Interactivity-Loss Avoidance 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

Since the expansion of their market and their challenging requirements, Massively Multiplayer Online Games are gaining increasing attention in the scientific community. One of the key factors in this kind of application is represented by the ability to rapidly deliver game events among the various players over the network. Employing in this context Mirrored Game Server architecture and adapting RED (Random Early Detection) techniques borrowed from network queuing management, we are able to show sensible benefits in upholding interactivity and scalability, whilst preserving game state consistency and game evolution fluency at the player's side.

Keywords
Massively Multiplayer Online Game, Online Entertainment, Event Delivery Service, Interactivity, Consistency.

EDIC: 7-CONS, 3-VRAR, 3-QUAL.

1. Introduction

In the last decade, thanks to their impressive progression in plunging players into terrifically realistic and capturing virtual worlds, online games have expanded their market with a persistent and accelerating growth. Nowadays, Massively Multiplayer Online Games (MMOGs) are further extending the boundaries of what has been defined "the tenth art" with the possibility of contemporary engaging millions of players located all over the world. Unfortunately, MMOG is a highly interactive application, characterized by strict real time requirements, harder than those accomplishable by traditional Internet protocols [1].

It is widely accepted that a suitable architecture able to efficiently manage large-scale distributed games may make use of a constellation of mirrored *Game State Servers* (GSSs), which are deployed over the network in a limited number [2]. GSSs maintain replicas of the same game state view. Having multiple replica servers allows each client to connect in a Client-Server fashion to the *closest mirror*, thus reducing the communication latency. Each GSS assembles all game events of its engaged players, creates

* Contact Author.

an updated game state, and then forwards it to all the other GSS peers. Prominent advantages of this approach are the absence of a single point of failure, having the networking complexity maintained by the servers, and the possibility to implement authentication. Nevertheless, synchronization schemes are still required to ensure the global consistency of the game state hold by the various servers.

Highly interactive applications as MMOGs are extremely sensitive to delays in event deliveries. In case of an intense traffic in the network or when excessive computational loads are slowing down some GSSs, the responsiveness of the system may be jeopardized. A proficient synchronization algorithm for MMOGs should be able to face both these two situations in order to preserve a high level of interactivity and an identical contemporary view of the game state among all the nodes in the system.

We present a novel synchronization mechanism able to uplift the playability degree of MMOGs by maintaining the event delivery delays under a human-perceptivity threshold. The core element at the basis of our scheme is the adoption of proactive queue management techniques [3] combined with the use of the semantics of the game [4]. In essence, events that can be considered *obsolete* since the arrival of *fresher* ones may be discarded at a given GSS utilizing a dropping probability which depends on the perceived responsiveness at that GSS. Limiting the number of game events in the system provides two beneficial results: it alleviates the processing burden and reduces the impact of network latency on playability. As a consequence, the game event delivery time results minimized.

The remainder of this paper is organized as follows. Section 2 reviews principal works in the field of online game interactivity. In Section 3, we discuss some design issues at the basis of our work and in Section 4 we present our novel scheme. The simulative environment and the corresponding results are provided, respectively, in Sections 5 and 6. Finally, Section 7 concludes the paper.

2. Related Work

Several research works have already brought contributions to the factual development of efficient synchronization schemes. *Compression* and *aggregation* consider networking having a dominant position when dealing with the delays and thus with the playability of a MMOG [5]. These schemes, however, pay

the achieved communication latency benefits with increased computational costs. Moreover, aggregation wastes time delaying the transmission of the available events while waiting for other ones 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 for games where generated events are relevant only for a small fraction of users [6]. A tradeoff relationship exists between the amount of computation spared at the destination by receiving only a limited number of packets and that one expended at the sending GSS to implement the filtering scheme.

Slightly detaching playability from the real responsiveness of the network, optimistic algorithms can be utilized to avoid delay perception. In case of lousy interactivity among GSSs, in fact, an optimistic approach executes events before really knowing if ordering would require processing other on-the-way events first. Game instances are thus processed without waiting for other possibly coming packets. *Rollback* based techniques are exploited to reestablish the consistency of the game state. *Time Warp* and *Trailing State Synchronization* represent exemplars of this class of algorithms [2, 7].

