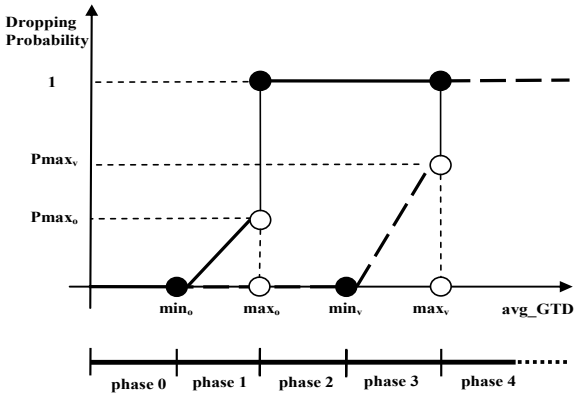


Fig. 1. Probability functions for discarding events.

packets before considering the probability for dropping any valid events; this is done by choosing max_o smaller than min_v.

Moreover, the degree of aggressiveness with which packets are dropped depends on the class they belong to, and can be decided by adjusting the values of $Pmax_o$ and $Pmax_v$.

The new ILA-RIO algorithm, which implements our scheme in all its phases, is shown in Figure 2, and is obtained by endowing the RED algorithm in Floyd and Jacobson [1993] with RIO features.

