On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures

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*—Online amusement applications, as distributed multiplayer videogames and interactive storytelling, are gaining increasing attention both from entertainment industry and from scientific community. Providing a pleasant experience to players requires a rapid delivery of game actions among the various nodes in the network. A high playability degree should be guaranteed independently of user's location, utilized device (PC, PDA, cellphone), type of connection (wired, wireless), and number of contemporary players. To this aim, we have devised an innovative approach to design the event delivery service for networked multiplayer game applications. Exploiting the semantics of the game, our scheme relaxes the ordering and reliability properties, upholding the interactivity level while preserving the game state consistency. The main contribution of our work is to show the benefits in event delivery synchronization obtainable employing, in this context, RED techniques borrowed from networking queuing management.**

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

I. INTRODUCTION

Many years elapsed since Higginbotham's tennis for two appeared in the world (Brookhaven National Laboratory, a US nuclear research lab in Upton, New York) as the first videogame, in 1958 [1]. That non-commercial first experience of interactive entertainment immediately gained a vast popularity and traced the path for the horde of commercial descendants that have invaded arcades and homes till our days. Nowadays, two main reasons above the others attract an increasing number of researchers and developers toward electronic amusements. The first one is the very high level of revenues generated every year, which surpasses even the cinematography industry [2]. The second reason, but clearly not less worthy, is represented by the correlation between problems that emerge in developing innovative game experiences and those typical of various other "conventional" research fields in Computer Science. Under this aspect, it is of particular interest to analyze one of the most innovative challenges in electronic amusements: on-line entertaining applications. Specifically, we concentrate on videogames in Internet, potentially engaging a very elevated number of players: namely, Massive Multiuser Online Games (MMOGs).

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

Critical issues regarding MMOGs, as well as distributed interactive cyberdrama generation, have identical counterparts in Distributed Interactive Simulations and in Networked Virtual Environments [3, 4, 5, 6, 7, 8, 9, 10]. Indeed, creating an enjoyable online game entertainment requires the convergence of solutions belonging to extremely diverse groups of different technical areas. Examples are represented by networking, computer graphics, animation, multimedia design, human-computer interaction, software engineering. Solutions designed in traditional fields of computer science could thus be easily extended to MMOGs. In particular, our focus here is centered on networking and computational load at the servers of the game platform architecture.

Customers have always been attracted by the possibility to expand their game experience sharing fun with other users. The ever-increasing popularity of the Internet and the exploding market of connectable handheld devices, always looking for new killer applications, push the game industry to propose effective distributed logical platforms proficient at engaging an unlimited number of contemporary users. This large and emerging market is driving researchers and experts to develop novel distributed solutions able to efficiently sustain interactive multiplayer networked game sessions [11, 12, 13, 14, 15, 16].

Focusing on a well-organized management of large-scale distributed games, a typical underlying architecture deploys over the network one or more Game State Servers (GSSs) communicating each other through a mirrored game server communication architecture. GSSs maintain part or the whole game state, take charge of event deliveries to/from other GSSs or to/from Input/Output Clients (I/O_Cs) connected with them, and can implement policies aimed at increasing the global performance of the system. Hence, in order to guarantee a uniform view of the game state among all GSSs, an efficient synchronization scheme needs to be employed.

Since a pleasant game experience for the final user is characterized by strict real time requirements in processing actions, playability in interactive gaming applications results extremely sensitive to delays in event deliveries. Loss of responsiveness in a MMOG can be caused both by an intense traffic load in the network that slows down message transmissions and by an excessive amount of events waiting to be processed by a single GSS. Consequently, a proficient

synchronization algorithm should be able to face these two situations, preserving a high level of interactivity among distributed players while maintaining an identical contemporary view of the game state in all the nodes of the system.

In this paper, we present a novel synchronization scheme, named ILA (Interactivity-Loss Avoidance), able to uplift the playability degree of online multiplayer games by maintaining the event delivery delays under a human-perceptivity threshold. This result is obtained discarding events that can be considered obsolete since the arrival of "fresher" ones, with a dropping probability which depends on the perceived responsiveness at GSSs. Limiting the number of messages in the system reduces both the processing and the network latency factors in the total delivery time experienced by the other events.

The core innovation in our work is the utilization of the RED mechanism imported from networking solutions and adapted here to preserve the interactivity degree in MMOGs. In essence, the idea is to monitor the interactivity level of the system and, when required, preempt the loss of interactivity discarding some events that have lost their importance during the game execution. As we show, this presents the prominent advantage to avoid more drops later in the attempt to restore an already disrupted interactivity. Consequently, players perceive a better playability thanks to a smoother progression of the executed events. ILA is specifically functional for Mirrored Game Server architectures and does not jeopardize the uniformity of the game state views at the various GSSs.

The remainder of this paper is organized as follows. Section II reviews some recent results in the field of interactivity performance for online gaming. In Section III we analyze some design issues as: i) the possible architectures to implement MMOGs (Subsection A), ii) the tradeoff between interactivity and consistency as the two main requirements in distributed interactive gaming (Subsection B), and iii) the notions of obsolescence and correlation as effective tools in relaxing the strict total order paradigm to the aim of augmenting interactivity (Subsection C). Section IV is concerned with interactivity maintenance techniques. After presenting some details about the inspiring RED (Subsection A) and Interactivity Restoring techniques (Subsection B), we present ILA, our novel approach to avoid interactivity loss (Subsection C). In Section V we describe the simulative environment adopted as the test bed and the metrics chosen to compare the various server synchronization schemes. Section VI presents simulation results. In particular, Subsection A shows the performance gain in utilizing an obsolescence based discarding mechanism. Subsection B, instead, discusses the performance obtained with the ILA algorithm. We conclude the paper with some comments and ongoing enhancements to our work in Section VII.

