FILA in Gameland, A Holistic Approach to a Problem of Many Dimensions

Stefano Ferretti,Università di Bologna
Claudio E. Palazzi, Università di Bologna and University of California, Los Angeles,
Marco Roccetti, Università di Bologna, and
Giovanni Pau and Mario Gerla, University of California, Los Angeles

Multiplayer online games have now become popular with millions across the globe, capturing the attention of both researchers and practitioners. Unfortunately, online games still have to deal with the limitations imposed by some unresolved issues. Interactivity, consistency, fairness, and scalability are the major requirements that need to be addressed efficiently in order to provide an appealing product to a huge number of potential customers worldwide. To answer this demand, we describe a holistic approach that can exploit the semantics of the game to satisfy the aforementioned requirements. We provide extensive and comparative results that demonstrate how our scheme copes efficiently with an elevated level of game traffic.

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

General Terms: Algorithms, Design, Human Factors, Measurement, Performance

Additional Key Words and Phrases: Multiplayer online games, fairness, interactivity, consistency, scalability

This work was partially supported by the Italian Ministry for Research via the ICTP/E-Grid Initiative and the Interlink Initiative, the National Science Foundation under grants CNS-0435515/ANI-0221528, and the UC-CoRe grant MICRO 05-06 private sponsor STMicroelectronics.

Authors' addresses: S. Ferretti, C.E. Palazzi, and M. Roccetti, Dip. di Scienze dell' Informazione, Univ. di Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italia; C.E. Palazzi, G. Pau, and M. Gerla, Computer Science Dept., Univ. of California, Los Angeles, Boelter Hall, Los Angeles, CA, 90095; email: {sferrett, roccetti}@cs.unibo.it {cpalazzi, gpau, gerla}@cs.ucla.edu

1. INTRODUCTION

A massively multiplayer online game (MMOG) can be defined as a computer game able to support a multitude of players who interact with each other within the same virtual world, across the Internet, and regardless of geographical location.

MMOGs trace their roots to the late 1970s, when the popularity of the Multi-User Dungeon (MUD), a text-based role-playing adventure game spread from Essex University to the entire world [http://www.ludd.luth.se/mud/aber/mud-history.html]. Since then, MMOGs have evolved, now embodying a large class that includes several kinds of games (e.g., car racing, first-person shooter, adventure, role-play, and strategic).

Indeed, in the past few years, the popularity of MMOGs over the Internet has increased exponentially [http://www.mmogchart.com]. Nevertheless, Internet latencies have limited the use of networked games, largely as a result of slow-paced applications, utilizing the client-server paradigm for network communication, and engaging players only when limited in number and located in the same region where the server is positioned.

The task of providing a pleasant experience to players becomes hugely more challenging when trying to deploy a large scale and highly interactive online game. Indeed, the outstanding key factors in developing networked MMOGs are as follows:

- *Interactivity* (or *responsiveness*): the degree to which this occurs in the game event exchange.
- *Consistency*: a shared view of the game state among all the engaged players.
- *Network fairness*: a guarantee that all players have the same chance of winning, regardless of their subjective network conditions;
- *Scalability*: assurance that the number of simultaneous players as well as their geographical distribution will be properly scaled.

Regarding the last point, the interest of companies in online gaming is due to the huge revenues that may be generated by a very large number of customers. Besides, humans are social beings who enjoy the presence of others in most of their amusements (e.g., team sports, movies in theatres) and in challenging their skills against real adversaries.

So in order to ensure success, every time a new solution for one of the first three key factors is proposed, scalability should be ensured (and verified) as well.

To generalize, developers should follow a holistic approach when designing a new MMOG, and take the whole set of requirements into consideration. Addressing only one could produce an unexpected and undesired result and jeopardize the others.

With the goal of supporting interactivity and fairness while preserving consistency, we propose a novel scheme called *Fairness and Interactivity-Loss Avoidance* (FILA) that facilitates fairness by increasing the degree of interactivity [Ferretti et al.

2006; Palazzi et al. 2005]. By doing so, FILA contradicts the general belief that interactivity and fairness are incompatible requirements in MMOGs. We obtained this result by exploiting the semantics of the game and discarding obsolete events with a probability that depends on the current degree of interactivity as perceived by players.

We will show how our mechanism can be adapted to satisfy the scalability requirement. In particular, we will demonstrate that FILA is particularly able to cope with the intense game traffic generated by a multitude of players sharing the same virtual arena.

The remainder of the article is organized as follows: Section 2 discusses MMOG's fundamental requirements; Section 3 presents a scalable gaming architecture and compares it with other traditional architectures for online games; Section 4 provides the basics of the FILA approach; Section 5 describes and evaluates the algorithms; Section 6 explains the simulation environment for our experiments; Section 7 presents experimental results; and Section 8 concludes.

2. PROBLEM STATEMENT

The four key requirements listed in the Introduction cannot be considered independent of each other. If we aim to improve only one of them, the others may be affected negatively. Before evaluating a new algorithm for MMOGs, we have to understand in a deep way the tradeoffs among interactivity, consistency, fairness, and scalability.