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

In particular, the packet's GTD is calculated (*sample_GTD*, line 1) as the difference in time between the generation of the game control event at the sender GSS and its delivery to the appropriate 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]   }
14]  }
15] }
```

Fig. 2. The ILA-RIO algorithm.

Table. I. Configuring the GSS

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:

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

where w is a sensitivity coefficient, with values (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 discarding operations are performed. Conversely, when *avg_GTD* is between min_o and max_o, the scheme is in phase 1 and lines 4-5 are executed.

Basically, a probability for dropping is computed in order to establish whether an obsolete event should be discarded. Such a probability increases until an event is discarded. This is done by exploiting a *counter* variable in order to have a uniform distribution of drops, following the method explained in Floyd and Jacobson [1993].

If *avg_GTD* grows beyond max_o, the scheme enters phase 2, or successive and all obsolete packets have to be discarded in order to re-establish 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 with a certain probability ($Pmax_v$), with a behavior analogous to that already explained in phase 1 are dropped, while in the latter case all events, with no distinction, are discarded.

4. SIMULATION ASSESSMENT

To evaluate our event-processing strategy, we simulated a general mirrored game server architecture made up of various GSS dispersed over the Internet. As previously mentioned, the events generated were totally ordered on the basis of a global notion of time. This can be achieved either by resorting to the variety of software solutions for clock synchronization proposed in the literature [Drummond and Babaoglu 1993; Cristian 1989; Mills. 1991] or by exploiting a technological device useful for synchronization like the GPS.

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

Based on results obtained by other authors [Park and Willinger 2000; Farber 2002], the values of the network latencies for each of the sending GSS and the receiving GSS_0 were obtained based on a lognormal distribution with the average and standard deviation values shown in Table I.

Furthermore, the rate of event generation, i.e., the interval of time between the departures of two subsequent events at each GSS, was sampled from a lognormal distribution (average equal to 30 ms and standard deviation equal to 10 ms). These values represent, approximately, the traffic generated by 5 to 10 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 previous ones obsolete was set, respectively, at 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 reliable communication.

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

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

Moreover, the other parameters 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 points can be made about the most appropriate values for the parameters cited above. The phase boundaries 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 screen is the first perceivable symptom of interactive degradation. The threshold for the more aggressive phase 2 should be chosen so that it can be surpassed when the lag results in annoyingly low-performance for the players.

We give the following reasons for our chosen values: the scientific literature declares that a delay of 50 ms (i.e., our min_o parameter) is not perceivable by players; a lag of 150 ms (i.e., our min_v and GIT parameters) results in disturbing the players' performance; and a delay of 225 ms (i.e., our max_v parameter) could represent an upper bound for playable interactions [Armitage 2003; Borella 2000; Pantel and Wolf 2002; Henderson 2001]. These limits hold for games based on vehicle chases, first-person shooters, and fast shoot 'em ups, but can be relaxed for strategic games (e.g., Starcraft, Age of Empire, etc. [Fitzek et al.. 2002]).

5. RESULTS

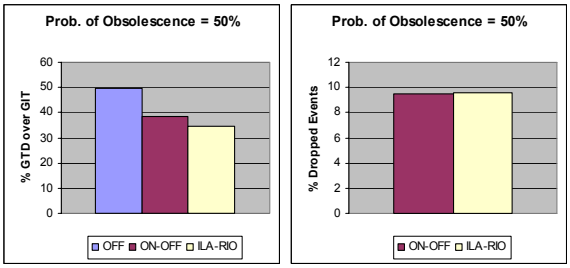
We intend to demonstrate the benefits that can be attained by implementing an event-discarding algorithm in case of a trend toward an increase in GTDs.

In a previous work we assessed the efficacy of the single ILA (RED-based) mechanism in a similar scenario [Palazzi et al. 2004]. In particular, we saw an improvement of 27% and 4% on the average GTD w.r.t. OFF and ON-OFF, respectively, and an improvement of 66% and 41% on the standard deviation w.r.t. OFF and ON-OFF.

As to the approach presented in this article, in Figures 3 and 4, we compare the ILA-RIO, ON-OFF, and OFF schemes for the following:

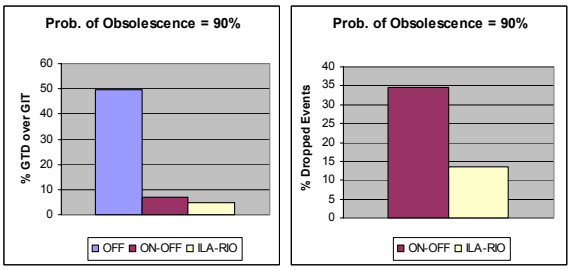
(1) the percentage of events that arrived at GSS_0 with a GTD value larger than the GIT; and
(2) the number of events dropped by ILA-RIO and ON-OFF.

Figures 3 and 4 refer to a specific event-trace configuration with different obsolescence probabilities for events.



(a) (b)

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



(a) (b)

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

It can be seen in Figures 3(a) and 4(a) that the ILA-RIO and ON-OFF schemes outperform the traditional OFF method in both configurations in terms of GTDs. Moreover, ILA-RIO further reduces the number of events with GTDs above the GIT w.r.t. ON-OFF. These results give a preliminary confirmation that ILA-RIO is able to guarantee a higher degree of interactivity compared to the two alternative approaches.