II. Related Work

Trying to improve the interactivity performance of a distributed game architecture, two main causes for delays have to be analyzed: network latencies and computational costs. Several research works have already brought contributions to the factual developing of efficient synchronization schemes. *Compression* and *aggregation* considers networking having a dominant position when dealing with the delays and thus with the playability of a MMOG [17]. In particular, packet compression tries to speed up transmissions by reducing bandwidth requirements. Indeed, minimizing the number of bits needed to represent a game information is a proficient method to diminish the traffic present in the network. Aggregation is another technique attempting to limit the bandwidth required by the application. Specifically, before being transmitted, packets are merged in larger ones thus reducing the overhead. Both compression and aggregation schemes, however, pay the latency benefits achieved with an increment in computational costs. Information compressed and aggregated, in fact, needs to be recovered with decompressing and disaggregating algorithms at the receiving server, thus increasing the time required to process each single event. Moreover, aggregation can origin further waste of time if a transmission is delayed while waiting for having available other events to aggregate.

In the attempt to reduce both the traffic load in the network and the computational cost to process each game event, *Interest Management* techniques have been devised [18]. In some game scenario, events generated are relevant only for a small fraction of the users. Therefore, implementing an area-of-interest scheme for filtering events, as well as a multicast protocol, could be put in good use to match every packet with the nodes that really need to receive it and, consequently, to reduce both the traffic on the channel and processing burden at each node [19]. Conversely, a tradeoff exists between the computation spared at the destination by receiving only a limited number of packets and that one expended at the sending GSS for implementing the filtering scheme. Games having interest-areas occupying a significant portion of the global virtual environment could hence be further delayed if Interest Management schemes would be implemented. For instance, think about games having a simple scenario wholly includable in the screen of the user's device (e.g. as in Pac-Man). In this case, all the game actions have to be forwarded to all the participants thus making useless a filter scheme while maintaining its computational cost.

Slightly detaching playability from the real responsiveness of the network, optimistic algorithms for synchronizing game state at servers can be utilized in order to avoid delay perception at destination. In case of lousy interactivity between GSSs, in fact, an optimistic approach executes events before really knowing if ordering would require to process other on-the-way events first. Game instances are thus processed without wasting any time in waiting for other eventually coming packets. On the other hand, this performance gain is paid with some occurrence of temporary consistency loss. *Standard Time Warp* and *Breathing Time Warp* represents typical exemplars of this family of algorithms [20, 21]. *Rollback* based techniques are exploited to reestablish the consistency of the game state. The problem, in this case, is that the use of this realignment techniques may further impact on the responsiveness of the system.

Dead Reckoning is another method that can cause some temporary incoherence between the factual game state and the assumed one at the server [22]. In fact, attempting to limit the

bandwidth required by the application, this scheme utilizes a reduced frequency in sending update packets while compensating the lack of information with prediction techniques. Obviously, predicted movements and actions are not always trustful. Therefore, even in this case, convergence techniques need sometimes to be exploited to recover from provisional instances of the game having some momentary inconsistencies. These eventual restoring actions further impact on interactivity and playability of the game.

All these mechanisms propose enhancements that can improve the performance of a MMOG. Nonetheless, we have exposed some limits and situations where those schemes could fail. We present here a novel algorithm, specifically designed for an efficient event delivery synchronization in multiplayer online games. Our scheme can be integrated with all the works mentioned above, adding further gains in reducing the delays of the system and bringing benefits, both for network and computational loads, even if singularly applied.

III. DESIGN ISSUES

A. Game Architectures

Typically, network architectures supporting MMOGs can be distinguished in three main categories: Centralized Client-Server, Peer-to-Peer, Mirrored Game Server. The Centralized Client-Server architecture represents a simple solution, but the unique bottleneck can limit its efficiency and scalability [14, 23, 24]. Having a unique server simplifies the maintenance of a correct game view in all the nodes of the system; conversely, it embodies a single point of failure. Fully distributed architectures, as Peer-to-Peer solutions, spread the traffic load among many nodes and result in a more scalable and failure-resilient system [25]. At the same time, identical copies of the current game state need to be stored at each node. This raises the necessity of devising some fully-distributed coordination scheme among clients to guarantee the coherence of all game views. Neither diverse networking delays nor any other factor should be able to compromise the uniformity of the game conditions. Moreover, with Peer-to-Peer architecture, IP multicast should be employed to reduce the bandwidth requirements, but this technology is neither generally available, nor enough mature for the kind of application we are considering.

Both Centralized Client-Server and Peer-to-Peer architectures present advantages and disadvantages in their employment to support MMOGs. Mirrored Game Servers represent an alternative architecture which efficiently collects the positive aspects of the other two [26]. Indeed, a hybrid architecture with multiple distributed servers present several benefits that could reveal it as the most appropriate solution for online multiplayer games. Having multiple replicas of the servers allows each client to connect in a classic client-server fashion to the closest mirror, thus reducing the communication latency. Mirrors are limited in number if compared to a fully distributed architecture and contain copy of the current game state. The connection may follow the Peer-to-Peer paradigm in order to exchange game state messages. Other advantages are the absence of a single point of failure, the networking complexity maintained by the servers, and the possibility to implement authentication. Even if synchronization schemes are still required to ensure the global consistency of the game state hold by the various servers, this requirement is made easier than in Peer-to-Peer architectures thanks to the lower number of nodes involved. All these reasons depict Mirrored Game Servers as the most appropriate architecture for MMOGs.