In particular, interactivity means having small delays between the generation of a game event and the time at which all the players display that event. Indeed, if interactivity is to be preserved, every class of game must have a game-specific *game interactivity threshold* (GIT) that represents the maximum delay allowable before displaying a game event on the players' screens. The typical GIT for fast-paced games (e.g., racing vehicles, first-person shooter) is 150 to 200ms, but this value can be increased to seconds in slow paced-games (e.g., strategic, role-playing games) [Pantel and Wolf 2002; Borella 2000; Zander.and Armitage 2004; Fitzek et al. 2002; Sheldon et al. 2003].

If we call $t^{g(e)}$ the generation time of event e and $t_i^{v(e)}$ the visualization time of the same event at player i, then interactivity is preserved at i during the delivery of e when the following condition is satisfied:

$$t_i^{v(e)} - t^{g(e)} \leq GIT. \quad (1)$$

Both consistency and network fairness require the same and simultaneous game state view in all the nodes of the system. Hence the same class of techniques is used to achieve each of them (or both). Indeed, the easiest way to guarantee consistency and fairness is to make the game proceed through discrete locksteps [Steinman 1995]. In other words, the game evolves through steps and players have to wait their turn to act. Obviously, this scheme cannot be applied to interactive games.

In order to show game events on the screens of all the players simultaneously, a scheme based on the introduction of artificial delays was devised recently [Zander et al. 2005; Gautier and Diot 1998; Pantel and Wolf 2002; Mauve et al. 2004; Kim et al. 2005]. This kind of solution is usually referred to as the *local lag* algorithm. With local lag, advances in the game are delayed for a sufficient amount of time to guarantee that all the players in the system process and perceive the game events at the same time and in the same order.

Indeed, since the time to generate each event is unique and considering the set of players, C, we can say that we have *event-related fairness* [Ferretti et al. 2006] for an event e if the following condition is satisfied, i.e., simply, that if there is a unique $t^{v(e)}$ value equal for all the players, then

$$t_i^{v(e)} = t^{v(e)}, \qquad \forall i \in C. \quad (2)$$

Since a single game event may experience different *overall delays* (OD) in its path from the source to all the diverse players, different amounts of artificial delay δ should be added by means of the local lag algorithm to simultaneously display the same event e on all the players' screens, to satisfy the following condition:

$$t^{g(e)} + OD_i(e) + \delta_i(e) = t^{v(e)} \quad \forall i \in C. \quad (3)$$

A possible value typically chosen for $t^{v(e)}$ is represented by the highest OD experienced in transmitting events among the nodes. When the highest OD is greater than GIT, however, fairness is preserved at the cost of jeopardizing interactivity for all the players. Conversely, if we use GIT as an upper bound of $t^{v(e)}$, then we can guarantee interactivity but not fairness.

Consequently, in order to maximize the possibility of obtaining both interactivity and fairness, $t^{v(e)}$ should be set as

$$t^{v(e)} = t^{g(e)} + GIT. \quad (4)$$

Yet the $OD_i(e)$ experienced by event e when it finally reaches client i is made up of several delay components such as physical latency, queuing time on routers along the path and on game servers, and processing time.

Therefore, even when network latency allows having values of OD, and hence also of $t^{v(e)}$ lower than GIT, a large number of players generating a huge amount of traffic may raise the values of the other two delay components, again leading to the crossroad between fairness and interactivity.

To conclude, the efficiency and applicability of popular, delayed-based algorithms like local lag depend strongly on the network conditions and on the degree of interactivity required by the game. Yet guaranteeing both interactivity and full fairness through local lag can sometimes only be achieved at the cost of limiting the scalability of the game by restricting the number of contemporaneous participants and the geographical extent of the target player market.

It has become evident that to find the most efficient tradeoffs among interactivity, consistency, fairness, and scalability, MMOGs require architectural solutions and algorithms.

3. ARCHITECTURAL SOLUTIONS

Typically, network architectures that support MMOGs can be distinguished on three main categories: the centralized client-server, fully distributed architecture, and the mirrored game server. The centralized client-server architecture represents the simplest solution for authentication procedures, security issues, and consistency maintenance [http://www.quakeforge.org]. Moreover, assuming there are N simultaneous players, the messages generated are on the order of $O(N)$. On the other hand, the unique bottleneck limits the efficiency and scalability of this solution.

Fully distributed architectures (peer-to-peer) spread the traffic load among many nodes, resulting in a more scalable and failure-resilient system [Gautier and Diot 1998]. However, identical copies of the current game state must be stored at each node; and this requires a complex coordination scheme among peers able to guarantee the coherence of all game state views. Moreover, with a fully distributed architecture, multicast should be employed to reduce the bandwidth requirements; but multicast technology is neither generally available nor mature enough for the specific application we are considering here (the messages exchanged could rise to the order of $O(N^2)$). Finally, authentication, cheating, and general consensus among all the peers are easier to address when a centralized architecture is employed.

Mirrored game server architectures represent a hybrid solution that efficiently embraces all the positive aspects of both centralized client-servers and fully distributed architectures [Cronin et al. 2004]. Based on this approach, game state servers (GSSs) are interconnected in a peer-to-peer fashion over the Internet and contain replicas of the same game state view. Players communicate with their closest GSS through the client-server paradigm. Each GSS gathers all the game events of its engaged players, updates the game state, and forwards it regularly to all its players and GSS peers.

