FILA, a Holistic Approach to Massive Online Gaming: Algorithm Comparison and Performance Analysis

Stefano Ferretti[1], Claudio E. Palazzi[1,2], Marco Roccetti[1] ,Giovanni Pau[2], Mario Gerla[2]

[1]Dipartimento di Scienze dell'Informazione, Università di Bologna,
Mura Anteo Zamboni 7, 40127 Bologna, Italia
Phone: +39-051-209-4503

[2]Computer Science Department, University of California Los Angeles,
Boelter Hall, Los Angeles CA, 90095, USA
Phone: +1-310-825-4367

{sferrett, roccetti}@cs.unibo.it
{cpalazzi, gpau, gerla}@cs.ucla.edu

ABSTRACT

The popularity of multiplayer online games has nowadays reached millions across the globe capturing the attention of both researchers and practitioners. Unfortunately, this kind of applications has still to deal with the limitations imposed by some unresolved issues. Interactivity, consistency, fairness, and scalability are the major requirements that need to be efficiently addressed in order to provide an appealing product to a huge number of potential customers all over the world. Answering to this demand, we describe a holistic approach able to exploit the semantics of the game to satisfy the aforementioned requirements. We provide extensive and comparative results that demonstrate how our scheme efficiently copes with an elevated game traffic level.

Categories and Subject Descriptors

K.8.0 [Computing Milieux]: Personal Computing — Games

General Terms

Algorithms, Measurement, Performance, Design, Human Factors.

Keywords

Multiplayer Online Games, Fairness, Interactivity, Consistency, Scalability.

1. INTRODUCTION

A Massively Multiplayer Online Game (MMOG) can be defined as a computer game able to support a multitude of players which interact each other within the same virtual world, across the Internet, and regardless of their geographical locations.

MMOGs trace their roots to the late '70s, when the popularity of the Multi-User Dungeon (MUD), a text based role playing adventure, escalated from the Essex University to all-over the world [1]. Since then, MMOGs have evolved embodying now a large class that includes several different kinds of games (e.g., car racing, first person shooter, adventure, role play game, strategic).

Indeed, in the past few years, the popularity of MMOGs has exponentially increased over the Internet [2]. Nevertheless, Internet latencies have damaged the use of networked games which significantly result as slow-paced applications, utilizing the client-server paradigm for the network communication, and efficiently 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 customers becomes apodictically more challenging when trying to deploy a large scale and highly interactive online game. Indeed, the prominent networking key factors in developing MMOGs are represented by:

- *Interactivity* (or *responsiveness*) degree in the game event exchange;
- *Consistency* of the game state view among all the engaged players;
- *Network Fairness* ensured in terms of guaranteeing the same possibility of victory to all the players regardless of their subjective network conditions;
- *Scalability* in the number of contemporary players as well as in their geographical distribution.

Regarding the last point, it should be noticed that the interest of companies in online gaming emerges from the huge revenues that may be generated by a very elevated number of customers. Besides, humans are social beings which enjoy the presence of others in most of their amusement activities (i.e., team sports, movies in theatres) and the competition in challenging their skills against real adversaries.

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

Generalizing this concept, developers should follow a holistic approach when designing a new MMOG, considering the whole set of requirements and aiming at the intersection of their solutions. Addressing only one among the aforementioned requirements, in fact, could produce the unexpected and undesired result of jeopardizing the others.

Aimed at supporting interactivity and fairness whilst preserving consistency, we have recently proposed *Fairness and Interactivity-Loss Avoidance* (FILA): a novel scheme able to facilitate fairness by aiming at increasing the interactivity degree [9, 26]. By doing this, FILA contradicts the general belief that interactivity and fairness would be 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 interactivity degree as perceived by players.

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

The remainder of the paper is organized as follows. Section 2 discusses MMOG 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 the algorithms evaluated in this work. Section 6 explains the simulative environment we have used for our experiments. Section 7 presents the experimental results. Finally, Section 8 concludes this work.