B. Interactivity vs Consistency

Distributed interactive gaming are characterized by two main requirements which cannot be considered independent one from the other: interactivity and consistency. The former refers to the delay between a game event generation in a node and the time at which the other nodes become aware of that event. Therefore, it includes both the network latency and the processing time. Having a high level of interactivity represents a fundamental quality for a MMOG. Hence, in order to assure an enjoyable playability to the final user, external stimuli generated by players need to be processed under a human-perceptivity threshold. This means that the time elapsed from the event generation at sending GSS and its processing time at the receiving GSS must maintain a low average value. Not only, in order to obtain a factual smooth progression in the game visualization on the player's screen, a low variance must be guaranteed too. Frequent changes in the perceived velocity of the game, depending on excessive traffic present at the GSS and regardless of the effective game evolution, result in annoying the customers, pushing them away from ever reattempting such an unpleasant experience.

Consistency regards the contemporary uniformity of the game state view in all the nodes belonging to the system [27]. Depending on the features of the game, consistency requirements may be absolute or partial. In the former case, each node must always have an identical view of the game state, while in the latter small discrepancies may occur. Whether a game requires *absolute* or *partial* consistency depends on the unique rules correlating the diverse player's area-of-interests.

The easiest way to guarantee absolute consistency is to make the game proceed through discrete *locksteps* [28]. At each step the system waits until having received all the actions generated by the final users; only at this moment a new instance of the game is produced and propagated to all the nodes. Having a single move allowed for each player and synchronizing all the agents before moving toward the next round, for sure grants absolute consistency but, on the other hand, impairs the interactivity of the system. Obtaining both absolute consistency and high interactivity would require the employment of almost unlimited network and computation resources (very high bandwidth, very low latencies, very high speed at server to process events). Consequently, in order to design an efficient game architecture, a trade-off between the two attributes needs to be found.

C. Obsolescence and Correlation

Absolute Consistency can be attained through the employment of a totally ordered event delivery scheme. On the other hand, this would imply an increment of the complexity and, most of all, in the total delays experienced by the system [29, 30]. Waiting for the next in order action to be processed

while having other events ready in queue may sensibly slow down the evolution of the game, thus jeopardizing interactivity. Exploiting the semantics of the application can be put in good use to relax the total order delivery requirement and augment interactivity [31]. Some events, in fact, can lose their significance as time passes: new actions could make irrelevant the previous ones. For example, player's moves are generally represented by final absolute position in the message exchanged by the various nodes and, in case of rapid succession of movements of a single agent, the event representing its last destination makes obsolete the older ones. *Obsolescence* can thus be defined as the relation between two received events *e1* and *e2*, generated at different times $t(e1) < t(e2)$, by which the existence of event *e2* diminishes the importance of processing also event *e1* (without affecting consistency). Dropping obsolete events before processing them, clearly reduces the computational cost at GSSs and speeds up the execution of fresher events. Consequently, exploiting obsolescence may result in an enhanced interactivity of the global system.

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 *e1* and *e2*, are correlated if the final game state depends on their execution order. Correlation have to be taken into account to determine the obsolescence of an event. In fact, it might be the case when *e3* would make obsolete a previous event *e1* but a further event *e2* (correlated to *e1*), temporary interleaved between *e1* and *e3*, breaks this relationship of obsolescence. However, they are the only events that really need to be delivered to the destined GSSs in the same order as generated. Total order delivery requirement can thus be relaxed in case of non-correlated events. Their semantic independence, in fact, allows different GSSs to process them in diverse order without affecting consistency. This means that non-correlated events can be processed as soon as they are received without wasting any time in waiting preceding ones, again augmenting interactivity.

The enhancement to the synchronization scheme proposed in this work to augment the interactivity degree improves primarily the way to exploit the obsolescence notion rather than correlation. The interested readers in a deeper analysis of correlation may refer to [27].

IV. Interactivity Maintenance

A. RED Technique

Random Early Detection (RED) algorithm is an active congestion avoidance mechanism enforced at routers [32]. Traditional queue managements employ simple "tail drop" schemes that drop packets only when the queue overflows. Conversely, RED algorithm randomly discards packets earlier to notify sources about the incipient congestion. In this way, a single loss experienced by a sender smoothly decreases the entire congestion level of the network and keeps low the average queue size. The rationale lies in the gained capability of better accommodating occasional bursts of packets and avoiding situations in which several connections decrease their sending rate at the same time. Summarizing, RED avoids severe congestion and maintains a stable traffic level in place of dealing with it after occurred.

Every time the router receives a packet, the RED algorithm calculates the new average queue size and the probability to discard the packet. The computing method utilizes a uniform random variable that behaves better than a geometric random variable. In fact, a uniformly distributed discarding function avoids global synchronization thus attaining an unwavering course of transmissions. The dropping probability is bounded by two thresholds of the queue size: min_{th} and max_{th}. Within this interval, the probability to drop a packet increases from 0 to a maximum discarding probability (max_p). Under min_{th}, no packet is dropped and beyond max_{th} all packets are discarded.