Dead Reckoning is a latency hiding method that utilizes a reduced frequency in sending update packets while compensating the lack of information with prediction techniques [8]. However, predicted actions are not always trustworthy. Therefore, convergence techniques have sometimes to be exploited to recover from provisional instances of the game having some momentary inconsistencies.

3. Background

3.1. Interactivity vs Consistency

Distributed multiplayer games are characterized by two main requirements which cannot be considered as 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. Indeed, in order to guarantee interactivity, the external stimuli generated by players have to be processed under a human-perceptivity threshold.

Consistency, instead, regards the contemporary uniformity of the game state view in all the nodes of

the system. The easiest way to guarantee full consistency is to make the game proceed through discrete *locksteps*. At each step, each node waits until having received all the actions from the other participants; then a new instance of the game is produced. However, the larger the delay while waiting for actions from all the players, the smaller the interactivity degree. This is a factual demonstration that a tradeoff relationship exists between interactivity and full consistency when designing online game schemes.

3.2. Obsolescence and Correlation

Full Consistency can be attained through the use of a completely reliable, totally ordered event delivery scheme [9]. On the other hand, this approach increments complexity and delays. Waiting for the arrival of the next action in the processing sequence, while having successive events ready in queue, may sensibly slow down the evolution of the game, jeopardizing interactivity.

In recent studies, we have demonstrated that exploiting the semantics of the game can be put to good use to relax the requirements for reliability and total order delivery, thus augmenting interactivity [4, 10]. Some events, in fact, can lose their significance as time passes: new actions may make the previous ones irrelevant. For example, where there is a rapid succession of movements by a single agent in a virtual world, the event representing the last destination supersedes the older ones. *Obsolescence* can hence be defined as the relation between two received events e' and e'', generated at different times $t' < t''$, by which the existence of event e'' diminishes the importance of e' and the need for its processing, without affecting consistency. Dropping obsolete events instead of processing and forwarding them to other nodes clearly reduces the delays, thus speeding up the execution of fresher events.

To define a game event as obsolete, we have to be sure that consistency would not be weakened. To this aim, we also have to take into consideration the notion of *correlation*. Two events, say e' and e^*, are correlated if the final game state depends on their execution order. Correlation has to be taken into account to determine the obsolescence of an event. In fact, it might be the case when e'' would make obsolete a previous event e' but a further event e^*, correlated to e' and temporary interleaved between e' and e'', breaks this relationship of obsolescence. Hence, correlated events are the only ones that really need to be delivered in the same order as generated. Instead, the total order delivery requirement can be

relaxed in case of non-correlated events.

Finally, it is worth mentioning that other alternative event ordering strategies exist (e.g., causal and FIFO orders). However, they are generally not suitable for capturing the peculiar notions of obsolescence and correlation. Interested readers may refer to [4] where examples are provided of correlated events that are also concurrent and hence cannot be delivered according to FIFO or causal ordering strategies.

4. A Novel Scheme for Interactivity-Loss Avoidance: ILA-RED

Hereinafter, we present two possible ways to exploit the notions of obsolescence and correlation to gain interactivity: Interactivity Restoring and Interactivity-Loss Avoidance.

4.1. Interactivity Restoring: *"Healing after illness"*

In a previous study, an obsolescence-based scheme was developed aimed at restoring interactivity *after* having already lost it [4]. Specifically, according to this scheme (hereinafter referred to as ON-OFF), during the game event delivery, each GSS measures the interval of time between the generation and the arrival of every game event it receives; the resulting value is named *Game Time Difference* (GTD). This GTD is then compared to a predefined constant named *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 obsolete messages so as to bring the GTD back within the GIT (referred to as phase ON). Since only obsolete events are discarded, this stabilization mechanism reduces the GTD without causing inconsistencies in the game evolution. Needless to say, a global conception of time must be maintained by all the GSSs, for example by exploiting a variety of solutions able to synchronize their physical clocks [11, 12], or by employing new technological synchronization devices such as GPS.

4.2. Interactivity-Loss Avoidance: *"Prevention is better than cure"*

Interactivity-Loss Avoidance takes inspiration from the well known *Random Early Detection* (RED) algorithm, a proactive congestion avoidance mechanism enforced at routers [3], and tries to preempt the loss of interactivity *before* it may happen (hereinafter the scheme is referred to as ILA-RED).