2. PROBLEM STATEMENT

The four key requirements listed in Section 1 cannot be considered as independent from each other. Aiming at improving only one of them, the others may be found negatively affected. Before evaluating any new algorithm for MMOGs, we have hence to deeply understand the tradeoff relationship that exists among interactivity, consistency, fairness, and scalability.

In particular, interactivity refers to having small delays between the generation of a game event and the time at which all the clients display that event. Indeed, every class of game is featured by a game specific *Game Interactivity Threshold* (*GIT*) that represents the maximum delay endurable before displaying a game event on players' screens if one wishes to preserve interactivity. The typical *GIT* for fast paced games (i.e., vehicle racing, first person shooter) amounts to 150-200ms but this value can be increased to even seconds in case of slow paced games (i.e., strategic, role play game) [6, 17-20].

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:

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

Both consistency and network fairness regards having the same and simultaneous game state view in all the nodes of the system. Therefore, 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 proceeding through discrete locksteps [14]. In other words, the game evolves through step and players have to wait their turn before making an action. Obviously, this scheme cannot be applied to interactive games.

In particular, a recent scheme has been devised that is based on the introduction of artificial delays in order to simultaneously visualize game events on all the players' screens [4-8]. This kind of solution is usually referred to as *local lag* algorithm. With local lag, game advancements are delayed for a sufficient amount of time in order to guarantee that all the clients in the system process and perceive the generated game events at the same time and in the same order.

Indeed, since the generation time of each event is unique and considering C, the set of players, we can say that we have *event-related fairness* [9] for an event e if the following condition is satisfied. Simply stated, if there is a unique $t^{v(e)}$ value equal for all the players:

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

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

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

A possible value typically chosen for $t^{v(e)}$ is represented by the highest *OD* experienced in transmitting events amongst 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 to $t^{v(e)}$, then we can guarantee interactivity but not fairness.

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

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

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

Therefore, even when the network latency would allow 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 value of the other two delay components, thus leading us again to the crossroad between fairness and interactivity.

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

It hence becomes evident as MMOGs require the use of architectural solutions and algorithms to find the most efficient tradeoff among interactivity, consistency, fairness and scalability.

3. ARCHITECTURAL SOLUTIONS

Typically, network architectures supporting MMOGs can be distinguished based on three main categories: centralized client-server, fully distributed, and mirrored game server. The centralized client-server architecture represents the simplest solution for authentication procedures, security issues, and consistency maintenance [10]. Moreover, assuming to have N simultaneous players, the generated messages are in the order of $O(N)$. On the other hand, the unique bottleneck limits the efficiency and scalability of this solution.

Fully distributed architectures (as peer-to-peer) spread the traffic load among many nodes and result in more scalable and failure-resilient system [5]. However, identical copies of the current game state need to be stored at each node; this requires some complex coordination scheme among peers able to guarantee the coherence of all game state views. Moreover, with 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 here considering. The exchanged messages could hence raise to the order of $O(N^2)$. Finally, authentication, cheating, and general consensus among all the peers are harder to be addressed than when a centralized architecture is employed.

Mirrored game server architectures represent a hybrid solution which efficiently embraces all the positive aspects of both centralized client-server and fully distributed architectures [12]. 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 in distributing the traffic over the system and reduces the processing burden at each node. Moreover, having each player connected to a close GSS reduces the impact of the player-dependent access technology (e.g., dial-up, cable, DSL) on the total delay experienced [13]. In this case, in fact, the communication among players results 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 one in finding better solutions for the tradeoff among interactivity, consistency, fairness and scalability.

Other advantages in employing a mirrored game server architecture are the absence of a single point of failure, the networking complexity maintained at server side, and the possibility to easily implement authentication procedures. 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 involved nodes. Assuming to have N players and M GSSs, for example, the generated game messages amount to $O(N+M)$, which is again $O(N)$ unless considering the unlikely case of having more servers than players.