B. Interactivity Restoring

In a precedent study, Ferretti and Roccetti demonstrate the interactivity benefits attainable exploiting the semantics of a game during its evolution to relax the total order delivery requirement [27]. The scheme proposed in that work foresees a Mirrored Game Server architecture and the concepts of obsolescence and correlation, as summarized respectively in Subsection A and C of Section III in this paper.

Specifically, player's actions are collected by the closer GSS and 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. 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, until the former value is lower than the latter, normal delivery operations are performed. Conversely, when the GTD value exceeds the GIT, the GSS turns on a stabilization mechanism which exploits the obsolescence notion to drop useless messages so as to bring the GTD back within the GIT.

As mentioned by the authors, this approach is based on the existence of a mechanism that synchronizes all the game system on a unique global conception of time. Finally, since only obsolete events are eventually discarded, this stabilization mechanism succeeds in reducing the GTD without causing inconsistencies in the game evolution.

C. Interactivity-Loss Avoidance: a Novel Scheme

Taking inspiration from the RED approach in case of incipient congestion [32], we propose to enhance the aforementioned Interactivity Restoring mechanism with the novel Interactivity-Loss Avoidance (ILA) approach. The main feature of this new scheme is the capability of avoiding interactivity loss before it happens, eventually discarding some packets when the level of interaction among GSSs descends significantly. In practice, ILA substitutes the previous binary dropping mechanism (OFF when interactivity is present and ON when interactivity is lost) with a continuously-working proactive mechanism that drops obsolete messages with a probability depending on the level of interactivity.

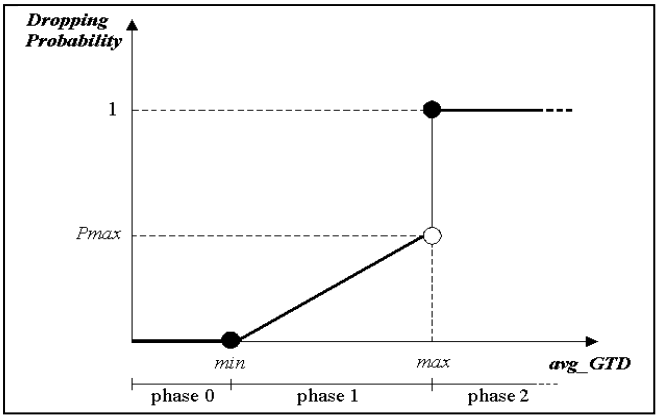

Figure 1. Discarding probability function.

Even if, similarly to RED, ILA utilizes a uniformly distributed dropping function, however, the parameter taken under delimited boundaries is the average GTD instead of the average queue size. This choice derives from the different goals of the two schemes: with RED we are trying to avoid buffer overflows, instead with ILA we want to preserve low delays in game command execution.

We focus now on some details of the algorithm. Upon each packet arrival, the GSS determines the GTD of the relative event, namely *sample_GTD*, and feeds a low pass filter to compute the updated average GTD, namely *avg_GTD*. Sporadic delays in game event deliveries, in fact, does not necessary compromise the perceived interactivity. 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.

As illustrated in Fig. 1, three parameters and three phases characterize the algorithm: respectively *min*, *max* and *Pmax* (parameters), and phase 0, phase 1 and phase 2 (phases). In the graph, the y-axis represents the dropping probability corresponding to the *avg_GTD* indicated by the x-axis. For values of *avg_GTD* in *[0, min)* (phase 0) the mechanism performs normal operations, with no event drops, while in *[min, max)* (phase 1) obsolete events are discarded with the computed probability *p*. Finally, when in *[max, ∞)* (phase 2), all obsolete events are thrown away. The dropping probability is computed as a function of *avg_GTD* and *Pmax*. Persistent situations of low interactivity result in large values of *avg_GTD* and, hence, in high discarding probabilities. An elevated dropping probability will make the GSS discard events without processing or forwarding them, thus helping in restoring an adequate level of time interaction between servers.

A pseudocoded version of our algorithm is given in Fig. 2 and it proceeds as follows. After an initialization phase, the algorithm repeats a block of operations each time a new packet arrives at the considered GSS. In particular, line 1 calculates the *sample_GTD* as explained in Section IV, Subsection B, while *avg_GTD*, in line 2, is computed employing the low pass filter given below, in (1). In this filter, *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. The higher the value of *w*, the higher is the relative weight of the last sample in the current average. In order to speed up computations, *w* should be chosen as a negative power of two, thus allowing to implement a shift operation in place of the multiplication.

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

When *avg_GTD* lies below *min*, the process stays in phase 0 and normal operations are performed. Conversely, when the value of *avg_GTD* is comprised between *min* and *max* (line 3), the scheme is in phase 1 and the discarding probability function has to be applied to obsolete events. Specifically, as shown in (2), the probability *p* could be calculated as a fraction of *Pmax*; this fraction linearly corresponds to the relative position of *avg_GTD* in the interval *[min, max)*.

$$p = \frac{Pmax \times (avg\_GTD - min)}{(max - min)} \quad (2)$$

Following the approach proposed by RED, (2) may be transformed into (5) to speed up its computation. This can be done by defining two constants *L1* and *L2*, as shown in (3) and (4), that need to be determined just once, during the very first initialization of the algorithm. The parameters can be chosen wisely so that *L1* results a power of two, and a shift operation can be utilized in place of the multiplication thus sparing cycles spent by the processor.

$$L1 = \frac{Pmax}{(max - min)} \quad (3)$$

$$L2 = \frac{Pmax \times min}{(max - min)} \quad (4)$$

In conclusion, (2) can thus be easily rewritten as follows and employed in line 4 of the ILA algorithm:

$$p = L1 \times avg\_GTD - L2 \quad (5)$$

```
0] for each event_packet arrival
1]      determine the sample_GTD
2]      calculate the new average delay avg_GTD
3]      if ( min ≤ avg_GTD < max ) then
4]          calculate probability p
5]          determine if one obsolete_event has to be discarded
6]      else if ( max ≤ avg_GTD ) then
7]          discard all obsolete_events
8]      endif
9] endfor
```

Figure 2. ILA Algorithm.