In particular, according to ILA-RED (see Fig. 1), when receiving a packet, each GSS determines the GTD of the relative event and feeds a low pass filter to update the average of the GTDs, called *avg_GTD*. When *avg_GTD* exceeds the *min* threshold (referred to as phase 1), the GSS drops obsolete events with a certain probability *p*, while neither processing nor forwarding them. If *avg_GTD* exceeds the subsequent *max* limit (referred to as phase 2), *p* is set equal to *1*, and all the obsolete events waiting for being processed are discarded; *max* represents the human perceptivity threshold and thus corresponds to the aforementioned GIT. The dropping probability *p* is a function of *avg_GTD* and *Pmax* as shown in (1).

$$p = \frac{Pmax \times (avg\_GTD - min)}{(max - min)} \quad (1)$$

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 preventing the loss of an adequate level of time interaction between servers. Interested readers can find a detailed rationale for the algorithm design choices in [10].

5. ILA-RED: A Simulation Assessment

To evaluate our scheme, we have simulated a general Mirrored Game Server architecture comprising a variable number of GSSs. 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 others are sending events to it. Following the literature [13], 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. 2. We carried out several simulation experiments (30) varying the number of sender servers from 4 to 7.

Based on the assumption that each client connects to the closest GSS to limit the impact of the client-GSS latency on the total delay experienced by game events, we have assumed to have the client-GSS latency portion already comprised in the network latencies among the GSSs. The event generation rate at each GSS follows a lognormal distribution as inspired by literature about games and varies from a normal traffic situation to an intensively loaded one [14]. In particular, different experiments have been conducted with an Average Interval Departure Time (AIDT) between two subsequent events varying

from 30ms (intense game traffic) to 60ms (moderate game traffic), while having the standard deviation constantly equal to 10ms. Finally, the average event size was set equal to 200 Bytes.

The probability that an event makes obsolete preceding ones was set to 90%. This represents a realistic scenario for a vast plethora of possible games (i.e. adventure, strategic, car race, flight simulator, etc.), where most of the events are just independent moves. In other words, critical (correlated) 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.

As a confirmation of this claim, an extensive study of players' behavior on Quake 3 is presented in [15]. In that paper, a measure of the average number of kill actions per minute as a function of the median ping time between client and server is reported. In particular, it is shown that when the 80% of players is located within 180ms of range from the server, an average of kill actions per minute is achieved which varies in the interval from 1.60 to 3.25. Considering the median of this interval (2.425) we obtain 0.04 kill events per second. With an AIDT of 60ms at client side [14, 16], 16.67 game events are generated every second. Therefore, the resulting percentage of kill events over the whole set of game actions amounts to just 0.24%. Pessimistically assuming that a kill event can be issued only after an average number of 40 correlated actions (e.g., various shots, movements of the character into the location where it will be shot or out of the position where it would have been shot) we get 9.6% of critical events. Therefore, 10% of correlated events and 90% of obsolescence probability represent a realistic configuration for online games simulations. Nevertheless, for the sake of completeness, we have also carried out a set of experiments with an obsolescence probability set equal to 50%. Obviously, an obsolescence probability of 50% implies an even more frantic game with an incredible increase of critical (correlated) game actions.

Parameters highlighted in Fig. 1 were set as follows: $Pmax$ = 0.2, min = 50ms and max = 150ms (equivalent to the GIT in the ON-OFF scheme). Special attention should be devoted to the chosen value for max. Recent studies, in fact, have demonstrated that players who are actively involved in a fast paced online game must experience network delays that lie below 150ms [15–17]. Further, based on these studies, it is now common knowledge that a game service provider should place game servers armed with

a network radius around 150-180ms on the Internet to satisfy their target player market. The contribution of our scheme becomes relevant whenever a distributed game architecture is deployed based on the above assumptions of topological locality. Obviously, if network latencies massively surpass the 150ms threshold, interactivity is hardly guaranteed. Nevertheless, also in this negative case, our proposed scheme is useful to alleviate the problem. Moreover, the 150ms limit holds for games like vehicle racing, first person shooter, etc., but could be increased to 300ms in case of strategic and role play games [17].

We have replicated each experiment to compare the outcomes of three different synchronization schemes: our proposed ILA-RED scheme, the ON-OFF mechanism [4], and the conventional OFF approach (having neither obsolete events drops nor other algorithms to restore interactivity).