All these reasons suggest mirrored game server architecture as the most appropriate in order to efficiently manage large-scale distributed games as they embody 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 some events which have become *obsolete* since the arrival of more recent ones that supersede their content. Interested readers may refer to [15] for a deeper discussion on the notion of obsolescence and a way to include it in game events. The second component takes advantage of this reduced transmission time to magnify the efficiency of a local lag-type of algorithm to ensure fairness. FILA utilizes (4) to determine the display time of a game event; it thus ensures fairness without compromising interactivity.

To calculate the appropriate δ in (3), *OD* should be determined for each player. For this reason, game events are marked at their creation with a generation timestamp and then sent to the destination. Obviously, a global concept of time has to be maintained in the system. This can be achieved through a variety of solutions that enable the synchronization of GSSs' physical clocks [21, 22], or by employing new technological synchronization devices such as GPS. Thanks to this, GSSs are able to monitor the *ODs* of their engaged players and make them available for the FILA algorithm.

The first part of FILA takes inspiration from Active Queuing Management techniques [25] and drops queued obsolete game events with a certain probability p_d when the average *OD* (*avgOD*) value increases putting at risk the interactivity of the system. The discarding probability p_d is directly proportional to *avgOD* and dependent on a constant *Pmax* as described in [23, 24]. Instead, the value for *avgOD*, at iteration n, is computed through the low-pass filter showed below, where w is a parameter that determines how close the average follows the *sample* trend:

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

More in detail, with FILA, all the game events are regularly processed and forwarded while *avgOD* is smaller than an alert threshold named *tmin*. As soon as *avgOD* exceeds *tmin*, the GSSs drop obsolete events with probability p_d, while neither processing nor forwarding them. Finally, if *avgOD* exceeds the subsequent

tmax (>*tmin*) threshold, then p_d is set equal to 1 and all obsolete events are discarded that are waiting for being processed.

This stabilization mechanism succeeds in reducing $OD_i(e)$ as the time spent in queue by a certain event is diminished by the spared processing time of preceding obsolete events which have been dropped without processing nor forwarding them. Moreover, since only obsolete events are discarded, FILA fully maintains consistency in the game evolution [9, 23].

To explain FILA in more details we use the clarifying help of Fig. 1 which provides the graphical definitions for some terms utilized in our explanation: *OD*, *ND* (*Network Delay*), and *LHD* (*Last Hop Delay*).

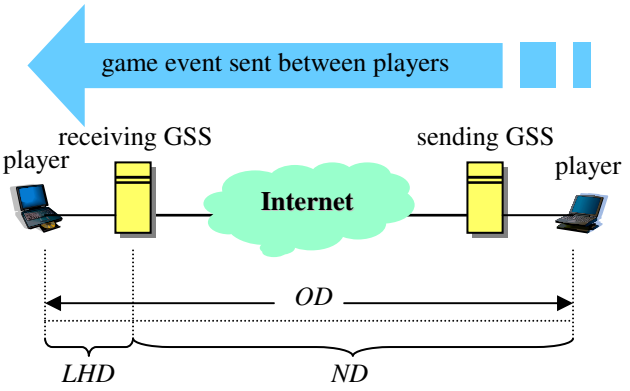


Figure 1. Delay definitions.

First of all, it should be noticed that FILA performs its operations on the receiving GSSs. This choice helps us in maintaining a simpler control of the exploited game platform. Under that circumstance, however, for each event *e*, GSSs can compute *ND(e)* but not *LHD(e)*. However an estimation of *LHD(e)* is necessary in order to compute *OD* and utilize it in the algorithm. For this reason, each GSS continuously monitors the latencies to each of its engaged players and maintains a variable named λ_{GSS}. The value of this variable represents the maximum among the latencies from the considered GSS to each of its connected clients (this set of clients is named *C_GSS*) and is as follows:

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

However, we cannot let some irremediably delay-affected client to excessively impact on the calculations performed by our scheme. Utilizing in FILA an excessively high λ_{GSS} generated by some player connected very far away from the GSS, in fact, would result in very high *sample* (and *avgOD*) value with respect to *GIT*. Consequently, FILA would increase the aggressiveness of its discarding function as perceived by *all* the players with no positive results (the "unlucky" player would still not be able to receive game events with delays below the interactivity threshold).