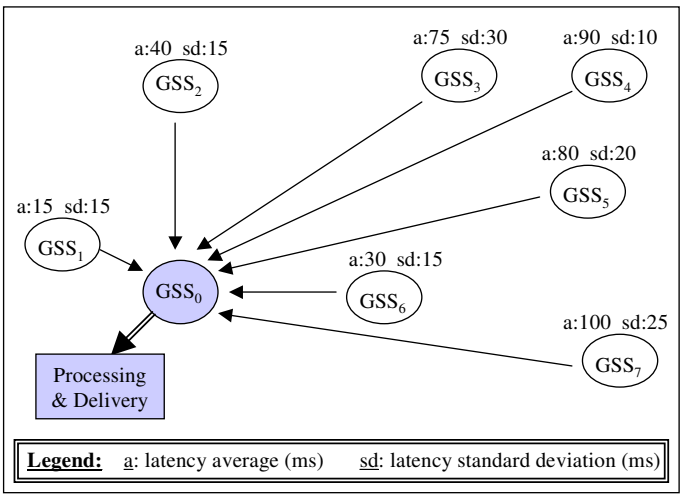

Figure 3. The adopted configuration.

So far, the probability function results in a geometric random variable distribution of the drops, while it would be desirable to discard events at a fairly regular intervals. Applying an incrementing counter to augment, at each iteration, the weight of the discarding probability, we can generate a uniform distribution of the dropping dispersion, thus being more resilient to temporary bursty periods [32].

To this aim, the probability p is compared with a randomly generated number R comprised in [0, 1], also taking into account the number of iterations elapsed since the last drop. Basically, in phase 1 an obsolete event is dropped when (6) is satisfied (line 5).

$$counter \geq (R/p) \quad (5)$$

Both *counter* and R are reinitialized each time obsolete packets are discarded. The dropping probability, instead, is computed with each new event arrival.

If *avg_GTD* grows beyond *max* (line 6), the scheme enters in phase 2 and all obsolete packets have to be discarded in the attempt of re-establishing interactivity (line 7).

V. Simulation Assessment

To evaluate our event processing strategy, we have simulated a general Mirrored Game Server architecture comprising various GSSs connected via diverse links over the Internet. Without any loss of generality, we assume that the events generated in the system may be totally ordered based on a global notion of time. It is out of the scope of this paper to propose a novel scheme to order events based on time, instead we claim that this goal may be accomplished exploiting a variety of different solutions proposed in literature [29, 30, 33, 34, 35], or employing technological synchronization devices, such as, for example, GPS.

The considered scenario includes a variable number of GSSs. For the sake of a deeper comprehension and without loss of generality, we have focused our attention on the event receiving aspect of a single GSS, pretending that the other GSSs are sending events to it. Fig. 3 depicts the adopted configuration of the network and shows the values assigned to the simulation parameters. GSS_0 is the receiving GSS and the others are the sending GSSs.

We carried out several simulation experiments with a number of servers varying in the range from four to seven. The involved GSSs for each different configuration is listed in Table I.

The values of the network latencies among the GSSs have been obtained based on a lognormal distribution having the average and standard deviation values as shown in Fig. 3 [36]. Also the average event size (200 Bytes), as well as the event generation rate at each GSS, is inspired by the games literature and varies from a normal traffic situation to an intense load one [37]. In particular, several experiments have been conducted with an interval of time between two subsequent event departures based on a lognormal distribution whose average was equal to 30ms and the standard deviation was set to 10ms.

By exploiting this configuration, we have generated a diverse trace file containing 1000 events for each GSS of the considered scenario. Each trace file also includes the information needed to identify (correlated and) obsolete events. In essence, in our simulations, we have set to 90% the probability that an event makes obsolete preceding events.

As in real commercial games, we have utilized UDP as the transmission protocol [38]. This protocol, in fact, responds better to the real time requirements of online game applications than TCP. Further, to circumvent the problems deriving from UDP's unreliability, we have implemented an application level retransmission scheme based on NACKs (Negative ACKnowledgments).

We have replicated each run to compare the outcomes of three different synchronization schemes: our proposed ILA scheme, the ON-OFF mechanism (Interactivity Restoring as reviewed in Subsection B of Section IV), the traditional OFF approach (having no discrimination of obsolete packets and no event discarding nor other algorithms to restore interactivity).

Focusing on the tuning of the ILA algorithm, we need to find an efficient tradeoff in adjusting the various parameters. In particular, we have chosen to 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.

TABLE I. Sending GSSs Involved in the Simulations.