6. ILA-RED: Simulation Results

We intend to demonstrate the benefits that can be attained by implementing an event-discarding algorithm in case of a trend toward an increase in GTDs. To this aim, in Fig. 3 we compare the percentage of events arrived at GSS_0 with a GTD value larger than the GIT; in other words, the *non-interactive events*. As observable, both ILA-RED and ON-OFF outperform the conventional OFF scheme, independently of the number of sending GSSs.

6.1. ILA-RED vs ON-OFF: A Comparative Evaluation

The interactivity results above were obtained by ILA-RED with a percentage of discarded events ranging from 10% to 14% (depending on the number of the servers). Slightly worse interactivity was obtained by ON-OFF at a cost of from 25% to 35% of discarded events. This is a very important advantage obtained when utilizing ILA-RED w.r.t. ON-OFF. In fact, even if obsolete events can be sacrificed to gain a better interactivity, they are still part of the game visual progression. Dropping too many obsolete events could result in jerky rendering and temporary interruptions of the image/video flow on the player's screen. These artifacts and unpredictable gaps in the game evolution could be annoying for customers and should be avoided whenever possible.

Another advantage of ILA-RED, when compared to ON-OFF, is represented by its ability in

guaranteeing a more uniform and fluent progression of the game. In essence, event drops in ILA-RED are not concentrated only during phase 2: they rather fall more uniformly over the whole game session through some phase 1 interludes. These benefits are confirmed by our measurements. In Fig. 4, the great difference between the histograms clearly shows that the preemptive, probabilistic dropping action of ILA-RED in phase 1 strongly reduces the number of times interactivity is lost and needs to be recovered by resorting to phase 2, where, instead, all obsolete events are discarded (with a probability equal to 1).

Finally, Fig. 5 shows the occurrence of bursts of non-interactive events for both ILA-RED and ON-OFF. This represents the number of times that two or more game events were consecutively received with a GTD greater than the GIT. As Fig. 5 demonstrates, the number of these bursts during the game is sensibly smaller when ILA-RED is employed.

6.2. Sensitivity Analysis as a Function of the Game Traffic Intensity

We aim at finding the breakeven point in the performance curves between ILA-RED and ON-OFF. As can be seen in Fig. 6, the difference in the number of discarded game events between ILA-RED and ON-OFF progressively diminishes as we increase the AIDT value. With AIDT = 60ms, in fact, these two synchronization schemes present comparable amounts of discarded events; moreover, the great advantage against the OFF scheme has been sensibly reduced (see Fig. 7). Nevertheless, obsolescence based discarding schemes still present lower average and standard deviation of the GTDs, as shown in Table 1.

Two main reasons lie at the basis of this outcome. First, having lower game event generation rates decreases both latencies in the network and queuing time at the receiving GSS, thus naturally improving the interactivity degree, yet without any external intervention. Second, both ILA-RED and ON-OFF schemes rely on discarding obsolete events when the interactivity level decreases. Consequently, to be effective, these schemes require droppable obsolete events queued at the GSS while the server is experiencing lousy interactivity. This accumulation of events waiting for being processed is more likely to happen with more intense game traffic (i.e., lower AIDT values).

6.3. Sensitivity Analysis as a Function of the Obsolescence Probability

ILA-RED requires the presence of droppable obsolete game events to take action. Therefore, not only are its benefits more evident in case of high game traffic, but its efficacy depends on the percentage of potential obsolete events. This claim is confirmed by Fig. 8, where we report the outcome of our simulations in terms of non-interactive events, as a function of different obsolescence probabilities (i.e., 50%, 90%). In order to understand more clearly where the performance improvement of both ON-OFF and ILA-RED over OFF scheme is coming from, we also present in Fig. 9 the average waiting time that game events experience in queue at GSS_0 before being processed. Finally, the corresponding percentages of discarded game events are reported in Fig. 10.

7. Conclusion and Future Work

To guarantee a pleasant game experience to online players engaged by MMOGs, a high interactivity degree, as well as a uniform view of the game, has to be provided. To this aim, an efficient synchronization scheme among Mirrored Game Servers should be implemented as basic solution. We have proposed a proactive event discarding mechanism, named ILA-RED, which relies on the discrimination of obsolete events as an innovative way to meet the aforementioned requirements. Experimental results confirmed the efficacy of our scheme.