The presence of multiple high performance GSSs helps to distribute the traffic over the system and reduce the processing burden at each node. Moreover, having each player connected to a close by GSS reduces the impact of the player-dependent access technology (e.g., dial-up, cable, DSL) on the total delay [Jehaes et al. 2003]. In this case, in fact, communication among players was mainly deployed over links physically connecting GSSs, which can exploit the fastest available technology (e.g., optical fibers) to reduce latency. As a result, this architecture helps in finding better solutions for the tradeoffs among interactivity, consistency, fairness, and scalability.

Absence of a single point of failure, networking complexity maintained at server side, and the possibility of implementing authentication procedures are other advantages in employing a mirrored game server architecture. Even if synchronization is still required to ensure the global consistency of the game state held by the various servers, this requirement is made easier than in fully distributed architectures, thanks to the lower number of nodes involved. Assuming N players and M GSSs, for example, the game messages generated amount to $O(N+M)$, which is again $O(N)$, unless considering the unlikely case of more servers than players.

All these reasons suggest that the mirrored game server architecture is the most appropriate to efficiently manage large-scale distributed games, as it embodies the advantages of both client-server and fully distributed paradigms.

4. FILA OVER A MIRRORED SERVER ARCHITECTURE

FILA can be thought of as comprised of two complementary subcomponents. The first one, enforced among GSSs, speeds up the delivery of *fresh* game events by dropping events that have become *obsolete* since the arrival of the more recent ones. Interested readers may refer to Ferretti and Roccetti [2004] for a deeper discussion of the notion of obsolescence and a way to include it in game events. The second component takes advantage of the reduced transmission time to magnify the efficiency of a local lag-type algorithm to ensure fairness. FILA utilizes the work of Zander et al. [2005] to determine the display time of a game event; and thus ensures fairness without compromising interactivity.

To calculate the appropriate δ in Eq. (3), the OD should be determined for each player. For this reason, game events are marked with a generation timestamp at their creation and then sent to the destination. Obviously, a global concept of time has to be maintained in the system, which can be achieved through a variety of solutions that enable the synchronization of GSSs' physical clocks [Mills 1991; Ramanathan et al. 1990], or by employing new synchronization devices such as GPS. Hence GSSs can monitor the ODs of their engaged players and make them available for the FILA algorithm.

The first component of FILA is inspired by the so-called active queuing management techniques [Floyd and Jacobson 1993]. It drops queued obsolete game events with probability p_d when the average OD (avgOD) value increases, putting the interactivity of the system at risk. The probability of discarding, i.e., p_d , is directly proportional to avgOD and depends on a constant *Pmax,* as described in the work of Palazzi et al. [2004] and Palazzi et al. [2006]. Instead, the value for avgOD at iteration n is computed through the low-pass filter shown below, where w is a parameter that determines how closely the average follows the *sample* trend:

$$avgOD_n = avgOD_{n-1} + w \times (sample_n - avgOD_{n-1}). \quad (5)$$

Moreover, with FILA, all game events are regularly processed and forwarded when avgOD is smaller than the threshold alert, called *tmin*. As soon as avgOD exceeds *tmin*, the GSS drops obsolete events with probability p_d, and does not process or forward them. Finally, if avgOD exceeds the subsequent *tmax* (>*tmin*) threshold, then p_d is set equal to 1, and all obsolete events waiting for processing are discarded.

This stabilization mechanism succeeds in reducing $OD_i(e)$ because the time spent in queue by an event is decreased due to a gain in processing time, which is the result of the preceding obsolete events being dropped without being processed or forwarded. Moreover, since only obsolete events are discarded, FILA maintains full consistency as the game evolves [Ferretti et al. 2006; Palazzi et al. 2004].

To explain FILA in more detail, see Figure 1, which provides the graphical definitions for some of the terms in our work, namely: *OD*, *ND* (network delay), and *LHD* (last hop delay).

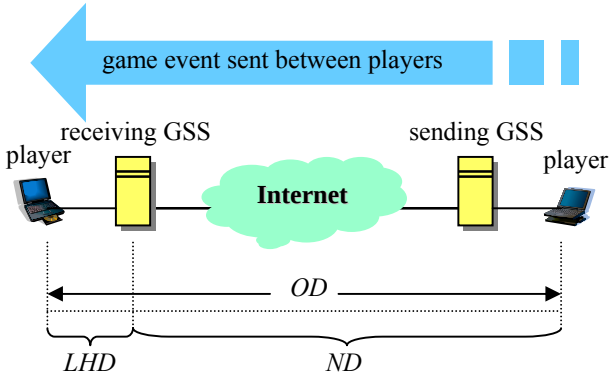


Fig.1. Delay definitions**.**

First, note that FILA performs its operations on the receiving GSS. This choice makes it easier to maintain control of the game platform. However, for each event *e*, the GSS can compute ND*(e)* but not LHD*(e)*. But it is necessary to estimate LHD*(e)* in order to compute OD and use it in the algorithm. For this reason, each GSS continuously monitors latencies for each of its players and maintains a variable called λ_{GSS}. The value of this variable represents the maximum among the latencies from the GSS to each of its connected clients (this set of clients is called *C_GSS*), as follows:

$$\lambda_{GSS} = \max_{i \in C\_GSS} \{ LHD_i \}. \quad (6)$$

However, we cannot let an irremediably delay-affected client impact our scheme's calculations too much. In FILA, utilizing an excessively high λ_{GSS} generated by a player very far from the GSS results in a very high sample (and avgOD) value with respect to GIT. Consequently, the aggressiveness of FILA's discarding function would increase, as perceived by *all* the players, with no positive results. (The "unlucky" player would still not be able to get game events with delays below the interactivity threshold.)