Number of sending GSSs	Corresponding sending GSSs employed
4	GSS_1, GSS_2, GSS_3, GSS_4
5	GSS_1, GSS_2, GSS_3, GSS_4, GSS_5
6	GSS_1, GSS_2, GSS_3, GSS_4, GSS_5, GSS_6
7	GSS_1, GSS_2, GSS_3, GSS_4, GSS_5, GSS_6, GSS_7

The values for the parameters highlighted in Fig. 1 have been chosen keeping in mind that 150ms of time elapsing between the generation of a player's action and its execution in the system could be considered as a threshold for human perception of an annoying delay [39, 40]. Since we want ILA mechanism take action before that limit, we have set *min* = 50ms, *max* = 150ms (equivalent to the GIT for the ON-OFF scheme) and *Pmax* = 0.2.

Our aim is to guarantee the best possible playability to MMOGs. Since this passes through ensuring an high level of interactivity in the network and the absence of interruptions, we have chosen to demonstrate the benefits attainable by the obsolescence based dropping mechanisms analyzing the following metrics:

- The number of events having a GTD higher than the GIT.
- The average of the GTD values, their standard deviation, the minimum and maximum values.
- The cumulative function of the GTD values.
- The number of obsolete events dropped (ILA and ON-OFF only).

VI. RESULTS

A. *Obsolescence Based Discarding Schemes: Performance Evaluation.*

We intend to demonstrate here the interactivity benefits attainable by implementing a discarding algorithm for obsolete events in case of an increasing trend of the GTD values. To this aim, in Fig. 4, we compare for ILA, ON-OFF and OFF schemes the percentage of events arrived at GSS_0 with a GTD value larger than the GIT. As observable, both ILA and ON-OFF mechanisms outperforms the traditional OFF mechanism, independently of the number of sending GSSs employed in the scenario.

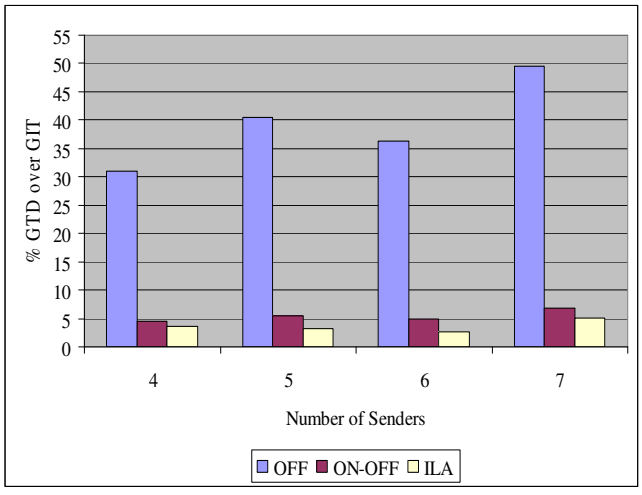

Figure 4. Percentage of events with GTD over GIT.

The trend of the outcomes, when we augment the number of servers utilized, is not monotonically increasing within the same synchronization scheme. First, this is due to the fact that we are considering a percentage rather than an absolute value. Second, this is also caused by the choice of conducting experiments using different selections of available servers. Specifically, the experimental scenario where six GSSs were involved was based on the experimental scenario with five servers plus the newcomer GSS_6 (see Table I). The adjoined GSS_6 has a latency value, 30ms, lower than the other GSSs, while the average latency for the other five sending GSSs, from GSS_1 to GSS_5, is 65ms (as shown in Fig. 3). For this reason, game events coming from GSS_6 have a much lower probability than the others to have a GTD value higher than the GIT. Even if the total traffic and thus also the sum of the delays are augmented, the percentage of events out of the interactivity threshold results slightly diminished. This is an obvious consequence of the fact that decreasing the latencies in the network reduces one of the causes of delay in event processing. As previously mentioned, the others are the queuing time at the receiving GSS (waiting to be processed) and the processing time at the receiving server.

The cumulative function of the GTDs embodies another tool proficient at evaluating the worth of ILA and ON-OFF schemes. Indeed, the more the line is concentrated in the left side of the chart, the higher is the percentage of events having a GTD lower than a certain threshold. In particular, Fig. 5 depicts the cumulative function of the GTDs in a scenario with seven sending GSSs, each one sending events to the receiving GSS_0. In this configuration, ILA has 93.86% of events with a GTD less or equal to the GIT of 150ms, ON-OFF hits the 89.40%, while OFF reaches only the 49.94%.

These results are coherent with the values of the average and the standard deviation of the GTDs considering all the events transmitted. Table II shows a sensible reduction in the values of both these metrics when ILA or ON-OFF are implemented in place of the traditional OFF scheme. Moreover, the two obsolescence based discarding schemes result more resilient to an increased event generation activity within our game architecture. This is evident if the case of seven sending GSSs is compared with the one employing only four sending GSSs. In this case, the average of the GTDs decreases from 19.72% (OFF) to 12.07% (ON-OFF) and 11.71% (ILA).

The reminder of our performance study aims at focusing on the advantages in utilizing a continuously acting scheme as ILA instead of a binary working approach like ON-OFF.

B. *ILA vs ON-OFF: a Comparative Evaluation*

We compare more in detail the two obsolescence based discarding schemes (ILA and ON-OFF), highlighting the benefits introduced by the proactive mechanism implemented by ILA. Indeed, in all the charts and tables presented so far (Fig. 4, Fig. 5, and Table II), ILA results always at the same level or, actually, slightly better than ON-OFF in the attempt of guaranteeing interactivity. Not only, Table II also shows that the standard deviation of the GTDs obtained employing ILA is always smaller than that obtained utilizing ON-OFF.