Finally, one of the main contributions of our work is represented by the original idea of adapting proactive queue management techniques to maintain interactivity in MMOGs. To this aim, the utilization of RED-like techniques should be considered as a proof-of-concept and several alternatives may be considered [18–21]. In particular, a scheme that permits to determine the target queue size as, for instance, Adaptive RED [22] may represent a good candidate for future investigations in the field of online games.

Acknowledgments

This research was conducted with a financial support from the Italian MIUR via the Interlink Project. We are indebted to the anonymous referees of the IEEE Transactions on Multimedia for their helpful review of this article.

References

[1] S. Wright, S. Tischer, “Architectural Considerations in Online Game Services over DSL Networks”, in Proc. IEEE International Conference on Communications - (ICC'04), IEEE Communications Society, Paris, France, 2004, pp.1380-1385.

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

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

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

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

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

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

[8] S. Srinivasan, “Efficient data consistency in HLA/DIS++,” in Proc. 1996 Winter Simulation Conf., 1996, pp. 946–951.

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

[10] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures”, in Proc. of the 1st IEEE NIME'04, GLOBECOM 2004, Dallas, TX, 2004, pp.157-165.

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

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

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

[14] J. Farber, “Network Game Traffic Modelling”, in Proc. of NetGames2002, Braunschweig, Germany, 2000, pp.53-57.

[15] G. Armitage, “An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3”, in Proc. of ICON, Sydney, Australia, 2003, pp.137-141.

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

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

[18] W. Feng, D. Kandlur, D. Saha, K. Shin, “A Self-Configuring RED Gateway”, in Proc. of IEEE Infocom, New York, NY, 1999, pp.1320-1328.

[19] W. Feng, D. Kandlur, D. Saha, K. Shin, “Blue: A New Class of Active Queue Management Algorithms,” Tech. Rep. UM CSE-TR-387-99, 1999.

[20] D. Lapsey, S. Low, “Random Early Marking for Internet Congestion Control”, in Proc. of IEEE Globecom, Rio de Janeiro, Brasil, 1999, pp.1747–1752.

[21] S. S. Kunniyur, R. Srikant, “An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management”, IEEE/ACM Transactions on Networking, vol.12, no.2, 2004, pp.286–299.

[22] S. Floyd and R. Gummadi and S. Shenker, “Adaptive RED: An Algorithm for Increasing the Robustness of RED”, http://www.icir.org/floyd/papers/adaptiveRed.pdf.

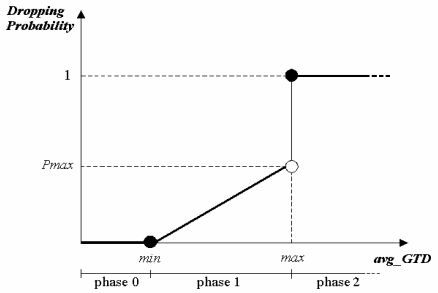

Figure 1: Discarding probability function.

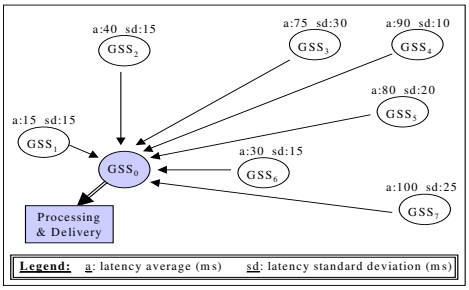

Figure 2: The adopted configuration.

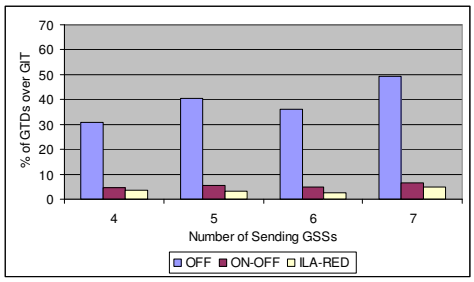

Figure 3: Percentage of events with GTD over GIT; AIDT = 30ms.

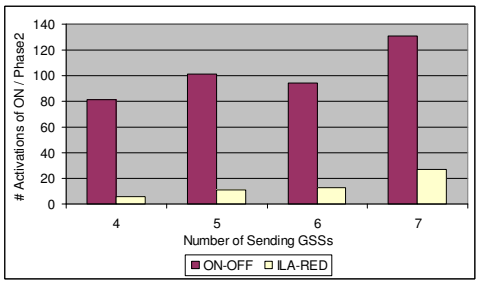

Figure 4: # of activations of phase ON and phase 2 for ON-OFF and ILA-RED respectively; AIDT = 30ms.