Hence we need to consider a *delay upper bound* (DUB), which is used by FILA to limit the impact of "unlucky" players on the algorithm. Toward this goal, we provide the computation of a formula for a fundamental parameter utilized by FILA to handle the impact of LHD*(e)* on the algorithm:

$$\sigma = min\{ \lambda_{GSS}, DUB \}. \quad (7)$$

How the parameter σ varies will depend on which version of our scheme, as detailed in Section 5, is used.

Instead, to determine *DUB*, we rely on a heuristic that computes its value dynamically, based on the general condition of the network during the game; the formula follows:

$$DUB = GIT - \max\{ND\}, \quad (8)$$

Where max{ND} represents the largest ND over all the connections in the entire network.

Obviously, each GSS has to communicate the largest ND at that server back to all the other peers. This allows global knowledge of the worst *ND* value at each GSS. Finally, the highest among the maximum *NDs* can be univocally determined by each GSS and used to determine the global *DUB*.

The second part of FILA is simply in charge of equalizing the delay differences among players via a local lag-type scheme that appropriately computes the δ value shown in Eq.(3) so as to satisfy Eq. (4) whenever possible.

5. IS FILA A GOOD SOLUTION?

We will now empirically demonstrate how the combination of the two subcomponents of FILA is effective in ensuring fairness and interactivity while allowing a scalable number of contemporaneous players. To this aim, four different schemes are taken into account: regular local lag (LL) and three versions of FILA (i.e., FILA-A, FILA-B, and FILA-C).

The first scheme, LL, embodies the traditional local lag scheme with no discarding mechanism for obsolete events. Even in this case, however, as for all the other schemes above, the algorithm is not allowed to introduce artificial delays if this results in jeopardizing interactivity (i.e., $t^{v(e)}$ cannot be set greater than $t^{g(e)} + GIT$).

FILA-A is the simplest among the three schemes. With this algorithm no *avgOD* is maintained and no p_d is calculated. To cohere with the basics of the algorithm anticipated in Section 4, we could say that, in FILA-A, *tmin* and *tmax* are both set equal to *GIT* (and both *w* and *Pmax* are always equal to 1). Moreover, at each iteration of Eq. (5), *sample* is set equal to the current *ND(e)*.

When a GSS receives a game event *e* from a player connected to one of its GSS peers, *e* still has to travel from that GSS to the final players. For this reason, our scheme takes into account the various $LHD_i(e)$ by reducing the threshold used by the algorithm. Therefore, with FILA-A, each GSS performs normal delivery and local lag operations for each game event *e* until the following condition holds:

$$ND(e) \le GIT - \sigma. \quad (9)$$

When Eq.(9) fails, *all* obsolete events in queue are discarded instead, and they are neither processed nor forwarded.

In FILA-B, the estimation of the impact of $LHD_i(e)$ is taken into account by diminishing one of the utilized thresholds instead, In particular, we set $tmax = GIT - \sigma$ and $tmin < tmax$. Even with this algorithm, at each iteration of Eq.(5), *sample* is set equal to the current *ND(e)*.

Finally, FILA-C takes the estimation of the $LHD_i(e)$ values directly into account when generating *sample*. Hence we have $tmax = GIT$ ($tmin < tmax$), and *sample* is determined by the arrival of a new event *e* by employing σ as follows:

$$sample(e) = ND(e) + \sigma. \quad (10)$$

Even when they are similar, the three versions of FILA still differ in substantial respects. In essence, FILA-A is an aggressive, yet slow, approach, in which all the queued obsolete events are discarded only after interactivity and fairness have already been lost. FILA-B and FILA-C try to avoid the loss of these two properties by preemptively discarding some obsolete game events when the trend in delay increases over the threshold alert, *tmin*. Hence we may expect to witness smaller dropping rates with FILA-B and FILA-C, as they need to drop all queued obsolete events less frequently than FILA-A.

This is a desirable property. In fact, even if obsolete events can be sacrificed, they are still part of the game's visual progression. Dropping too many obsolete events could result in jerky rendering caused by imprecise interpolations of the missing actions. As a result, annoying artifacts in the game's evolution may be generated.

When we contrast FILA-B with FILA-C, we see that the former includes σ in the *tmax* threshold, while the latter utilizes it to compute the *sample* value. The impact of σ on FILA-C should be smoothed by the low-pass filter (Eq.(5)). However, when *sample* is steadily augmented by σ, then even *avgOD* results are higher, since *sample* is used to compute *avgO*. Recall that the discarding probability p_d is directly proportional to *avgOD*, so we can see how the use of Eq.(10) results in higher p_d values. Hence we expect to find FILA-C more aggressive than FILA-B (with a larger number of dropped events).