For this reason, we need to consider a *Delay Upper Bound* (*DUB*) that is used by FILA to limit the impact of "unlucky" players on the algorithm. To this aim, we provide below the computation of a formula for a fundamental parameter utilized by FILA to handle the impact of *LHD(e)* on the algorithm:

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

The usage of this parameter σ varies depending on the employed version of our scheme as detailed in Section 5.

To determine *DUB*, instead, we rely on a heuristic that dynamically computes its value based on the general network condition during the game. Its formula is as follows:

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

where max{*ND*} represents the largest among the *NDs* experienced over all the connections within the entire network.

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

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

5. IS FILA A GOOD SOLUTION?

We are going now to 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 different 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 compared schemes, the algorithm is not allowed to introduce artificial delays if this would result 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 possible versions of this scheme. With this algorithm, in fact, no *avgOD* is maintained and no p_d is calculated. For coherence 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 (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* has still to travel from the considered GSS to the final players. For this reason, our scheme takes into consideration 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:

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

When (9) fails, instead, *all* obsolete events in queue are discarded while neither processing nor forwarding them.

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

Finally, FILA-C takes into direct account the estimation of the $LHD_i(e)$ values when generating *sample*. Therefore, 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 if similar, the three versions of FILA present substantial differences. In essence, FILA-A is an aggressive, yet late, approach according to which all the queued obsolete events are discarded only when interactivity and fairness have already been lost. FILA-B and FILA-C, instead, try to avoid the loss of these two properties by preemptively discarding some obsolete game events when the delay trend increases over the alert threshold *tmin*. We may hence 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.

Indeed, this is a desirable property. In fact, even if obsolete events can be sacrificed, they are still part of the game 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 evolution may be generated.

Focusing on contrasting FILA-B against FILA-C, we notice that the former includes σ in the *tmax* threshold, while the latter utilizes it to compute the *sample* value. Apparently, the impact of σ with FILA-C should be smoothened by the low pass filter (5). However, since *sample* is used to compute *avgOD*, if *sample* is steadily augmented by σ, then even *avgOD* results higher. Reminding now that the discarding probability p_d is directly proportional to *avgOD*, it becomes evident how the use of (10) results in higher p_d values. We hence expect to find FILA-C more aggressive than FILA-B (with a larger amount of dropped events).

6. EXPERIMENTAL ARCHITECTURE

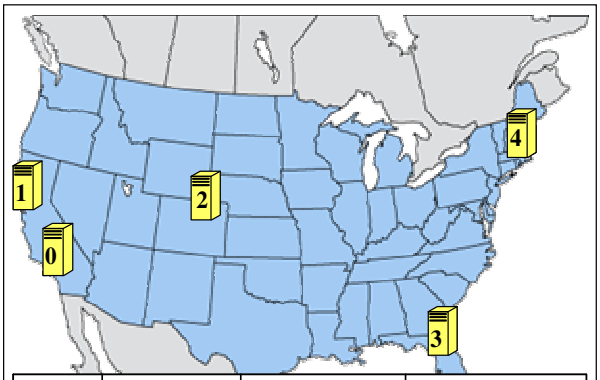
It is well known that MMOG service providers should appropriately position their game servers in such a way that their target player market would be located within a circle having 150-180ms of latency diameter [3]. Following this rule, we have simulated the deployment of five GSSs across U.S.A. positioned in optimal locations and presenting communication latencies among each other reasonably chosen to provide an adequate support for a highly interactive MMOG, in terms of both distance among the various nodes and number of customers that can be served. Clients are thus supposed to be distributed all over the North American continent connecting to one of those GSSs through various access technologies and enduring different access delays.

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

Following the literature [16], *ND* values among GSSs have been generated based on a lognormal distribution whose approximate average was taken from repeated runs of the *ping* application. More in detail, game events coming from players connected to the sending GSSs (i.e. GSS_1–GSS_4) and traveling towards GSS_0 experience latencies with an average as reported in Fig. 2 and a standard deviation of 10ms.