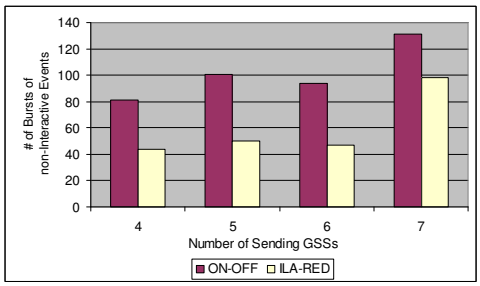

Figure 5: # of bursts of non-interactive events; AIDT = 30ms.

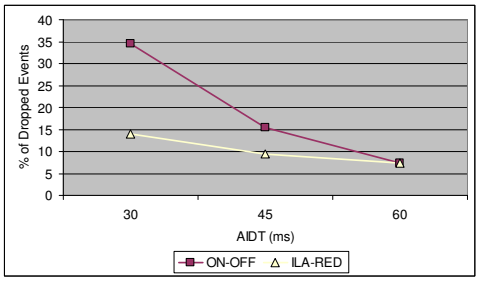

Figure 6: Percentage of dropped events; 7 sending GSSs.

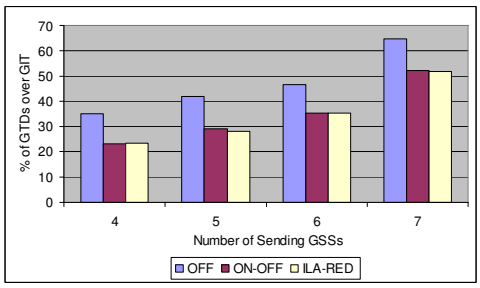

Figure 7: Percentage of events with GTD over GIT; AIDT = 60ms.

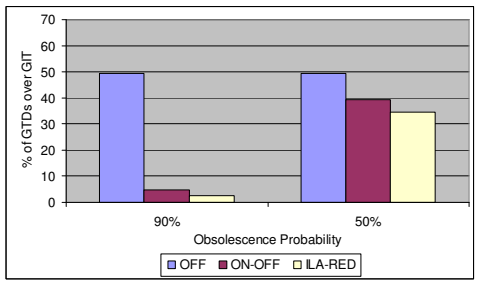

Figure 8: Percentage of events with GTD over GIT; AIDT = 30ms; 7 sending GSSs.

	4 sending GSSs			5 sending GSSs			6 sending GSSs			7 sending GSSs		
	OFF	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**
MAX	328	326	326	328	325	325	319	286	286	340	328	325
MIN	88	88	88	88	88	88	87	87	86	92	91	90
AVG	144	133	133	154	142	142	153	142	142	169	155	154
ST DEV	42	29	29	41	29	28	40	29	29	42	27	27

Table 1: Maximum, minimum, average and standard deviation of the GTDs (ms); AIDT = 60ms.

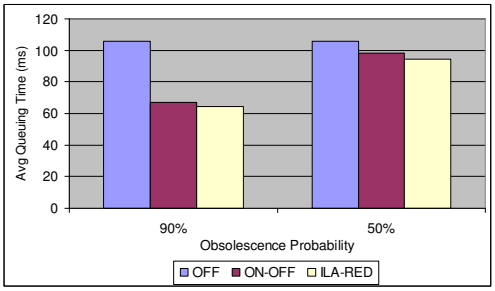

Figure 9: Average queuing time of game events at GSS_0.

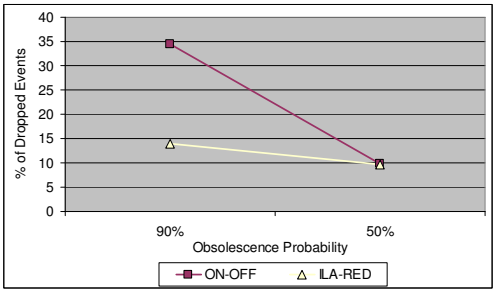

Figure 10: Percentage of dropped events; AIDT = 30ms; 7 sending GSSs.