Furthermore, while in the former event-trace configuration there are no significant differences between ILA-RIO and ON-OFF in the number of dropped events (Fig. 3(b)). In the second configuration, ILA-RIO greatly reduces this value (Fig. 4(b)). Therefore, the ILA-RIO scheme augments the evolution of the game's fluency.

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

Table II 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 number of dropped valid events diminish when the percentage of obsolete events is greater, since this suggests a lower percentage of valid events.

Table II. Percentage of Obsolete and Valid Discarded Events in ILA-RIO

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

In particular, while a small number of valid events were discarded, corresponding to the first event trace (probability of obsolescence equal to 50%), no valid event was 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 in phase 2. Simply stated, interactivity is promptly restored by dropping only obsolete events without the need to discard valid ones.

6. CONCLUSIONS AND FUTURE WORK

In this article we presented a new scheme for a fast event-delivery service for mirrored GSS, 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 degree of interactivity among players, while preserving only partial consistency in the system.

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

ACKNOWLEDGMENT

We wish to thank the ACM CiE editor for her careful reading of our paper and for all her useful suggestions in helping to improve the quality of this article.

REFERENCES

ARMITAGE, G. 2003. An experimental estimation of latency sensitivity in multiplayer Quake 3. In *Proceedings of the ICON Conference* (Sydney, Australia, 2003).

BORELLA, M. S. 2000. Source models for network game traffic. *Computer Communications* 23, 4 (2000), 403-410.

CHERITON, D.R. AND SKEEN, D. 1993. Understanding the limitations of causal and totally ordered multicast. In *Proceedings of the 14th Symposium on Operating System Principles* (SOSP '93, Asheville, NC, 1993). 44-57.

CRISTIAN, F. 1989. Probabilistic clock synchronization. *Distributed Computing 3*,.3 (1989), 146-158.

CRONIN, E., KURC, A. R., FILSTRUP, B., AND JAMIN, S. 2004. An efficient synchronization mechanism for mirrored game architectures. *Multimedia Tools and Applications 23*, 1 (2004), 7-30.

DRUMMOND, R. AND BABAOGLU, O. 1993. Low-cost clock synchronization. *Distributed Computing 6*,.3 (1993), 193-203.

EL RHALIBI, A. 2004. Peer-to-peer architecture and protocol for a massively multiplayer online game. In *Proceedings of the 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment* (NIME'04, GLOBECOM 2004, Dallas, TX, 2004).

FARBER, J. 2002. Network game traffic modelling. In *Proceedings of the NetGames 2002 Conference* (Braunschweig, Germany, 2002). 53-57.

FERRETTI, S. AND ROCCETTI, M. 2004. A novel obsolescence-based approach to event delivery synchronization in multiplayer games. *Int. J. Intelligent Games and Simulation.3*, 1 (2004), 7-19.

FERRETTI, S., ROCCETTI, M., AND CACCIAGUERRA, S. 2004. On distributing interactive storytelling: Issues of event synchronization and a solution. In *Proceedings of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment* (TIDSE 2004, Darmstadt, Germany). LNCS 3105, Springer Verlag, 219-231.

FITZEK, F., SCHULTE, G., AND REISSLEIN, M. 2002. System architecture for billing of multi-player games in a wireless environment using GSM/UMTS and WLAN services. In *Proceedings of the NetGames2002* Conference (Bruanschweig, Germany, April 2002) 58-64.

FLOYD, S. AND JACOBSON, V. 1993. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking 1*, 4 (1993), 397-413.

GRIWODZ, C. 2002. State replication for multiplayer games. In *Proceedings of the NetGames2002 Conference* (Braunschweig, Germany, 2002). 29-35.

HENDERSON, T. 2001. Latency and user behaviour on a multiplayer game server. In *Proceedings of the 3rd International Workshop on Networked Group Communication* (NGC01, London, Nov. 2001). 1-13.

MAUVE, M. 2000. Distributed interactive media. Ph.D. dissertation, University of Mannheim. Berlin, 2000.

.MILLS, D. L. 1991. Internet time synchronization: The network time protocol. *IEEE Trans. on Communications 39*, 10 (1991), 1482-1493. MAUVE, M. 2000. Distributed interactive media. Ph.D. dissertation, University of Mannheim. Berlin, 2000.

MINE, R. M., SHOCHET, J., AND HUGHSTON, R. 2003. Building a massively multiplayer game for the millions: Disney's Toontown online. *ACM Computers in Entertainment 1*,1 (2003), 15.

PALAZZI, C. E., FERRETTI, S., CACCIAGUERRA, S., AND ROCCETTI, M. 2004 On maintaining interactivity in event delivery synchronization for mirrored game architectures In *Proceedings of the 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment* (NIME'04, GLOBECOM 2004, Dallas, TX, 2004).

PANTEL, L. AND WOLF, L. C. 2002. On the impact of delay on real-tme multiplayer games. In *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (May 12-14, 2002, Miami, FL).

PARK, K. AND WILLINGER, W. 2000. *Self-Similar Network Traffic and Performance Evaluation*. 1st ed. Wiley-Interscience, 2000.

STEINMAN, J. S. 1995. Scalable parallel and distributed military simulations using the SPEEDES framework. In *Proceedings of the Second Electronic Simulation Conference* (ELECSIM95).

Received January 2005; accepted March 2005