6. EXPERIMENTAL ARCHITECTURE

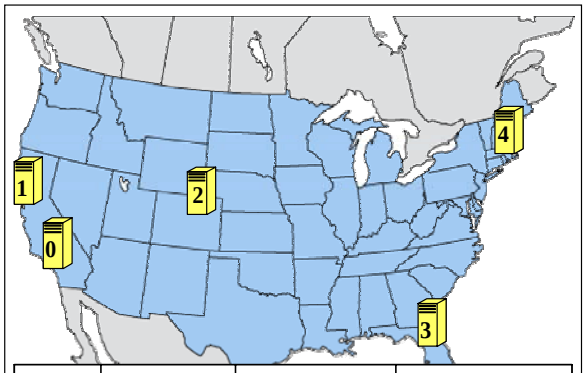
It is well known that MMOG service providers should position their game servers in such a way that their target player market is located within a circle with a 150 to 180ms latency diameter [Armitage 2003]. Following this rule, we have simulated the deployment of five GSS across North America, positioned in optimal locations with communication latencies that provide adequate support for a highly interactive MMOG in both the distances between the nodes and the number of customers that can be served. Thus, clients are supposed to be distributed all over the North America, connecting to a GSS through various access technologies with different access delays.

For the sake of deeper understanding, we focus our attention on the event-receiving aspect of a single GSS (GSS_0), supposing that the other GSSs are sending events to it without any loss of generality.

Following the literature [Park and Willinger.2000], *ND* values among GSSs were generated based on a lognormal distribution whose approximate average was taken from repeated runs of the *ping* application. Further, game events from players connected to the sending GSSs (i.e., GSS_1 to GSS_4) and traveling towards GSS_0 experienced latencies with an average reported in Figure 2 and a standard deviation of 10ms.

Further, several scenarios were considered in which the values of $\max_{i \in C\_GSS} \{LHD_i\}$ were chosen for each GSS within the following set [25ms, 50ms, 75ms, 100ms, 125ms, 150ms]. This choice simply derives from the consideration that clients should be located within a circle with a maximum latency diameter of 150ms. We assume that clients are connected to each GSS, are engaged in a fast-paced game, and generate a new action every 300ms, on average. Online game literature suggested that we utilize 200B as the size of the average game event [Farber 2002].

In MMOGs, not all queued or old game events become obsolete while the game evolves. Their content may be significant and drastically alter the evolution of the game state [Ferretti and Roccetti 2004]. Thus, not all queued game events can be discarded at a given GSS. The probability that an incoming event supersedes preceding ones, making them obsolete, is set to 90%. This represents a realistic scenario for a vast plethora of possible games (e.g., adventure, strategic, car race, flight simulator), where most of the events are just independent movements. In other words, critical game events that cannot become obsolete have to be considered only sporadically, e.g., during collisions or shots, and may represent even fewer than 10% of the whole set of game events. A coherent and more detailed numerical demonstration for this claim can be found in the work of Palazzi et al. [2006].



GSS	Avg latency	City	Road distance
0	0 ms	Los Angeles	0 km
1	20 ms	San Francisco	602 km
2	35 ms	Denver	1650 km
3	50 ms	Orlando	3870 km
4	65 ms	Boston	4896 km

Fig. 2. Game server deployment.

As to the choice of critical parameters in FILA-B and FILA-C algorithms, we have set w = 1/8 and $Pmax$ = 0.2 for all the simulations. The *tmin* and *tmax* variables, instead, change with the *GIT,* as explained in Section 5 and, in particular, *tmin* is always 100ms smaller than *tmax*.

Aimed at testing the scalability of our system, we analyzed different configurations with a varying number of players. In particular, we considered scenarios where each sending GSS was gathering from its engaged players and forwarding to GSS_0 an amount of game events generated following a lognormal distribution [Farber 2002; Borella 2000] with an average of, respectively, 30ms, 20ms, and 10ms. We call this parameter *average inter-departing time* (AIDT). Considering an average inter-generation time of 300ms between two subsequent game actions generated by the same player, the above AIDT values represent from 50 to 150 contemporaneous players.

We ran several sets of experiments varying the *GIT* from 150ms to 300ms in order to take into account different kinds of games [Pantel and Wolf 2002; Borella 2000; Zander and Armitage 2004;

Fitzek et al. 2002]. Each experiment was identically replicated to obtain a significant comparison among the outcomes of the three versions of FILA, plus the regular local lag algorithm. Zander et al. [2005] shows that lower latencies result in statistically higher mean kill rates, and thus in unfairness. Consequently, we have chosen to evaluate as a fairness parameter the percentage of events that were delivered by our monitored server, GSS_0, to *all* of its players. These game events had a visualization time suggested by Eq.(4) thus achieving both fairness and interactivity in the delivery to all players.

7. RESULTS

7.1 Interactivity and Fairness

The charts in Figure 3 show the percentage of game events that GSS_0 was able to deliver to all of its engaged players, satisfying both the interactivity and fairness requirements (consistency was maintained as well). Four graphs are presented reporting results from employing, respectively, (a) 150ms, (b) 200ms, (c) 250ms, and (d) 300ms as *GIT*. The AIDT value is equal to 30ms at each sending GSS for all the four sets of experiments (a), (b), (c), and (d). Each set is comprised of six experiments. Each experiment consists in the transmission of about 4000 game events that experienced, in the worst case, a maximal overall latency whose value is reported on the x-axis of each chart.

The outcomes of the three FILA versions are so similar that the three corresponding lines overlap in most of the configurations. In a few cases, however, FILA-C provides the best performance algorithm. At the same time, significantly higher percentages are ensured by each version of FILA with respect to LL over the various end-to-end latencies and *GIT* values.