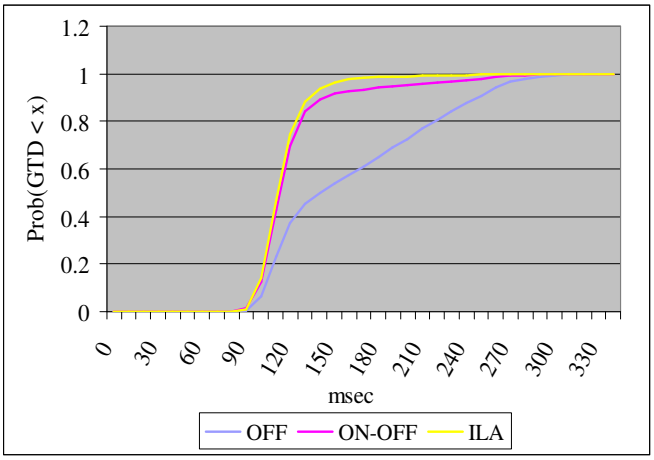

Figure 5. Cumulative function of the GTDs in a scenario with 7 GSSs.

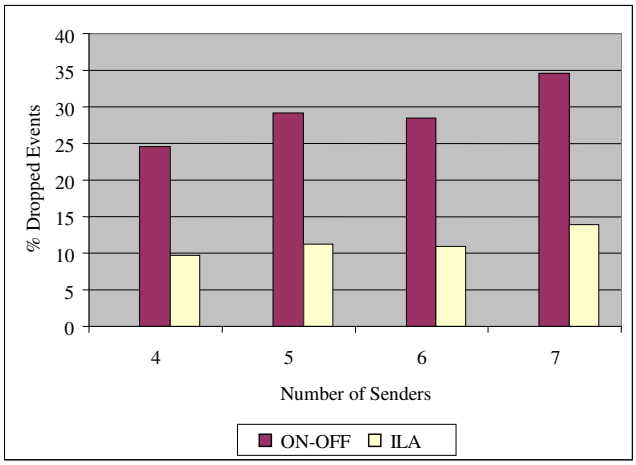

Figure 6. Percentage of dropped events.

This results in a more homogeneous flow of actions executed at the player's side, thus hiding to the customer the negative effects involved in utilizing a performance varying environment, likely the Internet, to support the game. The visually perceived evolution of the game results smoothened by ILA, thus providing a more pleasant game experience for the user.

Not only ILA obtains a slightly better interaction level with respect to ON-OFF, but the total number of discarded events to attain this positive result is definitively lower. In fact, Fig. 6 shows that the results in Fig. 4 and Table II are obtained by ILA at the cost of circa only 40% of the obsolete events dropped by ON-OFF.

This is a very important advantage obtained utilizing a proactive mechanism as ILA. In fact, even if obsolete events can be sacrificed to gain a better interactivity since consistency does not depend from them, they still are part of the game visual evolution. Dropping too many obsolete events, could result in sudden "jumps" and temporary interruptions of the images/video flow on the player's screen. These unpredictable gaps in the game plot could result annoying for customers and should be avoided every time it is possible.

In other words, we can say that even if both schemes ensures interactivity and consistency, ILA outperforms ON-OFF and founds an efficient tradeoff between the percentage of obsolete events to be discarded and a fluent visual progression of the game.

VII. Conclusion and Future Work

To guarantee a pleasant gaming experience to online players engaged by MMOGs, a high interactivity degree, as well as game consistency, has to be provided. Efficient synchronization schemes among Mirrored Game Servers are usually implemented as basic solutions. A proactive event discarding mechanism relying on the discrimination of obsolete events has been proposed as an innovative way to meet the aforementioned requirements. Inspired by the RED algorithm that manages queues at networking routers, ILA improves the fluency of the game progression on the player's screen, avoiding loss of interactivity at GSSs, instead of restoring it after having been disrupted. The benefits achievable in employing ILA as synchronization scheme among Mirrored Game Servers have been highlighted.

As a further enhancement of this work, we are currently studying the possibility of discarding also non-obsolete game events, under situations of particularly jeopardized interactivity. Taking inspiration from RIO technique, ILA scheme could thus be enhanced by employing two distinct discarding probability functions, respectively for obsolete and for non obsolete events [41]. In particular, it should be exploited when discarding all the obsolete events is not enough to restore an adequate level of interactivity, thus requiring, as last resort, to drop also some non-obsolete events. We are currently collecting data from simulations employing this enhanced ILA algorithm.

TABLE II. Maximum, Minimum, Average and Standard Deviation of the GTDs.

int = 30	4 GSSs			5 GSSs			6 GSSs			7 GSSs		
	OFF	**ON-OFF**	**ILA**	**OFF**	**ON-OFF**	**ILA**	**OFF**	**ON-OFF**	**ILA**	**OFF**	**ON-OFF**	**ILA**
MAX	324	324	325	325	324	277	318	319	278	345	345	300
MIN	88	88	86	88	88	88	87	88	88	93	93	93
AVG	142	116	111	153	120	115	148	119	114	170	130	124
ST.DEV	52	30	20	53	32	19	50	28	18	56	32	19

REFERENCES

[1] The First Video Game. Brookhaven National Laboratory (BNL), 2004. Web Site: http://www.bnl.gov/bnlweb/history/higinbotham.asp

[2] Kushner D., Masters of Doom, Random House, New York, N.Y, 2002.