Further, several scenarios were considered where 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 having a maximum latency diameter of 150ms. We assumed to have clients connected to each GSS, engaged in a fast-paced game, and generating a new action every 300ms in average. Online game literature also leaded us to utilize 200B as the average game event size [11].

In MMOGs, not all queued or old game events become obsolete during the game evolution. Their content may be significant and able to drastically alter the game state evolution [15]. 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, was set to 90%. This represents a realistic scenario for a vast plethora of possible games (i.e., 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, such as during collisions or shots, and may represent even less than the 10% of the whole set of game events. A coherent and more detailed numerical demonstration for this claim can be found in [24].



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

Figure 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 [11, 17] 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 have run several set of experiments varying the *GIT* from 150ms to 300ms in order to take into account different kinds of games [6, 17-19]. Each experiment was identically replicated to have a significant comparison among the outcomes of the three versions of FILA, plus the regular local lag algorithm. In [4], Zander *et al.* showed 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 as suggested by (4) thus achieving both fairness and interactivity in their delivery to all players.

7. RESULTS

7.1 Interactivity & Fairness

The charts in Fig. 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 (besides, consistency was maintained as well). In particular, four different graphs are presented reporting results coming from employing respectively: (a) 150ms, (b) 200ms, (c) 250ms, and (d) 300ms as *GIT*. The AIDT value, instead, was set equal to 30ms at each sending GSS for all the four set of experiments (a), (b), (c) and (d). Each set of experiment was comprised of six different experiments. Each experiment consisted in the transmission of about 4000 game events which experienced, in the worst case, a maximal overall latency whose value is reported on the x-axis of each provided chart.

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

Obviously, having a higher *GIT* improves the efficacy of all the evaluated schemes since larger local lags can be utilized. However, LL experiences a premature performance decrease 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. Obviously, in those configurations where the end-to-end latency is close to, or surpasses, *GIT* (end-to-end latency ≥ 140ms in Fig. 3.a and end-to-end latency ≥ 190ms in Fig. 3.b), all the schemes are unable to overwhelm network conditions thus achieving poor results. Even in this case, however, FILA behaves better than LL.

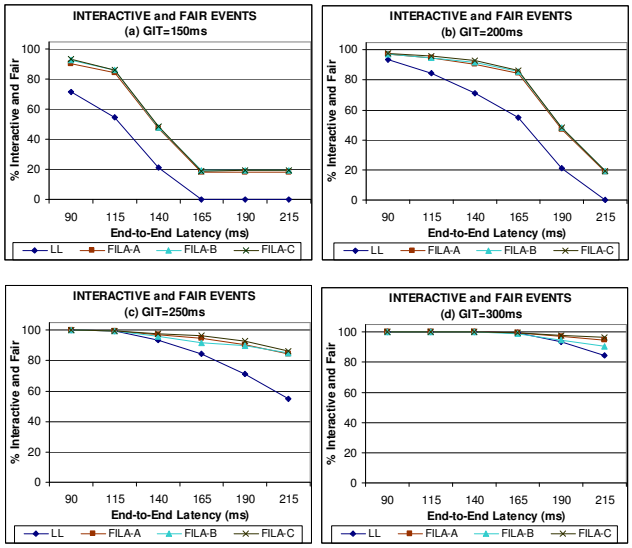


Figure 3 (a, b, c, d). Interactivity and fairness improvement with AIDT equal to 30ms.

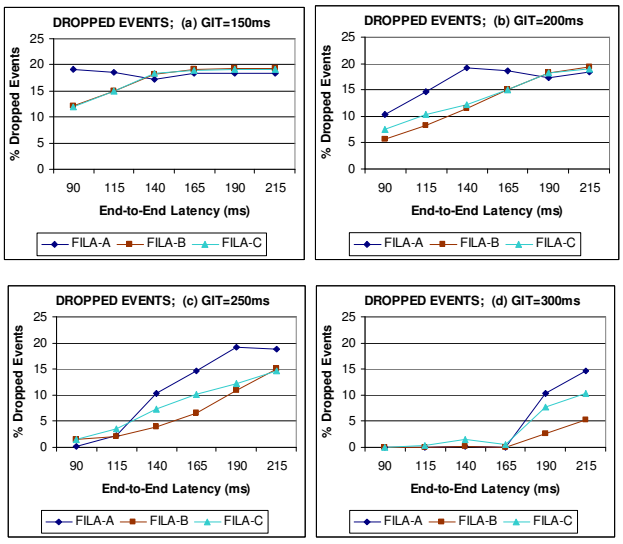


Figure 4 (a, b, c, d). Obsolete events dropped by FILA with AIDT equal to 30ms.