Since larger local lags can be utilized, it is obvious that a higher GIT will improve the efficacy of all the schemes. However, LL experiences a premature decrease in performance when the end-to-end latency increases, even if it is still far from the GIT. Instead, FILA ensures a good fairness level for a larger set of end-to-end latencies. However, in configurations where the end-to-end latency is close to, or surpasses, GIT (end-to-end latency ≥ 140ms in Figure 3a) and end-to-end latency ≥ 190ms in Figure 3b), none of the schemes are able to overwhelm network conditions, and so get poor results. Even in this case, however, FILA behaves better than LL.

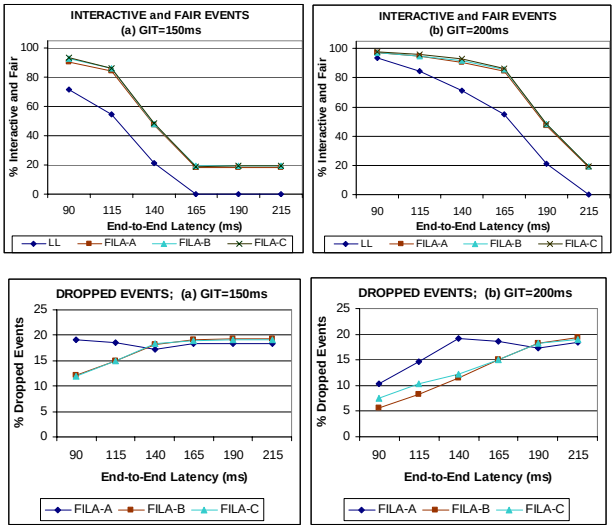


Fig. 3 (a, b, c, d). Improvement in interactivity and fairness with AIDT equal to 30ms.

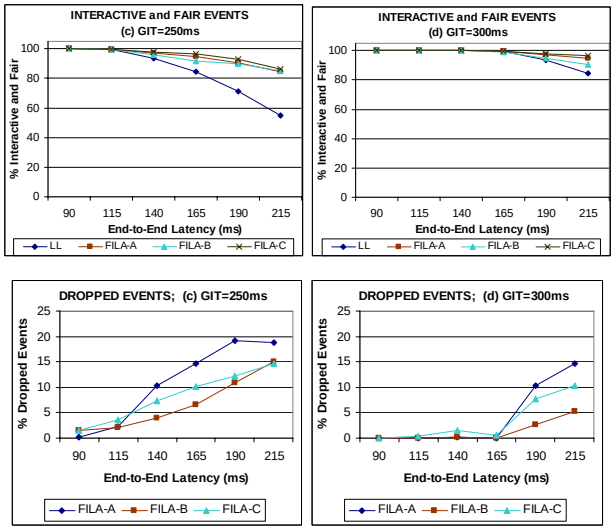


Fig. 4 (a, b, c, d). Obsolete events dropped by FILA with AIDT equal to 30ms.

7.2 Dropping Game Events

FILA rewards its better outcomes by dropping some obsolete events. Specifically, Figure 4 shows the percentage of game events that have been discarded by the various versions of FILA. However, in all these cases, less than 20% of the game events were dropped. This represents an acceptable value, since these events are exclusively obsolete ones.

FILA-A discards obsolete events only when interactivity and fairness have already been lost. Moreover, at that point, it discards all the obsolete events in the queue. Therefore, this version of FILA has the highest percentage of drops of the three in most configurations. And as expected, FILA-C behaves more aggressively than FILA-B, and generally discards a higher number of obsolete events.

The results are particularly meaningful if we focus on those scenarios where network latency is not irremediably high with respect to *GIT*. Considering the configurations where the maximal overall latency is lower than *GIT* by 35ms or more, we find that each FILA version always guarantees at least 84% of interactively and fairly delivered game events with less than 15% dropped events.

7.3 Scalability Issues

In order to test scalability, we decreased AIDT to generate scenarios with a higher level of game traffic in the network. In particular, Figures 5 and. 7 refer to the case with 20ms of AIDT, while Figures 6 and 8 correspond to the case where AIDT is equal to 10ms. Again, in each figure we show the outcomes for four different *GIT* values.

As expected, the higher the game traffic, the lower the interactivity and fairness degree provided by LL (see Figures 5 and 6). In contrast, not only does FILA manage higher traffic, but its performance actually improves when AIDT decreases.

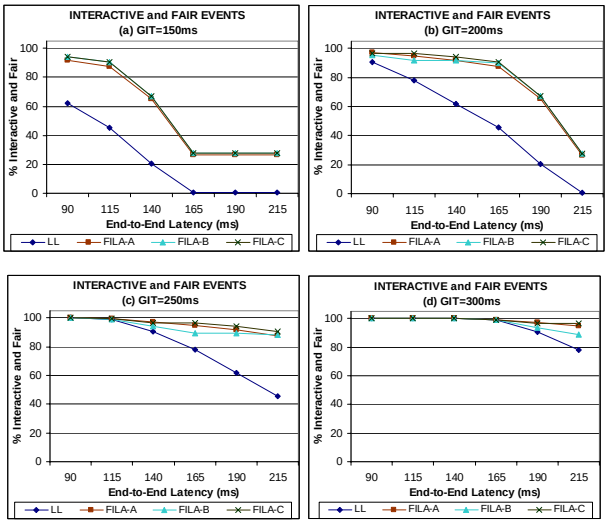


Fig. 5 (a, b, c, d). Improvements in interactivity and fairness with AIDT equal to 20ms.