[3] Smed J., Kaukoranta T., and Hakonen H., "Aspects of Networking in Multiplayer Computer Games", in Proc. of International Conference on Application and Development of Computer Games in the 21st Century, pp.74-81, Hong Kong, China, 2001.

[4] Cavazza M., Charles F., Mead S. J., "Emergent Situations in Interactive Storytelling", in Proc. of SAC2002, ACM, pp.1080-1085, Madrid, Spain, 2002.

[5] Macedonia M. R., A Network Software Architecture for Large Scale Virtual Environments, Ph.D. Thesis, Naval Postgraduate School, Monterey, CA, 1995.

[6] Neyland D. L., Virtual Combat: A Guide to Distributed Interactive Simulation, Stackpole Books, Mechanicsburg, PA, 1997.

[7] United States Department of Defence. Defence Modeling and Simulation Office, 2002. Web Site: http://www.dmso.mil/

[8] Frécon E. and Stenius M., "DIVE: a Scaleable Network Architecture for Distributed Virtual Environments", Distributed Systems Engineering, vol.5, no. 3, pp.91-100, 1998.

[9] Funkhouser T. A., "RING: a Client-Server System for Multi-User Virtual Environments", in Proc. of the 1995 Symposium on Interactive 3D Graphics, pp.85-92, Monterey, CA, 1995.

[10] Normand V., "The COVEN project: Exploring Applicative, Technical, and Usage Dimensions of Collaborative Virtual Environments", Presence, vol.8, no.2, pp.218-236, 1999.

[11] Cai W., Xavier P., Turner S. J. and Lee B., "A Scalable Architecture for Supporting Interactive Games on the Internet", in Proc. of the 16th Workshop on Parallel and Distributed Simulation, pp.54-61, Washington, DC, 2002.

[12] Griwodz C., "State Replication for Multiplayer Games", in Proc. of NetGames2002, pp.29-35, Braunschweig, Germany, 2002.

[13] Openskies Network Architecture Project, 2002. Web Site: http://www.openskies.net

[14] Quake Forge Project, 2002. Web Site: http://www.quakeforge.org

[15] 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.

[16] Armagetron, 2003. Web Site: http://armagetron.sourceforge.net/

[17] Singhal S. and Zyda M., Networked Virtual Environments: Design and Implementation, Addison Wesley, 1999.

[18] Morse K. L., Bic L. and Dillencourt M., "Interest Management in Large-Scale Virtual Environments", Presence, vol.9, no. 1, pp.52-68, 2000.

[19] Deering S., "Host Extensions for IP Multicasting", Internet RFC 1112, 1989. ftp://ftp.isi.edu/in-notes/rfc1112.txt

[20] Jefferson D. R., "Virtual Time", ACM Transaction on Programming Languages and Systems, vol.7, no.3, pp.404-425, 1985.

[21] Steinman J. S., Bagrodia R. and Jefferson D., "Breathing Time Warp", in Proc. of the 1993 Workshop on Parallel and Distributed Simulation, pp.109-118, San Diego, CA, 1993.

[22] Singhal S. K., Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments. Ph.D. Thesis, Stanford University, Stanford, CA, 1996.

[23] Everquest, 2003. Web Site: http://www.everquest.com

[24] Ultima Online, 2003. Web Site: http://www.uo.com

[25] Gautier L. and Diot C., "Design and Evaluation of MiMaze, a Multi-player Game on the Internet", 1998. ftp://ftpsop.inria.fr/rodeo/diot/ieee-mms.ps.gz

[26] 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.

[27] Ferretti S. and Roccetti M., "A Novel Obsolescence-Based Approach to Event Delivery Synchronization in Multiplayer Games", in International Journal of Intelligent Games and Simulation, vol.3, no.1, pp.7-19, 2004.

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

[29] 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), pp.44-57, Asheville, NC, 1993.

[30] Défago X., Schiper A. and Urban P., "Totally Ordered Broadcast and Multicast Algorithms: a Comprehensive Study", Technical Report, DSC/2000/036, Swiss Federal Ecole Politechnique Fédérale de Lausanne, Switzerland, 2000.

[31] 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, pp.219-231, Darmstadt, Germany, 2004.

[32] 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.

[33] Birman K., "A Response to Cheriton and Skeen's Criticism of Causally and Totally Ordered Communication", ACM Operating System Review 28, no.1, pp.11-21, 1994.

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

[35] Ramanathan P. Shin K.G, Butler R. W., "Fault Tolerant Clock Synchronization in Distributed Systems", IEEE Computer, vol.23, no.10, pp.33-42, 1990.

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

[37] Farber J., "Network Game Traffic Modelling" in Proc. of NetGames2002, pp.53-57, Braunschweig, Germany, 2002.

[38] Wright S. and Tischer S., "Architectural Considerations in Online Game Services over DSL Networks", in Proc. IEEE International Conference on Communications - Multimedia Technologies and Services Symposium (ICC'04), IEEE Communications Society, Paris, France, 2004.

[39] Armitage G., "An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3", in Proc. ICON, Sydney, Australia, 2003.

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

[41] Clark D. D., Fang W., "Explicit Allocation of Best-Effort Packet Delivery Service", IEEE/ACM Transactions on Networking, vol.6, no.4, pp.362-373, 1998.