7.2 On Dropping Game Events

FILA pays its better results with the drops of some obsolete events. Specifically, Fig. 4 shows the percentage of game events which have been discarded by the various versions of FILA. However, in all the considered cases, less than 20% of the game events have been dropped. This represents an acceptable value since these events are exclusively obsolete ones.

FILA-A resorts to discarding obsolete events only when interactivity and fairness have already been lost, moreover, at that point, it takes action by discarding all the obsolete events in the queue. Therefore, this version of FILA presents the higher dropping percentage among the three in most of the configurations. Moreover, as expected, FILA-C behaves more aggressively than FILA-B and generally discards a higher number of obsolete events.

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

7.3 Scalability Issues

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

As one can expect, the higher the game traffic, the lower the interactivity and fairness degree provided by LL (see Fig. 5 and Fig. 6). Instead, not only is FILA able to manage higher traffic, but its performance actually improves when AIDT decreases.

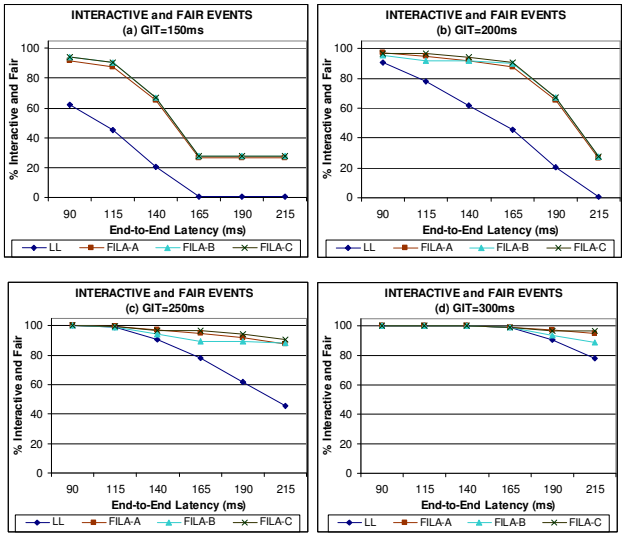


Figure 5 (a, b, c, d). Interactivity and fairness improvement with AIDT equal to 20ms.

This surprising result has a simple explanation. Higher rates in game event transmissions result in generating larger queues at GSSs of packets that have not yet been processed. This amounts to a crucial problem for LL since the queuing time at router increases putting at risk the performance of the system without having any countermeasures. With FILA, instead, a larger queue of game events at a certain GSS represents also 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 traveling towards the various clients.

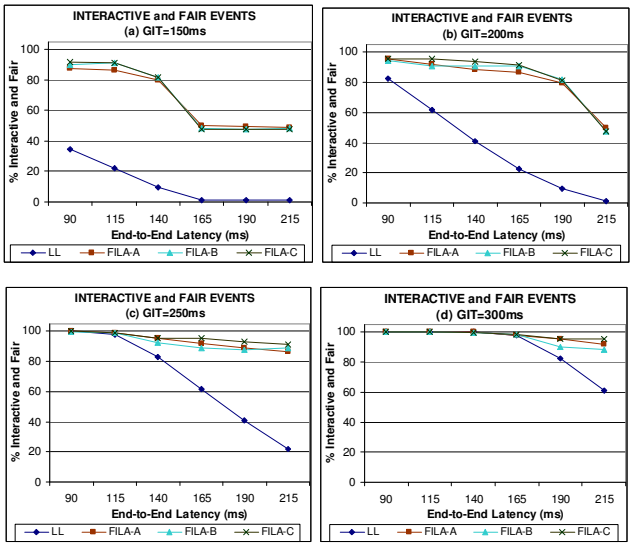


Figure 6 (a, b, c, d). Interactivity and fairness improvement with AIDT equal to 10ms.