This surprising result has a simple explanation. Higher rates in game event transmissions result in generating larger queues of packets at GSSs that have not yet been processed. This amounts to a crucial problem for LL, since the queuing time at the router increases, putting the performance of the system at risk without any countermeasures. With FILA, however, a larger queue of game events at a GSS also represents a resource. In fact, obsolete game events in queue can be discarded, thus reducing the time that a subsequent event *e* will experience in its travel towards various clients.

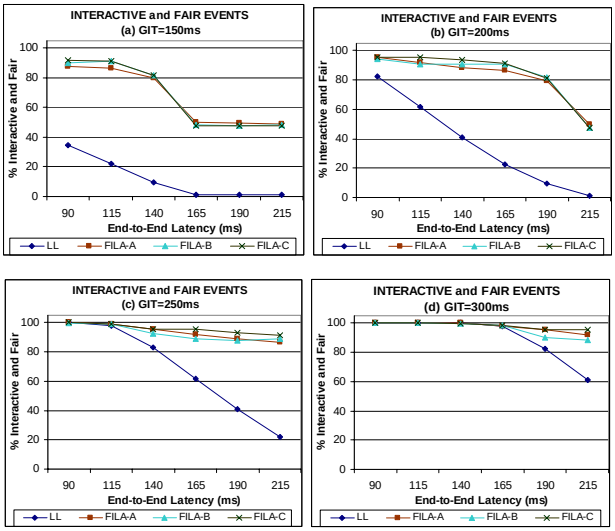


Fig. 6 (a, b, c, d). Improvements in interactivity and fairness with AIDT equal to 10ms.

As proof for our rationale, we can see in Figures 7 and 8 that the number of obsolete game events dropped by the three versions of FILA increases when AIDT decreases. This is caused by higher *avgOD* values due to increased traffic, but also by the presence of more game events in queue that FILA can exploit in order to drop the obsolete ones.

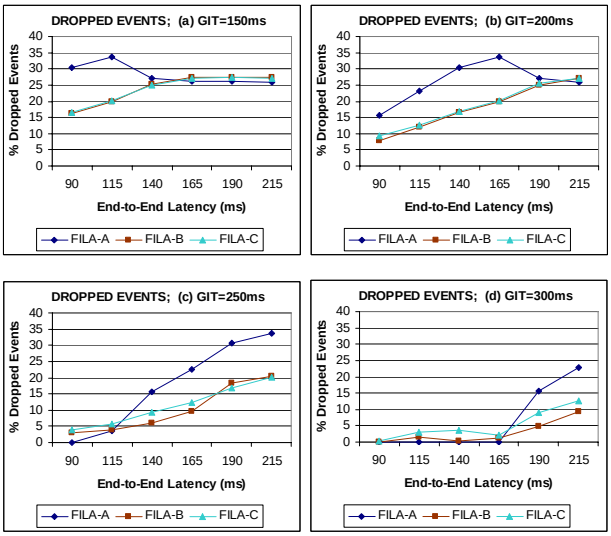


Fig. 7 (a, b, c, d). Obsolete events dropped by FILA with AIDT equal to 20ms.

Finally, and similarly to the scenario with 30ms of AIDT, even with AIDT equal to 20ms and 10ms, FILA-A generally discards more events than the other two FILA algorithms. The only case where this claim is contradicted is when the latency is much lower than the *GIT* being considered. In such case, due to the high volume of traffic in the game network, the *avgOD* can steadily surpass the *tmin* thresholds of FILA-B and FILA-C, causing them to execute some preemptive drops. At the same time, the large divergence between latency and *GIT* makes the situation where

avgOD exceeds *GIT* and activates the drop function for FILA-A a very rare event. However, even with these configurations, the percentage of discarded game events still remains very small for all the FILA configurations.

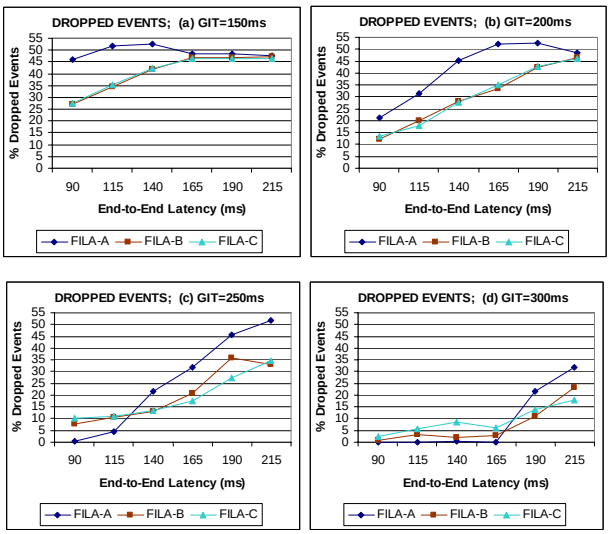


Fig. 8 (a, b, c, d). Obsolete events dropped by FILA with AIDT equal to 10ms.

8. CONCLUSION

Interactivity, consistency, fairness, and scalability are fundamental requirements that cannot be ignored when designing a new online game. Aside from the quality of the game, the ability to holistically sustain these requirements is crucial to make MMOG a success. Unfortunately, as highlighted in this work, these requirements involve tradeoffs that make the contemporaneous achievement of all of them a hard task.

To this aim, we designed an event delivery scheme, FILA, among replicated game servers that discards obsolete events to ensure interactivity and fairness, while preserving consistency. Since only obsolete events are discarded, there is no risk that different percentages in dropped obsolete events at different servers could result in unfairness [Zander and Armitage 2004].

We have provided extensive experimental results that demonstrate the efficacy of FILA with various client-to-client latency ranges. We have also contrasted different version of FILA, showing the advantages and disadvantages of each of them. Finally, we have presented a scalability evaluation of our schemes by increasing the game traffic level in the system.

REFERENCES

ARMITAGE, G. 2003. An experimental estimation of latency sensitivity in Multiplayer Quake 3. In *Proceedings of the* 11th *IEEE International Conference on Networks* (ICON 2003, Sydney, Australia).137-141.

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

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

EARLY MUD HISTORY. http://www.ludd.luth.se/mud/aber/mud-history.html.

FARBER, J. 2002. Network game traffic modelling. In *Proceedings of the* 1st *Workshop on Network and System Support for Games* (NetGames2002, Braunschweig, Germany). ACM, New York. 53-57.

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

FERRETTI, S., PALAZZI, C. E., ROCCETTI, M., GERLA ,M., AND PAU, G. 2006. *Buscar el Levante por el Poniente*: In search of fairness through interactivity in massively multiplayer online games. In *Proceedings of the* 2nd *IEEE International Workshop on Networking Issues in Multimedia Entertainment* (NIME'06, CCNC, Las Vegas, NV, Jan.).

FITZEK, F., SCHULTE, G., AND REISSLEIN, M. 2002. System architecture for billing of multiplayer games in a wireless environment using GSM/UMTS and WLAN services. In *Proceedings of the* 1st *Workshop on Network and System Support for Games* (NetGames2002, Braunschweig, Germany). ACM, New York, 58-64.

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

GAUTIER, L. AND DIOT, C. 1998. Design and evaluation of MiMaze, a multiplayer game on the Internet. *In Proceedings of IEEE Multimedia* (ICMCS'98, Austin, TX*)*. 233-236.

JEHAES, T., DE VLEESCHAUWER, D., COPPENS, T., VAN DOORSELAER, B., DECKERS, E., NAUDTS, W., SPRUYT, K., AND SMETS, R. 2003. Access network delay in networked games. In *Proceedings of the* 2nd *Workshop on Network and System Support for Games (*NetGames 2003, Redwood City, CA). ACM, New York, 63-71.

KIM, S., KUESTER, F., AND KIM, K. H. 2005. A global timestamp-Bbased Aapproach to Eenhanced data consistency and fairness in collaborative virtual environments. *Multimedia Syst. 10,* 3 (2005), 220-229.

MAUVE, M., VOGEL, J., HILT, V., AND EFFELSBERG, W. 2004. Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Trans. Multimedia 6,* 1 (2004), 47-57.

MILLS, D. L. 1991. Internet time synchronization: The network time protocol. *IEEE Trans. Commun. 39,* 10 (1991), 1482-1493.

MMOGCHART.COM. http://www.mmogchart.com.

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).157-165.

PALAZZI, C. E., FERRETTI, S., CACCIAGUERRA, S., AND ROCCETTI, M. 2005. A RIO-like technique for interactivity loss avoidance in fast-paced multiplayer online games. *ACM J. Comput. Entertainment 3,* 2 (2005), 1-11.

PALAZZI, C. E., FERRETTI, S., CACCIAGUERRA, S., AND ROCCETTI, M. Interactivity-loss avoidance in event delivery synchronization for mirrored game architectures. *IEEE Trans. Multimedia 8*, 4, (2006), 874-879.

PANTEL, L. AND WOLF, L. C. 2002. On the impact of delay on real-time multiplayer games. *In Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (Miami, FL). 23-29.

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

QUAKE FORGE PROJECT. http://www.quakeforge.org.

RAMANATHAN, P., SHIN, K. G., AND BUTLER, R. W. 1990. Fault tolerant clock synchronization in distributed systems. *IEEE Computer 23,* 10 (1990), 33-42.

SHELDON, N., GIRARD, E., BORG, S., CLAYPOOL, M., AND AGU, E. 2003. The effect of latency on user performance in Warcraft III. In *Proceedings of the 2nd Workshop on Network and System Support for Games* (NetGames 2003, Redwood City, CA). ACM, New York.3-14.

STEINMAN, J. S. 1995. Scalable parallel and distributed military simulations using the SPEEDES framework. In *Proceedings of the 2nd Electronic Simulation Conference* (ELECSIM95).

ZANDER, S., LEEDER, I., AND ARMITAGE, G. 2005. Achieving fairness in multiplayer network games through automated latency balancing. In *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (ACE 2005, Valencia, Spain).

ZANDER, S. AND ARMITAGE, G. 2004. Empirically measuring the QoS sensitivity of interactive online game players. In *Proceedings of Australian Telecommunications Networks & Applications Conference* (ATNAC2004, Sydney, Australia). 511-518.