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

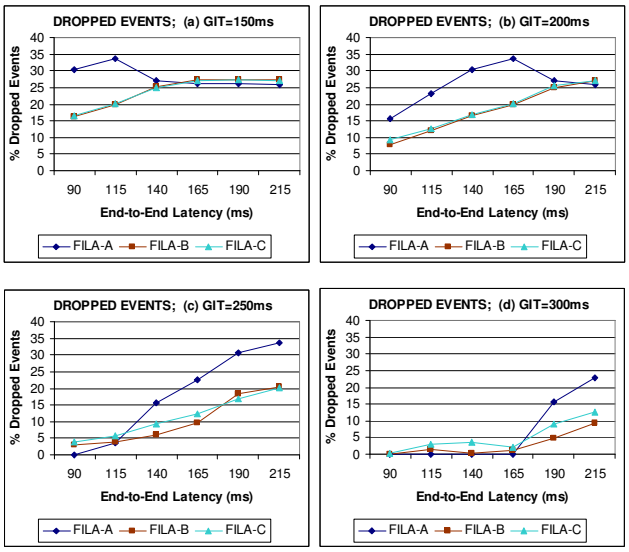


Figure 7 (a, b, c, d). Obsolete events dropped by FILA with AIDT equal to 20ms.

Finally, 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 when this claim is contradicted is when the latency is much lower than the considered *GIT*. In this case, since the high traffic volume in the game network, the *avgOD* can steadily surpass the *tmin*

threshold of FILA-B and FILA-C, making them executing some preemptive drops. At the same time, the large divergence between latency and *GIT* makes the situation where *avgOD* exceeds *GIT* and activates the dropping functionality for FILA-A a very rare event. However, even with these configurations, the percentage of discarded game events remains still very small for all the FILA configurations.

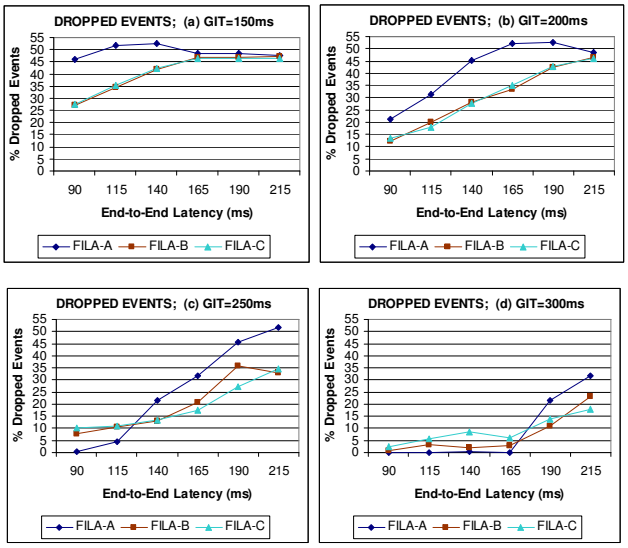


Figure 8 (a, b, c, d). Obsolete events dropped by FILA with AIDT equal to 10ms.

8. CONCLUSION

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

To this aim, we have designed an event delivery scheme, FILA, enforced among replicated game servers which discards obsolete events to ensure interactivity and fairness, whilst preserving consistency. Since only obsolete events are discarded, there is no risk that different dropping percentages at different servers could result in some unfairness [18].

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, exposing advantages and disadvantages of each of them. Finally, we have presented a scalability evaluation of the compared schemes by increasing the game traffic level in the system.

9. ACKNOWLEDGMENTS

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

10. REFERENCES

[1] Early MUD History, http://www.ludd.luth.se/mud/aber/mud-history.html

[2] MMOGCHART.COM, http://www.mmogchart.com

[3] Armitage G. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3. *In Proceedings of the 11th IEEE International Conference on Networks (ICON 2003)*. Sydney, Australia, 2003, 137-141.

[4] Zander S., Leeder I., Armitage G. 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, 2005.

[5] Gautier L., Diot C. Design and Evaluation of MiMaze, a Multiplayer Game on the Internet. *In Proceedings of IEEE Multimedia (ICMCS'98)*. Austin, TX, USA, 1998, 233-236.

[6] Pantel L., Wolf L. C. 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, USA, 2002, 23-29.

[7] Mauve M., Vogel J., Hilt V., Effelsberg W. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia, 6,* 1, (2004), 47-57.

[8] Kim S., Kuester F., Kim K. H. A Global Timestamp-Based Approach to Enhanced Data Consistency and Fairness in Collaborative Virtual Environments. *Multimedia Systems, Springer-Verlag, 10,* 3, (2005), 220-229.

[9] Ferretti S., Palazzi C. E., Roccetti M., Gerla M., Pau G. *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 2006*. Las Vegas, USA, January 2006.

[10] Quake Forge Project, http://www.quakeforge.org

[11] Farber J. Network Game Traffic Modelling. *In Proceedings of the 1st Workshop on Network and System Support for Games (NetGames2002), ACM SIG MM*. Braunschweig, Germany, 2002, 53-57.

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

[13] Jehaes T., De Vleeschauwer D., Coppens T., Van Doorselaer B., Deckers E., Naudts W., Spruyt K., Smets R. Access

Network Delay in Networked Games. *In Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames 2003), ACM SIGCOMM and SIG MM*. Redwood City, CA, USA, 2003, 63-71.

[14] Steinman J. S. Scalable Parallel and Distributed Military Simulations Using the SPEEDES Framework. *In Proceedings of 2nd Electronic Simulation Conference (ELECSIM95)*, Internet, 1995.

[15] Ferretti S., Roccetti M., A Novel Obsolescence-based approach to Event Delivery Synchronization in Multiplayer Games, *International Journal of Intelligent Games and Simulation, 3,* 1, (2004), 7-19.

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

[17] Borella M. S. Source Models for Network Game Traffic. *Computer Communications, Elsevier, 23,* 4, (2000), 403-410.

[18] Zander S., Armitage G. Empirically Measuring the QoS Sensitivity of Interactive Online Game Players. *In Proceedings of Australian Telecommunications Networks & Applications Conference 2004 (ATNAC2004)*. Sydney, Australia, 2004, 511-518.

[19] Fitzek F., Schulte G., Reisslein M. System Architecture for Billing of Multi-Player Games in a Wireless Environment Using GSM/UMTS and WLAN Services. *In Proceedings of the 1st Workshop on Network and System Support for Games (NetGames2002), ACM SIG MM*. Braunschweig, Germany, 2002, 58-64.

[20] Sheldon N., Girard E., Borg S., Claypool M., Agu E. 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), ACM SIGCOMM and SIG MM*. Redwood City, CA, USA, 2003. 3-14.

[21] Mills D. L. Internet Time Synchronization: the Network Time Protocol. *IEEE Transactions on Communications, 39,* 10, (1991), 1482-1493.

[22] Ramanathan P., Shin K. G., Butler R. W. Fault Tolerant Clock Synchronization in Distributed Systems. *IEEE Computer, 23,* 10, (1990), 33-42.

[23] Palazzi C. E., Ferretti S., Cacciaguerra S., Roccetti M. 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, USA, 2004, 157-165.

[24] Palazzi C. E., Ferretti S., Cacciaguerra S., Roccetti M. Interactivity-Loss Avoidance in Event Delivery Synchronization for Mirrored Game Architectures. Accepted with minor revisions in *IEEE Transactions on Multimedia*.

[25] Floyd S., Jacobson V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking, 1,* 4, (1993), 397-413.

[26] Palazzi C. E., Ferretti S., Cacciaguerra S., Roccetti M., ''A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games", *ACM Journal of Computers in Entertainment, ACM Press, 3,* 2, (2005), 1-11.